Mini2440 之 U-boot 使用及移植详细手册
2010-4-16

简介

有很多网友基于 mini2440 移植了 U-boot，不过大都是功能比较简单，我们发现网友 Tekkaman 移植的 U-boot 集成了很多其他版本 U-boot 的优点，比如支持 SD 卡、优盘、开机 Logo、USB 下载等，这使得 U-boot 更加方便易用且实用，这和我们公司对产品设计的追求理念是一致的。遂与其取得联系，并于 2009 年底并提供了各个版本的 mini2440 样品以供测试，因此你现在可以在 Tekkaman 的代码仓库中下载到支持当前所有 Nand Flash 容量版本 (64M/128M/256M/1GB)的 U-boot 源代码，它完全适用于 mini2440/micro2440。

对于大部分初学者而言，仅仅会下载和编译使用别人移植好的软件或许还是不够的，因此我们根据网友 Tekkaman 的博客文章为材料，并做了适当的整理和补充，得以形成此手册，以便其他人学习参考，在此我们十分感谢 Tekkaman 对此付出的努力和贡献。也许有不少网友已经浏览过他的博客，上面已经有一部分移植的说明文档，但此手册介绍的内容将会更加系统和细致，因此十分适合打算采用 U-boot 的初学者。

其实我们公司也提供了其他开源的 bootloader，比如 vboot 和 nboot，它们都十分短小精悍，分别适用于 Linux 和 WinCE 系统，我们认为这对于研究 bootloader 原理和实际的项目应用更加合适。基于习惯(Linux 爱好者和 Windows 用户向来是有水火不相容的斗争意识)考虑，vboot 是基于 ARM Gcc 开发制作的，nboot 是基于 ADS 开发制作的。

本手册目前主要由友善之臂整理、审阅和发布，我们希望本手册遵循开源软件之自由和共享的核心精神，但为了避免争议，友善之臂保留本手册的一切解释权，本公司及作者并不提供关于该手册的任何技术支持。如果你对本手册的内容有任何疑问，可以到 arm9 之家论坛反馈和交流讨论，我们将会在以后尽量更正，更新后的手册也将在该论坛发布。

另外，本手册是以首页的日期为版本标志的。

广州友善之臂计算机科技有限公司
网址：http://www.arm9.net
ARM9 之家论坛
网址：http://www.arm9home.net

第1章　BootLoader 的概念与功能

1.1 嵌入式 Linux 软件结构与分布

一般情况下嵌入式 Linux 系统中的软件主要分为以下几部分：

1) **引导加载程序**：其中包括内部 ROM 中的固化启动代码和 BootLoader 两部分。内部固化 ROM 是厂家在芯片生产时候固化的，作用基本上是引导 BootLoader。有的芯片比较复杂，比如 Omap3 在 flash 中没有代码的时候有许多启动方式：USB、UART 或以太网等等。而 S3C24x0 则很简单，只有 Norboot 和 Nandboot。

   s3c24x0

2) **Linux kernel 和 drivers**。

3) **文件系统**。包括根文件系统和建立于 Flash 内存设备之上的文件系统（EXT4、UBI、CRAMFS 等等）。它是提供管理系统的各种配置文件以及系统执行用户应用程序的良好运行环境及载体。

4) **应用程序**。用户自定义的应用程序，存放于文件系统之中。

在 Flash 存储器中，他们的分布一般如下：

BootLoader	BootLoader 参数区	Linux Kernel	根文件系统	其他文件系统……(被挂载到根文件系统或者作为
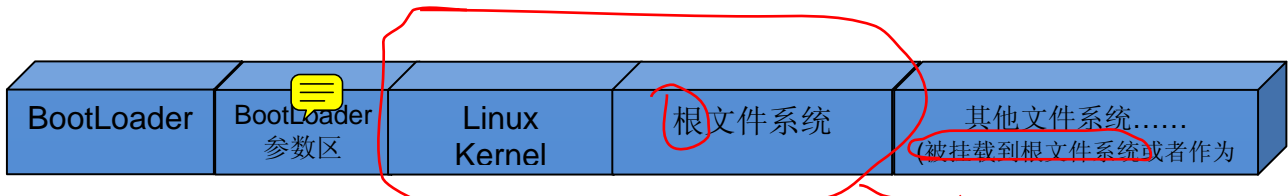
但是以上只是大部分情况下的分布，也有一些可能根文件系统是 initramfs，被一起压缩到了内核映像里，或者没有 Bootloader 参数区，等等。

1.2 在嵌入式 Linux 中 BootLoader 的必要性

bootloader　　　kernel,1　　　　　　2　　cpu　　　　cache,mmu.

Linux 内核的启动除了内核映像必须在主存的适当位置，CPU 还必须具备一定的条件：

1. CPU 寄存器的设置：	R0＝0； R1＝Machine ID（即 Machine Type Number，定义在 linux/arch/arm/tools/mach-types）； R2＝内核启动参数在 RAM 中起始基地址；
2. CPU 模式：	必须禁止中断（IRQs 和 FIQs）； CPU 必须 SVC 模式；
3. Cache 和 MMU 的设置：	MMU 必须关闭； 指令 Cache 可以打开也可以关闭； 数据 Cache 必须关闭；

但是在 CPU 刚上电启动的时候，一般连内存控制器都没有初始化过，根本无法在主存中

运行程序，更不可能处在 Linux 内核启动环境中。为了初始化 CPU 及其他外设，使得 Linux 内核可以在系统主存中运行，并让系统符合 Linux 内核启动的必备条件，必须要有一个先于内核运行的程序，他就是所谓的引导加载程序（Boot Loader）。

　　而 BootLoader 并不是 Linux 才需要，而是几乎所有运行操作系统的设备都需要。我们的 PC 的 BOIS 就是 Boot Loader 的一部分（只是前期引导，后面一般还有外存中的各种 Boot Loader），对于 Linux PC 来说，Boot Loader = BIOS + GRUB/LILO。

1.3 Boot Loader 的功能和选择

　　综上所述：BootLoader 是在操作系统内核启动之前运行的一段小程序。通过这段程序，我们可以初始化硬件设备，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境，最后从别处（Flash、以太网、UART）载入内核映像到主存并跳到入口地址。

　　由于 BootLoader 需要直接操作硬件，所以它严重依赖于硬件，而且依据所引导的操作系统的不同，也有不同的选择。对于嵌入式世界中更是如此。就 S3C24x0 而言，如果是引导 Linux，一般选用韩国的 mizi 公司设计的 vivi 或者 DENX 软件工程中心的 Das U-boot，如果是引导 Win CE，就选用 Eboot。如果是开发 StrongARM 构架下的 LART，就可选用由 Jan-Derk Bakker 和 Erik Mouw 发布的 Blob(Boot Loader Object)。如果是要引导 eCos 系统，可以选用同是 Redhat 公司开发的 Redboot。bootloader,

　　所以在嵌入式世界中建立一个通用的 BootLoader 几乎是不可能的，而可能的是让一个 Boot Loader 代码支持多种不同的构架和操作系统，并让她有很好的可移植性。U-boot 就是支持多平台多操作系统的一个杰出代表。这也是 U-boot 的优势所在，因为如果在开发 S3C2440 时熟悉了 U-boot，再转到别的平台的时候，就可以很快地完成这个平台下 U-boot 的移植。而且 U-boot 的代码结构越来越合理，对于新功能的添加也十分容易。

　　＊推荐阅读：嵌入式系统 Boot Loader 技术内幕

第2章　U-boot 简介

2.1 U-boot 的起源

U-Boot 是 Das U-Boot 的简称，其含义是 Universal Boot Loader，是遵循 GPL 条款的开放源码项目。最早德国 DENX 软件工程中心的 Wolfgang Denk 基于 8xxROM 和 FADSROM 的源码创建了 PPCBoot 工程项目，此后不断添加处理器的支持。而后，Sysgo Gmbh 把 PPCBoot 移植到 ARM 平台上，创建了 ARMBoot 工程项目。最终，以 PPCBoot 工程和 ARMBoot 工程为基础，创建了 U-Boot 工程，2002 年 12 月 17 日第一个版本 U-Boot-0.2.0 发布，同时 PPCBoot 和 ARMBoot 停止维护。

而今，U-Boot 作为一个主流、通用的 BootLoader，成功地被移植到包括 PowerPC、ARM、X86 、MIPS、NIOS、XScale 等主流体系结构上的百种开发板，成为功能最多、灵活性最强，并且开发最积极的开源 BootLoader。目前。U-Boot 仍然由 DENX 的 Wolfgang Denk 维护。

2.2 U-boot 的开发情况和资源

最早 U-boot 的版本号是由 X.Y.Z 来表示的，从 0.2.0 一直发展到 1.3.4。之后便开始使用年份加月份的表示方法，从 2008.11 到现在的 2010.3 平均每 3 个多月出一个新版本。每次代码的结构和定义都会有一些修正和改进，其代码越来越规整，功能越来越强，但是移植的难度反而越来越小，需要修改的地方越来越少。

U-boot不仅有主线版本，在 U-boot的Git代码仓库中还有各个CPU构架的分支版本，这些分支会在一定的时候将修改汇入主线。

下面总结一下关于 U-boot 源代码的网络资源：

官方链接	
德国 DENX 软件工程中心主页	http://www.denx.de/
U-boot 官方主页 (注意其中的 邮件列表链接)	http://www.denx.de/wiki/U-Boot/WebHome
U-boot 官方源码 FTP 下载	ftp://ftp.denx.de/pub/u-boot/
U-boot 官方 Git 代码仓库	http://git.denx.de/?p=u-boot.git
针对 S3C2440 的修改	
Openmoko 手机的 U-boot 源码 Git	http://git.openmoko.org/?p=u-boot.git;a=shortlog;h=refs/heads/stable

buserror 的 U-boot 源码 Git （针对 mini2440）	http://repo.or.cz/w/u-boot-openmoko/mini2440.git
Tekkaman Ninja 的 U-boot 源码 Git （针对 **mini2440**）	http://github.com/tekkamanninja

第3章   开发环境搭建

3.1 交叉编译工具链的安装

编译U-boot给mini2440 的时候，必须使用交叉编译工具链。你可以使用 友善之臂提供的交叉编译工具（gcc版本 4.3.2），也可以使用crosstool-0.43 或crosstool-ng自己编译一个。至于如何用工具自己编译交叉编译工具链，请看 Tekkaman Ninja 的博客的相关文章：

> 用crosstool0.43 建立 ARM－Linux 交叉编译环境
>
> 用crosstool-ng建立Linux交叉编译环境（以S3C2440（armv4t）为例）

在编译好交叉编译工具链后，要在环境变量的 PATH 中添加编译工具的路径（也就是 arm-*-linux-*-gcc 所在的路径），这样在编译时系统才找得到编译器的命令。在 Ubuntu 下的修改方法如下：

vi ~/.profile，　　　bash

在最后加上：PATH="<交叉编译工具的路径>:$PATH"。

3.2 网路服务的设置

在使用 U-boot 的时候常常会用到宿主机的 TFTP 和 NFS 这两种网络服务，所以最好在开发前设置好。下面以 Ubuntu 下使用 apt-get 安装为例，简单介绍一下安装配置过程：

3.2.1 安装配置 TFTP 服务

安装配置 TFTP 服务的大致步骤如下：

（1）安装 tftp-hpa、tftpd-hpa 和 openbsd-inetd 程序；

（2）修改配置文件/etc/inetd.conf；

（3）根据配置文件的路径，建立 tftp 目录，并修改目录权限；

（4）重启 tftp 服务；

（5）本地传输测试。

以下是一个安装和配置主要步骤的脚本，大家参考：

```
#!/bin/sh
TFTPDIR=<你想要的tftp目录路径>

    echo install tftp server ...

    sudo apt-get install tftp-hpa tftpd-hpa
```

```
        if [ "$?" = "0" ]
        then
                echo "install tftp-hpa and tftpd-hpa  OK!!"
        else
                echo "install tftp-hpa and tftpd-hpa  error !!!"
#               exit 1
        fi

        sudo apt-get install openbsd-inetd
        if [ "$?" = "0" ]
        then
                echo "install openbsd-inetd OK!!"
        else
                echo "install openbsd-inetd error !!!"
#               exit 1
        fi

        echo modify /etc/inetd.conf
#tftp    dgram    udp    wait root    /usr/sbin/in.tftpd /usr/sbin/in.tftpd -c -s <你想要的tftp目录路径>
        sudo vi /etc/inetd.conf


#建立tftp目录，并修改目录权限：
        mkdir -p $TFTPDIR
        if [ "$?" = "0" ]
        then
                echo "make tftp dir $TFTPDIR OK!!"
        else
                echo "make tftp dir $TFTPDIR error !!!"
#               exit 1
        fi
        sudo chmod 777 $TFTPDIR

#重启tftp server
        sudo /etc/init.d/openbsd-inetd restart
```

3.2.2 安装配置 NFS 服务

安装配置 NFS 服务的大致步骤如下：

（1）安装 NFS 内核服务；

（2）重新配置 portmap 服务，修改/etc/hosts.deny 和/etc/hosts.allow 配置文件，重启 portmap 服务；

（3）修改 NFS 服务的配置文件/etc/exports，添加服务目录和配置，重新导入配置；

（4）重启 NFS 服务，并检查可挂载的目录；

（5）在本地挂载测试。

以下是一个安装和配置主要步骤的脚本，大家参考：

```
#!/bin/sh
        echo install tftp server ...
                sudo apt-get install nfs-kernel-server
                if [ "$?" = "0" ]
```

```
                    then
                              echo "install nfs-kernel-server  OK!!"
                    else
                              echo "install nfs-kernel-server error !!!"
#                             exit 1
                    fi

          sudo dpkg-reconfigure portmap
          #对Should portmap be bound to the loopback address? 选N.

          sudo vi /etc/hosts.deny
          #portmap:ALL
          #lockd:ALL
          #mountd:ALL
          #rquotad:ALL
          #statd:ALL

          sudo vi /etc/hosts.allow
          #portmap: 192.168.1.
          #lockd: 192.168.1.
          #rquotad: 192.168.1.
          #mountd: 192.168.1.
          #statd: 192.168.1.

          sudo service portmap restart

          sudo vi /etc/exports
          #/home/tekkaman/development/share
192.168.1.0/24(rw,nohide,insecure,no_wdelay,no_root_squash,no_subtree_check,sync)
          #特别要注意上面的IP的形式，以前是形如 192.168.1.*，现在是IP/掩码为数的形式。用旧的格式可能会
出问题
          #具体的说明,建议看man手册：  man exports
          sudo exportfs -r

          sudo /etc/init.d/nfs-kernel-server restart
          showmount -e 127.0.0.1
```

3.3 串口终端程序的安装配置

在使用 U-boot 的时候，必然会用到串口与开发板进行通信，所以串口终端程序必不可少。下面简单介绍一下 Linux 下常用的串口终端：minicom 和 C-kermit 的安装配置（以Ubuntu 下使用 apt-get 安装为例）。

3.3.1 C-kermit 的安装配置（推荐安装）

在 Linux 下是通过串口传输文件到开发板，就属 C-kermit 比较好用。

（1）安装 ckermit 程序；

（2）编写 ckermit 的配置文件~/.kermrc。

下面是一个很简单的安装和配置脚本，供大家参考：

```
#!/bin/sh
```

```
        echo install C-kermit ...
                sudo apt-get install ckermit
                if [ "$?" = "0" ]
                then
                        echo "install ckermit  OK!!"
                else
                        echo "install ckermit error !!!"
#                       exit 1
                fi

#如果是USB转串口，就是类似/dev/ttyUSB0 的设备，如果是原生的硬件串口，就是类似/dev/ttyS0 的设备节
点。
#根据你使用的串口，设备节点编号可能有变，你可以ls /dev/tty*看看你用的到底有什么设备节点。
        cat >~/.kermrc <<EOF
set line /dev/ttyUSB0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name u-boot.bin
set rec pack 1000
set send pack 1000
set window 5
c

EOF
```

3.3.2 minicom 的安装配置

minicom 是在 Linux 系统下比较常用的串口终端工具，简单的安装配置步骤如下：

（1）安装 minicom 程序；

（2）使用 minicom -s 命令生成配置文件~/.minirc.dfl。

```
#!/bin/sh
        echo install Minicom ...
                sudo apt-get install minicom
                if [ "$?" = "0" ]
                then
                        echo "install minicom  OK!!"
                else
                        echo "install minicom error !!!"
#                       exit 1
                fi

        minicom -s
```

运行命令 minicom -s 后，屏幕出现如下界面：

```
    +----[配置]----+
    | 文件名和路径    |
    | 文件传输协议    |
```

```
          | 串口设置      |
          | 调制解调器和拨号——|——
          | 屏幕和键盘    |
          | 设置保存为dfl |
          | 设置保存为.. |
          | 退出       |
          | 退出Minicom  |
          +-----------+
```

1.使用方向键选择"串口设置",出现具体的配置：

```
+----------------------------------------------------+
|A -  串口设备： /dev/ttyUSB0                          |
|B - 锁文件的位置： /var/lock                          |
|C -  调入程序：                                       |
|D -  调出程序      ：                                 |
|E -   Bps/Par/Bits   : 115200 8N1                    |
|F - 硬件数据流控制： 否                               |
|G - 软件数据流控制   :否                              |
|   希望修改哪个设置？                                 |
+----------------------------------------------------+
```

使用相应的字母键配置，比如修改设备节点则输入 A，光标转移到"串口设备"选项后，可以对其值进行修改，完成后回车确定。如果选择了 E 则出现：

```
+--------------[普通参数]--------------+
|                                      |
|   当前: 115200 8N1                   |
|速度          参数     数据           |
|A: <next>       L: None     S: 5      |
|B: <prev>       M: Even     T: 6      |
|C:  9600        N: Odd      U: 7      |
|D:  38400       O: Mark     V: 8      |
|E: 115200       P: Space              |
|                                      |
|停止位                                |
|W: 1         Q: 8-N-1                 |
|X: 2         R: 7-E-1                 |
|                                      |
|                                      |
| 退出Mini| 选择, 或按 <Enter> 退出? |
+------------------------------------+
```

选择 E 和 Q，设置好后回车，回到"配置"菜单后再进入"调制解调器和拨号"，清除以下几项的数据：

A - 初始化字符串 ......
B - 重置字符串 ....
K - 停机字符串 ......

清除完后，回车退到"配置"菜单后再进入"设置保存为 dfl"，就会保存配置到 ~/.minirc.dfl。然后选择"退出"退出配置状态，就可以通过串口连接开发板了。

还有一个图形化的串口终端：gtkterm，图形配置，简单易用，但无法传输文件。

第4章 熟悉 U-boot 的使用与烧写

要开发和移植 U-boot，首先要对 U-boot 有一定的了解，起码要会使用。所以这里首先熟悉一下 U-boot 的使用以及如何将 U-boot 烧入 mini2440。

当然在这之前首先必须保证你的开发板上已经有了U-boot。如果没有，就请先烧入一个已经编译好的U-boot。已经针对mini2440 编译好的 U-boot-2009.11 的bin文件在Tekkaman Ninja 的 博客中有，可以直接烧入，链接：u-boot.bin_tekkaman_2009.11.tar.bz2

把它烧到mini2440 的NAND或者NOR Flash的起始地址即可，烧入的方法请参考 友善之臂官方网提供的《MINI2440 用户手册》。

4.1 烧写 U-boot 到 mini2440 开发板

4.2 常用 U-boot 命令详解

U-boot 发展到现在，其命令行模式已经非常接近 Linux 下的 shell 了，在 Tekkaman Ninja 编译的 U-boot-2009.11 中的命令行模式下支持"Tab"键的命令补全和命令的历史记录功能。而且如果命令的前几个字符和别的命令不重复，那就只需要打出这几个字符即可，比如查看 U-boot 的版本号命令是" version"，但是在所有其它命令中没有任何一个的命令是由"v"开头的，所以只需要输入"v"即可。

```
[u-boot@MINI2440]# version

U-Boot 2009.11 ( 4 月 04 2010 - 12:09:25)
[u-boot@MINI2440]# v

U-Boot 2009.11 ( 4 月 04 2010 - 12:09:25)
[u-boot@MINI2440]# base
Base Address: 0x00000000
[u-boot@MINI2440]# ba
Base Address: 0x00000000
```

下面简单介绍常用的命令。

4.2.1 获取帮助

命令：help 或？

功能：查看当前 U-boot 支持的所有命令。

```
[u-boot@MINI2440]# help
?      - alias for 'help'
askenv  - get environment variables from stdin
```

```
base    - print or set address offset
bdinfo  - print Board Info structure
bmp     - manipulate BMP image data
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootelf - Boot from an ELF image in memory
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
bootvx  - Boot vxWorks from an ELF image
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
date    - get/set/reset date & time
dcache  - enable or disable data cache
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
editenv - edit environment variable
eeprom  - EEPROM sub-system
erase   - erase FLASH memory
exit    - exit script
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
flinfo  - print FLASH memory information
fsinfo  - print information about filesystems
fsload  - load binary file from a filesystem image
go      - start application at address 'addr'
help    - print online help
i2c     - I2C sub-system
icache  - enable or disable instruction cache
iminfo  - print header information for application image
imls    - list all images found in flash
imxtract- extract a part of a multi-image
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loadx   - load binary file over serial line (xmodem mode)
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
ls      - list files in a directory (default /)
md      - memory display
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub-system
mtest   - simple RAM read/write test
mw      - memory write (fill)
nand    - NAND sub-system
nboot   - boot from NAND device
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
protect - enable or disable FLASH write protection
rarpboot- boot image via network using RARP/TFTP protocol
reginfo - print register information
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
showvar - print local hushshell variables
```

```
sleep   - delay execution for some time
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
unzip   - unzip a memory region
usb     - USB sub-system
usbboot - boot from USB device
version - print monitor version
```

如果想获取某条命令的详细帮助，可使用：

help <想要查的指令>

或者 ？ <想要查的指令>，

甚至 h <想要查的指令缩写>。

以 bmp 指令为例：

```
[u-boot@MINI2440]# help bmp
bmp - manipulate BMP image data

Usage:
bmp info <imageAddr>        - display image info
bmp display <imageAddr> [x y] - display image at x,y
[u-boot@MINI2440]# ? bmp
bmp - manipulate BMP image data

Usage:
bmp info <imageAddr>        - display image info
bmp display <imageAddr> [x y] - display image at x,y
[u-boot@MINI2440]# h bm
bmp - manipulate BMP image data

Usage:
bmp info <imageAddr>        - display image info
bmp display <imageAddr> [x y] - display image at x,y
```

4.2.2 环境变量与相关指令

和 shell 类似，U-Boot 也有环境变量（environment variables，简称 ENV），U-boot 默认的一些环境变量如下：

环 境 变 量	解 释 说 明
bootdelay	执行自动启动（bootcmd 中的命令）的等候秒数
baudrate	串口控制台的波特率
netmask	以太网的网络掩码
ethaddr	以太网的 MAC 地址
bootfile	默认下载文件名 tftp

bootargs	传递给 Linux 内核的启动参数
bootcmd	自动启动时执行命令
serverip	文件服务器端的 IP 地址
ipaddr	本地 IP 地址
stdin	标准输入设备，一般是串口
stdout	标准输出，一般是串口，也可是 LCD（VGA）
stderr	标准出错，一般是串口，也可是 LCD（VGA）

查看当前 U-boot 的 ENV 值可使用 printenv 命令：

```
[u-boot@MINI2440]# printenv
bootargs=noinitrd root=/dev/nfs rw nfsroot=192.168.0.1:/home/tekkaman/working/nfs/rootfs
ip=192.168.0.2:192.168.0.1::255.255.255.0 console=ttySAC0,115200 init=/linuxrc mem=64M
bootcmd=nfs 0x30008000 192.168.0.1:/home/tekkaman/working/nfs/zImage.img;bootm
bootdelay=1
baudrate=115200
ethaddr=08:08:11:18:12:27
ipaddr=192.168.0.2
serverip=192.168.0.1
gatewayip=192.168.0.1
netmask=255.255.255.0
tekkaman=bmp d 70000
 stdin=serial
stdout=serial
stderr=serial
ethact=dm9000

Environment size: 470/131068 bytes
```

你会发现有些上面提到的 ENV 没有显示，还有一个名为"tekkaman"的 ENV。原因是如果没有设置这个环境变量就它不会被打印出，也可以自己定义 ENV，并在命令中使用${ENV}来调用它。同时还可以删除这个 ENV。设置 ENV 的命令是 setenv，格式为：

setenv name value

第 1 个参数 name 是环境变量的名称。

第 2 个参数 value 是要设置的值，如果没有第 2 个参数，表示删除这个环境变量。

范例：先将"tekkaman"参数删除，再设置，最后在一个命令串中调用。

```
[u-boot@MINI2440]#  printenv tekkaman
tekkaman=bmp d 70000
[u-boot@MINI2440]# setenv tekkaman
[u-boot@MINI2440]# printenv tekkaman
## Error: "tekkaman" not defined
[u-boot@MINI2440]# setenv tekkaman echo "I am Tekkaman Ninja!"
[u-boot@MINI2440]# printenv tekkaman
tekkaman=echo I am Tekkaman Ninja!
[u-boot@MINI2440]# echo I Love Linux ;${tekkaman}
```

```
I Love Linux
I am Tekkaman Ninja!
```

当设置或改动了 ENV，它只保存在了内存中。如果需要它保存在存放 ENV 的固态存储器中，请使用：saveenv。

```
[u-boot@MINI2440]# saveenv
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x6000000000002 --   0% complete.
Writing to Nand... done
[u-boot@MINI2440]#
```

如果在启动时，U-boot 打印出："Warning - bad CRC, using default environment"，说明 U-boot 没有在存放 ENV 的固态存储器中找到有效的 ENV，只好使用编译时定义的默认 ENV。**如果 U-boot 存放 ENV 的固态存储器的驱动是没问题**的，那只要运行 saveenv 就可以把当前系统的所有 ENV 写入固态存储器，下次启动就不会有这个警告了。

ENV 可以放在多种固体存储器中，对于 mini2440 来说 Nor Flash、Nand Flash 或 EEPROM 都可以，这依赖 include/configs 下的配置文件是如何定义的。例如：

Nor Flash：

```
#define CONFIG_ENV_IS_IN_FLASH 1
#define CONFIG_ENV_OFFSET 0X40000
#define CONFIG_ENV_SIZE              0x20000      /* Total Size of Environment Sector */
```

Nand Flash：

```
#define CONFIG_ENV_IS_IN_NAND 1
#define CONFIG_ENV_OFFSET 0X40000
#define CONFIG_ENV_SIZE              0x20000      /* Total Size of Environment Sector */
```

EEPROM：

```
#define CONFIG_ENV_IS_IN_EEPROM    1   /* use EEPROM for environment vars */
#define CONFIG_ENV_OFFSET      0x000   /* environment starts at offset 0 */
#define CONFIG_ENV_SIZE             0x400   /* 1KB  */
```

CONFIG_ENV_OFFSET   ：是在整个存储器中的偏移地址；

CONFIG_ENV_SIZE   ：是指用来保存 ENV 的分区大小。

注意 CONFIG_ENV_OFFSET 和 CONFIG_ENV_SIZE 的设置，请不要覆盖了其他分区

4.2.3 串口传输命令

命令：

loadb   - load binary file over serial line **(kermit mode)**

loadx   - load binary file over serial line **(xmodem mode)**

loady   - load binary file over serial line **(ymodem mode)**

功能：以不同的协议从串口获取 Host 传输过来的文件.。

格式：

Load? [off] [baud]

第 1 个参数 off 是下载到 SDRAM 的地址。如果不填，就使用默认配置：CONFIG_SYS_LOAD_ADDR

第 2 个参数是波特率，一般不填，用默认的 115200.

在 windows 下的超级终端可以用这些协议发送文件，但是在 ubuntu 下基本只能用 kermit 协议。以下是使用 C-kermit 来发送一个文件到 mini2440。

```
[u-boot@MINI2440]# loadb
## Ready for binary (kermit) download to 0x30008000 at 115200 bps...
```

上面已经启动了 U-boot 的 kermit 传输协议，这时按下 **Ctrl + \ ,** 再按 **c**, 切换到 C-kermit 的命令行模式，输入命令：**send <文件路径>**，回车。

```
[u-boot@MINI2440]# loadb
## Ready for binary (kermit) download to 0x30008000 at 115200 bps...

(Back at MAGI-Linux)
--------------------------------------------------
C-Kermit 8.0.211, 10 Apr 2004, for Linux
 Copyright (C) 1985, 2004,
  Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
(/home/tekkaman/桌面/) C-Kermit>send /home/tekkaman/development/share/zImage.img
```

C-kermit 就开始传送，并且显示一个传送界面，并动态显示传送进度。

```
C-Kermit 8.0.211, 10 Apr 2004, MAGI-Linux

  Current Directory: /home/tekkaman/       ~L   ~]
Communication Device: /dev/ttyUSB0
 Communication Speed: 115200
          Parity: none
      RTT/Timeout: 01 / 02
        SENDING: /home/tekkaman/development/share/zImage.img => zImage.img
      File Type: BINARY
      File Size: 2277540
    Percent Done: 19  /////////-
              ...10...20...30...40...50...60...70...80...90..100
 Estimated Time Left: 00:03:35
 Transfer Rate, CPS: 8536
      Window Slots: 1 of 1
      Packet Type: D
     Packet Count: 557
    Packet Length: 1000
     Error Count: 0
      Last Error:
    Last Message:

X to cancel file, Z to cancel group, <CR> to resend last packet,
```

```
E to send Error packet, ^C to quit immediately, ^L to refresh screen.
```

传送完毕后，输入 **c** ，回到 U-boot 的串口界面。

```
[u-boot@MINI2440]# loadb
## Ready for binary (kermit) download to 0x30008000 at 115200 bps...

(Back at MAGI-Linux)
--------------------------------------------------
C-Kermit 8.0.211, 10 Apr 2004, for Linux
 Copyright (C) 1985, 2004,
  Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
(/home/tekkaman/桌面/) C-Kermit>send /home/tekkaman/development/share/zImage.img
(/home/tekkaman/桌面/) C-Kermit>c
Connecting to /dev/ttyUSB0, speed 115200
 Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
--------------------------------------------------
## Total Size      = 0x0022c0a4 = 2277540 Bytes
## Start Addr      = 0x30008000
[u-boot@MINI2440]#
```

4.2.4 网络命令

只要 U-boot 的网卡驱动没问题，就可以通过网络来传输文件到开发板，这比串口快多了。你可以直接用**交叉网线**连接开发板和电脑，也可以用普通直连网线通过路由器连到电脑，不过记得配置好网络，关闭防火墙。

先测试网络是否畅通，在开发板使用 ping 命令，看看同电脑的网络连接是否畅通：

```
[u-boot@MINI2440]# ping 192.168.1.100
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
host 192.168.1.100 is alive
```

如果出现：

```
[u-boot@MINI2440]# ping 192.168.1.100
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
ping failed; host 192.168.1.100 is not alive
```

这种无法 ping 通的问题，原因可能是：

1、U-boot 网卡驱动有问题；

2、U-boot 网络协议延时配置有问题；

3、网络参数配置问题，比如 IP 配置等，Host 和 Target 都有可能有问题。Host 最好关闭 IPv6。

如果还找不到原因，可用 Wireshark 抓包看看。

如果网络畅通，就可以使用下面的命令从 tftp 服务目录或者 nfs 服务目录下载文件到 SDRAM 了。

命令：

dhcp ： 使用 DHCP/TFTP 协议获取文件

rarpboot ： 使用 RARP/TFTP 协议获取文件

nfs ： 使用 NFS 协议获取文件

tftpboot ： 使用 TFTP 协议获取文件

bootp ： 使用 BOOTP/TFTP 协议获取文件

以上命令的格式都为：指令 [目的 SDRAM 地址] [[主机 IP:]文件名]

> 注意：
>
> 要使用 dhcp、rarpboot 或 bootp 要路由器或 Host 支持的这些协议和服务。
>
> 如果没有输入[目的 SDRAM 地址]，系统就是用编译时定义的 CONFIG_SYS_LOAD_ADDR
>
> 在使用如果 tftpboot 和 nfs 命令没有定义[主机 IP:]，则使用 ENV 中的 serverip
>
> 其它命令必需定义[主机 IP:]，否则使用提供动态 IP 服务的主机 IP。

以下是使用范例，请注意红色字体：

```
[u-boot@MINI2440]# nfs 0x30008000 192.168.1.100:/home/tekkaman/development/share/u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
File transfer via NFS from server 192.168.1.100; our IP address is 192.168.1.101
Filename '/home/tekkaman/development/share/u-boot.bin'.
Load address: 0x30008000
Loading: #################################################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# tftp u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'u-boot.bin'.
Load address: 0x30008000
Loading: T ################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# dhcp 192.168.1.100:u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
```

```
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 192.168.1.101
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'u-boot.bin'.
Load address: 0x30008000
Loading: #################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# bootp 192.168.1.100:u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 192.168.1.101
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'u-boot.bin'.
Load address: 0x30008000
Loading: #################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# rarpboot 192.168.1.100:u-boot.bin
```

4.2.5 Nand Flash 操作指令

常用的 Nand Flash 指令如下：

指令	功能
nand info	显示可使用的 Nand Flash *nand flash*
nand device [dev]	显示或设定当前使用的 Nand Flash
nand read addr off size	Nand Flash 读取命令，从 Nand 的 off 偏移地址处读取 size 字节的数据到 SDRAM 的 addr 地址。
nand write addr off size	Nand Flash 烧写命令，将 SDRAM 的 addr 地址处的 size 字节的数据烧写到 Nand 的 off 偏移地址。
nand write[.yaffs[1]] addr off size	烧写 yaffs 映像专用的命令，.yaffs1 for 512+16 NAND
nand erase [clean] [off size]	Nand Flash 擦除命令，擦除 Nand Flash 的 off 偏移地址处的 size 字节的数据
nand bad	显示 Nand Flash 的坏块
nand dump[.oob] off	显示 Nand Flash 中的数据（16 进制） *512byte*
nand scrub *oob*	彻底擦除整块 Nand Flash 中的数据，包括 OOB。可以擦除软件坏块标志。
nand markbad off	标示 Nand 的 off 偏移地址处的块为坏块

使用范例：

```
[u-boot@MINI2440]# nand info

Device 0: NAND 128MiB 3,3V 8-bit, sector size 128 KiB
[u-boot@MINI2440]# nand device 0
Device 0: NAND 128MiB 3,3V 8-bit... is now current device
[u-boot@MINI2440]# nand read 0x30008000 0x60000 200000

NAND read: device 0 offset 0x60000, size 0x200000
 2097152 bytes read: OK
[u-boot@MINI2440]# nand bad

Device 0 bad blocks:
  030a0000
  030c0000
  030e0000
  07ee0000
[u-boot@MINI2440]# nand markbad 0x500000
block 0x00500000 successfully marked as bad
[u-boot@MINI2440]# nand bad

Device 0 bad blocks:
  00500000
  030a0000
  030c0000
  030e0000
  07ee0000
[u-boot@MINI2440]# nand scrub

NAND scrub: device 0 whole chip
Warning: scrub option will erase all factory set bad blocks!
     There is no reliable way to recover them.
     Use this command only for testing purposes if you
     are sure of what you are doing!

Really scrub this NAND flash? <y/N>
Erasing at 0x2f4000008000000 --   0% complete.
NAND 128MiB 3,3V 8-bit: MTD Erase failure: -5

NAND 128MiB 3,3V 8-bit: MTD Erase failure: -5

NAND 128MiB 3,3V 8-bit: MTD Erase failure: -5
Erasing at 0x7ea000008000000 --   0% complete.
NAND 128MiB 3,3V 8-bit: MTD Erase failure: -5
Erasing at 0x7fe000008000000 --   0% complete.
OK
[u-boot@MINI2440]# nand bad

Device 0 bad blocks:
  030a0000
  030c0000
  030e0000
  07ee0000
[u-boot@MINI2440]# nand dump 0x8000
Page 00008000 dump:
      ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
    (略)
```

```
OOB:
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
[u-boot@MINI2440]# tftp u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'u-boot.bin'.
Load address: 0x30008000
Loading: T #################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# nand write 0x30008000 0 40000

NAND write: device 0 offset 0x0, size 0x40000
Writing at 0x2000000020000 -- 100% is complete. 262144 bytes written: OK
[u-boot@MINI2440]# nand dump 0x8000
Page 00008000 dump:
        00 00 53 e1 01 00 00 2a  15 40 e0 e3 19 00 00 ea
    (略)
        60 30 97 e5 03 00 54 e1  f6 ff ff ba 00 40 a0 e3
OOB:
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        ff ff ff ff ff ff ff ff
        65 a9 6b f3 ff 33 fc 30
        f3 33 cf 33 0f f0 ff 00
        cc 0f 59 55 57 96 a5 5b
```

nboot 指令也是一条 Nand Flash 读取指令，它是将 Nand Flash 的 offset 偏移地址的内核映像读取到 SDRAM 的 loadAddr 位置。它会自动读取到内核映像（使用 mkimage 处理过的）的结束，所以不用给出读取大小。

格式：nboot loadAddr dev offset

使用范例：

```
[u-boot@MINI2440]# tftp 192.168.1.100:zImage.img
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'zImage.img'.
Load address: 0x30008000
Loading: T #############################################################
        #############################################################
```

```
            #########################
done
Bytes transferred = 2277540 (22c0a4 hex)
[u-boot@MINI2440]# nand erase 0x100000 300000

NAND erase: device 0 offset 0x100000, size 0x300000
Erasing at 0x3e000001800000 --   0% complete.
OK
[u-boot@MINI2440]# nand write 0x30008000   0x100000 300000

NAND write: device 0 offset 0x100000, size 0x300000
Writing at 0x3e000000020000 -- 100% is complete. 3145728 bytes written: OK
[u-boot@MINI2440]# nand device 0
Device 0: NAND 128MiB 3,3V 8-bit... is now current device
[u-boot@MINI2440]# nboot 30008000 0 0x100000

Loading from NAND 128MiB 3,3V 8-bit, offset 0x100000
  Image Name:   tekkaman
  Created:      2010-03-29  12:59:51 UTC
  Image Type:   ARM Linux Kernel Image (uncompressed)
  Data Size:    2277476 Bytes =  2.2 MB
  Load Address: 30008000
  Entry Point:  30008040

 [u-boot@MINI2440]# bootm 30008000
## Booting kernel from Legacy Image at 30008000 ...
  Image Name:   tekkaman
  Created:      2010-03-29  12:59:51 UTC
  Image Type:   ARM Linux Kernel Image (uncompressed)
  Data Size:    2277476 Bytes =  2.2 MB
  Load Address: 30008000
  Entry Point:  30008040
  Verifying Checksum ... OK
  XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
（略）
```

4.2.6 内存/寄存器操作指令

nm   修改内存值 (指定地址)

格式： nm [.b, .w, .l] address

mm   修改内存值(地址自动加一)

格式： mm [.b, .w, .l] address

md　　　显示内存值

格式：　md [.b, .w, .l] address [# of objects]

mw　　用指定的数据填充内存

格式：　mw [.b, .w, .l] address value [count]

cp　　　内存的拷贝（包括内存与 Nor Flash 间的数据拷贝）

格式：cp [.b, .w, .l] source target count

上面是查看和修改内存值的指令，可以查看和修改 SDRAM 和寄存器值。

[.b, .w, .l]代表了查看和修改形式：bit、word、long

使用范例：

```
[u-boot@MINI2440]# md.b 0x30008000  20
30008000: cc 33 fe 33 cc b3 4c 33 ac 33 de 33 5c 13 cc 33    .3.3..L3.3.3\..3
30008010: cc 32 cc 31 dc 33 cf 33 cc 33 4e 33 8f 13 cc 33    .2.1.3.3.3N3...3
[u-boot@MINI2440]# md.w 0x30008000  20
30008000: 33cc 33fe b3cc 334c 33ac 33de 135c 33cc    .3.3..L3.3.3\..3
30008010: 32cc 31cc 33dc 33cf 33cc 334e 138f 33cc    .2.1.3.3.3N3...3
30008020: 338c 33cd 33cc 7bcc 3bcc 33cc 135e 734c    .3.3.3.{.;.3^.Ls
30008030: 7bdc 37cc 31dc 33c4 038c 33e8 77cc 13cc    .{.7.1.3...3.w..
[u-boot@MINI2440]# md.l 0x30008000  20
30008000: 33fe33cc 334cb3cc 33de33ac 33cc135c    .3.3..L3.3.3\..3
30008010: 31cc32cc 33cf33dc 334e33cc 33cc138f    .2.1.3.3.3N3...3
30008020: 33cd338c 7bcc33cc 33cc3bcc 734c135e    .3.3.3.{.;.3^.Ls
30008030: 37cc7bdc 33c431dc 33e8038c 13cc77cc    .{.7.1.3...3.w..
30008040: 234c77ce 33dc339c 33ec3ece f3cc36ec    .wL#.3.3.>.3.6..
30008050: 37dc33cc 73cc3f5c 17dd314c 33cc62e8    .3.7\?.sL1...b.3
30008060: b6cc33dc 33c233cc 33cc32cc 33cc3f68    .3...3.3.2.3h?.3
30008070: 73cc31cc b3cc33cc 33cc37c9 33df13cc    .1.s.3...7.3...3
[u-boot@MINI2440]# nm 0x30008000
30008000: 33fe33cc ? 12345678
30008000: 12345678 ? 34567890
30008000: 34567890 ? q
[u-boot@MINI2440]# nm.b 0x30008000
30008000: 90 ? 11
30008000: 11 ? 12
30008000: 12 ? q
[u-boot@MINI2440]# mm 0x30008000
30008000: 34567812 ? 54321123
30008004: 334cb3cc ? 12345678
30008008: 33de33ac ? 21234543
```

```
3000800c: 33cc135c ? q
[u-boot@MINI2440]# md.b  0x30008000  20
30008000: 23 11 32 54 78 56 34 12 43 45 23 21 5c 13 cc 33    #.2TxV4.CE#!\..3
30008010: cc 32 cc 31 dc 33 cf 33 cc 33 4e 33 8f 13 cc 33    .2.1.3.3.3N3...3
[u-boot@MINI2440]# mw.b 0x30008000 aa  10
[u-boot@MINI2440]# mw.b 0x30008010 55  10
[u-boot@MINI2440]# md.b  0x30008000 20
30008000: aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa    ................
30008010: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55    UUUUUUUUUUUUUUUU
[u-boot@MINI2440]# cp.b 0x30008000 0x30008010  10
[u-boot@MINI2440]# md.b  0x30008000 20
30008000: aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa    ................
30008010: aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa    ................
```

你可以试着修改 LED 相连的 GPIO 寄存器的数据寄存器值，可以控制 LED 的点亮！

先熄灭后点亮 LED1 的范例：（这个实验要结合芯片数据手册和 mini2440 的原理图来理解）

```
 [u-boot@MINI2440]# md 0x56000014 1
56000014: 00000600    ....
[u-boot@MINI2440]# nm.w  0x56000014
56000014: 0600 ? 620  （熄灭）
56000014: 0620 ? 600  （点亮）
```

4.2.7 Nor Flash 指令

Nor Flash 的命令经常用于烧写数据到 Nor Flash 。

flinfo 打印 Flash 存储器的信息，并列出所有 Sector。

flinfo N 单独打 Flash 存储器 N Block 的信息。（在有多块 Nor Flash 时使用）

nor flash   N
nor flash

```
[u-boot@MINI2440]# flinfo

Bank # 1: SST: 1x SST39VF1601 (2MB)
  Size: 2 MB in 32 Sectors
  Sector Start Addresses:
   00000000 (RO) 00010000 (RO) 00020000 (RO) 00030000 (RO) 00040000
   00050000      00060000 (RO) 00070000 (RO) 00080000      00090000
   000A0000      000B0000      000C0000      000D0000      000E0000
   000F0000      00100000      00110000      00120000      00130000
   00140000      00150000      00160000      00170000      00180000
   00190000      001A0000      001B0000      001C0000      001D0000
   001E0000      001F0000
[u-boot@MINI2440]# flinfo  1

Bank # 1: SST: 1x SST39VF1601 (2MB)
  Size: 2 MB in 32 Sectors
  Sector Start Addresses:
   00000000 (RO) 00010000 (RO) 00020000 (RO) 00030000 (RO) 00040000
   00050000      00060000 (RO) 00070000 (RO) 00080000      00090000
```

```
    000A0000      000B0000      000C0000      000D0000      000E0000
    000F0000      00100000      00110000      00120000      00130000
    00140000      00150000      00160000      00170000      00180000
    00190000      001A0000      001B0000       001C0000       001D0000
    001E0000      001F0000
[u-boot@MINI2440]# flinfo  2
Only FLASH Banks # 1 ... # 1 supported
```

后面带有(RO)的说明这个 Sector 已经写保护了。

因为 Nor Flash 的读取接口和 SDRAM 是一样的，所以 Nor Flash 的读取也是使用 md 命令。范例如下：

```
[u-boot@MINI2440]# md.b 0x0 20
00000000: 12 00 00 ea 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5    ................
00000010: 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5    ...............
[u-boot@MINI2440]# md 0x0 20
00000000: ea000012 e59ff014 e59ff014 e59ff014    ................
00000010: e59ff014 e59ff014 e59ff014 e59ff014    ...............
00000020: 33f80260 33f802c0 33f80320 33f80380    `..3...3 ..3...3
00000030: 33f803e0 33f80440 33f804a0 deadbeef    ...3@..3...3....
00000040: 33f80000 33f80000 33fbe8dc 3400374c    ...3...3...3L7.4
00000050: e10f0000 e3c0001f e38000d3 e129f000    .............).
00000060: e3a00453 e3a01000 e5801000 e3e01000    S...............
00000070: e59f0488 e5801000 e59f1484 e59f0484    ................
```

但由于 Nor Flash 的烧写时序和 SDRAM 的写入不同，烧写 Nor Flash 不能使用 mm 等命令，只能使用 cp 命令从内存拷贝到 Nor Flash，而且之前必须解除保护并擦除！命令如下：

protect：对 Flash 写保护的操作，可以使能和解除写保护。

格式：

protect on/off start end

protect on/off start +end

protect on/off N:SF[-SL]

protect on/off bank N

protect on/off all

第 1 个参数 on 代表使能写保护；off 代表解除写保护。

第 2、3 参数是指定 Flash 写保护操作范围

start end 是照起始地址和结束地址定义范围，start 是擦除块的起始地址；end 是擦除末尾块的结束地址。

例如：擦除 Sector 2 和 Sector 3 区域命令为 erase 20000 3ffff 。

start +end 是照起始地址和操作字节数定义范围，这种方式最常用。start 是擦除块的起始地址；end 是擦除的字节数。

例如：擦除 Sector 2 和 Sector 3 区域命令为 erase 20000 +20000

N:SF[-SL]是按照组和扇区，N 表示 Flash 的 Block 号，SF 表示擦除起始 Sector 号，SL 表示擦除结束 Sector 号。

例如：擦除 Block1 的 Sector 2 和 Sector 3 区域命令为 erase 1:2-3。

bank N 是擦除整个 Block，擦除 Block 号为 N 的整个 Flash。

all 是擦除全部 Flash。

注意：Nor Flash 擦除的最小单位是 Sector，也就是 0x10000 字节，如果你定义的大小不满 1 Sector 或超过 Sector 的边界，那么被定义到的 Sector 会被全部擦除。

erase ：擦除 Flash 的命令

格式：

erase start end

erase start +end

erase N:SF[-SL]

erase bank N

erase all

参数是指定 Flash 擦除操作范围，跟写保护的方式相同。

范例将 mini2440 的 Nor Flash 的 Sector 16 写保护，再解除保护，擦除数据，最后将起始的 20 字节拷贝到 Sector 16。

```
[u-boot@MINI2440]# flinfo  1

Bank # 1: SST: 1x SST39VF1601 (2MB)
 Size: 2 MB in 32 Sectors
 Sector Start Addresses:
  00000000 (RO) 00010000 (RO) 00020000 (RO) 00030000 (RO) 00040000
  00050000     00060000 (RO) 00070000 (RO) 00080000     00090000
  000A0000     000B0000     000C0000     000D0000     000E0000
  000F0000     00100000     00110000     00120000     00130000
  00140000     00150000     00160000     00170000     00180000
  00190000     001A0000     001B0000     001C0000     001D0000
  001E0000     001F0000
 [u-boot@MINI2440]# protect on 1:16-16
 Protect Flash Sectors 16-16 in Bank # 1
 [u-boot@MINI2440]# flinfo  1

 Bank # 1: SST: 1x SST39VF1601 (2MB)
  Size: 2 MB in 32 Sectors
  Sector Start Addresses:
   00000000 (RO) 00010000 (RO) 00020000 (RO) 00030000 (RO) 00040000
```

```
    00050000     00060000 (RO) 00070000 (RO) 00080000     00090000
    000A0000     000B0000     000C0000     000D0000     000E0000
    000F0000     00100000 (RO) 00110000     00120000     00130000
    00140000     00150000     00160000     00170000     00180000
    00190000     001A0000     001B0000     001C0000     001D0000
    001E0000     001F0000
[u-boot@MINI2440]# protect off 0x100000  0x10ffff
Un-Protect Flash Sectors 16-16 in Bank # 1
[u-boot@MINI2440]# flinfo  1

Bank # 1: SST: 1x SST39VF1601 (2MB)
  Size: 2 MB in 32 Sectors
  Sector Start Addresses:
    00000000 (RO) 00010000 (RO) 00020000 (RO) 00030000 (RO) 00040000
    00050000     00060000 (RO) 00070000 (RO) 00080000     00090000
    000A0000     000B0000     000C0000     000D0000     000E0000
    000F0000     00100000     00110000     00120000     00130000
    00140000     00150000     00160000     00170000     00180000
    00190000     001A0000     001B0000     001C0000     001D0000
    001E0000     001F0000
[u-boot@MINI2440]# erase 0x100000 +20
Erasing sector 16 ... ok.
Erased 1 sectors
[u-boot@MINI2440]# cp.b 0x0 0x100000 0x20
Copy to Flash... done
[u-boot@MINI2440]# md.b 100000 20
00100000: 12 00 00 ea 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5    ................
00100010: 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5 14 f0 9f e5    ................
```

4.2.8 USB 操作指令

指令	功能
usb reset	初始化 USB 控制器
usb stop [f]	关闭 USB 控制器
usb tree	已连接的 USB 设备树
usb info [dev]	显示 USB 设备[dev]的信息
usb storage	显示已连接的 USB 存储设备
usb dev [dev]	显示和设置当前 USB 存储设备
usb part [dev]	显示 USB 存储设备[dev]的分区信息

usb read addr blk# cnt	读取 USB 存储设备数据

　　我将一个 4G 的 kingstonU 盘（可引导盘）插入 mini2440，然后读取他的头 512 字节（MBR）：

```
[u-boot@MINI2440]#   usb reset
(Re)start USB...
USB:   scanning bus for devices... 2 USB Device(s) found
     scanning bus for storage devices... 1 Storage Device(s) found
[u-boot@MINI2440]# usb tree

Device Tree:
  1  Hub (12 Mb/s, 0mA)
  |  OHCI Root Hub
  |
  +-2  Mass Storage (12 Mb/s, 100mA)
       Kingston DT 101 II 0019E02CB6EB5B8B1B120051

[u-boot@MINI2440]# usb info
1: Hub,  USB Revision 1.10
 -  OHCI Root Hub
 - Class: Hub
 - PacketSize: 8  Configurations: 1
 - Vendor: 0x0000  Product 0x0000 Version 0.0
   Configuration: 1
   - Interfaces: 1 Self Powered 0mA
     Interface: 0
     - Alternate Setting 0, Endpoints: 1
     - Class Hub
     - Endpoint 1 In Interrupt MaxPacket 2 Interval 255ms

2: Mass Storage,  USB Revision 2.0
 - Kingston DT 101 II 0019E02CB6EB5B8B1B120051
 - Class: (from Interface) Mass Storage
 - PacketSize: 64  Configurations: 1
 - Vendor: 0x0951  Product 0x1613 Version 1.0
   Configuration: 1
   - Interfaces: 1 Bus Powered 100mA
     Interface: 0
     - Alternate Setting 0, Endpoints: 2
     - Class Mass Storage, Transp. SCSI, Bulk only
     - Endpoint 1 In Bulk MaxPacket 64
     - Endpoint 2 Out Bulk MaxPacket 64

[u-boot@MINI2440]# usb storage
```

```
   Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
        Type: Removable Hard Disk
        Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)
[u-boot@MINI2440]# usb dev 0


USB device 0:
   Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
        Type: Removable Hard Disk
        Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)
... is now current device
[u-boot@MINI2440]# usb part 0
print_part of 0


Partition Map for USB device 0  --  Partition Type: DOS


Partition    Start Sector    Num Sectors    Type
    4              63         7935937         c
[u-boot@MINI2440]# usb read 0x30008000 0 200


USB read: device 0 block # 0, count 512 ... ........................
512 blocks read: OK
[u-boot@MINI2440]# md.b 0x30008000  200
30008000: fa 31 c0 8e d8 8e c0 8e d0 bc 00 7c fb fc 89 e6    .1.........|....
30008010: bf 00 06 b9 00 01 f3 a5 ea dc 06 00 00 10 00 01    ................
30008020: 00 00 7c 00 00 00 00 00 00 00 00 00 00 80 3f 00    ..|..........?.
30008030: ff 00 ed 01 1e 0e 1f 3a 16 10 00 74 06 1f ea 36    .......:...t...6
30008040: e7 00 f0 3d fb 54 75 05 8c d8 fb eb 1d 80 fc 08    ...=.Tu.........
30008050: 75 1b e8 81 00 8a 36 13 00 fe ce 8b 0e 15 00 86    u.....6.........
30008060: cd c0 e1 06 0a 0e 11 00 31 c0 f8 eb 65 80 fc 02    ........1...e...
30008070: 72 cb 80 fc 04 77 c6 60 80 cc 40 50 be 00 00 c7    r....w.`..@P....
30008080: 04 10 00 30 e4 89 44 02 89 5c 04 8c 44 06 66 31    ...0..D..\..D.f1
30008090: c0 66 89 44 0c 88 f0 f6 26 11 00 88 cf 88 eb c0    .f.D....&.......
300080a0: ef 06 81 e1 3f 00 01 c8 48 89 c7 a1 13 00 f7 26    ....?...H......&
300080b0: 11 00 f7 e3 01 f8 81 d2 00 00 89 44 08 89 54 0a    ...........D..T.
300080c0: 58 30 c0 8a 16 10 00 e8 0c 00 88 26 03 00 61 a1    X0.........&..a.
300080d0: 02 00 1f ca 02 00 9c ff 1e 22 00 c3 80 fa 8f 7f    ........."......
300080e0: 04 88 16 2d 06 be 87 07 e8 8d 00 be be 07 31 c0    ...-..........1.
300080f0: b9 04 00 f6 04 80 74 03 40 89 f5 81 c6 10 00 e2    ......t.@.......
30008100: f2 48 74 02 cd 18 bf 05 00 be 1d 06 c7 44 02 01    .Ht..........D..
30008110: 00 66 8b 46 08 66 89 44 08 b8 00 42 8a 16 2d 06    .f.F.f.D...B..-.
30008120: cd 13 73 0d 4f 74 49 30 e4 8a 16 2d 06 cd 13 eb    ..s.OtI0...-....
30008130: d8 a1 fe 7d 3d 55 aa 75 37 fa 66 a1 4c 00 66 a3    ...}=U.u7.f.L.f.
30008140: 3f 06 be 13 04 8b 04 48 89 04 c1 e0 06 8e c0 31    ?......H.......1
30008150: ff be 1d 06 b9 60 00 fc f3 a5 c7 06 4c 00 17 00    .....`......L...
```

```
30008160: a3 4e 00 fb 8a 16 2d 06 89 ee fa ea 00 7c 00 00    .N....-......|..
30008170: be aa 07 e8 02 00 eb fe ac 20 c0 74 09 b4 0e bb     ......... .t....
30008180: 07 00 cd 10 eb f2 c3 53 74 61 72 74 20 62 6f 6f    .......Start boo
30008190: 74 69 6e 67 20 66 72 6f 6d 20 55 53 42 20 64 65     ting from USB de
300081a0: 76 69 63 65 2e 2e 2e 0d 0a 00 42 6f 6f 74 20 66     vice......Boot f
300081b0: 61 69 6c 65 64 00 00 00 ea eb d4 ca 00 00 00 00     ailed...........
300081c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
300081d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
300081e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 01    ................
300081f0: 01 00 0c fe 7f ec 3f 00 00 00 c1 17 79 00 55 aa    ......?.....y.U.
```

在所有的命令使用前，必须先插入 USB 设备，然后使用：usb reset，以初始化 USB 控制器，获取设备信息。

4.2.9 SD 卡(MMC)指令

```
 [u-boot@MINI2440]# ? mmc
mmc - MMC sub-system

Usage:
mmc init [dev] - init MMC sub system
mmc device [dev] - show or set current device
```

SD 卡的使用命令比较简单，只有初始化和设备信息的显示，读写是通过文件系统命令实现的。

使用和 USB 类似，在所有的命令使用前，必须先插入 SD 卡，然后使用：mmc init，以初始化 MMC 控制器，获取设备信息。

我在 mini2440 中插入 1GB SD 卡：

```
[u-boot@MINI2440]# mmc init
mmc: Probing for SDHC ...
mmc: SD 2.0 or later card found
trying to detect SD Card...
Manufacturer:    0x00, OEM "Product name:      " ", revision 0.0
Serial number:    7864775
Manufacturing date: 11/2006
CRC:         0x4f, b0 = 1
READ_BL_LEN=6, C_SIZE_MULT=7, C_SIZE=4095
size = 0
SD Card detected RCA: 0x2 type: SD
mmc1 is available
[u-boot@MINI2440]# mmc device
mmc1 is current device
```

4.2.10　FAT 文件系统指令

fatinfo：显示文件系统的相关信息

格式：fatinfo \<interface> \<dev[:part]>

Interface：代表接口，如 usb、mmc；

dev：代表设备编号，如 0、1……；

part：代表存储设备中的分区，如 1、2……。

fatload：从 FAT32 文件系统中读取二进制文件到 SDRAM。

格式：fatload \<interface> \<dev[:part]> \<addr> \<filename> [bytes]

Interface、dev 和 part 同上；

addr：代表写入 SDRAM 的地址；

filename：代表存储设备中的文件名；

bytes：代表从存储设备中读取的文件大小，可不填；如果填的数据比文件小，就只读取 bytes 字节，如果填的数据比文件大，也只读取文件的大小。

fatls：列出 FAT32 文件系统中目录里的文件。

格式：fatls \<interface> \<dev[:part]> [directory]

Interface、dev 和 part 同上；

directoryr：代表所要查看的目录，可不填，默认为/。

这些指令基本上要和 U 盘或者 SD 卡同时使用，主要用于读取这些移动存储器上的 FAT32 分区。

使用范例：

```
[u-boot@MINI2440]# usb part 0
print_part of 0


Partition Map for USB device 0  --  Partition Type: DOS


Partition   Start Sector   Num Sectors   Type
  4            63         7935937      c
[u-boot@MINI2440]# fatinfo usb 0:4
Interface:  USB
 Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
      Type: Removable Hard Disk
      Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)
Partition 4: Filesystem: FAT32 "7600_16385_"
```

```
[u-boot@MINI2440]# fatls  usb 0:4
        boot/
        efi/
        sources/
        support/
        upgrade/
    43   autorun.inf
 383562   bootmgr
 111880   setup.exe
 256220   u-boot.bin

4 file(s), 5 dir(s)

[u-boot@MINI2440]# fatls  usb 0:4 /boot/
        ./
        ../
        fonts/
        zh-cn/
  262144   bcd
 3170304   boot.sdi
    1024   bootfix.bin
   97280   bootsect.exe
    4096   etfsboot.com
  485440   memtest.exe


6 file(s), 4 dir(s)
[u-boot@MINI2440]# fatload  usb 0:4 0x30008000 u-boot.bin
reading u-boot.bin
.......................

256220 bytes read
 [u-boot@MINI2440]# fatload  usb 0:4 0x30008000 u-boot.bin  200
reading u-boot.bin

512 tes read
```

4.2.11 系统引导指令

boot 和 bootd 都是运行 ENV"bootcmd"中指定的指令。

bootm 指令是专门用于启动在 SDRAM 中的用 U-boot 的 mkimage 工具处理过的内核映像。

格式：bootm [addr [arg ...]]

addr 是内核映像所在的 SDRAM 中的地址

当启动的是 Linux 内核时，'arg' 可以使 initrd 的地址。

范例：

```
[u-boot@MINI2440]# setenv bootcmd tftp\;bootm
[u-boot@MINI2440]# saveenv
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x6000000000002 --   0% complete.
Writing to Nand... done
[u-boot@MINI2440]# boot
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'zImage.img'.
Load address: 0x30008000
Loading: T #################################################################
        #################################################################
        #######################
done
Bytes transferred = 2277540 (22c0a4 hex)
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
（略）


U-Boot 2009.11（4 月 04 2010 - 12:09:25)

 modified by tekkamanninja (tekkamanninja@163.com)
 Love Linux forever!!

I2C:  ready
DRAM:  64 MB
Flash:  2 MB
NAND:  128 MiB
Video: 240x320x16 20kHz 62Hz
In:   serial
Out:   serial
```

```
Err:   serial
Net:   dm9000
U-Boot 2009.11 ( 4 月 04 2010 - 12:09:25)
modified by tekkamanninja
(tekkamanninja@163.com)
Love Linux forever!!
Hit any key to stop autoboot:  0
[u-boot@MINI2440]# bootd
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'zImage.img'.
Load address: 0x30008000
Loading: T ###############################################################
        ###############################################################
        #########################
done
Bytes transferred = 2277540 (22c0a4 hex)
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...
（略）
```

4.2.13其他指令

run   -运一个 ENV 定义的命令脚本。

使用范例：

```
[u-boot@MINI2440]# setenv a_run_test  echo $bootfile \; version
[u-boot@MINI2440]# run a_run_test
zImage.img

U-Boot 2009.11 ( 4æœˆ 04 2010 - 12:09:25)
```

reset   - 重启 CPU

其他指令，可以用"help"查到用法。

4.3 下载与烧写

使用 U-boot 将映像文件烧写到板上的 Flash，一般步骤是：

（1）通过网络、串口、**U 盘**、**SD** 卡等方式将文件传输到 **SDRAM**；

（2）使用 **Nand Flash** 或 **Nor Flash** 相关的读写命令将 **SDRAM** 中的数据烧入 **Flash**。

下面是烧写范例：

如果使用 SD 卡和 U 盘形式更新 U-boot，那么首先 SD 卡和 U 盘中必须有 FAT32 文件系统，并在里面存放了 u-boot.bin 文件。

4.3.1 通过 SD 卡烧入 Nand Flash：

```
[u-boot@MINI2440]# mmc init
mmc: Probing for SDHC ...
mmc: SD 2.0 or later card found
trying to detect SD Card...
Manufacturer:     0x00, OEM "Product name:      "     ", revision 0.0
Serial number:     7864775
Manufacturing date: 11/2006
CRC:           0x4f, b0 = 1
READ_BL_LEN=6, C_SIZE_MULT=7, C_SIZE=4095
size = 0
SD Card detected RCA: 0x2 type: SD
mmc1 is available
[u-boot@MINI2440]# fatload mmc 1 0x30008000 u-boot.bin
reading u-boot.bin

256220 bytes read
[u-boot@MINI2440]# nand erase 0 0x40000

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x2000000000004 --   0% complete.
OK
[u-boot@MINI2440]# nand write 0x30008000 0 0x40000

NAND write: device 0 offset 0x0, size 0x40000
Writing at 0x2000000020000 -- 100% is complete. 262144 bytes written: OK
```

4.3.2 通过 U 盘烧入 Nor Flash：

```
[u-boot@MINI2440]# usb start
(Re)start USB...
USB:  scanning bus for devices... 2 USB Device(s) found
      scanning bus for storage devices... 1 Storage Device(s) found
[u-boot@MINI2440]# usb storage
  Device 0: Vendor: Kingston Rev: PMAP Prod: DT 101 II
        Type: Removable Hard Disk
```

```
        Capacity: 3875.0 MB = 3.7 GB (7936000 x 512)
[u-boot@MINI2440]# usb part 0
print_part of 0

Partition Map for USB device 0  --  Partition Type: DOS

Partition   Start Sector   Num Sectors   Type
   4                63     7935937        c
 [u-boot@MINI2440]# fatload usb 0:4 0x30008000 u-boot.bin
reading u-boot.bin
........................

256220 bytes read
[u-boot@MINI2440]# protect off all
Un-Protect Flash Bank # 1
[u-boot@MINI2440]# erase 0x0 0x3ffff
Erasing sector  0 ... ok.
Erasing sector  1 ... ok.
Erasing sector  2 ... ok.
Erasing sector  3 ... ok.
Erased 4 sectors
[u-boot@MINI2440]# cp.b 0x30008000 0x0  0x3ffff
Copy to Flash... done
```

4.3.3 通过 TFTP 服务烧入 Nand Flash：

tftp   u-boot.bin

```
[u-boot@MINI2440]# tftpboot 30008000 192.168.1.100:u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'u-boot.bin'.
Load address: 0x30008000
Loading: T ##################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# nand erase 0 0x40000
NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x2000000000004 --   0% complete.
OK
[u-boot@MINI2440]# nand write 0x30008000 0 0x40000

NAND write: device 0 offset 0x0, size 0x40000
Writing at 0x2000000020000 -- 100% is complete. 262144 bytes written: OK
```

4.3.4 通过 NFS 服务烧入 Nand Flash：

```
[u-boot@MINI2440]# nfs 30008000 192.168.1.100:/home/tekkaman/development/share/u-boot.bin
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
```

```
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
File transfer via NFS from server 192.168.1.100; our IP address is 192.168.1.101
Filename '/home/tekkaman/development/share/u-boot.bin'.
Load address: 0x30008000
Loading: #################################################
done
Bytes transferred = 256220 (3e8dc hex)
[u-boot@MINI2440]# nand erase 0 0x40000
NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x2000000000004 --   0% complete.
OK
[u-boot@MINI2440]# nand write 0x30008000 0 0x40000

NAND write: device 0 offset 0x0, size 0x40000
Writing at 0x2000000020000 -- 100% is complete. 262144 bytes written: OK
```

4.4 内核引导

内核的引导步骤如下：

（1）用 U-boot 的 mkimage 工具处理内核映像 zImage。

（2）通过网络、串口、U 盘、SD 卡等方式将处理过的内核映像传输到 SDRAM 的一定位置（一般使用 0x30008000）

（3）然后使用"bootm"等内核引导命令来启动内核。

为什么要用 U-boot 的 mkimage 工具处理内核映像 zImage？

因为在用 bootm 命令引导内核的时候，bootm 需要读取一个 64 字节的文件头，来获取这个内核映象所针对的 CPU 体系结构、OS、加载到内存中的位置、在内存中入口点的位置以及映象名等等信息。这样 bootm 才能为 OS 设置好启动环境，并跳入内核映象的入口点。而 mkimage 就是添加这个文件头的专用工具。具体的实现请看 U-boot 中 bootm 的源码和 mkimage 的源码。

mkimage 工具的使用：

参数说明：

-A 指定 CPU 的体系结构，可用值有：alpha、**arm** 、x86、ia64、mips、mips64、ppc 、s390、sh、sparc 、sparc64、m68k 等

-O 指定操作系统类型，可用值有：openbsd、netbsd、freebsd、4_4bsd、**linux**、svr4、esix、solaris、irix、sco、dell、ncr、lynxos、vxworks、psos、qnx、u-boot、rtems、artos

-T 指定映象类型，可用值有：standalone、kernel、ramdisk、multi、firmware、script、filesystem

-C 指定映象压缩方式，可用值有：gzip   mkzimge   bzip2   kernel

　　　 none 不压缩(一般使用这个，因为 zImage 是已经被 bzip2 压缩过的自解压内核)

gzip 用 gzip 的压缩方式

bzip2 用 bzip2 的压缩方式

-a 指定映象在内存中的加载地址，映象下载到内存中时，要按照用 mkimage 制作映象时，这个参数所指定的地址值来下载

-e 指定映象运行的入口点地址，这个地址就是-a 参数指定的值加上 0x40（因为前面有个 mkimage 添加的 0x40 个字节的头）

-n 指定映象名

-d 指定制作映象的源文件

以下是制作内核映像的命令示例：

mkimage -n 'tekkaman' -A arm -O linux -T kernel -C none -a 0x30008000 -e 0x30008040 -d zImage zImage.img

以下是使用范例：

4.4.1 通过 SD 卡引导内核：

首先 SD 卡中必须有 FAT32 文件系统，并在里面存放了处理过的内核映像文件。

```
[u-boot@MINI2440]# mmc init
mmc: Probing for SDHC ...
mmc: SD 2.0 or later card found
trying to detect SD Card...
Manufacturer:    0x00, OEM "Product name:    "    ", revision 0.0
Serial number:    7864775
Manufacturing date: 11/2006
CRC:          0x4f, b0 = 1
READ_BL_LEN=6, C_SIZE_MULT=7, C_SIZE=4095
size = 0
SD Card detected RCA: 0x2 type: SD
mmc1 is available
[u-boot@MINI2440]# fatload mmc 1 30008000 zImage.img
reading zImage.img

2277540 bytes read
[u-boot@MINI2440]# bootm 30008000
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...
```

```
Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
（略）
```

4.4.2 通过 TFTP 服务引导内核

```
[u-boot@MINI2440]# tftpboot 0x30008000 192.168.1.100:zImage.img
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.100; our IP address is 192.168.1.101
Filename 'zImage.img'.
Load address: 0x30008000
Loading: T #################################################################
        #################################################################
        #########################
done
Bytes transferred = 2277540 (22c0a4 hex)
[u-boot@MINI2440]# bootm 30008000
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
（略）
```

4.4.3 通过 NFS 服务引导内核：

```
[u-boot@MINI2440]# nfs 30008000 192.168.1.100:/home/tekkaman/development/share/zImage.img
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
```

```
Using dm9000 device
File transfer via NFS from server 192.168.1.100; our IP address is 192.168.1.101
Filename '/home/tekkaman/development/share/zImage.img'.
Load address: 0x30008000
Loading: #################################################################
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        #####################################################
done
Bytes transferred = 2277540 (22c0a4 hex)
[u-boot@MINI2440]# bootm 30008000
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
 （略）
```

kernel

4.4.4 通过 Nand Flash 引导内核：

　　首先要将处理过的内核映像文件烧入 Nand Flash 的一定位置（由内核分区表决定）。以后每次启动时用 Nand Flash 的读取命令先将这个内核映像文件读到内存的一定位置（由制作内核映像时的-a 参数决定），再使用 bootm 命令引导内核。

　　内核映像文件的烧入：

```
[u-boot@MINI2440]# nfs 30008000 192.168.1.100:/home/tekkaman/development/share/zImage.img
dm9000 i/o: 0x20000300, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 08:08:11:18:12:27
operating at 100M full duplex mode
Using dm9000 device
File transfer via NFS from server 192.168.1.100; our IP address is 192.168.1.101
Filename '/home/tekkaman/development/share/zImage.img'.
Load address: 0x30008000
Loading: #################################################################
        #################################################################
```

```
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        ########################################################
done
Bytes transferred = 2277540 (22c0a4 hex)
[u-boot@MINI2440]# nand erase 0x80000 0x300000

NAND erase: device 0 offset 0x80000, size 0x300000
Erasing at 0x36000001800000 --   0% complete.
OK
[u-boot@MINI2440]# nand write 30008000 0x80000 300000

NAND write: device 0 offset 0x80000, size 0x300000

Writing at 0x36000000020000 -- 100% is complete. 3145728 bytes written: OK
```

内核引导：

```
[u-boot@MINI2440]# nand read 30008000 0x80000 300000

NAND read: device 0 offset 0x80000, size 0x300000
 3145728 bytes read: OK
[u-boot@MINI2440]# bootm 30008000
## Booting kernel from Legacy Image at 30008000 ...
   Image Name:   tekkaman
   Created:      2010-03-29  12:59:51 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2277476 Bytes =  2.2 MB
   Load Address: 30008000
   Entry Point:  30008040
   Verifying Checksum ... OK
   XIP Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 2.6.33.1 (tekkaman@MAGI-Linux) (gcc version 4.3.2 (crosstool-NG-1.6.1-tekkaman) ) #5 Mon
Mar 29 20:58:50 CST 2010
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: MINI2440
（略）
```

第5章　U-boot 源码简要分析

本次移植使用的是 U-boot-2009.11。

先来看看源码目录结构，再按照代码的执行顺序简单地分析源码。

5.1 U-boot 源码整体框架

源码解压以后，我们可以看到以下的文件和文件夹：

cpu	与处理器相关的文件。每个子目录中都包括 cpu.c 和 interrupt.c、start.S、u-boot.lds。 **cpu.c** 初始化 CPU、设置指令 Cache 和数据 Cache 等 **interrupt.c** 设置系统的各种中断和异常 **start.S** 是 U-boot 启动时执行的第一个文件，它主要做最早其的系统初始化，代码重定向和设置系统堆栈，为进入 U-boot 第二阶段的 C 程序奠定基础 **u-boot.lds** 链接脚本文件，对于代码的最后组装非常重要
board	已经支持的所有开发板相关文件，其中包含 SDRAM 初始化代码、Flash 底层驱动、板级初始化文件。 其中的 **config.mk** 文件定义了 TEXT_BASE，也就是代码在内存的其实地址，非常重要。
common	与处理器体系结构无关的通用代码，U-boot 的命令解析代码 /common/command.c、所有命令的上层代码 **cmd_*.c**、U-boot 环境变量处理代码 env_*.c、等都位于该目录下
drivers	包含几乎所有外围芯片的驱动，网卡、USB、串口、LCD、Nand Flash 等等
disk **fs** **net**	支持的 CPU 无关的重要子系统： 磁盘驱动的分区处理代码 文件系统：FAT、JFFS2、EXT2 等 网络协议：NFS、TFTP、RARP、DHCP 等等
include	头文件，包括各 CPU 的寄存器定义，文件系统、网络等等 configs 子目录下的文件是与目标板相关的配置头文件
doc	U-Boot 的说明文档，在修改配置文件的时候可能用得上
lib_arm lib_avr32　　lib_mips lib_nios lib_blackfin　lib_nios2 lib_generic　lib_ppc lib_i386　　lib_sh lib_m68k　　lib_sparc lib_microblaze	处理器体系相关的初始化文件 比较重要的是其中的 board.c 文件，几乎是 U-boot 的所有架构第二阶段代码入口函数和相关初始化函数存放的地方。
api examples	外部扩展应用程序的 API 和范例

nand_spl onenand_ipl post	一些特殊构架需要的启动代码和上电自检程序代码
libfdt	支持平坦设备树(flattened device trees)的库文件
tools	编译 S-Record 或 U-Boot 映像等相关工具，制作 bootm 引导 的内核映像文件工具 mkimage 源码就在此
Makefile MAKEALL config.mk rules.mk mkconfig	控制整个编译过程的主 Makefile 文件和规则文件
CHANGELOG CHANGELOG-before-U-Boot-1.1.5 COPYING CREDITS MAINTAINERS README	一些介绍性的文档、版权说明

标为红色的是移植时比较重要的文件或文件夹。

5.2 U-boot 代码的大致执行流程（以 S3C24x0 为例）

从链接脚本文件 u-boot.lds 中可以找到代码的起始:

```
OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
{
        . = 0x00000000;

        . = ALIGN(4);
        .text :
        {
                cpu/arm920t/start.o        (.text)
                *(.text)
        }
…...
```

从中知道程序的入口点是_start，定位于 cpu/arm920t /start.S（即 u-boot 启动的第一阶段）。

下面我们来仔细分析一下 start.S。（请对照数据手册阅读源码）

`#include <common.h>`
`#include <config.h>`
 	
`/*`
`*****************************************************************`
`*`
`* Jump vector table as in table 3.1 in [1]`
`*`
`*****************************************************************`
`*/`
 	
`.globl _start`
`_start:    b        start_code`
`       ldr      pc, _undefined_instruction`
`       ldr      pc, _software_interrupt`
`       ldr      pc, _prefetch_abort`
`       ldr      pc, _data_abort`
	//位于\include 目录下是一个包含其他头文件的
	头文件
	//位于\include\linux 目录下
	
	u-boot 的主入口，跳入了后面的 start_code

<table>
<tr>
<td>

```
            ldr         pc, _not_used
            ldr         pc, _irq
            ldr         pc, _fiq

_undefined_instruction:      .word undefined_instruction
_software_interrupt: .word software_interrupt
_prefetch_abort:     .word prefetch_abort
_data_abort:                 .word data_abort
_not_used:                   .word not_used
_irq:                        .word irq
_fiq:                        .word fiq

        .balignl 16,0xdeadbeef


/*
*********************************************************************
*
* Startup Code (called from the ARM reset exception vector)
*
* do important init only if we don't start from memory!
* relocate armboot to ram
* setup stack
* jump to second stage
*
*********************************************************************
*/

_TEXT_BASE:
        .word       TEXT_BASE

.globl _armboot_start
_armboot_start:
        .word _start

/*
* These are defined in the board-specific linker script.
*/
.globl _bss_start
_bss_start:
        .word __bss_start

.globl _bss_end
_bss_end:
        .word _end

#ifdef CONFIG_USE_IRQ
/* IRQ stack memory (calculated at run-time) */
.globl IRQ_STACK_START
IRQ_STACK_START:
        .word       0x0badc0de

/* IRQ stack memory (calculated at run-time) */
.globl FIQ_STACK_START
FIQ_STACK_START:
        .word 0x0badc0de
#endif


/*
* the actual start code
*/

start_code:
        /*
         * set the cpu to SVC32 mode
         */
        mrs     r0, cpsr
        bic     r0, r0, #0x1f
        orr     r0, r0, #0xd3
        msr     cpsr, r0

        bl          coloured_LED_init
        bl          red_LED_on
```

</td>
<td>

这些是跳转向量表，和芯片的体系结构有关

ldr 语句的意思是将第二个操作数（如：
_undefined_instruction）指向的地址数据传给
PC

.word 为定义一个 4 字节的空间
undefined_instruction 为地址，即后面标号所
对的偏移地址数据

16 字节对齐，并以 0xdeadbeef 填充，它是个
Magic number 。

这些和上面的一样，定义一个 4 字节的空间存
放地址

代码从这里开始执行！！

让系统进入 SVC（管理员模式）

这些都是为 AT91RM9200 写的

</td>
</tr>
</table>

```
#if         defined(CONFIG_AT91RM9200DK) || defined(CONFIG_AT91RM9200EK)
        /*
         * relocate exception table
         */
        ldr     r0, =_start
        ldr     r1, =0x0
        mov     r2, #16
copyex:
        subs    r2, r2, #1
        ldr     r3, [r0], #4
        str     r3, [r1], #4
        bne     copyex
#endif

#if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410)
        /* turn off the watchdog */

# if defined(CONFIG_S3C2400)
# define pWTCON  0x15300000
# define INTMSK  0x14400008          /* Interupt-Controller base addresses */
# define CLKDIVN 0x14800014          /* clock divisor register */
#else
# define pWTCON  0x53000000
# define INTMSK  0x4A000008          /* Interupt-Controller base addresses */
# define INTSUBMSK         0x4A00001C
# define CLKDIVN 0x4C000014          /* clock divisor register */
# endif

        ldr     r0, =pWTCON
        mov     r1, #0x0
        str     r1, [r0]

        /*
         * mask all IRQs by setting all bits in the INTMR - default
         */
        mov     r1, #0xffffffff
        ldr     r0, =INTMSK
        str     r1, [r0]
# if defined(CONFIG_S3C2410)
        ldr     r1, =0x3ff
        ldr     r0, =INTSUBMSK
        str     r1, [r0]
# endif

        /* FCLK:HCLK:PCLK = 1:2:4 */
        /* default FCLK is 120 MHz ! */
        ldr     r0, =CLKDIVN
        mov     r1, #3
        str     r1, [r0]
#endif      /* CONFIG_S3C2400 || CONFIG_S3C2410 */

        /*
         * we do sys-critical inits only at reboot,
         * not when booting from ram!
         */
#ifndef CONFIG_SKIP_LOWLEVEL_INIT
        bl      cpu_init_crit
#endif

#ifndef CONFIG_SKIP_RELOCATE_UBOOT
relocate:                               /* relocate U-Boot to RAM       */
        adr     r0, _start              /* r0 <- current position of code   */
        ldr     r1, _TEXT_BASE          /* test if we run from flash or RAM */
        cmp     r0, r1                  /* don't reloc during debug     */
        beq     stack_setup

        ldr     r2, _armboot_start
        ldr     r3, _bss_start
        sub     r2, r3, r2              /* r2 <- size of armboot        */
        add     r2, r0, r2              /* r2 <- source end address     */

copy_loop:
        ldmia   r0!, {r3-r10}           /* copy from source address [r0]  */
        stmia   r1!, {r3-r10}           /* copy to   target address [r1]    */
```

系统时钟的寄存器地址定义

关闭看门狗

关闭所有中断

设置时钟的分频比

跳入 **cpu_init_crit** ，这是一个系统初始化函数，他还会调用 **board/*/lowlevel_init.S** 中的 **lowlevel_init** 函数。
主要是对系统总线的初始化，初始化了连接存储器的位宽、速度、刷新率等重要参数。经过这个函数的正确初始化，**Nor Flash**、**SDRAM** 才可以被系统使用。下面的代码重定向就依赖它。

代码重定向，它首先检测自己是否已经在内存中：
如果是直接跳到下面的堆栈初始化代码 stack_setup。
如果不是就将自己从 Nor Flash 中拷贝到内存中

```
        cmp     r0, r2                          /* until source end addreee [r2]  */
        ble     copy_loop
#endif    /* CONFIG_SKIP_RELOCATE_UBOOT */

        /* Set up the stack                                            */
stack_setup:
        ldr     r0, _TEXT_BASE    /* upper 128 KiB: relocated uboot   */
        sub     r0, r0, #CONFIG_SYS_MALLOC_LEN   /* malloc area        */
        sub     r0, r0, #CONFIG_SYS_GBL_DATA_SIZE /* bdinfo            */
#ifdef CONFIG_USE_IRQ
        sub     r0, r0, #(CONFIG_STACKSIZE_IRQ+CONFIG_STACKSIZE_FIQ)
#endif
        sub     sp, r0, #12                     /* leave 3 words for abort-stack   */

clear_bss:
        ldr     r0, _bss_start                  /* find start of bss segment       */
        ldr     r1, _bss_end                    /* stop here                        */
        mov     r2, #0x00000000                 /* clear                            */

clbss_l:strr2, [r0]             /* clear loop...             */
        add     r0, r0, #4
        cmp     r0, r1
        ble     clbss_l

        ldr     pc, _start_armboot

_start_armboot:     .word start_armboot


/*
**************************************************************************
*
* CPU_init_critical registers
*
* setup important registers
* setup memory timing
*
**************************************************************************
*/


#ifndef CONFIG_SKIP_LOWLEVEL_INIT
cpu_init_crit:
        /*
         * flush v4 I/D caches
         */
        mov     r0, #0
        mcr     p15, 0, r0, c7, c7, 0 /* flush v3/v4 cache */
        mcr     p15, 0, r0, c8, c7, 0 /* flush v4 TLB */

        /*
         * disable MMU stuff and caches
         */
        mrc     p15, 0, r0, c1, c0, 0
        bic     r0, r0, #0x00002300 @ clear bits 13, 9:8 (--V- --RS)
        bic     r0, r0, #0x00000087 @ clear bits 7, 2:0 (B--- -CAM)
        orr     r0, r0, #0x00000002 @ set bit 2 (A) Align
        orr     r0, r0, #0x00001000 @ set bit 12 (I) I-Cache
        mcr     p15, 0, r0, c1, c0, 0

        /*
         * before relocating, we have to setup RAM timing
         * because memory timing is board-dependend, you will
         * find a lowlevel_init.S in your board directory.
         */
        mov     ip, lr

        bl      lowlevel_init

        mov     lr, ip
        mov     pc, lr
#endif /* CONFIG_SKIP_LOWLEVEL_INIT */
......
```

自拷贝循环

请注意看英文注释

堆栈初始化代码（为第二阶段的 C 语言做准备）

对 BSS 段清零（为第二阶段的 C 语言做准备）
BSS 段（bss segment）通常是用来存放程序中未初始化的全局变量的一块内存区域。BSS 是英文 Block Started by Symbol 的简称。BSS 段属于静态内存分配。在编译时，编译器已经为他们分配好了空间，只不过他们的值为 0，为了节省空间，在 bin 或 ELF 文件中不占空间。

编译器会计算出 _bss_start 和 _bss_end 的值，不是定义的

跳入第二阶段的 C 语言代码入口
_start_armboot （已经被重定向到内存）

前面所说的 cpu_init_crit 系统初始化函数

操作 CP15 协处理器，

调用 board/*/lowlevel_init.S 中的 lowlevel_init 函数，对系统总线的初始化，初始化了连接存储器的位宽、速度、刷新率等重要参数。经过

| | 这个函数的正确初始化，Nor Flash、SDRAM 才可以被系统使用。 |
| | 后面的代码略，主要是中断相关代码，但是 U-boot 基本不使用中断所以暂且略过。 |

现在我们再来看看 lib_arm/board.c 中的第二阶段入口函数 start_armboot：

| ```c
void start_armboot (void)
{
        init_fnc_t **init_fnc_ptr;
        char *s;
#if defined(CONFIG_VFD) || defined(CONFIG_LCD)
        unsigned long addr;
#endif

        /* Pointer is writable since we allocated a register for it */
        gd = (gd_t*)(_armboot_start - CONFIG_SYS_MALLOC_LEN -
sizeof(gd_t));
        /* compiler optimization barrier needed for GCC >= 3.4 */
        __asm__ __volatile__("": : :"memory");

        memset ((void*)gd, 0, sizeof (gd_t));
        gd->bd = (bd_t*)((char*)gd - sizeof(bd_t));
        memset (gd->bd, 0, sizeof (bd_t));

        gd->flags |= GD_FLG_RELOC;

        monitor_flash_len = _bss_start - _armboot_start;
``` | **gd_t 和 bd_t 这两个数据结构比较重要，建议大家看看。**
分配一个存储全局数据的区域，地址给指针 gd

全局数据的区清零
给 gd->bd（指针）赋值（在 gd 的前面）并清零

gd->flags 赋值，表示已经重定向（在内存中）
monitor_flash_len 为 u-boot 代码长度。 |
| ```c
        for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
                if ((*init_fnc_ptr)() != 0) {
                        hang ();
                }
        }
``` | **初始化循环：**
init_sequence 是一个初始化函数集的函数指针数组（后面讲解）
如果有任何一个函数失败就进入死循环。
这个始化函数集比较重要，建议大家认真跟踪一下。 |
| ```c
        /* armboot_start is defined in the board-specific linker script */
        mem_malloc_init (_armboot_start - CONFIG_SYS_MALLOC_LEN,
                        CONFIG_SYS_MALLOC_LEN);
``` | **初始化堆空间，清零。** |
| ```c
#ifndef CONFIG_SYS_NO_FLASH
        /* configure available FLASH banks */
        display_flash_config (flash_init ());
#endif /* CONFIG_SYS_NO_FLASH */
``` | **初始化 Nor Flash 相关参数，并显示其大小。** |
| ```c
#ifdef CONFIG_VFD
#       ifndef PAGE_SIZE
#        define PAGE_SIZE 4096
#       endif
        /*
         * reserve memory for VFD display (always full pages)
         */
        /* bss_end is defined in the board-specific linker script */
        addr = (_bss_end + (PAGE_SIZE - 1)) & ~(PAGE_SIZE - 1);
        vfd_setmem (addr);
        gd->fb_base = addr;
#endif /* CONFIG_VFD */

#ifdef CONFIG_LCD
        /* board init may have inited fb_base */
        if (!gd->fb_base) {
``` | 初始化 VFD 存储区（LCD 显示相关） |

| ```
#            ifndef PAGE_SIZE
#              define PAGE_SIZE 4096
#            endif
             /*
              * reserve memory for LCD display (always full pages)
              */
             /* bss_end is defined in the board-specific linker script */
             addr = (_bss_end + (PAGE_SIZE - 1)) & ~(PAGE_SIZE - 1);
             lcd_setmem (addr);
             gd->fb_base = addr;
         }
#endif /* CONFIG_LCD */
``` | 初始化 LCD 显存 |
| ```
#if defined(CONFIG_CMD_NAND)
         puts ("NAND:  ");
         nand_init();              /* go init the NAND */
#endif
``` | **初始化 Nand Flash 控制器，并显示其容量大小。** |
| ```
#if defined(CONFIG_CMD_ONENAND)
         onenand_init();
#endif
``` | 初始化 OneNand |
| ```
#ifdef CONFIG_HAS_DATAFLASH
         AT91F_DataflashInit();
         dataflash_print_info();
#endif
``` | 初始化 DataFlash |
| ```
         /* initialize environment */
         env_relocate ();
``` | **初始化环境变量，如果认为没有找到存储其中的，就用默认值并打印："*** Warning - bad CRC, using default environment"。**这是我们常看到的。 |
| ```
#ifdef CONFIG_VFD
         /* must do this after the framebuffer is allocated */
         drv_vfd_init();
#endif /* CONFIG_VFD */
``` | 初始化 VFD（LCD 显示相关） |
| ```
#ifdef CONFIG_SERIAL_MULTI
         serial_initialize();
#endif
``` | 初始化串口。 |
| ```
         /* IP Address */
         gd->bd->bi_ip_addr = getenv_IPaddr ("ipaddr");

         stdio_init ();       /* get the devices list going. */

         jumptable_init ();
``` | 从环境变量里获取 IP 地址<br>初始化标准输入输出设备。比如：串口、LCD、键盘等等<br>初始化全局数据表中的跳转表 gd->jt。<br>跳转表是一个函数指针数组，定义了 u-boot 中基本的常用的函数库，gd->jt 是这个函数指针数组的首指针。 |
| ```
#if defined(CONFIG_API)
         /* Initialize API */
         api_init ();
#endif
``` | 初始化 API，用于为 U-boot 编写的"应用程序" |
| ```
         console_init_r ();           /* fully init console as a device */
``` | 初始化 console，平台无关，不一定是串口哦，如果把标准输出设为 vga，字符会显示在 LCD 上。 |
| ```
#if defined(CONFIG_ARCH_MISC_INIT)
         /* miscellaneous arch dependent initialisations */
         arch_misc_init ();
#endif
#if defined(CONFIG_MISC_INIT_R)
         /* miscellaneous platform dependent initialisations */
         misc_init_r ();
``` | 平台相关的其他初始化，有的平台有 |

```c
#endif

        /* enable exceptions */
        enable_interrupts ();

        /* Perform network card initialisation if necessary */
#ifdef CONFIG_DRIVER_TI_EMAC
        /* XXX: this needs to be moved to board init */
extern void davinci_eth_set_mac_addr (const u_int8_t *addr);
        if (getenv ("ethaddr")) {
                uchar enetaddr[6];
                eth_getenv_enetaddr("ethaddr", enetaddr);
                davinci_eth_set_mac_addr(enetaddr);
        }
#endif

#if defined(CONFIG_DRIVER_SMC91111) || defined
(CONFIG_DRIVER_LAN91C96)
        /* XXX: this needs to be moved to board init */
        if (getenv ("ethaddr")) {
                uchar enetaddr[6];
                eth_getenv_enetaddr("ethaddr", enetaddr);
                smc_set_mac_addr(enetaddr);
        }
#endif /* CONFIG_DRIVER_SMC91111 || CONFIG_DRIVER_LAN91C96 */

        /* Initialize from environment */
        if ((s = getenv ("loadaddr")) != NULL) {
                load_addr = simple_strtoul (s, NULL, 16);
        }
#if defined(CONFIG_CMD_NET)
        if ((s = getenv ("bootfile")) != NULL) {
                copy_filename (BootFile, s, sizeof (BootFile));
        }
#endif

#ifdef BOARD_LATE_INIT
        board_late_init ();
#endif

#ifdef CONFIG_GENERIC_MMC
        puts ("MMC:   ");
        mmc_initialize (gd->bd);
#endif

#ifdef CONFIG_BITBANGMII
        bb_miiphy_init();
#endif
#if defined(CONFIG_CMD_NET)
#if defined(CONFIG_NET_MULTI)
        puts ("Net:   ");
#endif
        eth_initialize(gd->bd);
#if defined(CONFIG_RESET_PHY_R)
        debug ("Reset Ethernet PHY\n");
        reset_phy();
#endif
#endif
        /* main_loop() can return to retry autoboot, if so just run it again. */
        for (;;) {
```

中断使能（一般不使用，很多平台此函数是空的）

TI 芯片中的内置 MAC 初始化（平台相关）

一种网卡芯片初始化（平台相关）

获取 bootfile 参数

一些板级初始化（有的板子有）

SD 卡/MMC 控制器初始化

MII 相关初始化

网卡初始化

进入主循环，其中会读取 bootdelay 和 bootcmd

在 bootdelay 时间内按下键进入命令行，否则执行 bootcmd 的命令。

`        main_loop ();`
`    }`
 	
`    /* NOTREACHED - no way out of command loop except booting */`
`}`	

标有红色的是比较重要的地方。大致的 U-boot 启动流程就简单介绍到这。

第6章　U-boot 在 mini2440 上的移植

通过上面的叙述，大家应该比较了解 U-boot 的大致情况，下面开始移植工作了。

我们要做的工作是移植，就是根据不同的地方做修改。U-Boot 一直都没有支持 S3C2440，移植仍是用 U-Boot 支持的友善之臂 SBC2410 的文件作蓝本来移植。所以移植所要做的就是针对 S3C2440 和 S3C2410 的不同，以及 SBC2410 和 mini2440 开发板的外设不同作相应的修改，并增加新的功能。

移植之前必须对 S3C2440 和 S3C2410 有所了解，移植过程和芯片关系紧密。特别是 Nandboot 的原理和 Norboot 的内部 ram 映射原理必须搞清楚。

S3C2440 和 S3C2410 的区别主要是 2440 的主频更高，接口方面，增加了摄像头接口和 AC'97 音频接口；寄存器方面，除了新增模块的寄存器外，NAND FIASH 控制器的寄存器有较大的变化，芯片时钟频率控制寄存器有一定的变化，其他寄存器基本是兼容的。

下面大部分以补丁的形式介绍移植过程:

6.1 建立开发板文件，测试编译环境

6.1.1 修改顶层 Makefile

目的：定义交叉编译工具链和开发板配置选项。

```
diff -aurNp u-boot-2009.11/Makefile u-boot-2009.11_tekkaman/Makefile
--- u-boot-2009.11/Makefile 2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/Makefile     2010-03-28 17:16:12.000000000 +0800
@@ -157,7 +157,7 @@ sinclude $(obj)include/autoconf.mk
 # load ARCH, BOARD, and CPU configuration
 include $(obj)include/config.mk
 export      ARCH CPU BOARD VENDOR SOC
-
+CROSS_COMPILE =  arm-tekkaman-linux-gnueabi-
 # set default to nothing for native builds
 ifeq ($(HOSTARCH),$(ARCH))
 CROSS_COMPILE ?=
@@ -3046,6 +3046,9 @@ smdk2400_config            :          unconfig

 smdk2410_config  :          unconfig
    @$(MKCONFIG) $(@:_config=) arm arm920t smdk2410 samsung s3c24x0
+
```

```
+mini2440_config  :       unconfig
+   @$(MKCONFIG) $(@:_config=) arm arm920t mini2440 tekkamanninja s3c24x0


 SX1_stdout_serial_config \
 SX1_config:              unconfig
```

开发板配置选项中各项的含义如下:

arm	CPU 的架构(ARCH)
arm920t	CPU 的类型(CPU),其对应于 cpu/arm920t 子目录
tekkamanninja	开发者/或经销商(vender),对应于 board/tekkamanninja 目录
mini2440	开发板的型号(BOARD),对应于 board/tekkamanninja/mini2440 目录
s3c24x0	片上系统(SOC)定义

6.1.2 在/board 中建立 mini2440 目录和文件

在/board 目录中建立开发板 mini2440 的目录,并复制 sbc2410x 的文件到此,做适当修改。目的:以 sbc2410x 为蓝本,加快移植进度。

由于上一步板子的 vender 中填了 tekkamanninja ,所以开发板 mini2440 目录一定要建在/board 子目录中的 tekkamanninja 目录下 ,否则编译出错。

```
cd board
mkdir -p tekkamanninja/mini2440
cp -arf sbc2410x/* tekkamanninja/mini2440/
cd tekkamanninja/mini2440/
mv sbc2410x.c mini2440.c
```

修改mini2440 目录下的Makefile文件 (现在先不用动,后面还会修改此文件):

```
@@ -25,7 +25,7 @@ include $(TOPDIR)/config.mk

 LIB     = $(obj)lib$(BOARD).a

-COBJS         := sbc2410x.o flash.o
+COBJS         := mini2440.o  flash.o
 SOBJS:= lowlevel_init.o

 SRCS  := $(SOBJS:.o=.S) $(COBJS:.o=.c)
```

6.1.3 在 include/configs/中建立开发板配置文件

因为 sbc2410x 和 mini2440 最接近,所以以 sbc2410x 的配置为蓝本。

```
cp include/configs/sbc2410x.h include/configs/mini2440.h
```

6.1.4 测试编译环境

在 U-boot 源码的根目录下：

```
make mini2440_config
Configuring for mini2440 board...
make
```

可能出现的问题:

如果出现：

（1）配置出错

```
    make mini2440_config
    Makefile:????: *** 遗漏分隔符 。 停止。
```

请在 U-boot 的根目录下 Makefile 的

"@$(MKCONFIG) $(@:_config=) arm arm920t mini2440 tekkamanninja s3c24x0 "

前加上"Tab"键，这是 Makefile 的规则：所有命令都必须以"Tab"开头。

（2）如果编译时出现以下错误（这是编译器的问题，没出错就不要修改）：

```
uses hardware FP, whereas u-boot uses software FP
```

修正的方法：

```
diff -aurNp u-boot-2009.11/cpu/arm920t/config.mk u-boot-2009.11_tekkaman/cpu/arm920t/config.mk
--- u-boot-2009.11/cpu/arm920t/config.mk            2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/config.mk    2010-03-28 17:16:12.000000000 +0800
@@ -21,7 +21,8 @@
 # MA 02111-1307 USA
 #

-PLATFORM_RELFLAGS += -fno-common -ffixed-r8 -msoft-float
+PLATFORM_RELFLAGS += -fno-common -ffixed-r8
+#-msoft-float

 PLATFORM_CPPFLAGS += -march=armv4
 # ============================================================================
```

以上测试通过后，说明编译环境和基本的开发板的代码创建都没有问题。现在编译出来的都是蓝本 SBC2410 的，下面按照代码的执行流程来针对 mini2440 做修改。

6.2 第一阶段:探索启动代码

首先进入/cpu/arm920t/start.S

6.2.1 关闭为 AT9200 写的 LED 跳转

```
diff -aurNp u-boot-2009.11/cpu/arm920t/start.S u-boot-2009.11_tekkaman/cpu/arm920t/start.S
--- u-boot-2009.11/cpu/arm920t/start.S   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/start.S       2010-03-28 17:16:12.000000000 +0800
@@ -114,8 +114,8 @@ start_code:
        orr     r0, r0, #0xd3
        msr     cpsr, r0

-       bl      coloured_LED_init
-       bl      red_LED_on
+@      bl      coloured_LED_init
+@      bl      red_LED_on

 #if     defined(CONFIG_AT91RM9200DK) || defined(CONFIG_AT91RM9200EK)
        /*
```

6.2.2 修改 CPU 频率初始化设置

2410 和 2440 相比一个不同的地方就是 PLL 的初始化参数不一样，在数据手册可以查到。这里一开始就将频率升到 405MHz。其中还包括了中断掩码的修正。

```
diff -aurNp u-boot-2009.11/cpu/arm920t/start.S u-boot-2009.11_tekkaman/cpu/arm920t/start.S
--- u-boot-2009.11/cpu/arm920t/start.S   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/start.S       2010-03-28 17:16:12.000000000 +0800
@@ -131,7 +131,7 @@ copyex:
        bne     copyex
 #endif

-#if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410)
+#if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410)|| defined(CONFIG_S3C2440)
        /* turn off the watchdog */

 # if defined(CONFIG_S3C2400)
@@ -144,6 +144,12 @@ copyex:
 #  define INTSUBMSK  0x4A00001C
 #  define CLKDIVN      0x4C000014     /* clock divisor register */
 # endif
+#define CLK_CTL_BASE        0x4C000000     /* tekkaman */
+#define MDIV_405     0x7f << 12        /* tekkaman */
+#define PSDIV_405    0x21             /* tekkaman */
+#define MDIV_200     0xa1 << 12       /* tekkaman */
+#define PSDIV_200    0x31             /* tekkaman */
+

        ldr     r0, =pWTCON
        mov     r1, #0x0
@@ -156,17 +162,53 @@ copyex:
        ldr     r0, =INTMSK
        str     r1, [r0]
 # if defined(CONFIG_S3C2410)
-       ldr     r1, =0x3ff
+       ldr     r1, =0x7ff
        ldr     r0, =INTSUBMSK
        str     r1, [r0]
```

```
 # endif

+#if defined(CONFIG_S3C2440)
+       ldr     r1, =0x7fff
+       ldr     r0, =INTSUBMSK
+       str     r1, [r0]
+#endif
+
+
+#if defined(CONFIG_S3C2440)
+       /* FCLK:HCLK:PCLK = 1:4:8 */
+       ldr     r0, =CLKDIVN
+       mov     r1, #5
+       str     r1, [r0]
+
+       mrc     p15, 0, r1, c1, c0, 0
+       orr     r1, r1, #0xc0000000
+       mcr     p15, 0, r1, c1, c0, 0
+
+
+       mov     r1, #CLK_CTL_BASE
+       mov     r2, #MDIV_405
+       add     r2, r2, #PSDIV_405
+       str     r2, [r1, #0x04]            /* MPLLCON tekkaman */
+
+#else
        /* FCLK:HCLK:PCLK = 1:2:4 */
        /* default FCLK is 120 MHz ! */
        ldr     r0, =CLKDIVN
        mov     r1, #3
        str     r1, [r0]
-#endif  /* CONFIG_S3C2400 || CONFIG_S3C2410 */
+
+
+       mrc     p15, 0, r1, c1, c0, 0
+       orr     r1, r1, #0xc0000000
+       mcr     p15, 0, r1, c1, c0, 0       /*write ctrl register tekkaman*/
+
+
+       mov     r1, #CLK_CTL_BASE    /* tekkaman*/
+       mov     r2, #MDIV_200
+       add     r2, r2, #PSDIV_200
+       str     r2, [r1, #0x04]
+#endif
+#endif /* CONFIG_S3C2400 || CONFIG_S3C2410  || CONFIG_S3C2440*/

        /*
         * we do sys-critical inits only at reboot,
```

6.2.3 修改 lowlevel_init.S 文件

为了匹配mini2440 的存储器配置（总线上连接的Nor Flash 和SDRAM），需要修改 lowlevel_init.S文件。这个所连接的Nor Flash位数有关。至于SDRAM的参数，可以从芯片手册查到。据说有人将其 64MB的内存升到了 128MB，其参数就是在这里修改的，有需要可以看 MINI2440: Auto probe for SDRAM size 。

以下的 64MB 内存的参数修改：

```
diff -aurNp u-boot-2009.11/board/sbc2410x/lowlevel_init.S u-boot-
2009.11_tekkaman/board/sbc2410x/lowlevel_init.S
--- u-boot-2009.11/board/sbc2410x/lowlevel_init.S        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/board/sbc2410x/lowlevel_init.S    2010-03-28 17:16:12.000000000 +0800
@@ -116,10 +116,18 @@
 /* REFRESH parameter */
 #define REFEN                          0x1      /* Refresh enable */
 #define TREFMD                         0x0      /* CBR(CAS before RAS)/Auto refresh */
-#define Trp                    0x0      /* 2clk */
 #define Trc                    0x3      /* 7clk */
 #define Tchr                   0x2      /* 3clk */
+
+#if defined(CONFIG_S3C2440)
+#define Trp                    0x2      /* 4clk */
+#define REFCNT                         1012
+#else
+#define Trp                    0x0      /* 2clk */
 #define REFCNT                         0x0459
+#endif
+
+
 /***********************************/

 _TEXT_BASE:
```

在这个 lowlevel_init.S 有一个小 bug，使得无法使用 OpenJTAG 下载到内存中直接运行，修正如下：

```
diff -aurNp u-boot-2009.11/board/sbc2410x/lowlevel_init.S u-boot-
2009.11_tekkaman/board/sbc2410x/lowlevel_init.S
--- u-boot-2009.11/board/sbc2410x/lowlevel_init.S        2009-12-16 06:20:54.000000000 +0800
@@ -131,8 +139,10 @@ lowlevel_init:
        /* make r0 relative the current location so that it */
        /* reads SMRDATA out of FLASH rather than memory ! */
        ldr    r0, =SMRDATA
-       ldr     r1, _TEXT_BASE
+       ldr     r1, =lowlevel_init
        sub     r0, r0, r1
+       adr     r3, lowlevel_init            /* r3 <- current position of code   */
+       add     r0, r0, r3
        ldr     r1, =BWSCON  /* Bus Width Status Controller */
        add     r2, r0, #13*4
 0:
```

6.2.4 修改代码重定向部分

Tekkaman Ninja从 2009.08 开始就在启动时增加了启动时检测自身是否已经在SDRAM中(通过OpenJTAG载入)，以及芯片是Norboot还是Nandboot的机制，来决定代码重定向的方式，使得编译出的bin文件可以同时烧入Nand Flash和Nor flash，以及被OpenJTAG载入进行调试。至于这部分的原理，在Tekkaman Ninja的博客文章 《在U-boot下实现自动识别启动Flash的原理（针对S3C24x0)》 中有详细叙述。

```
diff -aurNp u-boot-2009.11/cpu/arm920t/start.S u-boot-2009.11_tekkaman/cpu/arm920t/start.S
--- u-boot-2009.11/cpu/arm920t/start.S   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/start.S        2010-03-28 17:16:12.000000000 +0800
@@ -176,13 +218,189 @@ copyex:
         bl       cpu_init_crit
 #endif

-#ifndef CONFIG_SKIP_RELOCATE_UBOOT
-relocate:                               /* relocate U-Boot to RAM          */
+/*************** CHECK_CODE_POSITION ****************************************/
        adr      r0, _start              /* r0 <- current position of code   */
        ldr      r1, _TEXT_BASE          /* test if we run from flash or RAM */
        cmp      r0, r1                  /* don't reloc during debug       */
        beq      stack_setup
+/*************** CHECK_CODE_POSITION ****************************************/
+
+/*************** CHECK_BOOT_FLASH ****************************************/
+        ldr      r1, =( (4<<28)|(3<<4)|(3<<2) )           /* address of Internal SRAM  0x4000003C*/
+        mov      r0, #0                  /* r0 = 0 */
+        str      r0, [r1]
+
+
+        mov      r1, #0x3c               /* address of men  0x0000003C*/
+        ldr      r0, [r1]
+        cmp      r0, #0
+        bne      relocate
+
+        /* recovery  */
+        ldr      r0, =(0xdeadbeef)
+        ldr      r1, =( (4<<28)|(3<<4)|(3<<2) )
+        str      r0, [r1]
+/*************** CHECK_BOOT_FLASH ****************************************/
+
+/*************** NAND_BOOT ****************************************/
+
+#define LENGTH_UBOOT 0x60000
+#define NAND_CTL_BASE 0x4E000000
+
+#ifdef CONFIG_S3C2440
+/* Offset */
+#define oNFCONF 0x00
+#define oNFCONT 0x04
+#define oNFCMD 0x08
+#define oNFSTAT 0x20
+
+        @ reset NAND
+        mov      r1, #NAND_CTL_BASE
+        ldr      r2, =( (7<<12)|(7<<8)|(7<<4)|(0<<0) )
+        str      r2, [r1, #oNFCONF]
+        ldr      r2, [r1, #oNFCONF]
+
+        ldr      r2, =( (1<<4)|(0<<1)|(1<<0) )     @ Active low CE Control
+        str      r2, [r1, #oNFCONT]
+        ldr      r2, [r1, #oNFCONT]
+
+        ldr      r2, =(0x6)        @ RnB Clear
+        str      r2, [r1, #oNFSTAT]
+        ldr      r2, [r1, #oNFSTAT]
+
+        mov      r2, #0xff@ RESET command
```

```
+        strb    r2, [r1, #oNFCMD]
+
+        mov     r3, #0    @ wait
+nand1:
+        add     r3, r3, #0x1
+        cmp     r3, #0xa
+        blt     nand1
+
+nand2:
+        ldr     r2, [r1, #oNFSTAT]       @ wait ready
+        tst     r2, #0x4
+        beq     nand2
+
+
+        ldr     r2, [r1, #oNFCONT]
+        orr     r2, r2, #0x2      @ Flash Memory Chip Disable
+        str     r2, [r1, #oNFCONT]
+
+        @ get read to call C functions (for nand_read())
+        ldr     sp, DW_STACK_START          @ setup stack pointer
+        mov     fp, #0    @ no previous frame, so fp=0
+
+        @ copy U-Boot to RAM
+        ldr     r0, =TEXT_BASE
+        mov     r1, #0x0
+        mov     r2, #LENGTH_UBOOT
+        bl      nand_read_ll
+        tst     r0, #0x0
+        beq     ok_nand_read
+
+bad_nand_read:
+loop2:
+        b       loop2    @ infinite loop
+ok_nand_read:
+        @ verify
+        mov     r0, #0
+        ldr     r1, =TEXT_BASE
+        mov     r2, #0x400        @ 4 bytes * 1024 = 4K-bytes
+go_next:
+        ldr     r3, [r0], #4
+        ldr     r4, [r1], #4
+        teq     r3, r4
+        bne     notmatch
+        subs    r2, r2, #4
+        beq     stack_setup
+        bne     go_next

+notmatch:
+loop3:
+        b       loop3    @ infinite loop
+#endif
+
+#ifdef  CONFIG_S3C2410
+
+/* Offset */
+#define oNFCONF 0x00
+#define oNFCMD 0x04
+#define oNFSTAT 0x10
+
+        @ reset NAND
```

```
+        mov      r1, #NAND_CTL_BASE
+        ldr      r2, =0xf830        @ initial value
+        str      r2, [r1, #oNFCONF]
+        ldr      r2, [r1, #oNFCONF]
+        bic      r2, r2, #0x800   @ enable chip
+        str      r2, [r1, #oNFCONF]
+        mov      r2, #0xff            @ RESET command
+        strb     r2, [r1, #oNFCMD]
+
+
+        mov      r3, #0   @ wait
+nand1:
+        add      r3, r3, #0x1
+        cmp      r3, #0xa
+        blt      nand1
+
+nand2:
+        ldr      r2, [r1, #oNFSTAT]        @ wait ready
+        tst      r2, #0x1
+        beq      nand2
+
+        ldr      r2, [r1, #oNFCONF]
+        orr      r2, r2, #0x800    @ disable chip
+        str      r2, [r1, #oNFCONF]
+
+        @ get read to call C functions (for nand_read())
+        ldr      sp, DW_STACK_START        @ setup stack pointer
+        mov      fp, #0   @ no previous frame, so fp=0
+
+        @ copy U-Boot to RAM
+        ldr      r0, =TEXT_BASE
+        mov      r1, #0x0
+        mov      r2, #LENGTH_UBOOT
+        bl       nand_read_ll
+        tst      r0, #0x0
+        beq      ok_nand_read
+
+bad_nand_read:
+loop2:
+        b        loop2    @ infinite loop
+
+
+ok_nand_read:
+        @ verify
+        mov      r0, #0
+        ldr      r1, =TEXT_BASE
+        mov      r2, #0x400        @ 4 bytes * 1024 = 4K-bytes
+go_next:
+        ldr      r3, [r0], #4
+        ldr      r4, [r1], #4
+        teq      r3, r4
+        bne      notmatch
+        subs     r2, r2, #4
+        beq      stack_setup
+        bne      go_next
+
+notmatch:
+loop3:
+        b        loop3    @ infinite loop
+
```

```
+#endif
+/*************** NAND_BOOT ***********************************************/
+
+/*************** NOR_BOOT ***********************************************/
+relocate:                                   /* relocate U-Boot to RAM        */
+    /*********** CHECK_FOR_MAGIC_NUMBER**************/
+        ldr      r1, =(0xdeadbeef)
+        cmp      r0, r1
+        bne      loop3
+    /*********** CHECK_FOR_MAGIC_NUMBER**************/
+        adr      r0, _start                 /* r0 <- current position of code  */
+        ldr      r1, _TEXT_BASE             /* test if we run from flash or RAM */
        ldr      r2, _armboot_start
        ldr      r3, _bss_start
        sub      r2, r3, r2                 /* r2 <- size of armboot         */
@@ -193,7 +411,7 @@ copy_loop:
        stmia    r1!, {r3-r10}              /* copy to   target address [r1]  */
        cmp      r0, r2                     /* until source end addreee [r2]  */
        ble      copy_loop
-#endif /* CONFIG_SKIP_RELOCATE_UBOOT */
+/*************** NOR_BOOT ***********************************************/

        /* Set up the stack                                         */
 stack_setup:
```

在上面添加的代码中有一个跳转：`bl  nand_read_ll`，它跳入是新增的 C 语言文件（board/tekkamanninja/mini2440/nand_read.c）中的函数，这个文件原本是用 vivi 的代码，好来经过了 openmoko 的修改，并支持不同的 Nand Flash 芯片，我又多加了几个个芯片 ID 以支持所有 mini2440 的 Nand Flash。代码如下：

```
/*
 * nand_read.c: Simple NAND read functions for booting from NAND
 *
 * This is used by cpu/arm920/start.S assembler code,
 * and the board-specific linker script must make sure this
 * file is linked within the first 4kB of NAND flash.
 *
 * Taken from GPLv2 licensed vivi bootloader,
 * Copyright (C) 2002 MIZI Research, Inc.
 *
 * Author: Hwang, Chideok <hwang@mizi.com>
 * Date  : $Date: 2004/02/04 10:37:37 $
 *
 * u-boot integration and bad-block skipping (C) 2006 by OpenMoko, Inc.
 * Author: Harald Welte <laforge@openmoko.org>
 */

#include <common.h>
#include <linux/mtd/nand.h>


#define __REGb(x)      (*(volatile unsigned char *)(x))
#define __REGw(x)      (*(volatile unsigned short *)(x))
#define __REGi(x)      (*(volatile unsigned int *)(x))
#define NF_BASE             0x4e000000
#if defined(CONFIG_S3C2410)
#define NFCONF              __REGi(NF_BASE + 0x0)
#define NFCMD          __REGb(NF_BASE + 0x4)
```

```c
#define NFADDR              __REGb(NF_BASE + 0x8)
#define NFDATA              __REGb(NF_BASE + 0xc)
#define NFSTAT              __REGb(NF_BASE + 0x10)
#define NFSTAT_BUSY 1
#define nand_select()    (NFCONF &= ~0x800)
#define nand_deselect()(NFCONF |= 0x800)
#define nand_clear_RnB()        do {} while (0)
#elif defined(CONFIG_S3C2440) || defined(CONFIG_S3C2442)
#define NFCONF              __REGi(NF_BASE + 0x0)
#define NFCONT              __REGi(NF_BASE + 0x4)
#define NFCMD           __REGb(NF_BASE + 0x8)
#define NFADDR              __REGb(NF_BASE + 0xc)
#define NFDATA              __REGb(NF_BASE + 0x10)
#define NFDATA16     __REGw(NF_BASE + 0x10)
#define NFSTAT              __REGb(NF_BASE + 0x20)
#define NFSTAT_BUSY 1
#define nand_select()    (NFCONT &= ~(1 << 1))
#define nand_deselect()(NFCONT |= (1 << 1))
#define nand_clear_RnB()        (NFSTAT |= (1 << 2))
#endif

static inline void nand_wait(void)
{
        int i;

        while (!(NFSTAT & NFSTAT_BUSY))
                for (i=0; i<10; i++);
}

struct boot_nand_t {
        int page_size;
        int block_size;
        int bad_block_offset;
//      unsigned long size;
};

#if 0
#if defined(CONFIG_S3C2410) || defined(CONFIG_MINI2440)
/* configuration for 2410 with 512byte sized flash */
#define NAND_PAGE_SIZE          512
#define BAD_BLOCK_OFFSET 5
#define NAND_BLOCK_MASK         (NAND_PAGE_SIZE - 1)
#define NAND_BLOCK_SIZE         0x4000
#else
/* configuration for 2440 with 2048byte sized flash */
#define NAND_5_ADDR_CYCLE
#define NAND_PAGE_SIZE          2048
#define BAD_BLOCK_OFFSET NAND_PAGE_SIZE
#define NAND_BLOCK_MASK         (NAND_PAGE_SIZE - 1)
#define NAND_BLOCK_SIZE         (NAND_PAGE_SIZE * 64)
#endif

/* compile time failure in case of an invalid configuration */
#if defined(CONFIG_S3C2410) && (NAND_PAGE_SIZE != 512)
#error "S3C2410 does not support nand page size != 512"
#endif
#endif

static int is_bad_block(struct boot_nand_t * nand, unsigned long i)
{
```

```c
        unsigned char data;
        unsigned long page_num;

        nand_clear_RnB();
        if (nand->page_size == 512) {
                NFCMD = NAND_CMD_READOOB; /* 0x50 */
                NFADDR = nand->bad_block_offset & 0xf;
                NFADDR = (i >> 9) & 0xff;
                NFADDR = (i >> 17) & 0xff;
                NFADDR = (i >> 25) & 0xff;
        } else if (nand->page_size == 2048) {
                page_num = i >> 11; /* addr / 2048 */
                NFCMD = NAND_CMD_READ0;
                NFADDR = nand->bad_block_offset & 0xff;
                NFADDR = (nand->bad_block_offset >> 8) & 0xff;
                NFADDR = page_num & 0xff;
                NFADDR = (page_num >> 8) & 0xff;
                NFADDR = (page_num >> 16) & 0xff;
                NFCMD = NAND_CMD_READSTART;
        } else {
                return -1;
        }
        nand_wait();
        data = (NFDATA & 0xff);
        if (data != 0xff)
                return 1;

        return 0;
}

static int nand_read_page_ll(struct boot_nand_t * nand, unsigned char *buf, unsigned long addr)
{
        unsigned short *ptr16 = (unsigned short *)buf;
        unsigned int i, page_num;

        nand_clear_RnB();

        NFCMD = NAND_CMD_READ0;

        if (nand->page_size == 512) {
                /* Write Address */
                NFADDR = addr & 0xff;
                NFADDR = (addr >> 9) & 0xff;
                NFADDR = (addr >> 17) & 0xff;
                NFADDR = (addr >> 25) & 0xff;
        } else if (nand->page_size == 2048) {
                page_num = addr >> 11; /* addr / 2048 */
                /* Write Address */
                NFADDR = 0;
                NFADDR = 0;
                NFADDR = page_num & 0xff;
                NFADDR = (page_num >> 8) & 0xff;
                NFADDR = (page_num >> 16) & 0xff;
                NFCMD = NAND_CMD_READSTART;
        } else {
                return -1;
        }
        nand_wait();

#if defined(CONFIG_S3C2410)
```

```c
        for (i = 0; i < nand->page_size; i++) {
                *buf = (NFDATA & 0xff);
                buf++;
        }
#elif defined(CONFIG_S3C2440) || defined(CONFIG_S3C2442)
        for (i = 0; i < (nand->page_size>>1); i++) {
                *ptr16 = NFDATA16;
                ptr16++;
        }
#endif

        return nand->page_size;
}

static unsigned short nand_read_id()
{
        unsigned short res = 0;
        NFCMD = NAND_CMD_READID;
        NFADDR = 0;
        res = NFDATA;
        res = (res << 8) | NFDATA;
        return res;
}

extern unsigned int dynpart_size[];

/* low level nand read function */
int nand_read_ll(unsigned char *buf, unsigned long start_addr, int size)
{
        int i, j;
        unsigned short nand_id;
        struct boot_nand_t nand;

        /* chip Enable */
        nand_select();
        nand_clear_RnB();

        for (i = 0; i < 10; i++)
                ;
        nand_id = nand_read_id();
        if (0) { /* dirty little hack to detect if nand id is misread */
                unsigned short * nid = (unsigned short *)0x31fffff0;
                *nid = nand_id;
        }

    if (nand_id == 0xec76 ||            /* Samsung K91208 */
        nand_id == 0xad76 ) {    /*Hynix HY27US08121A*/
                nand.page_size = 512;
                nand.block_size = 16 * 1024;
                nand.bad_block_offset = 5;
        //      nand.size = 0x4000000;
        } else if (nand_id == 0xecf1 ||    /* Samsung K9F1G08U0B */
                   nand_id == 0xecda ||  /* Samsung K9F2G08U0B */
                   nand_id == 0xecd3 )   { /* Samsung K9K8G08 */
                nand.page_size = 2048;
                nand.block_size = 128 * 1024;
                nand.bad_block_offset = nand.page_size;
        //      nand.size = 0x8000000;
        } else {
                return -1; // hang
```

```
        }
        if ((start_addr & (nand.block_size-1)) || (size & ((nand.block_size-1))))
                return -1;          /* invalid alignment */

        for (i=start_addr; i < (start_addr + size);) {
#ifdef CONFIG_S3C2410_NAND_SKIP_BAD
                if (i & (nand.block_size-1)== 0) {
                        if (is_bad_block(&nand, i) ||
                          is_bad_block(&nand, i + nand.page_size)) {
                                /* Bad block */
                                i += nand.block_size;
                                size += nand.block_size;
                                continue;
                        }
                }
#endif
                j = nand_read_page_ll(&nand, buf, i);
                i += j;
                buf += j;
        }

        /* chip Disable */
        nand_deselect();

        return 0;
}
```

在添加了这个文件之后，记得要在 Makefile 里加上对这个文件的编译。

```
diff -aurNp u-boot-2009.11/board/sbc2410x/Makefile u-boot-2009.11_tekkaman/board/sbc2410x/Makefile
--- u-boot-2009.11/board/sbc2410x/Makefile      2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/board/sbc2410x/Makefile 2010-03-28 17:16:12.000000000 +0800
@@ -25,7 +25,7 @@ include $(TOPDIR)/config.mk

 LIB     = $(obj)lib$(BOARD).a

-COBJS           := sbc2410x.o flash.o
+COBJS           := nand_read.o mini2440.o  flash.o
 SOBJS:= lowlevel_init.o

 SRCS  := $(SOBJS:.o=.S) $(COBJS:.o=.c)
```

6.2.5 增加 LED 的点亮操作

作用是显示代码进度，对 Debug 有帮助。代码在跳转到第二阶段代码 start_armboot 函数前会亮起一个 LED 灯。（还记得前面有调用 C 语言的 nand_read_ll 函数吧，初始化堆栈的定义就在此）

```
diff -aurNp u-boot-2009.11/cpu/arm920t/start.S u-boot-2009.11_tekkaman/cpu/arm920t/start.S
--- u-boot-2009.11/cpu/arm920t/start.S   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/start.S      2010-03-28 17:16:12.000000000 +0800
@@ -217,8 +435,29 @@ clbss_l:str     r2, [r0]            /* clear loop...

        ldr        pc, _start_armboot
```

```
-_start_armboot:              .word start_armboot
+#if defined(CONFIG_MINI2440_LED)
+#define GPIO_CTL_BASE 0x56000000
+#define oGPIO_B 0x10
+#define oGPIO_CON 0x0
+/* R/W, Configures the pins of the port */
+#define oGPIO_DAT 0x4
+#define oGPIO_UP 0x8
+/* R/W, Pull-up disable register */
+        mov     r1, #GPIO_CTL_BASE
+        add     r1, r1, #oGPIO_B
+        ldr     r2, =0x295551
+        str     r2, [r1, #oGPIO_CON]
+        mov     r2, #0xff
+        str     r2, [r1, #oGPIO_UP]
+        ldr     r2, =0x1c1
+        str     r2, [r1, #oGPIO_DAT]
+#endif

+_start_armboot:              .word start_armboot
+#define STACK_BASE 0x33f00000
+#define STACK_SIZE 0x10000
+        .align   2
+DW_STACK_START:  .word    STACK_BASE+STACK_SIZE-4

 /*
  **************************************************************************
```

到这里，启动的第一阶段就修改完了，但是在 U-boot-1.3.3 之后，这些本应放在 bin 文件前 4K 的代码会被放到后面，以至启动失败。所以必须手动修改链接时使用的.lds 文件，使得这些代码被放在 bin 文件的最前面：

```
diff -aurNp u-boot-2009.11/cpu/arm920t/u-boot.lds u-boot-2009.11_tekkaman/cpu/arm920t/u-boot.lds
--- u-boot-2009.11/cpu/arm920t/u-boot.lds          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/u-boot.lds    2010-03-28 17:16:12.000000000 +0800
@@ -40,6 +40,8 @@ SECTIONS
        .text :
        {
                cpu/arm920t/start.o        (.text)
+               board/tekkamanninja/mini2440/lowlevel_init.o    (.text)
+               board/tekkamanninja/mini2440/nand_read.o        (.text)
                *(.text)
        }
```

6.3 第二阶段:修改初始化代码

代码运行到了第二阶段代码 lib_arm/board.c 中的 start_armboot 函数，开始了系统的全面初始化。

6.3.1 修改 lib_arm/board.c 文件

　　这个文件的修改主要是关闭为AT9200写的代码，增加LED的点亮，在初始化console后和进入命令行之前各点亮一个LED（3、4）(第二个LED的点亮在其中的board_init函数中)，增加打印信息（for LCD console）。

```
diff -aurNp u-boot-2009.11/lib_arm/board.c u-boot-2009.11_tekkaman/lib_arm/board.c
--- u-boot-2009.11/lib_arm/board.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/lib_arm/board.c   2010-03-28 17:16:12.000000000 +0800
@@ -49,6 +49,7 @@
 #include <nand.h>
 #include <onenand_uboot.h>
 #include <mmc.h>
+#include <s3c2410.h>    //tekkamanninja for LED

 #ifdef CONFIG_BITBANGMII
 #include <miiphy.h>
@@ -86,7 +87,7 @@ extern void rtl8019_get_enetaddr (uchar
 #include <i2c.h>
 #endif


-
+#if 0
 /***************************************************************************
  * Coloured LED functionality
 ***************************************************************************
@@ -110,6 +111,7 @@ void inline __blue_LED_on(void) {}
 void blue_LED_on(void) __attribute__((weak, alias("__blue_LED_on")));
 void inline __blue_LED_off(void) {}
 void blue_LED_off(void) __attribute__((weak, alias("__blue_LED_off")));
+#endif

 /***************************************************************************
  * Init Utilities                                                        *
@@ -135,7 +137,13 @@ static int init_baudrate (void)

 static int display_banner (void)
 {
-        printf ("\n\n%s\n\n", version_string);
+#if defined(CONFIG_MINI2440_LED)
+        struct s3c24x0_gpio * const gpio = s3c24x0_get_base_gpio();
+        gpio->GPBDAT = 0x101; //tekkamanninja
+#endif
+        printf ("\n\n%s\n\n", version_string);
+        printf (" modified by tekkamanninja (tekkamanninja@163.com)\n");
+        printf (" Love Linux forever!!\n\n");
        debug ("U-Boot code: %08lX -> %08lX  BSS: -> %08lX\n",
            _armboot_start, _bss_start, _bss_end);
 #ifdef CONFIG_MODEM_SUPPORT
@@ -272,7 +280,9 @@ void start_armboot (void)
 #if defined(CONFIG_VFD) || defined(CONFIG_LCD)
        unsigned long addr;
 #endif
-
+#if defined(CONFIG_MINI2440_LED)
+        struct s3c24x0_gpio * const gpio = s3c24x0_get_base_gpio();
+#endif
```

```
        /* Pointer is writable since we allocated a register for it */
        gd = (gd_t*)(_armboot_start - CONFIG_SYS_MALLOC_LEN - sizeof(gd_t));
        /* compiler optimization barrier needed for GCC >= 3.4 */
@@ -434,6 +444,15 @@ extern void davinci_eth_set_mac_addr (co
        reset_phy();
 #endif
 #endif
+#if defined(CONFIG_MINI2440_LED)
+               gpio->GPBDAT = 0x0; //tekkamanninja
+#endif
+
+#if defined(CONFIG_CFB_CONSOLE)
+       printf ("%s\n", version_string);
+       printf ("modified by tekkamanninja\n(tekkamanninja@163.com)\n");
+       printf ("Love Linux forever!!\n");
+#endif
        /* main_loop() can return to retry autoboot, if so just run it again. */
        for (;;) {
                main_loop ();
```

大家可以看到 lib_arm/board.c 中的 start_armboot 函数中调用了很多初始化函数，这些
函数分布在不同的文件中，后面会分别来修改。

6.3.2 修改 board/tekkamanninja/mini2440/mini2440.c 文件。

这个文件负责板级初始化的任务，修改的地方主要包括：增加 LCD 初始化函数、修改
GPIO 设置（这个和开发板的外设连接有关，比如 LCD 和 LED）、LED 的点亮（第二个）、
屏蔽已不使用的 Nand 控制器初始化代码，还有添加网卡芯片（DM9000）的初始化函数。

```
diff -aurNp u-boot-2009.11/board/sbc2410x/sbc2410x.c u-boot-2009.11_tekkaman/
board/tekkamanninja/mini2440/mini2440.c
--- u-boot-2009.11/board/sbc2410x/sbc2410x.c   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/board/tekkamanninja/mini2440/mini2440.c         2010-03-28
17:16:12.000000000 +0800
@@ -31,6 +31,7 @@
 #include <common.h>
 #include <netdev.h>
 #include <s3c2410.h>
+#include <video_fb.h>

 #if defined(CONFIG_CMD_NAND)
 #include <linux/mtd/nand.h>
@@ -45,9 +46,20 @@ DECLARE_GLOBAL_DATA_PTR;
 #define M_PDIV          0x4
 #define M_SDIV          0x1
 #elif FCLK_SPEED==1         /* Fout = 202.8MHz */
-#define M_MDIV          0x5c
-#define M_PDIV          0x4
-#define M_SDIV          0x0
+
+#if defined(CONFIG_S3C2410)
+/* Fout = 202.8MHz */
+#define M_MDIV          0xA1
+#define M_PDIV          0x3
+#define M_SDIV          0x1
```

```
+#endif
+
+#if defined(CONFIG_S3C2440)
+/* Fout = 405MHz */
+#define M_MDIV 0x7f
+#define M_PDIV 0x2
+#define M_SDIV 0x1
+#endif
 #endif

 #define USB_CLOCK 1
@@ -57,8 +69,17 @@ DECLARE_GLOBAL_DATA_PTR;
 #define U_M_PDIV     0x3
 #define U_M_SDIV     0x1
 #elif USB_CLOCK==1
+
+#if defined(CONFIG_S3C2410)
 #define U_M_MDIV     0x48
 #define U_M_PDIV     0x3
+#endif
+
+#if defined(CONFIG_S3C2440)
+#define U_M_MDIV 0x38
+#define U_M_PDIV 0x2
+#endif
+
 #define U_M_SDIV     0x2
 #endif

@@ -96,13 +117,21 @@ int board_init (void)

        /* set up the I/O ports */
        gpio->GPACON = 0x007FFFFF;
+
+#if defined(CONFIG_MINI2440)
+       gpio->GPBCON = 0x00295551;
+#else
        gpio->GPBCON = 0x00044556;
+#endif
+
        gpio->GPBUP = 0x000007FF;
+
        gpio->GPCCON = 0xAAAAAAAA;
-       gpio->GPCUP = 0x0000FFFF;
+       gpio->GPCUP = 0xFFFFFFFF;
        gpio->GPDCON = 0xAAAAAAAA;
-       gpio->GPDUP = 0x0000FFFF;
-       gpio->GPECON = 0xAAAAAAAA;
+       gpio->GPDUP = 0xFFFFFFFF;
+
+   gpio->GPECON = 0xAAAAAAAA;
        gpio->GPEUP = 0x0000FFFF;
        gpio->GPFCON = 0x000055AA;
        gpio->GPFUP = 0x000000FF;
@@ -115,18 +144,82 @@ int board_init (void)
        gpio->EXTINT1=0x22222222;
        gpio->EXTINT2=0x22222222;

+#if defined(CONFIG_S3C2410)
        /* arch number of SMDK2410-Board */
```

```
        gd->bd->bi_arch_number = MACH_TYPE_SMDK2410;
+#endif
+
+#if defined(CONFIG_S3C2440)
+/* arch number of S3C2440-Board */
+        gd->bd->bi_arch_number = MACH_TYPE_MINI2440 ;
+#endif
+

        /* adress of boot parameters */
        gd->bd->bi_boot_params = 0x30000100;

        icache_enable();
        dcache_enable();
-
+#if     defined(CONFIG_MINI2440_LED)
+        gpio->GPBDAT = 0x00000181;
+#endif
        return 0;
 }


+
+
+#define MVAL          (0)
+#define MVAL_USED  (0)              //0=each frame   1=rate by MVAL
+#define INVVDEN           (1)              //0=normal      1=inverted
+#define BSWP          (0)            //Byte swap control
+#define HWSWP             (1)            //Half word swap control
+
+
+//TFT 240320
+#define LCD_XSIZE_TFT_240320      (240)
+#define LCD_YSIZE_TFT_240320      (320)
+
+//TFT240320
+#define HOZVAL_TFT_240320(LCD_XSIZE_TFT_240320-1)
+#define LINEVAL_TFT_240320        (LCD_YSIZE_TFT_240320-1)
+
+//Timing parameter for NEC3.5"
+#define VBPD_240320          (3)
+#define VFPD_240320          (10)
+#define VSPW_240320              (1)
+
+#define HBPD_240320              (5)
+#define HFPD_240320          (2)
+#define HSPW_240320_NEC          (36)  //Adjust the horizontal displacement of the
screen :tekkamanninja@163.com
+#define HSPW_240320_TD           (23)  //64MB nand mini2440 is 36 ,128MB is 23
+                                  //+ :  -->   - : <--
+#define CLKVAL_TFT_240320 (3)
+//FCLK=101.25MHz,HCLK=50.625MHz,VCLK=6.33MHz
+
+
+void board_video_init(GraphicDevice *pGD)
+{
+        struct s3c24x0_lcd * const lcd    = s3c24x0_get_base_lcd();
+        struct s3c2410_nand * const nand = s3c2410_get_base_nand();
+   /* FIXME: select LCM type by env variable */
+
+        /* Configuration for GTA01 LCM on QT2410 */
```

```
+        lcd->LCDCON1 = 0x00000378; /* CLKVAL=4, BPPMODE=16bpp, TFT, ENVID=0 */
+        lcd->LCDCON2 =
(VBPD_240320<<24)|(LINEVAL_TFT_240320<<14)|(VFPD_240320<<6)|(VSPW_240320);
+        lcd->LCDCON3 = (HBPD_240320<<19)|(HOZVAL_TFT_240320<<8)|(HFPD_240320);
+
+        if ( (nand->NFCONF) & 0x08 )    {
+        lcd->LCDCON4 = (MVAL<<8)|(HSPW_240320_TD);
+        }
+        else      {
+         lcd->LCDCON4 = (MVAL<<8)|(HSPW_240320_NEC);
+        }
+
+        lcd->LCDCON5 = 0x00000f09;
+        lcd->LPCSEL  = 0x00000000;
+}
+
+
 int dram_init (void)
 {
         gd->bd->bi_dram[0].start = PHYS_SDRAM_1;
@@ -135,6 +228,7 @@ int dram_init (void)
         return 0;
 }

+#if 0
 #if defined(CONFIG_CMD_NAND)
 extern ulong nand_probe(ulong physadr);

@@ -180,6 +274,7 @@ void nand_init(void)
         printf ("%4lu MB\n", nand_probe((ulong)nand) >> 20);
 }
 #endif
+#endif

 #ifdef CONFIG_CMD_NET
 int board_eth_init(bd_t *bis)
@@ -188,6 +283,9 @@ int board_eth_init(bd_t *bis)
 #ifdef CONFIG_CS8900
         rc = cs8900_initialize(0, CONFIG_CS8900_BASE);
 #endif
+#ifdef CONFIG_DRIVER_DM9000
+        rc = dm9000_initialize(bis);
+#endif
         return rc;
 }
 #endif
```

　　到这里第二阶段的初始化的主线代码就修改完了，下面是各子系统的初始化和功能代码的修改。

6.4 第三阶段：完善目标板外设驱动

6.4.1 Nand Flash 相关代码的修改

在 U-boot 启动的第一阶段，初始化了 Nand Flash 控制器。但到第二阶段 start_armboot 函数还是会再次初始化 Nand Flash 控制器。因为第二阶段和第一阶段的代码基本是独立的，第一阶段的代码基本只起到代码重定位的作用，到了第二阶段才是真正 U-boot 的开始，以前的初始化过程还会重做一遍，比如始化 Nand Flash 控制器、CPU 频率等。

因为 S3C2440 和 S3C2410 之间的很大差别就是：S3C2410 的 Nand Flash 控制器只支持 512B+16B 的 Nand Flash，而 S3C2440 还支持 2KB+64B 的大容量 Nand Flash。所以在 Nand Flash 控制器上寄存器和控制流程上的差别很明显，底层驱动代码的修改也是必须的。具体的差别还是需要对比芯片数据手册的，下面是关于 Nand Flash 底层驱动代码的修改：

```
diff -aurNp u-boot-2009.11/drivers/mtd/nand/s3c2410_nand.c u-boot-
2009.11_tekkaman/drivers/mtd/nand/s3c2410_nand.c
--- u-boot-2009.11/drivers/mtd/nand/s3c2410_nand.c      2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/mtd/nand/s3c2410_nand.c        2010-03-28 17:16:12.000000000
+0800
@@ -24,6 +24,9 @@
 #include <s3c2410.h>
 #include <asm/io.h>

+#define        NF_BASE                 0x4e000000
+
+#if defined(CONFIG_S3C2410)
 #define S3C2410_NFCONF_EN       (1<<15)
 #define S3C2410_NFCONF_512BYTE    (1<<14)
 #define S3C2410_NFCONF_4STEP      (1<<13)
@@ -35,24 +38,41 @@

 #define S3C2410_ADDR_NALE 4
 #define S3C2410_ADDR_NCLE 8
+#endif
+
+#if defined(CONFIG_S3C2440)
+#define S3C2410_NFCONT_EN         (1<<0)
+#define S3C2410_NFCONT_INITECC    (1<<4)
+#define S3C2410_NFCONT_nFCE       (1<<1)
+#define S3C2410_NFCONT_MAINECCLOCK (1<<5)
+#define S3C2410_NFCONF_TACLS(x)   ((x)<<12)
+#define S3C2410_NFCONF_TWRPH0(x)  ((x)<<8)
+#define S3C2410_NFCONF_TWRPH1(x)  ((x)<<4)
+
+#define S3C2410_ADDR_NALE 0x08
+#define S3C2410_ADDR_NCLE 0x0c
+#endif
+
+ulong IO_ADDR_W = NF_BASE;

 static void s3c2410_hwcontrol(struct mtd_info *mtd, int cmd, unsigned int ctrl)
 {
-        struct nand_chip *chip = mtd->priv;
+//      struct nand_chip *chip = mtd->priv;
```

```
        struct s3c2410_nand *nand = s3c2410_get_base_nand();

        debugX(1, "hwcontrol(): 0x%02x 0x%02x\n", cmd, ctrl);

        if (ctrl & NAND_CTRL_CHANGE) {
-               ulong IO_ADDR_W = (ulong)nand;
+               IO_ADDR_W = (ulong)nand;

                if (!(ctrl & NAND_CLE))
                        IO_ADDR_W |= S3C2410_ADDR_NCLE;
                if (!(ctrl & NAND_ALE))
                        IO_ADDR_W |= S3C2410_ADDR_NALE;

-               chip->IO_ADDR_W = (void *)IO_ADDR_W;
+//              chip->IO_ADDR_W = (void *)IO_ADDR_W;

+#if defined(CONFIG_S3C2410)
                if (ctrl & NAND_NCE)
                        writel(readl(&nand->NFCONF) & ~S3C2410_NFCONF_nFCE,
                                &nand->NFCONF);
@@ -60,9 +80,19 @@ static void s3c2410_hwcontrol(struct mtd
                        writel(readl(&nand->NFCONF) | S3C2410_NFCONF_nFCE,
                                &nand->NFCONF);
        }
+#endif
+#if defined(CONFIG_S3C2440)
+               if (ctrl & NAND_NCE)
+                       writel(readl(&nand->NFCONT) & ~S3C2410_NFCONT_nFCE,
+                               &nand->NFCONT);
+               else
+                       writel(readl(&nand->NFCONT) | S3C2410_NFCONT_nFCE,
+                               &nand->NFCONT);
+       }
+#endif

        if (cmd != NAND_CMD_NONE)
-               writeb(cmd, chip->IO_ADDR_W);
+               writeb(cmd, (void *)IO_ADDR_W);
}

static int s3c2410_dev_ready(struct mtd_info *mtd)
@@ -77,7 +107,13 @@ void s3c2410_nand_enable_hwecc(struct mt
{
        struct s3c2410_nand *nand = s3c2410_get_base_nand();
        debugX(1, "s3c2410_nand_enable_hwecc(%p, %d)\n", mtd, mode);
+#if defined(CONFIG_S3C2410)
        writel(readl(&nand->NFCONF) | S3C2410_NFCONF_INITECC, &nand->NFCONF);
+#endif
+
+#if defined(CONFIG_S3C2440)
+       writel(readl(&nand->NFCONT) | S3C2410_NFCONT_INITECC, &nand->NFCONT);
+#endif
}

static int s3c2410_nand_calculate_ecc(struct mtd_info *mtd, const u_char *dat,
@@ -116,6 +152,7 @@ int board_nand_init(struct nand_chip *na

        writel(readl(&clk_power->CLKCON) | (1 << 4), &clk_power->CLKCON);

+#if defined(CONFIG_S3C2410)
```

```
        /* initialize hardware */
        twrph0 = 3;
        twrph1 = 0;
@@ -129,7 +166,24 @@ int board_nand_init(struct nand_chip *na

        /* initialize nand_chip data structure */
        nand->IO_ADDR_R = nand->IO_ADDR_W = (void *)&nand_reg->NFDATA;
+#endif
+#if defined(CONFIG_S3C2440)
+       twrph0 = 4;
+       twrph1 = 2;
+       tacls = 0;
+
+       cfg = 0;
+       cfg |= S3C2410_NFCONF_TACLS(tacls - 1);
+       cfg |= S3C2410_NFCONF_TWRPH0(twrph0 - 1);
+       cfg |= S3C2410_NFCONF_TWRPH1(twrph1 - 1);
+       writel(cfg, &nand_reg->NFCONF);
+
+       cfg = (0<<13)|(0<<12)|(0<<10)|(0<<9)|(0<<8)|(0<<6)|(0<<5)|(1<<4)|(0<<1)|(1<<0);
+       writel(cfg, &nand_reg->NFCONT);
+
+       /* initialize nand_chip data structure */
+       nand->IO_ADDR_R = nand->IO_ADDR_W = (void *)&nand_reg->NFDATA;
+#endif
        /* read_buf and write_buf are default */
        /* read_byte and write_byte are default */
```

6.4.2 添加 Yaffs(2)镜像烧写功能

　　修改好了 Nand Flash 驱动，接下来修改相关的 Yaffs(2)映像烧写代码，由于现在很多使用 Nand Flash 的系统，在 Linux 下都用 Yaffs(2)作为存储数据的文件系统，甚至是根文件系统。所以在 BootLoader 下能够烧写 Yaffs(2) 映像文件变得很必要。

　　对于 Yaffs(2)映像烧写的支持其实就是在烧写时，写入数据的同时，将镜像文件中的 oob 数据也写入到 Nand Flash 的 Spare 区。这和 Yaffs 文件系统原理以及 Nand Flash 的结构有关，请参考相关资料。

　　下面是需要修改的 4 个文件的补丁：

```
diff -aurNp u-boot-2009.11/common/cmd_nand.c u-boot-2009.11_tekkaman/common/cmd_nand.c
--- u-boot-2009.11/common/cmd_nand.c 2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/common/cmd_nand.c     2010-03-28 17:16:12.000000000 +0800
@@ -389,6 +389,25 @@ int do_nand(cmd_tbl_t * cmdtp, int flag,
                        else
                                ret = nand_write_skip_bad(nand, off, &size,
                                                          (u_char *)addr);
+#if defined(ENABLE_CMD_NAND_YAFFS)
+               }else if ( s != NULL &&
+                       (!strcmp(s, ".yaffs") || !strcmp(s, ".yaffs1"))){
+                       if(read)  {
+                               printf("nand read.yaffs[1] is not provide temporarily!");
+                       } else   {
+                               nand->rw_oob = 1;
+#if defined(ENABLE_CMD_NAND_YAFFS_SKIPFB)
```

```
+                               nand->skipfirstblk = 1;
+#else
+                               nand->skipfirstblk = 0;
+#endif
+                               ret = nand_write_skip_bad(nand,off,&size,(u_char *)addr);
+#if defined(ENABLE_CMD_NAND_YAFFS_SKIPFB)
+                               nand->skipfirstblk = 0;
+#endif
+                               nand->rw_oob = 0;
+                       }
+#endif
                } else if (!strcmp(s, ".oob")) {
                        /* out-of-band data */
                        mtd_oob_ops_t ops = {
@@ -496,6 +515,11 @@ U_BOOT_CMD(nand, CONFIG_SYS_MAXARGS, 1,
        "    to/from memory address 'addr', skipping bad blocks.\n"
        "nand erase [clean] [off size] - erase 'size' bytes from\n"
        "    offset 'off' (entire device if not specified)\n"
+#if defined(ENABLE_CMD_NAND_YAFFS)
+       "nand read[.yaffs[1]] is not provide temporarily!\n"
+       "nand write[.yaffs[1]]    addr off size - write the `size' byte yaffs image starting\n"
+       "    at offset `off' from memory address `addr' (.yaffs1 for 512+16 NAND)\n"
+#endif
        "nand bad - show bad blocks\n"
        "nand dump[.oob] off - dump page\n"
        "nand scrub - really clean NAND erasing bad blocks (UNSAFE)\n"
```

```
diff -aurNp u-boot-2009.11/drivers/mtd/nand/nand_base.c u-boot-
2009.11_tekkaman/drivers/mtd/nand/nand_base.c
--- u-boot-2009.11/drivers/mtd/nand/nand_base.c          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/mtd/nand/nand_base.c     2010-03-28 17:16:12.000000000 +0800
@@ -2008,7 +2008,30 @@ static int nand_write(struct mtd_info *m
 {
        struct nand_chip *chip = mtd->priv;
        int ret;
-
+#if defined(ENABLE_CMD_NAND_YAFFS)
+       /*Thanks for hugerat's code!*/
+
+       int oldopsmode = 0;
+       if(mtd->rw_oob==1)      {
+               size_t oobsize = mtd->oobsize;
+               size_t datasize = mtd->writesize;
+               int i = 0;
+               uint8_t oobtemp[oobsize];
+               int datapages = 0;
+               datapages = len/(datasize);
+               for(i=0;i<(datapages);i++)        {
+                       memcpy((void *)oobtemp,
+                               (void *)(buf+datasize*(i+1)),
+                               oobsize);
+                       memmove((void *)(buf+datasize*(i+1)),
+                               (void *)(buf+datasize*(i+1)+oobsize),
+                               (datapages-(i+1))*(datasize)+(datapages-1)*oobsize);
+                       memcpy((void *)(buf+(datapages)*(datasize+oobsize)-oobsize),
+                               (void *)(oobtemp),
+                               oobsize);
+               }
+       }
+#endif
```

```
        /* Do not allow reads past end of device */
        if ((to + len) > mtd->size)
                return -EINVAL;
@@ -2019,14 +2042,30 @@ static int nand_write(struct mtd_info *m

        chip->ops.len = len;
        chip->ops.datbuf = (uint8_t *)buf;
-       chip->ops.oobbuf = NULL;

+#if defined(ENABLE_CMD_NAND_YAFFS)
+       /*Thanks for hugerat's code!*/
+       if(mtd->rw_oob!=1)      {
+         chip->ops.oobbuf = NULL;
+       } else  {
+         chip->ops.oobbuf = (uint8_t *)(buf+len); //
+         chip->ops.ooblen = mtd->oobsize;
+         oldopsmode = chip->ops.mode;
+         chip->ops.mode = MTD_OOB_RAW; //
+       }
+#else
+       chip->ops.oobbuf = NULL;
+#endif
        ret = nand_do_write_ops(mtd, to, &chip->ops);

        *retlen = chip->ops.retlen;

        nand_release_device(mtd);

+#if defined(ENABLE_CMD_NAND_YAFFS)
+       /*Thanks for hugerat's code!*/
+       chip->ops.mode = oldopsmode; //
+#endif
        return ret;
 }
```

```
diff -aurNp u-boot-2009.11/drivers/mtd/nand/nand_util.c u-boot-
2009.11_tekkaman/drivers/mtd/nand/nand_util.c
--- u-boot-2009.11/drivers/mtd/nand/nand_util.c   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/mtd/nand/nand_util.c        2010-03-28 17:16:12.000000000 +0800
@@ -480,6 +480,26 @@ int nand_write_skip_bad(nand_info_t *nan
        size_t left_to_write = *length;
        size_t len_incl_bad;
        u_char *p_buffer = buffer;
+#if defined(ENABLE_CMD_NAND_YAFFS)
+       /*Thanks for hugerat's code*/
+
+       if(nand->rw_oob==1)    {
+               size_t oobsize = nand->oobsize;
+               size_t datasize = nand->writesize;
+               int datapages = 0;
+
+
+               if (((*length)%(nand->oobsize+nand->writesize)) != 0) {
+                   printf ("Attempt to write error length data!\n");
+                   return -EINVAL;
+               }
+
+               datapages = *length/(datasize+oobsize);
+               *length = datapages*datasize;
```

```
+                left_to_write = *length;
+//              nand->skipfirstblock=1;
+        }
+#endif

        /* Reject writes, which are not page aligned */
        if ((offset & (nand->writesize - 1)) != 0 ||
@@ -494,7 +514,9 @@ int nand_write_skip_bad(nand_info_t *nan
                printf ("Attempt to write outside the flash area\n");
                return -EINVAL;
        }
-
+
+#if !defined(ENABLE_CMD_NAND_YAFFS)
+/*by hugerat*/
        if (len_incl_bad == *length) {
                rval = nand_write (nand, offset, length, buffer);
                if (rval != 0)
@@ -503,7 +525,7 @@ int nand_write_skip_bad(nand_info_t *nan

                return rval;
        }
-
+#endif
        while (left_to_write > 0) {
                size_t block_offset = offset & (nand->erasesize - 1);
                size_t write_size;
@@ -516,12 +538,21 @@ int nand_write_skip_bad(nand_info_t *nan
                        offset += nand->erasesize - block_offset;
                        continue;
                }
-
+#if defined(ENABLE_CMD_NAND_YAFFS)
+                /*Thanks for hugerat's code*/
+                if(nand->skipfirstblk==1){
+                        nand->skipfirstblk=0;
+                        printf ("Skip the first good block %llx\n",
+                                offset & ~(nand->erasesize - 1));
+                        offset += nand->erasesize - block_offset;
+                        continue;
+                }
+#endif
                if (left_to_write < (nand->erasesize - block_offset))
                        write_size = left_to_write;
                else
                        write_size = nand->erasesize - block_offset;
-
+                printf("\rWriting at 0x%llx -- ",offset);        /*Thanks for hugerat's code*/
                rval = nand_write (nand, offset, &write_size, p_buffer);
                if (rval != 0) {
                        printf ("NAND write to offset %llx failed %d\n",
@@ -531,8 +562,18 @@ int nand_write_skip_bad(nand_info_t *nan
                }

                left_to_write -= write_size;
+                printf("%d%% is complete.",100-(left_to_write/(*length/100)));/*Thanks for hugerat's code*/
                offset      += write_size;
+#if defined(ENABLE_CMD_NAND_YAFFS)
+                /*Thanks for hugerat's code*/
+                if(nand->rw_oob==1)    {
```

```
+                         p_buffer += write_size+(write_size/nand->writesize*nand->oobsize);
+                  } else   {
+                         p_buffer += write_size;
+                  }
+#else
                  p_buffer      += write_size;
+#endif
         }

         return 0;
```

```
diff -aurNp u-boot-2009.11/include/linux/mtd/mtd.h u-boot-2009.11_tekkaman/include/linux/mtd/mtd.h
--- u-boot-2009.11/include/linux/mtd/mtd.h          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/linux/mtd/mtd.h    2010-03-28 17:16:12.000000000 +0800
@@ -129,6 +129,12 @@ struct mtd_info {
          */
         u_int32_t writesize;

+#if defined(ENABLE_CMD_NAND_YAFFS)
+                 /*Thanks for hugerat's code*/
+         u_char rw_oob;
+         u_char skipfirstblk;
+#endif
+
         u_int32_t oobsize;   /* Amount of OOB data per block (e.g. 16) */
         u_int32_t oobavail;  /* Available OOB bytes per block */
```

6.4.3 修改 Nor Flash 写入功能的代码

在虽然 S3C2440 和 S3C2410 对于 Nor Flash 的链接都是一样的，但是 SBC2410 使用的 AMD 的 Nor Flash 芯片，而 mini2440 使用的 SST 的 Nor Flash。这两款芯片在写入时所使用的块大小、时序和指令代码有差别，所以必须根据芯片的数据手册进行修改。主要的差别请看数据手册的对比：

   SST39VF1601：

TABLE 6: SOFTWARE COMMAND SEQUENCE

Command Sequence	1st Bus Write Cycle		2nd Bus Write Cycle		3rd Bus Write Cycle		4th Bus Write Cycle		5th Bus Write Cycle		6th Bus Write Cycle	
	Addr[1]	Data[2]	Addr[1]	Data[2]	Addr[1]	Data[2]	Addr[1]	Data[2]	Addr[1]	Data[2]	Addr[1]	Data[2]
Word-Program	5555H	AAH	2AAAH	55H	5555H	A0H	WA[3]	Data				
Sector-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	SA$_X$[4]	30H
Block-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	BA$_X$[4]	50H
Chip-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	5555H	10H
Erase-Suspend	XXXXH	B0H										
Erase-Resume	XXXXH	30H										
Query Sec ID[5]	5555H	AAH	2AAAH	55H	5555H	88H						
User Security ID Word-Program	5555H	AAH	2AAAH	55H	5555H	A5H	WA[6]	Data				
User Security ID Program Lock-Out	5555H	AAH	2AAAH	55H	5555H	85H	XXH[6]	0000H				
Software ID Entry[7,8]	5555H	AAH	2AAAH	55H	5555H	90H						
CFI Query Entry	5555H	AAH	2AAAH	55H	5555H	98H						
Software ID Exit[9,10] /CFI Exit/Sec ID Exit	5555H	AAH	2AAAH	55H	5555H	F0H						
Software ID Exit[9,10] /CFI Exit/Sec ID Exit	XXH	F0H										

 Am29LV160：

Table 9. Am29LV160D Command Definitions

Command Sequence (Note 1)			Cycles	First		Second		Third		Fourth		Fifth		Sixth	
				Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Read (Note 6)			1	RA	RD										
Reset (Note 7)			1	XXX	F0										
Autoselect (Note 8)	Manufacturer ID	Word	4	555	AA	2AA	55	555	90	X00	01				
		Byte		AAA		555		AAA							
	Device ID, Top Boot Block	Word	4	555	AA	2AA	55	555	90	X01	22C4				
		Byte		AAA		555		AAA		X02	C4				
	Device ID, Bottom Boot Block	Word	4	555	AA	2AA	55	555	90	X01	2249				
		Byte		AAA		555		AAA		X02	49				
	Sector Protect Verify (Note 9)	Word	4	555	AA	2AA	55	555	90	(SA) X02	XX00 XX01				
		Byte		AAA		555		AAA		(SA) X04	00 01				
CFI Query (Note 10)		Word	1	55	98										
		Byte		AA											
Program		Word	4	555	AA	2AA	55	555	A0	PA	PD				
		Byte		AAA		555		AAA							
Unlock Bypass		Word	3	555	AA	2AA	55	555	20						
		Byte		AAA		555		AAA							
Unlock Bypass Program (Note 11)			2	XXX	A0	PA	PD								
Unlock Bypass Reset (Note 12)			2	XXX	90	XXX	00								
Chip Erase		Word	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
		Byte		AAA		555		AAA		AAA		555		AAA	
Sector Erase		Word	6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA	30
		Byte		AAA		555		AAA		AAA		555			
Erase Suspend (Note 13)			1	XXX	B0										
Erase Resume (Note 14)			1	XXX	30										

        除了上面的不同以外，SST39VF1601 每个 SECTOR 的大小都是一样的，而 Am29LV160 的头几块比较小。需要做的修改集中在 board/tekkamanninja/mini2440/flash.c 这个文件上，修改如下：

```
diff -aurNp u-boot-2009.11/board/sbc2410x/flash.c u-boot-
2009.11_tekkaman/board/tekkamanninja/mini2440/flash.c
--- u-boot-2009.11/board/sbc2410x/flash.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/board/tekkamanninja/mini2440/flash.c     2010-03-28
17:16:12.000000000 +0800
@@ -35,12 +35,18 @@ flash_info_t flash_info[CONFIG_SYS_MAX_F
 #define CMD_UNLOCK1                    0x000000AA
 #define CMD_UNLOCK2                    0x00000055
 #define CMD_ERASE_SETUP                0x00000080
-#define CMD_ERASE_CONFIRM              0x00000030
+//#define CMD_ERASE_CONFIRM            0x00000030
+#define CMD_ERASE_CONFIRM              0x00000050
 #define CMD_PROGRAM                    0x000000A0
 #define CMD_UNLOCK_BYPASS              0x00000020

+#ifdef  CONFIG_SST_VF1601
+#define MEM_FLASH_ADDR1                (*(volatile u16 *)(CONFIG_SYS_FLASH_BASE + (0x000005555 <<
1)))
+#define MEM_FLASH_ADDR2                (*(volatile u16 *)(CONFIG_SYS_FLASH_BASE + (0x000002AAA <<
1)))
+#else
 #define MEM_FLASH_ADDR1                (*(volatile u16 *)(CONFIG_SYS_FLASH_BASE + (0x00000555 <<
1)))
 #define MEM_FLASH_ADDR2                (*(volatile u16 *)(CONFIG_SYS_FLASH_BASE + (0x000002AA <<
1)))
+#endif

 #define BIT_ERASE_DONE                 0x00000080
 #define BIT_RDY_MASK                   0x00000080
@@ -69,6 +75,9 @@ ulong flash_init (void)
 #elif defined(CONFIG_AMD_LV800)
                        (AMD_MANUFACT & FLASH_VENDMASK) |
                        (AMD_ID_LV800B & FLASH_TYPEMASK);
+#elif defined(CONFIG_SST_VF1601)
+                       (SST_MANUFACT & FLASH_VENDMASK) |
+                       (SST_ID_xF1601 & FLASH_TYPEMASK);
 #else
 #error "Unknown flash configured"
 #endif
@@ -80,6 +89,7 @@ ulong flash_init (void)
                else
                        panic ("configured too many flash banks!\n");
                for (j = 0; j < flash_info[i].sector_count; j++) {
+#ifndef CONFIG_SST_VF1601
                        if (j <= 3) {
                                /* 1st one is 16 KB */
                                if (j == 0) {
@@ -90,9 +100,7 @@ ulong flash_init (void)
                                /* 2nd and 3rd are both 8 KB */
                                if ((j == 1) || (j == 2)) {
                                        flash_info[i].start[j] =
-                                               flashbase + 0x4000 + (j -
-                                                                            1) *
-                                               0x2000;
+                                               flashbase + 0x4000 + (j -1) *0x2000;
                                }

                                /* 4th 32 KB */
@@ -104,6 +112,11 @@ ulong flash_init (void)
```

```
                                        flash_info[i].start[j] =
                                                flashbase + (j - 3) * MAIN_SECT_SIZE;
                                }
+#else
+                                       flash_info[i].start[j] =
+                                               flashbase + (j) * MAIN_SECT_SIZE;
+#endif
+
                }
                size += flash_info[i].size;
        }
@@ -130,6 +143,9 @@ void flash_print_info (flash_info_t * in
        case (AMD_MANUFACT & FLASH_VENDMASK):
                printf ("AMD: ");
                break;
+       case (SST_MANUFACT & FLASH_VENDMASK):
+               printf ("SST: ");
+               break;
        default:
                printf ("Unknown Vendor ");
                break;
@@ -142,6 +158,10 @@ void flash_print_info (flash_info_t * in
        case (AMD_ID_LV800B & FLASH_TYPEMASK):
                printf ("1x Amd29LV800BB (8Mbit)\n");
                break;
+       case (SST_ID_xF1601 & FLASH_TYPEMASK):
+               printf ("1x SST39VF1601 (2MB)\n");
+               break;
+
        default:
                printf ("Unknown Chip Type\n");
                goto Done;
@@ -169,10 +189,10 @@ void flash_print_info (flash_info_t * in

 int flash_erase (flash_info_t * info, int s_first, int s_last)
 {
-       ushort result;
+//      ushort result;
        int iflag, cflag, prot, sect;
        int rc = ERR_OK;
-       int chip;
+//      int chip;

        /* first look for protection bits */

@@ -182,12 +202,17 @@ int flash_erase (flash_info_t * info, in
        if ((s_first < 0) || (s_first > s_last)) {
                return ERR_INVAL;
        }
-
+#ifdef CONFIG_SST_VF1601
+       if ((info->flash_id & FLASH_VENDMASK) !=
+         (SST_MANUFACT & FLASH_VENDMASK)) {
+               return ERR_UNKNOWN_FLASH_VENDOR;
+       }
+#else
        if ((info->flash_id & FLASH_VENDMASK) !=
          (AMD_MANUFACT & FLASH_VENDMASK)) {
                return ERR_UNKNOWN_FLASH_VENDOR;
        }
```

```diff
-
+#endif
        prot = 0;
        for (sect = s_first; sect <= s_last; ++sect) {
                if (info->protect[sect]) {
@@ -226,6 +251,7 @@ int flash_erase (flash_info_t * info, in
                        MEM_FLASH_ADDR2 = CMD_UNLOCK2;
                        *addr = CMD_ERASE_CONFIRM;

+#if 0
                        /* wait until flash is ready */
                        chip = 0;

@@ -267,11 +293,31 @@ int flash_erase (flash_info_t * info, in
                        printf ("protected!\n");
                }
        }
+#endif
+               /* wait until flash is ready */
+               while(1){
+                   unsigned short i;
+                   i = *((volatile unsigned short *)addr) & 0x40;
+                   if(i != (*((volatile unsigned short *)addr) & 0x40))
+                       continue;
+                   if((*((volatile unsigned short *)addr)) & 0x80)
+                       break;
+               }
+               printf ("ok.\n");
+
+           } else {   /* it was protected */
+               printf ("protected!\n");
+           }
+       }
+
+
+
+

        if (ctrlc ())
                printf ("User Interrupt!\n");

-    outahere:
+//    outahere:
        /* allow flash to settle - wait 10 ms */
        udelay_masked (10000);

@@ -294,7 +340,7 @@ static int write_hword (flash_info_t * i
        ushort result;
        int rc = ERR_OK;
        int cflag, iflag;
-       int chip;
+//       int chip;

        /*
         * Check if Flash is (sufficiently) erased
@@ -317,13 +363,15 @@ static int write_hword (flash_info_t * i

        MEM_FLASH_ADDR1 = CMD_UNLOCK1;
        MEM_FLASH_ADDR2 = CMD_UNLOCK2;
-       MEM_FLASH_ADDR1 = CMD_UNLOCK_BYPASS;
```

```
-        *addr = CMD_PROGRAM;
+//      MEM_FLASH_ADDR1 = CMD_UNLOCK_BYPASS;
+//      *addr = CMD_PROGRAM;
+        MEM_FLASH_ADDR1 = CMD_PROGRAM;
+
         *addr = data;

         /* arm simple, non interrupt dependent timer */
         reset_timer_masked ();
-
+#if 0
         /* wait until flash is ready */
         chip = 0;
         do {
@@ -352,6 +400,18 @@ static int write_hword (flash_info_t * i

         if (chip == ERR || *addr != data)
                 rc = ERR_PROG_ERROR;
+#endif
+
+        /* wait until flash is ready */
+    while(1){
+        unsigned short i = *(volatile unsigned short *)addr & 0x40;
+        if(i != (*(volatile unsigned short *)addr & 0x40))   //D6 == D6
+            continue;
+        if((*(volatile unsigned short *)addr & 0x80) == (data & 0x80)){
+            rc = ERR_OK;
+            break;    //D7 == D7
+        }
+    }

         if (iflag)
                 enable_interrupts ();
```

6.4.4 修改网络相关代码

　　以前的 U-boot 对于网络延时部分有问题，需要修改许多地方。但是现在的 U-boot 网络部分已经基本不需要怎么修改了，只有在 DM9000 的驱动和 NFS 的 TIMEOUT 参数上需要稍微修改一下：

　　对于 DM9000 的驱动，只是屏蔽了一点代码：

```
diff -aurNp u-boot-2009.11/drivers/net/dm9000x.c u-boot-2009.11_tekkaman/drivers/net/dm9000x.c
--- u-boot-2009.11/drivers/net/dm9000x.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/net/dm9000x.c    2010-03-28 17:16:12.000000000 +0800
@@ -364,12 +364,13 @@ static int dm9000_init(struct eth_device
        while (!(phy_read(1) & 0x20)) {   /* autonegation complete bit */
                udelay(1000);
                i++;
-               if (i == 10000) {
-                       printf("could not establish link\n");
-                       return 0;
+               if (i == 1000) {
+//                     printf("could not establish link\n");
+//                     return 0;
+                       break;
                }
```

```
        }

        /* see what we've got */
        lnk = phy_read(17) >> 12;
        printf("operating at ");
```

对于 NFS，增加了延时，否则会出现"*** ERROR: Cannot mount"的错误。

```
diff -aurNp u-boot-2009.11/net/nfs.c u-boot-2009.11_tekkaman/net/nfs.c
--- u-boot-2009.11/net/nfs.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/net/nfs.c 2010-03-28 17:16:12.000000000 +0800
@@ -33,7 +33,7 @@

 #define HASHES_PER_LINE 65          /* Number of "loading" hashes per line    */
 #define NFS_RETRY_COUNT 30
-#define NFS_TIMEOUT 2000UL
+#define NFS_TIMEOUT (10*2000UL)

 static int fs_mounted = 0;
 static unsigned long rpc_id = 0;
```

6.4.5 添加串口 Xmodem 传输协议（可不修改）

对于使用串口传输数据到内存的操作，有可能会用到Xmodem协议。但是原本的kermit协议传输就挺好用的，速度也比较快，所以可添加此功能。修改的方法是参考 www.100ask.net 的方法。

```
diff -aurNp u-boot-2009.11/common/cmd_load.c u-boot-2009.11_tekkaman/common/cmd_load.c
--- u-boot-2009.11/common/cmd_load.c  2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/common/cmd_load.c       2010-03-28 17:16:12.000000000 +0800
@@ -34,6 +34,9 @@
 DECLARE_GLOBAL_DATA_PTR;

 #if defined(CONFIG_CMD_LOADB)
+#if defined(ENABLE_CMD_LOADB_X)
+static ulong load_serial_xmodem (ulong offset);
+#endif
 static ulong load_serial_ymodem (ulong offset);
 #endif

@@ -475,7 +478,19 @@ int do_load_serial_bin (cmd_tbl_t *cmdtp
            }
        }

+#if defined(ENABLE_CMD_LOADB_X)
+       if (strcmp(argv[0],"loadx")==0) {
+               printf ("## Ready for binary (xmodem) download "
+                       "to 0x%08lX at %d bps...\n",
+                       offset,
+                       load_baudrate);
+
+               addr = load_serial_xmodem (offset);
+
+       } else if (strcmp(argv[0],"loady")==0) {
+#else
        if (strcmp(argv[0],"loady")==0) {
```

```
+#endif
                  printf ("## Ready for binary (ymodem) download "
                            "to 0x%08lX at %d bps...\n",
                            offset,
@@ -963,6 +978,66 @@ static int getcxmodem(void) {
                  return (getc());
          return -1;
  }
+
+#if defined(ENABLE_CMD_LOADB_X)
+static ulong load_serial_xmodem (ulong offset)
+{
+        int size;
+        char buf[32];
+        int err;
+        int res;
+        connection_info_t info;
+        char xmodemBuf[1024];
+        ulong store_addr = ~0;
+        ulong addr = 0;
+
+        size = 0;
+        info.mode = xyzModem_xmodem;
+        res = xyzModem_stream_open (&info, &err);
+        if (!res) {
+
+                while ((res =
+                        xyzModem_stream_read (xmodemBuf, 1024, &err)) > 0) {
+                        store_addr = addr + offset;
+                        size += res;
+                        addr += res;
+#ifndef CFG_NO_FLASH
+                        if (addr2info (store_addr)) {
+                                int rc;
+
+                                rc = flash_write ((char *) xmodemBuf,
+                                store_addr, res);
+                                if (rc != 0) {
+                                flash_perror (rc);
+                                return (~0);
+                                }
+                        } else
+#endif
+                        {
+                                memcpy ((char *) (store_addr), xmodemBuf,
+                                        res);
+                        }
+
+                }
+        } else {
+                printf ("%s\n", xyzModem_error (err));
+        }
+
+        xyzModem_stream_close (&err);
+        xyzModem_stream_terminate (false, &getcxmodem);
+
+
+        flush_cache (offset, size);
+
+        printf ("## Total Size      = 0x%08x = %d Bytes\n", size, size);
```

```
+        sprintf (buf, "%X", size);
+        setenv ("filesize", buf);
+
+        return offset;
+}
+#endif
+
+
 static ulong load_serial_ymodem (ulong offset)
 {
        int size;
@@ -1078,6 +1153,16 @@ U_BOOT_CMD(
        " with offset 'off' and baudrate 'baud'"
 );


+#if defined(ENABLE_CMD_LOADB_X)
+U_BOOT_CMD(
+        loadx, 3, 0,    do_load_serial_bin,
+        "load binary file over serial line (xmodem mode)",
+        "[ off ] [ baud ]\n"
+        "    - load binary file over serial line"
+        " with offset 'off' and baudrate 'baud'"
+);
+#endif
+
 U_BOOT_CMD(
        loady, 3, 0,        do_load_serial_bin,
        "load binary file over serial line (ymodem mode)",
```

6.4.6 添加 <mark>LCD</mark> <mark>显示</mark>功能

对于这个 LCD 的支持是参考 Openmoko 的代码移植的。Openmoko 的 GTA2 使用的是
S3C2442 的 CPU，在 LCD 控制器上是一样的。而 GTA2 在 U-boot 的可以在 LCD 上显示字
符，而且对于软件分层的 U-boot 来说，<mark>只要将底层驱动移植过来</mark>并<mark>调整好初始化参数</mark>就可以
在 LCD 上<mark>显示</mark> console。

由于友善之臂使用了两种屏，他们<mark>在一个参数上有很小的差别</mark>，而屏有不可以通过程序
去识别，所以只能简单的通过 Nand Flash 配置来区分导入的参数（64MB Nand 的板子使用
的都是 NEC 的，其他的都是<mark>统宝</mark>的）。

这个功能的移植<mark>修改了</mark> <mark>5</mark> <mark>个文件</mark>（包括 drivers/video/Makefile，以及前面已经修改过的
board/tekkamanninja/mini2440/mini2440.c 文件），<mark>在/drivers/video/下添加一个驱动文件</mark>
<mark>s3c2410_fb.c</mark>。

```
diff -aurNp u-boot-2009.11/drivers/video/cfb_console.c u-boot-2009.11_tekkaman/drivers/video/cfb_console.c
--- u-boot-2009.11/drivers/video/cfb_console.c    2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/video/cfb_console.c        2010-03-28 17:16:12.000000000 +0800
@@ -281,8 +281,11 @@ void   console_cursor (int state);
 #define VIDEO_LOGO_LUT_OFFSET  LINUX_LOGO_LUT_OFFSET
 #define VIDEO_LOGO_COLORS        LINUX_LOGO_COLORS
 #endif  /* CONFIG_VIDEO_BMP_LOGO */
-#define VIDEO_INFO_X                (VIDEO_LOGO_WIDTH)
-#define VIDEO_INFO_Y                (VIDEO_FONT_HEIGHT/2)
+#define VIDEO_INFO_X                (0)
+#define VIDEO_INFO_Y                (VIDEO_LOGO_HEIGHT)
```

```
+//#define VIDEO_INFO_X              (VIDEO_LOGO_WIDTH)
+//#define VIDEO_INFO_Y              (VIDEO_FONT_HEIGHT/2)
+
 #else   /* CONFIG_VIDEO_LOGO */
 #define VIDEO_LOGO_WIDTH 0
 #define VIDEO_LOGO_HEIGHT          0
```

```
diff -aurNp u-boot-2009.11/drivers/video/Makefile u-boot-2009.11_tekkaman/drivers/video/Makefile
--- u-boot-2009.11/drivers/video/Makefile2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/video/Makefile     2010-03-28 17:16:12.000000000 +0800
@@ -38,6 +38,7 @@ COBJS-$(CONFIG_VIDEO_SM501) += sm501.o
 COBJS-$(CONFIG_VIDEO_SMI_LYNXEM) += smiLynxEM.o
 COBJS-$(CONFIG_VIDEO_VCXK) += bus_vcxk.o
 COBJS-y += videomodes.o
+COBJS-y += s3c2410_fb.o

 COBJS        := $(COBJS-y)
 SRCS  := $(COBJS:.o=.c)
```

```
diff -aurNp u-boot-2009.11/drivers/video/videomodes.c u-boot-2009.11_tekkaman/drivers/video/videomodes.c
--- u-boot-2009.11/drivers/video/videomodes.c   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/video/videomodes.c         2010-03-28 17:16:12.000000000 +0800
@@ -95,6 +95,7 @@ const struct ctfb_vesa_modes vesa_modes[
        {0x319, RES_MODE_1280x1024, 15},
        {0x31A, RES_MODE_1280x1024, 16},
        {0x31B, RES_MODE_1280x1024, 24},
+       {0x211, RES_MODE_240x320, 16},
 };
 const struct ctfb_res_modes res_mode_init[RES_MODES_COUNT] = {
        /* x       y pixclk  le     ri up   lo  hs vs s  vmode */
@@ -104,6 +105,7 @@ const struct ctfb_res_modes res_mode_ini
        {960, 720, 13100, 160, 40, 32, 8, 80, 4, 0, FB_VMODE_NONINTERLACED},
        {1152, 864, 12004, 200, 64, 32, 16, 80, 4, 0, FB_VMODE_NONINTERLACED},
        {1280, 1024, 9090, 200, 48, 26, 1, 184, 3, 0, FB_VMODE_NONINTERLACED},
+       {240,  320,  158025, 26,   6,    1,   11,  37,   2, 0, FB_VMODE_NONINTERLACED},
 };

 /**********************************************************************/
```

```
diff -aurNp u-boot-2009.11/drivers/video/videomodes.h u-boot-2009.11_tekkaman/drivers/video/videomodes.h
--- u-boot-2009.11/drivers/video/videomodes.h   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/video/videomodes.h         2010-03-28 17:16:12.000000000 +0800
@@ -22,8 +22,8 @@
  */


-#ifndef CONFIG_SYS_DEFAULT_VIDEO_MODE
-#define CONFIG_SYS_DEFAULT_VIDEO_MODE        0x301
+#ifndef CFG_SYS_DEFAULT_VIDEO_MODE
+#define CFG_SYS_DEFAULT_VIDEO_MODE 0x211
 #endif

 /* Some mode definitions */
@@ -78,9 +78,11 @@ struct ctfb_vesa_modes {
 #define RES_MODE_960_720  3
 #define RES_MODE_1152x8644
 #define RES_MODE_1280x1024         5
-#define RES_MODES_COUNT            6
+#define RES_MODE_240x320 6

-#define VESA_MODES_COUNT 19
```

+#define RES_MODES_COUNT   7
+
+#define VESA_MODES_COUNT 20

```
extern const struct ctfb_vesa_modes vesa_modes[];
extern const struct ctfb_res_modes res_mode_init[];
```

/drivers/video/s3c2410_fb.c：

```c
/*
 * (C) Copyright 2006 by OpenMoko, Inc.
 * Author: Harald Welte <laforge@openmoko.org>
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <common.h>

#if defined(CONFIG_VIDEO_S3C2410)

#include <video_fb.h>
#include "videomodes.h"
#include <s3c2410.h>
/*
 * Export Graphic Device
 */
GraphicDevice smi;

#define VIDEO_MEM_SIZE  0x200000       /* 240x320x16bit = 0x25800 bytes */

extern void board_video_init(GraphicDevice *pGD);

/*****************************************************************************
 *
 * Init video chip with common Linux graphic modes (lilo)
 */
void *video_hw_init (void)
{
    struct s3c24x0_lcd * const lcd = s3c24x0_get_base_lcd();
    GraphicDevice *pGD = (GraphicDevice *)&smi;
    int videomode;
    unsigned long t1, hsynch, vsynch;
    char *penv;
    int tmp, i, bits_per_pixel;
    struct ctfb_res_modes *res_mode;
    struct ctfb_res_modes var_mode;
```

```c
//    unsigned char videoout;

    /* Search for video chip */
    printf("Video: ");

   tmp = 0;

        videomode = CFG_SYS_DEFAULT_VIDEO_MODE;
        /* get video mode via environment */
        if ((penv = getenv ("videomode")) != NULL) {
            /* deceide if it is a string */
            if (penv[0] <= '9') {
                videomode = (int) simple_strtoul (penv, NULL, 16);
                tmp = 1;
            }
        } else {
            tmp = 1;
        }
        if (tmp) {
            /* parameter are vesa modes */
            /* search params */
            for (i = 0; i < VESA_MODES_COUNT; i++) {
                if (vesa_modes[i].vesanr == videomode)
                    break;
            }
            if (i == VESA_MODES_COUNT) {
                printf ("no VESA Mode found, switching to mode 0x%x ",
CFG_SYS_DEFAULT_VIDEO_MODE);
                i = 0;
            }
            res_mode =
                (struct ctfb_res_modes *) &res_mode_init[vesa_modes[i].
                                        resindex];
            bits_per_pixel = vesa_modes[i].bits_per_pixel;
        } else {

            res_mode = (struct ctfb_res_modes *) &var_mode;
            bits_per_pixel = video_get_params (res_mode, penv);
        }

        /* calculate hsynch and vsynch freq (info only) */
        t1 = (res_mode->left_margin + res_mode->xres +
            res_mode->right_margin + res_mode->hsync_len) / 8;
        t1 *= 8;
        t1 *= res_mode->pixclock;
        t1 /= 1000;
        hsynch = 1000000000L / t1;
        t1 *=
            (res_mode->upper_margin + res_mode->yres +
            res_mode->lower_margin + res_mode->vsync_len);
        t1 /= 1000;
        vsynch = 1000000000L / t1;

        /* fill in Graphic device struct */
        sprintf (pGD->modeIdent, "%dx%dx%d %ldkHz %ldHz", res_mode->xres,
            res_mode->yres, bits_per_pixel, (hsynch / 1000),
            (vsynch / 1000));
        printf ("%s\n", pGD->modeIdent);
        pGD->winSizeX = res_mode->xres;
        pGD->winSizeY = res_mode->yres;
```

```c
        pGD->plnSizeX = res_mode->xres;
        pGD->plnSizeY = res_mode->yres;

        switch (bits_per_pixel) {
        case 8:
            pGD->gdfBytesPP = 1;
            pGD->gdfIndex = GDF__8BIT_INDEX;
            break;
        case 15:
            pGD->gdfBytesPP = 2;
            pGD->gdfIndex = GDF_15BIT_555RGB;
            break;
        case 16:
            pGD->gdfBytesPP = 2;
            pGD->gdfIndex = GDF_16BIT_565RGB;
            break;
        case 24:
            pGD->gdfBytesPP = 3;
            pGD->gdfIndex = GDF_24BIT_888RGB;
            break;
        }

        /* statically configure settings */
        pGD->winSizeX = pGD->plnSizeX = 240;
        pGD->winSizeY = pGD->plnSizeY = 320;
        pGD->gdfBytesPP = 2;
        pGD->gdfIndex = GDF_16BIT_565RGB;

        pGD->frameAdrs = LCD_VIDEO_ADDR;
        pGD->memSize = VIDEO_MEM_SIZE;

        board_video_init(pGD);

        lcd->LCDSADDR1 = pGD->frameAdrs >> 1;

        /* This marks the end of the frame buffer. */
        lcd->LCDSADDR2 = (lcd->LCDSADDR1&0x1fffff) + (pGD->winSizeX+0) * pGD->winSizeY;
        lcd->LCDSADDR3 = (pGD->winSizeX & 0x7ff);

        /* Clear video memory */
        memset((void *)pGD->frameAdrs, 0, pGD->memSize);

        /* Enable  Display  */
        lcd->LCDCON1 |= 0x01;   /* ENVID = 1 */

        return ((void*)&smi);
}

void
video_set_lut (unsigned int index,      /* color number */
        unsigned char r, /* red */
        unsigned char g, /* green */
        unsigned char b  /* blue */
                )
{
}

#endif /* CONFIG_VIDEO_S3C2410 */
```

6.4.7 添加 SD 卡（MMC）读取功能

SD卡的支持参考了 buserror 的Git代码仓库中的源码，他也是为mini2440 移植的。它使用的代码也是Openmoko的GTA2 源码。因为GTA2 可以在U-boot中使用SD卡更新系统。将其SD卡底层驱动代码搬过来，经过简单的修改就可以使用了。

这个功能需要修改 5 个文件，添加 3 个驱动代码文件。

```
diff -aurNp u-boot-2009.11/common/cmd_mem.c u-boot-2009.11_tekkaman/common/cmd_mem.c
--- u-boot-2009.11/common/cmd_mem.c 2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/common/cmd_mem.c    2010-03-28 17:16:12.000000000 +0800
@@ -32,6 +32,9 @@
 #ifdef CONFIG_HAS_DATAFLASH
 #include <dataflash.h>
 #endif
+#if defined(CONFIG_CMD_MMC)
+#include <mmc.h>
+#endif
 #include <watchdog.h>

 #include <u-boot/md5.h>
@@ -404,6 +407,46 @@ int do_mem_cp ( cmd_tbl_t *cmdtp, int fl
        }
 #endif

+#if defined(CONFIG_CMD_MMC)
+       if (mmc2info(dest)) {
+               int rc;
+
+               puts ("Copy to MMC... ");
+               switch (rc = mmc_write ((uchar *)addr, dest, count*size)) {
+               case 0:
+                       putc ('\n');
+                       return 1;
+               case -1:
+                       puts ("failed\n");
+                       return 1;
+               default:
+                       printf ("%s[%d] FIXME: rc=%d\n",__FILE__,__LINE__,rc);
+                       return 1;
+               }
+               puts ("done\n");
+               return 0;
+       }
+
+       if (mmc2info(addr)) {
+               int rc;
+
+               puts ("Copy from MMC... ");
+               switch (rc = mmc_read (addr, (uchar *)dest, count*size)) {
+               case 0:
+                       putc ('\n');
+                       return 1;
+               case -1:
+                       puts ("failed\n");
+                       return 1;
+               default:
```

```diff
+                       printf ("%s[%d] FIXME: rc=%d\n",__FILE__,__LINE__,rc);
+                       return 1;
+               }
+               puts ("done\n");
+               return 0;
+       }
+#endif
+
 #ifdef CONFIG_HAS_DATAFLASH
        /* Check if we are copying from RAM or Flash to DataFlash */
        if (addr_dataflash(dest) && !addr_dataflash(addr)){
```

```diff
diff -aurNp u-boot-2009.11/common/cmd_mmc.c u-boot-2009.11_tekkaman/common/cmd_mmc.c
--- u-boot-2009.11/common/cmd_mmc.c 2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/common/cmd_mmc.c     2010-03-28 17:16:12.000000000 +0800
@@ -50,7 +50,7 @@ int do_mmc (cmd_tbl_t *cmdtp, int flag,
                        return 1;
                }

-               if (mmc_legacy_init(dev) != 0) {
+               if (mmc_init(dev) != 0) {
                        puts("No MMC card found\n");
                        return 1;
                }
```

```diff
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/Makefile u-boot-
2009.11_tekkaman/cpu/arm920t/s3c24x0/Makefile
--- u-boot-2009.11/cpu/arm920t/s3c24x0/Makefile          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/Makefile    2010-03-28 17:16:12.000000000 +0800
@@ -30,7 +30,7 @@ COBJS-y += speed.o
 COBJS-y         += timer.o
 COBJS-y         += usb.o
 COBJS-y         += usb_ohci.o
-
+COBJS-y         += mmc.o

 SRCS  := $(SOBJS:.o=.S) $(COBJS-y:.o=.c)
 OBJS  := $(addprefix $(obj),$(SOBJS) $(COBJS-y))
```

```diff
diff -aurNp u-boot-2009.11/include/mmc.h u-boot-2009.11_tekkaman/include/mmc.h
--- u-boot-2009.11/include/mmc.h          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/mmc.h    2010-03-28 17:16:12.000000000 +0800
@@ -169,7 +169,7 @@
 #define MMC_RSP_R6     (MMC_RSP_PRESENT|MMC_RSP_CRC|MMC_RSP_OPCODE)
 #define MMC_RSP_R7     (MMC_RSP_PRESENT|MMC_RSP_CRC|MMC_RSP_OPCODE)

-
+#if 0
 struct mmc_cid {
        unsigned long psn;
        unsigned short oid;
@@ -218,7 +218,7 @@ struct mmc_csd
        u8      crc:7;
        u8      one:1;
 };
-
+#endif
 struct mmc_cmd {
        ushort cmdidx;
        uint resp_type;
@@ -268,8 +268,10 @@ struct mmc {
```

```
 int mmc_register(struct mmc *mmc);
 int mmc_initialize(bd_t *bis);
-int mmc_init(struct mmc *mmc);
-int mmc_read(struct mmc *mmc, u64 src, uchar *dst, int size);
+//int mmc_init(struct mmc *mmc);
+//int mmc_read(struct mmc *mmc, u64 src, uchar *dst, int size);
+int mmc_init(int verbose);
+int mmc_read(ulong src, uchar *dst, int size);
 struct mmc *find_mmc_device(int dev_num);
 void print_mmc_devices(char separator);
```

```
diff -aurNp u-boot-2009.11/include/part.h u-boot-2009.11_tekkaman/include/part.h
--- u-boot-2009.11/include/part.h 2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/part.h    2010-03-28 17:16:12.000000000 +0800
@@ -62,6 +62,7 @@ typedef struct block_dev_desc {
 #define IF_TYPE_MMC               6
 #define IF_TYPE_SD            7
 #define IF_TYPE_SATA              8
+#define IF_TYPE_SDHC              9

 /* Part types */
 #define PART_TYPE_UNKNOWN         0x00
```

添加的 3 个驱动代码文件：

/cpu/arm920t/s3c24x0/mmc.c：

```
/*
 * u-boot S3C2410 MMC/SD card driver
 * (C) Copyright 2006 by OpenMoko, Inc.
 * Author: Harald Welte <laforge@openmoko.org>
 *
 * based on u-boot pxa MMC driver and linux/drivers/mmc/s3c2410mci.c
 * (C) 2005-2005 Thomas Kleffel
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <config.h>
#include <common.h>
#include <mmc.h>
#include <asm/arch/mmc.h>
#include <asm/errno.h>
#include <asm/io.h>
#include <s3c2410.h>
```

```c
#include <part.h>
#include <fat.h>

#if defined(CONFIG_MMC) && defined(CONFIG_MMC_S3C)

#ifdef DEBUG
#define pr_debug(fmt, args...) printf(fmt, ##args)
#else
#define pr_debug(...) do { } while(0)
#endif

#define CONFIG_MMC_WIDE

static struct s3c2410_sdi *sdi;

static block_dev_desc_t mmc_dev;

block_dev_desc_t * mmc_get_dev(int dev)
{
        return ((block_dev_desc_t *)&mmc_dev);
}

/*
 * FIXME needs to read cid and csd info to determine block size
 * and other parameters
 */
static uchar mmc_buf[MMC_BLOCK_SIZE];
static mmc_csd_t mmc_csd;
static int mmc_ready = 0;
static int wide = 0;


#define CMD_F_RESP  0x01
#define CMD_F_RESP_LONG   0x02

#define CMD_F_RESP_R7 CMD_F_RESP

static u_int32_t *mmc_cmd(ushort cmd, ulong arg, ushort flags)
{
        static u_int32_t resp[5];

        u_int32_t ccon, csta;
        u_int32_t csta_rdy_bit = S3C2410_SDICMDSTAT_CMDSENT;

        memset(resp, 0, sizeof(resp));

        debug("mmc_cmd CMD%d arg=0x%08x flags=%x\n", cmd, arg, flags);

        sdi->SDICSTA = 0xffffffff;
        sdi->SDIDSTA = 0xffffffff;
        sdi->SDIFSTA = 0xffffffff;

        sdi->SDICARG = arg;

        ccon = cmd & S3C2410_SDICMDCON_INDEX;
        ccon |= S3C2410_SDICMDCON_SENDERHOST|S3C2410_SDICMDCON_CMDSTART;

        if (flags & CMD_F_RESP) {
                ccon |= S3C2410_SDICMDCON_WAITRSP;
                csta_rdy_bit = S3C2410_SDICMDSTAT_RSPFIN; /* 1 << 9 */
```

```c
        }

        if (flags & CMD_F_RESP_LONG)
                ccon |= S3C2410_SDICMDCON_LONGRSP;

        sdi->SDICCON = ccon;

        while (1) {
                csta = sdi->SDICSTA;
                if (csta & csta_rdy_bit)
                        break;
                if (csta & S3C2410_SDICMDSTAT_CMDTIMEOUT) {
                        printf("===============> MMC CMD Timeout\n");
                        sdi->SDICSTA |= S3C2410_SDICMDSTAT_CMDTIMEOUT;
                        break;
                }
        }

        debug("final MMC CMD status 0x%x\n", csta);

        sdi->SDICSTA |= csta_rdy_bit;

        if (flags & CMD_F_RESP) {
                resp[0] = sdi->SDIRSP0;
                resp[1] = sdi->SDIRSP1;
                resp[2] = sdi->SDIRSP2;
                resp[3] = sdi->SDIRSP3;
        }

        return resp;
}

#define FIFO_FILL(host) ((host->SDIFSTA & S3C2410_SDIFSTA_COUNTMASK) >> 2)

static int mmc_block_read(uchar *dst, ulong src, ulong len)
{
        u_int32_t dcon, fifo;
        u_int32_t *dst_u32 = (u_int32_t *)dst;
        u_int32_t *resp;

        if (len == 0)
                return 0;

        debug("mmc_block_rd dst %lx src %lx len %d\n", (ulong)dst, src, len);

        /* set block len */
        resp = mmc_cmd(MMC_CMD_SET_BLOCKLEN, len, CMD_F_RESP);
        sdi->SDIBSIZE = len;

        //sdi->SDIPRE = 0xff;

        /* setup data */
        dcon = (len >> 9) & S3C2410_SDIDCON_BLKNUM;
        dcon |= S3C2410_SDIDCON_BLOCKMODE;
        dcon |= S3C2410_SDIDCON_RXAFTERCMD|S3C2410_SDIDCON_XFER_RXSTART;
        if (wide)
                dcon |= S3C2410_SDIDCON_WIDEBUS;
#if defined(CONFIG_S3C2440) || defined(CONFIG_S3C2442)
        dcon |= S3C2440_SDIDCON_DS_WORD | S3C2440_SDIDCON_DATSTART;
#endif
```

```c
        sdi->SDIDCON = dcon;

        /* send read command */
        resp = mmc_cmd(MMC_CMD_READ_BLOCK, (mmc_dev.if_type == IF_TYPE_SDHC) ? (src >> 9) :
src, CMD_F_RESP);

        while (len > 0) {
                u_int32_t sdidsta = sdi->SDIDSTA;
                fifo = FIFO_FILL(sdi);
                if (sdidsta & (S3C2410_SDIDSTA_FIFOFAIL|
                                S3C2410_SDIDSTA_CRCFAIL|
                                S3C2410_SDIDSTA_RXCRCFAIL|
                                S3C2410_SDIDSTA_DATATIMEOUT)) {
                        printf("mmc_block_read: err SDIDSTA=0x%08x\n", sdidsta);
                        return -EIO;
                }

                while (fifo--) {
                        //debug("dst_u32 = 0x%08x\n", dst_u32);
                        *(dst_u32++) = sdi->SDIDAT;
                        if (len >= 4)
                                len -= 4;
                        else {
                                len = 0;
                                break;
                        }
                }
        }

        debug("waiting for SDIDSTA  (currently 0x%08x\n", sdi->SDIDSTA);
        while (!(sdi->SDIDSTA & (1 << 4))) {}
        debug("done waiting for SDIDSTA (currently 0x%08x\n", sdi->SDIDSTA);

        sdi->SDIDCON = 0;

        if (!(sdi->SDIDSTA & S3C2410_SDIDSTA_XFERFINISH))
                debug("mmc_block_read; transfer not finished!\n");

        return 0;
}

static int mmc_block_write(ulong dst, uchar *src, int len)
{
        printf("MMC block write not yet supported on S3C2410!\n");
        return -1;
}


int mmc_read(ulong src, uchar *dst, int size)
{
        ulong end, part_start, part_end, part_len, aligned_start, aligned_end;
        ulong mmc_block_size, mmc_block_address;

        if (size == 0)
                return 0;

        if (!mmc_ready) {
                printf("Please initialize the MMC first\n");
                return -1;
        }
```

```c
        mmc_block_size = MMC_BLOCK_SIZE;
        mmc_block_address = ~(mmc_block_size - 1);

        src -= CFG_MMC_BASE;
        end = src + size;
        part_start = ~mmc_block_address & src;
        part_end = ~mmc_block_address & end;
        aligned_start = mmc_block_address & src;
        aligned_end = mmc_block_address & end;

        /* all block aligned accesses */
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        if (part_start) {
                part_len = mmc_block_size - part_start;
                debug("ps src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_read(mmc_buf, aligned_start, mmc_block_size)) < 0)
                        return -1;

                memcpy(dst, mmc_buf+part_start, part_len);
                dst += part_len;
                src += part_len;
        }
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        for (; src < aligned_end; src += mmc_block_size, dst += mmc_block_size) {
                debug("al src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_read((uchar *)(dst), src, mmc_block_size)) < 0)
                        return -1;
        }
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        if (part_end && src < end) {
                debug("pe src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_read(mmc_buf, aligned_end, mmc_block_size)) < 0)
                        return -1;

                memcpy(dst, mmc_buf, part_end);
        }
        return 0;
}

int mmc_write(uchar *src, ulong dst, int size)
{
        ulong end, part_start, part_end, part_len, aligned_start, aligned_end;
        ulong mmc_block_size, mmc_block_address;

        if (size == 0)
                return 0;

        if (!mmc_ready) {
                printf("Please initialize the MMC first\n");
                return -1;
        }

        mmc_block_size = MMC_BLOCK_SIZE;
```

```c
        mmc_block_address = ~(mmc_block_size - 1);

        dst -= CFG_MMC_BASE;
        end = dst + size;
        part_start = ~mmc_block_address & dst;
        part_end = ~mmc_block_address & end;
        aligned_start = mmc_block_address & dst;
        aligned_end = mmc_block_address & end;

        /* all block aligned accesses */
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        if (part_start) {
                part_len = mmc_block_size - part_start;
                debug("ps src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                (ulong)src, dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_read(mmc_buf, aligned_start, mmc_block_size)) < 0)
                        return -1;

                memcpy(mmc_buf+part_start, src, part_len);
                if ((mmc_block_write(aligned_start, mmc_buf, mmc_block_size)) < 0)
                        return -1;

                dst += part_len;
                src += part_len;
        }
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        for (; dst < aligned_end; src += mmc_block_size, dst += mmc_block_size) {
                debug("al src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_write(dst, (uchar *)src, mmc_block_size)) < 0)
                        return -1;

        }
        debug("src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
        src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
        if (part_end && dst < end) {
                debug("pe src %lx dst %lx end %lx pstart %lx pend %lx astart %lx aend %lx\n",
                src, (ulong)dst, end, part_start, part_end, aligned_start, aligned_end);
                if ((mmc_block_read(mmc_buf, aligned_end, mmc_block_size)) < 0)
                        return -1;

                memcpy(mmc_buf, src, part_end);
                if ((mmc_block_write(aligned_end, mmc_buf, mmc_block_size)) < 0)
                        return -1;

        }
        return 0;
}

ulong mmc_bread(int dev_num, ulong blknr, ulong blkcnt, void *dst)
{
        int mmc_block_size = MMC_BLOCK_SIZE;
        ulong src = blknr * mmc_block_size + CFG_MMC_BASE;

        mmc_read(src, dst, blkcnt*mmc_block_size);
        return blkcnt;
}
```

```c
/* MMC_DEFAULT_RCA should probably be just 1, but this may break other code
   that expects it to be shifted. */
static u_int16_t rca = MMC_DEFAULT_RCA >> 16;

static u_int32_t mmc_size(const struct mmc_csd *csd)
{
        u_int32_t block_len, mult, blocknr;

        block_len = csd->read_bl_len << 12;
        mult = csd->c_size_mult1 << 8;
        blocknr = (csd->c_size+1) * mult;

        return blocknr * block_len;
}

struct sd_cid {
        char            pnm_0; /* product name */
        char            oid_1;  /* OEM/application ID */
        char            oid_0;
        uint8_t         mid;    /* manufacturer ID */
        char            pnm_4;
        char            pnm_3;
        char            pnm_2;
        char            pnm_1;
        uint8_t         psn_2; /* product serial number */
        uint8_t         psn_1;
        uint8_t         psn_0; /* MSB */
        uint8_t         prv;    /* product revision */
        uint8_t         crc;    /* CRC7 checksum, b0 is unused and set to 1 */
        uint8_t         mdt_1; /* manufacturing date, LSB, RRRRyyyy yyyymmmm */
        uint8_t         mdt_0; /* MSB */
        uint8_t         psn_3; /* LSB */
};

static void print_mmc_cid(mmc_cid_t *cid)
{
        printf("MMC found. Card desciption is:\n");
        printf("Manufacturer ID = %02x%02x%02x\n",
                cid->id[0], cid->id[1], cid->id[2]);
        printf("HW/FW Revision = %x %x\n",cid->hwrev, cid->fwrev);
        cid->hwrev = cid->fwrev = 0;    /* null terminate string */
        printf("Product Name = %s\n",cid->name);
        printf("Serial Number = %02x%02x%02x\n",
                cid->sn[0], cid->sn[1], cid->sn[2]);
        printf("Month = %d\n",cid->month);
        printf("Year = %d\n",1997 + cid->year);
}

static void print_sd_cid(const struct sd_cid *cid)
{
        printf("Manufacturer:       0x%02x, OEM \"%c%c\"\n",
           cid->mid, cid->oid_0, cid->oid_1);
        printf("Product name:       \"%c%c%c%c%c\", revision %d.%d\n",
           cid->pnm_0, cid->pnm_1, cid->pnm_2, cid->pnm_3, cid->pnm_4,
           cid->prv >> 4, cid->prv & 15);
        printf("Serial number:      %u\n",
           cid->psn_0 << 24 | cid->psn_1 << 16 | cid->psn_2 << 8 |
           cid->psn_3);
        printf("Manufacturing date: %d/%d\n",
           cid->mdt_1 & 15,
```

```c
                2000+((cid->mdt_0 & 15) << 4)+((cid->mdt_1 & 0xf0) >> 4));
        printf("CRC:            0x%02x, b0 = %d\n",
            cid->crc >> 1, cid->crc & 1);
}

int mmc_init(int verbose)
{
        int retries, rc = -ENODEV;
        int is_sd = 0;
        u_int32_t *resp;
        struct s3c24x0_clock_power * const clk_power = s3c24x0_get_base_clock_power();
        block_dev_desc_t *mmc_blkdev_p = &mmc_dev;

        sdi = s3c2410_get_base_sdi();

        debug("mmc_init(PCLK=%u)\n", get_PCLK());

        clk_power->CLKCON |= (1 << 9);

        sdi->SDIBSIZE = 512;
#if defined(CONFIG_S3C2410)
        /* S3C2410 has some bug that prevents reliable operation at higher speed */
        //sdi->SDIPRE = 0x3e;  /* SDCLK = PCLK/2 / (SDIPRE+1) = 396kHz */
        sdi->SDIPRE = 0x02;  /* 2410: SDCLK = PCLK/2 / (SDIPRE+1) = 11MHz */
        sdi->SDIDTIMER = 0xffff;
#elif defined(CONFIG_S3C2440) || defined(CONFIG_S3C2442)
        sdi->SDIPRE = 0x05;  /* 2410: SDCLK = PCLK / (SDIPRE+1) = 11MHz */
        sdi->SDIDTIMER = 0x7fffff;
#endif
        sdi->SDIIMSK = 0x0;
        sdi->SDICON = S3C2410_SDICON_FIFORESET|S3C2410_SDICON_CLOCKTYPE;
        udelay(125000); /* FIXME: 74 SDCLK cycles */

        mmc_csd.c_size = 0;

        /* reset */
        retries = 10;
        resp = mmc_cmd(MMC_CMD_RESET, 0, 0);

        mmc_dev.if_type = IF_TYPE_UNKNOWN;
        if(verbose)
                puts("mmc: Probing for SDHC ...\n");

        /* Send supported voltage range */
        /* SD cards 1.x do not answer to CMD8 */
    resp = mmc_cmd(MMC_CMD_IF_COND, ((1 << 8) | 0xAA), CMD_F_RESP_R7);
    if (!resp[0]) {
        /*
         * ARC: No answer let's try SD 1.x
         */
        if(verbose)
            puts("mmc: No answer to CMD8 trying SD\n");
        mmc_blkdev_p->if_type = IF_TYPE_SD;
    } else {
        /*
         * ARC: probably an SDHC card
         */
        mmc_blkdev_p->if_type = IF_TYPE_SDHC;
        if(verbose)
            puts("mmc: SD 2.0 or later card found\n");
```

```c
        /* Check if the card supports this voltage */
        if (resp[0] != ((1 << 8) | 0xAA)) {
                pr_debug("mmc: Invalid voltage range\n");
                return -ENODEV;
        }
    }

        /*
         * ARC: HC (30) bit set according to response to
         * CMD8 command
         */

        pr_debug("mmc: Sending ACMD41 %s HC set\n",
                        ((mmc_blkdev_p->if_type ==
                         IF_TYPE_SDHC) ? "with" : "without"));

        printf("trying to detect SD Card...\n");
        while (retries--) {
                udelay(100000);
                resp = mmc_cmd(55, 0x00000000, CMD_F_RESP);
                resp = mmc_cmd(41, (mmc_blkdev_p->if_type == IF_TYPE_SDHC)? (0x00300000 |
(1<<30)) : 0x00300000, CMD_F_RESP);

                if (resp[0] & (1 << 31)) {
                        is_sd = 1;
                        break;
                }
        }

        /*
        * ARC: check for HC bit, if its not set
        * sd card is SD
        */
        if (is_sd && (resp[0] & 0xc0000000) == 0x80000000) {
            mmc_dev.if_type = IF_TYPE_SD;
        }

        if (retries == 0 && !is_sd) {
                retries = 10;
                printf("failed to detect SD Card, trying MMC\n");
                mmc_blkdev_p->if_type = IF_TYPE_MMC;
                resp = mmc_cmd(MMC_CMD_SEND_OP_COND, 0x00ffc000, CMD_F_RESP);
                while (retries-- && resp && !(resp[4] & 0x80)) {
                        debug("resp %x %x\n", resp[0], resp[1]);
                        udelay(50);
                        resp = mmc_cmd(1, 0x00ffff00, CMD_F_RESP);
                }
        }

        /* try to get card id */
        resp = mmc_cmd(MMC_CMD_ALL_SEND_CID, 0, CMD_F_RESP|CMD_F_RESP_LONG);
        if (resp) {
                if (!is_sd) {
                        /* TODO configure mmc driver depending on card
                           attributes */
                        mmc_cid_t *cid = (mmc_cid_t *)resp;

                        if (verbose)
                                print_mmc_cid(cid);
```

```c
                        sprintf((char *) mmc_dev.vendor,
                                "Man %02x%02x%02x Snr %02x%02x%02x",
                                cid->id[0], cid->id[1], cid->id[2],
                                cid->sn[0], cid->sn[1], cid->sn[2]);
                        sprintf((char *) mmc_dev.product,"%s",cid->name);
                        sprintf((char *) mmc_dev.revision,"%x %x",
                                cid->hwrev, cid->fwrev);
                }
                else {
                        struct sd_cid *cid = (struct sd_cid *) resp;

                        if (verbose)
                                print_sd_cid(cid);
                        sprintf((char *) mmc_dev.vendor, "Man %02x OEM %c%c \"%c%c%c%c%c\"", cid-
>mid, cid->oid_0, cid->oid_1, cid->pnm_0, cid->pnm_1, cid->pnm_2, cid->pnm_3, cid->pnm_4);
                        sprintf((char *) mmc_dev.product, "%d",
                            cid->psn_0 << 24 | cid->psn_1 << 16 |
                            cid->psn_2 << 8 | cid->psn_3);
                        sprintf((char *) mmc_dev.revision, "%d.%d",
                            cid->prv >> 4, cid->prv & 15);
                }

                /* fill in device description */
                if (mmc_dev.if_type == IF_TYPE_UNKNOWN)
                        mmc_dev.if_type = IF_TYPE_MMC;
                mmc_dev.part_type = PART_TYPE_DOS;
                mmc_dev.dev = 0;
                mmc_dev.lun = 0;
                mmc_dev.type = 0;
                /* FIXME fill in the correct size (is set to 32MByte) */
                mmc_dev.blksz = 512;
                mmc_dev.lba = 0x10000;
                mmc_dev.removable = 0;
                mmc_dev.block_read = mmc_bread;

                /* MMC exists, get CSD too */
                resp = mmc_cmd(MMC_CMD_SET_RCA, MMC_DEFAULT_RCA, CMD_F_RESP);
                if (is_sd)
                        rca = resp[0] >> 16;

                resp = mmc_cmd(MMC_CMD_SEND_CSD, rca<<16,
CMD_F_RESP|CMD_F_RESP_LONG);
                if (resp) {
                        mmc_csd_t *csd = (mmc_csd_t *)resp;
                        memcpy(&mmc_csd, csd, sizeof(csd));
                        rc = 0;
                        mmc_ready = 1;
                        /* FIXME add verbose printout for csd */
                        printf("READ_BL_LEN=%u, C_SIZE_MULT=%u, C_SIZE=%u\n",
                                csd->read_bl_len, csd->c_size_mult1, csd->c_size);
                        printf("size = %u\n", mmc_size(csd));
                }
        }

        resp = mmc_cmd(MMC_CMD_SELECT_CARD, rca<<16, CMD_F_RESP);

        if (verbose)
            printf("SD Card detected RCA: 0x%x type: %s\n",
                rca, ((mmc_dev.if_type == IF_TYPE_SDHC) ? "SDHC" : ((mmc_dev.if_type ==
IF_TYPE_SD) ? "SD" : "MMC")));
```

```
#ifdef CONFIG_MMC_WIDE
        if (is_sd) {
                resp = mmc_cmd(55, rca<<16, CMD_F_RESP);
                resp = mmc_cmd(6, 0x02, CMD_F_RESP);
                wide = 1;
        }
#endif

        fat_register_device(&mmc_dev,1); /* partitions start counting with 1 */

        return rc;
}

int
mmc_ident(block_dev_desc_t *dev)
{
        return 0;
}

int
mmc2info(ulong addr)
{
        /* FIXME hard codes to 32 MB device */
        if (addr >= CFG_MMC_BASE && addr < CFG_MMC_BASE + 0x02000000)
                return 1;

        return 0;
}

#endif   /* defined(CONFIG_MMC) && defined(CONFIG_MMC_S3C) */
```

include/asm-arm/arch-s3c24x0/mmc.h：

```
/*
 *  linux/drivers/mmc/mmc_pxa.h
 *
 *  Author: Vladimir Shebordaev, Igor Oblakov
 *  Copyright:  MontaVista Software Inc.
 *
 *  $Id: mmc_pxa.h,v 0.3.1.6 2002/09/25 19:25:48 ted Exp ted $
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License version 2 as
 *  published by the Free Software Foundation.
 */
#ifndef __MMC_PXA_P_H__
#define __MMC_PXA_P_H__

#include <asm/arch/regs-sdi.h>

#define MMC_DEFAULT_RCA                 (1<<16)

#define MMC_BLOCK_SIZE                  512
#define MMC_CMD_RESET                   0
#define MMC_CMD_SEND_OP_COND            1
#define MMC_CMD_ALL_SEND_CID            2
#define MMC_CMD_SET_RCA                 3
#define MMC_CMD_SELECT_CARD             7
```

```c
#define MMC_CMD_IF_COND         8
#define MMC_CMD_SEND_CSD            9
#define MMC_CMD_SEND_CID            10
#define MMC_CMD_SEND_STATUS         13
#define MMC_CMD_SET_BLOCKLEN        16
#define MMC_CMD_READ_BLOCK          17
#define MMC_CMD_RD_BLK_MULTI        18
#define MMC_CMD_WRITE_BLOCK         24

#define MMC_MAX_BLOCK_SIZE          512

#define MMC_R1_IDLE_STATE       0x01
#define MMC_R1_ERASE_STATE          0x02
#define MMC_R1_ILLEGAL_CMD          0x04
#define MMC_R1_COM_CRC_ERR          0x08
#define MMC_R1_ERASE_SEQ_ERR        0x01
#define MMC_R1_ADDR_ERR             0x02
#define MMC_R1_PARAM_ERR            0x04

#define MMC_R1B_WP_ERASE_SKIP       0x0002
#define MMC_R1B_ERR                 0x0004
#define MMC_R1B_CC_ERR              0x0008
#define MMC_R1B_CARD_ECC_ERR        0x0010
#define MMC_R1B_WP_VIOLATION        0x0020
#define MMC_R1B_ERASE_PARAM         0x0040
#define MMC_R1B_OOR                 0x0080
#define MMC_R1B_IDLE_STATE          0x0100
#define MMC_R1B_ERASE_RESET         0x0200
#define MMC_R1B_ILLEGAL_CMD         0x0400
#define MMC_R1B_COM_CRC_ERR         0x0800
#define MMC_R1B_ERASE_SEQ_ERR       0x1000
#define MMC_R1B_ADDR_ERR            0x2000
#define MMC_R1B_PARAM_ERR           0x4000

typedef struct mmc_cid
{
        /* FIXME: BYTE_ORDER */
        uchar   year:4,
                month:4;
        uchar   sn[3];
        uchar   fwrev:4,
                hwrev:4;
        uchar   name[6];
        uchar   id[3];
} mmc_cid_t;

typedef struct mmc_csd
{
        uchar   ecc:2,
                file_format:2,
                tmp_write_protect:1,
                perm_write_protect:1,
                copy:1,
                file_format_grp:1;
        uint64_t content_prot_app:1,
                rsvd3:4,
                write_bl_partial:1,
                write_bl_len:4,
                r2w_factor:3,
                default_ecc:2,
```

```
                wp_grp_enable:1,
                wp_grp_size:5,
                erase_grp_mult:5,
                erase_grp_size:5,
                c_size_mult1:3,
                vdd_w_curr_max:3,
                vdd_w_curr_min:3,
                vdd_r_curr_max:3,
                vdd_r_curr_min:3,
                c_size:12,
                rsvd2:2,
                dsr_imp:1,
                read_blk_misalign:1,
                write_blk_misalign:1,
                read_bl_partial:1;

        ushort  read_bl_len:4,
                ccc:12;
        uchar   tran_speed;
        uchar   nsac;
        uchar   taac;
        uchar   rsvd1:2,
                spec_vers:4,
                csd_structure:2;
} mmc_csd_t;


#endif /* __MMC_PXA_P_H__ */
```

include/asm-arm/arch-s3c24x0/regs-sdi.h :

```
/* linux/include/asm/arch-s3c2410/regs-sdi.h
 *
 * Copyright (c) 2004 Simtec Electronics <linux@simtec.co.uk>
 *                http://www.simtec.co.uk/products/SWLINUX/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 *
 * S3C2410 MMC/SDIO register definitions
 *
 *  Changelog:
 *    18-Aug-2004 Ben Dooks     Created initial file
 *    29-Nov-2004 Koen Martens   Added some missing defines, fixed duplicates
 *    29-Nov-2004 Ben Dooks       Updated Koen's patch
*/

#ifndef __ASM_ARM_REGS_SDI
#define __ASM_ARM_REGS_SDI "regs-sdi.h"

#define S3C2440_SDICON_SDRESET     (1<<8)
#define S3C2440_SDICON_MMCCLOCK    (1<<5)
#define S3C2410_SDICON_BYTEORDER   (1<<4)
#define S3C2410_SDICON_SDIOIRQ     (1<<3)
#define S3C2410_SDICON_RWAITEN     (1<<2)
#define S3C2410_SDICON_FIFORESET   (1<<1)
#define S3C2410_SDICON_CLOCKTYPE   (1<<0)
```

```c
#define S3C2410_SDICMDCON_ABORT      (1<<12)
#define S3C2410_SDICMDCON_WITHDATA   (1<<11)
#define S3C2410_SDICMDCON_LONGRSP    (1<<10)
#define S3C2410_SDICMDCON_WAITRSP    (1<<9)
#define S3C2410_SDICMDCON_CMDSTART   (1<<8)
#define S3C2410_SDICMDCON_SENDERHOST (1<<6)
#define S3C2410_SDICMDCON_INDEX      (0x3f)

#define S3C2410_SDICMDSTAT_CRCFAIL   (1<<12)
#define S3C2410_SDICMDSTAT_CMDSENT   (1<<11)
#define S3C2410_SDICMDSTAT_CMDTIMEOUT (1<<10)
#define S3C2410_SDICMDSTAT_RSPFIN    (1<<9)
#define S3C2410_SDICMDSTAT_XFERING   (1<<8)
#define S3C2410_SDICMDSTAT_INDEX     (0xff)

#define S3C2440_SDIDCON_DS_BYTE      (0<<22)
#define S3C2440_SDIDCON_DS_HALFWORD  (1<<22)
#define S3C2440_SDIDCON_DS_WORD      (2<<22)
#define S3C2410_SDIDCON_IRQPERIOD    (1<<21)
#define S3C2410_SDIDCON_TXAFTERRESP  (1<<20)
#define S3C2410_SDIDCON_RXAFTERCMD   (1<<19)
#define S3C2410_SDIDCON_BUSYAFTERCMD (1<<18)
#define S3C2410_SDIDCON_BLOCKMODE    (1<<17)
#define S3C2410_SDIDCON_WIDEBUS      (1<<16)
#define S3C2410_SDIDCON_DMAEN        (1<<15)
#define S3C2410_SDIDCON_STOP         (1<<14)
#define S3C2440_SDIDCON_DATSTART     (1<<14)
#define S3C2410_SDIDCON_DATMODE      (3<<12)
#define S3C2410_SDIDCON_BLKNUM       (0x7ff)

/* constants for S3C2410_SDIDCON_DATMODE */
#define S3C2410_SDIDCON_XFER_READY   (0<<12)
#define S3C2410_SDIDCON_XFER_CHKSTART (1<<12)
#define S3C2410_SDIDCON_XFER_RXSTART (2<<12)
#define S3C2410_SDIDCON_XFER_TXSTART (3<<12)

#define S3C2410_SDIDCNT_BLKNUM_MASK  (0xFFF)
#define S3C2410_SDIDCNT_BLKNUM_SHIFT (12)

#define S3C2410_SDIDSTA_RDYWAITREQ   (1<<10)
#define S3C2410_SDIDSTA_SDIOIRQDETECT (1<<9)
#define S3C2410_SDIDSTA_FIFOFAIL     (1<<8)       /* reserved on 2440 */
#define S3C2410_SDIDSTA_CRCFAIL      (1<<7)
#define S3C2410_SDIDSTA_RXCRCFAIL    (1<<6)
#define S3C2410_SDIDSTA_DATATIMEOUT  (1<<5)
#define S3C2410_SDIDSTA_XFERFINISH   (1<<4)
#define S3C2410_SDIDSTA_BUSYFINISH   (1<<3)
#define S3C2410_SDIDSTA_SBITERR      (1<<2)       /* reserved on 2410a/2440 */
#define S3C2410_SDIDSTA_TXDATAON     (1<<1)
#define S3C2410_SDIDSTA_RXDATAON     (1<<0)

#define S3C2440_SDIFSTA_FIFORESET    (1<<16)
#define S3C2440_SDIFSTA_FIFOFAIL     (3<<14)  /* 3 is correct (2 bits) */
#define S3C2410_SDIFSTA_TFDET        (1<<13)
#define S3C2410_SDIFSTA_RFDET        (1<<12)
#define S3C2410_SDIFSTA_TFHALF       (1<<11)
#define S3C2410_SDIFSTA_TFEMPTY      (1<<10)
#define S3C2410_SDIFSTA_RFLAST       (1<<9)
#define S3C2410_SDIFSTA_RFFULL       (1<<8)
#define S3C2410_SDIFSTA_RFHALF       (1<<7)
```

```
#define S3C2410_SDIFSTA_COUNTMASK     (0x7f)

#define S3C2410_SDIIMSK_RESPONSECRC   (1<<17)
#define S3C2410_SDIIMSK_CMDSENT       (1<<16)
#define S3C2410_SDIIMSK_CMDTIMEOUT    (1<<15)
#define S3C2410_SDIIMSK_RESPONSEND    (1<<14)
#define S3C2410_SDIIMSK_READWAIT      (1<<13)
#define S3C2410_SDIIMSK_SDIOIRQ       (1<<12)
#define S3C2410_SDIIMSK_FIFOFAIL      (1<<11)
#define S3C2410_SDIIMSK_CRCSTATUS     (1<<10)
#define S3C2410_SDIIMSK_DATACRC       (1<<9)
#define S3C2410_SDIIMSK_DATATIMEOUT   (1<<8)
#define S3C2410_SDIIMSK_DATAFINISH    (1<<7)
#define S3C2410_SDIIMSK_BUSYFINISH    (1<<6)
#define S3C2410_SDIIMSK_SBITERR       (1<<5)        /* reserved 2440/2410a */
#define S3C2410_SDIIMSK_TXFIFOHALF    (1<<4)
#define S3C2410_SDIIMSK_TXFIFOEMPTY   (1<<3)
#define S3C2410_SDIIMSK_RXFIFOLAST    (1<<2)
#define S3C2410_SDIIMSK_RXFIFOFULL    (1<<1)
#define S3C2410_SDIIMSK_RXFIFOHALF    (1<<0)

#endif /* __ASM_ARM_REGS_SDI */
```

6.5 第四阶段：修正配置文件

6.5.1 添加 CONFIG_S3C2440 条件定义

对于 S3C2440，很多代码是借用 S3C2410 的，所以要在所有条件编译中有 CONFIG_S3C2410 的地方添加 CONFIG_S3C2440，这样这些代码才会编译进来。一个简单的方法就是在代码中搜索出所有的 CONFIG_S3C2410，并根据实际情况修改。在有些地方不仅要加入 CONFIG_S3C2440，还必须根据两个芯片的不同来分布做出修改，比如 PLL 的操作代码。对于 U-boot-2009.11 的修改如下：

```
diff -aurNp u-boot-2009.11/common/serial.c u-boot-2009.11_tekkaman/common/serial.c
--- u-boot-2009.11/common/serial.c      2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/common/serial.c 2010-03-28 17:16:12.000000000 +0800
@@ -59,7 +59,7 @@ struct serial_device *__default_serial_c
 #else
                return &serial0_device;
 #endif
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined(CONFIG_S3C2440)
 #if defined(CONFIG_SERIAL1)
        return &s3c24xx_serial0_device;
 #elif defined(CONFIG_SERIAL2)
@@ -148,7 +148,7 @@ void serial_initialize (void)
 #if defined (CONFIG_STUART)
        serial_register(&serial_stuart_device);
 #endif
-#if defined(CONFIG_S3C2410)
+#if defined(CONFIG_S3C2410) || defined(CONFIG_S3C2440)
        serial_register(&s3c24xx_serial0_device);
```

```
                 serial_register(&s3c24xx_serial1_device);
                 serial_register(&s3c24xx_serial2_device);
```

```diff
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/interrupts.c u-boot-
2009.11_tekkaman/cpu/arm920t/s3c24x0/interrupts.c
--- u-boot-2009.11/cpu/arm920t/s3c24x0/interrupts.c       2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/interrupts.c 2010-03-28 17:16:12.000000000 +0800
@@ -33,7 +33,7 @@

 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif
 #include <asm/proc-armv/ptrace.h>
```

```diff
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/timer.c u-boot-
2009.11_tekkaman/cpu/arm920t/s3c24x0/timer.c
--- u-boot-2009.11/cpu/arm920t/s3c24x0/timer.c  2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/timer.c       2010-03-28 17:16:12.000000000 +0800
@@ -32,13 +32,14 @@
 #include <common.h>
 #if defined(CONFIG_S3C2400) || \
     defined(CONFIG_S3C2410) || \
+    defined(CONFIG_S3C2440) || \
     defined(CONFIG_TRAB)

 #include <asm/io.h>

 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif

@@ -188,6 +189,7 @@ ulong get_tbclk(void)
         tbclk = timer_load_val * 100;
 #elif defined(CONFIG_SBC2410X) || \
     defined(CONFIG_SMDK2410) || \
+    defined(CONFIG_MINI2440) || \
     defined(CONFIG_VCMA9)
         tbclk = CONFIG_SYS_HZ;
 #else
@@ -229,4 +231,5 @@ void reset_cpu(ulong ignored)

 #endif /* defined(CONFIG_S3C2400)  ||
         defined (CONFIG_S3C2410) ||
+        defined(CONFIG_S3C2440)  ||
         defined (CONFIG_TRAB) */
```

```diff
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/usb.c u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/usb.c
--- u-boot-2009.11/cpu/arm920t/s3c24x0/usb.c   2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/usb.c       2010-03-28 17:16:12.000000000 +0800
@@ -24,11 +24,11 @@
 #include <common.h>

 #if defined(CONFIG_USB_OHCI_NEW) && defined(CONFIG_SYS_USB_OHCI_CPU_INIT)
-# if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410)
+# if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410) || defined(CONFIG_S3C2440)
```

```
 #if defined(CONFIG_S3C2400)
 # include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined(CONFIG_S3C2440)
 # include <s3c2410.h>
 #endif
```

```
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/usb_ohci.c u-boot-
2009.11_tekkaman/cpu/arm920t/s3c24x0/usb_ohci.c
--- u-boot-2009.11/cpu/arm920t/s3c24x0/usb_ohci.c       2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/usb_ohci.c 2010-03-28 17:16:12.000000000 +0800
@@ -40,7 +40,7 @@

 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410)|| defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif
```

```
diff -aurNp u-boot-2009.11/drivers/i2c/s3c24x0_i2c.c u-boot-2009.11_tekkaman/drivers/i2c/s3c24x0_i2c.c
--- u-boot-2009.11/drivers/i2c/s3c24x0_i2c.c       2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/i2c/s3c24x0_i2c.c 2010-03-28 17:16:12.000000000 +0800
@@ -29,7 +29,7 @@
 #include <common.h>
 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif

@@ -61,7 +61,7 @@ static int GetI2CSDA(void)
 {
        struct s3c24x0_gpio *gpio = s3c24x0_get_base_gpio();

-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
        return (readl(&gpio->GPEDAT) & 0x8000) >> 15;
 #endif
 #ifdef CONFIG_S3C2400
@@ -80,7 +80,7 @@ static void SetI2CSCL(int x)
 {
        struct s3c24x0_gpio *gpio = s3c24x0_get_base_gpio();

-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
        writel((readl(&gpio->GPEDAT) & ~0x4000) | (x & 1) << 14, &gpio->GPEDAT);
 #endif
 #ifdef CONFIG_S3C2400
@@ -132,7 +132,7 @@ void i2c_init(int speed, int slaveadd)
        }

        if ((readl(&i2c->IICSTAT) & I2CSTAT_BSY) || GetI2CSDA() == 0) {
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
                ulong old_gpecon = readl(&gpio->GPECON);
 #endif
```

```
 #ifdef CONFIG_S3C2400
@@ -141,7 +141,7 @@ void i2c_init(int speed, int slaveadd)
                /* bus still busy probably by (most) previously interrupted
                   transfer */

-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
                /* set I2CSDA and I2CSCL (GPE15, GPE14) to GPIO */
                writel((readl(&gpio->GPECON) & ~0xF0000000) | 0x10000000,
                    &gpio->GPECON);
@@ -167,7 +167,7 @@ void i2c_init(int speed, int slaveadd)
                udelay(1000);

                /* restore pin functions */
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
                writel(old_gpecon, &gpio->GPECON);
 #endif
 #ifdef CONFIG_S3C2400
```

```
diff -aurNp u-boot-2009.11/drivers/rtc/s3c24x0_rtc.c u-boot-2009.11_tekkaman/drivers/rtc/s3c24x0_rtc.c
--- u-boot-2009.11/drivers/rtc/s3c24x0_rtc.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/rtc/s3c24x0_rtc.c 2010-03-28 17:16:12.000000000 +0800
@@ -32,7 +32,7 @@

 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined(CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif
```

```
diff -aurNp u-boot-2009.11/drivers/serial/serial_s3c24x0.c u-boot-
2009.11_tekkaman/drivers/serial/serial_s3c24x0.c
--- u-boot-2009.11/drivers/serial/serial_s3c24x0.c        2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/serial/serial_s3c24x0.c    2010-03-28 17:16:12.000000000 +0800
@@ -21,7 +21,7 @@
 #include <common.h>
 #if defined(CONFIG_S3C2400) || defined(CONFIG_TRAB)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif
```

```
diff -aurNp u-boot-2009.11/drivers/usb/host/ohci-hcd.c u-boot-2009.11_tekkaman/drivers/usb/host/ohci-hcd.c
--- u-boot-2009.11/drivers/usb/host/ohci-hcd.c    2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/drivers/usb/host/ohci-hcd.c        2010-03-28 17:16:12.000000000 +0800
@@ -67,6 +67,7 @@
 #if defined(CONFIG_ARM920T) || \
    defined(CONFIG_S3C2400) || \
    defined(CONFIG_S3C2410) || \
+   defined(CONFIG_S3C2440) || \
    defined(CONFIG_S3C6400) || \
    defined(CONFIG_440EP) || \
    defined(CONFIG_PCI_OHCI) || \
```

```
diff -aurNp u-boot-2009.11/include/common.h u-boot-2009.11_tekkaman/include/common.h
--- u-boot-2009.11/include/common.h    2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/common.h        2010-03-28 17:16:12.000000000 +0800
```

```
@@ -496,7 +496,7 @@ ulong   get_OPB_freq (void);
 ulong   get_PCI_freq (void);
 #endif
 #if defined(CONFIG_S3C2400) || defined(CONFIG_S3C2410) || \
-        defined(CONFIG_LH7A40X) || defined(CONFIG_S3C6400)
+        defined(CONFIG_LH7A40X) || defined(CONFIG_S3C6400) || defined(CONFIG_S3C2440)
 ulong   get_FCLK (void);
 ulong   get_HCLK (void);
 ulong   get_PCLK (void);
```

```
diff -aurNp u-boot-2009.11/include/serial.h u-boot-2009.11_tekkaman/include/serial.h
--- u-boot-2009.11/include/serial.h          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/serial.h   2010-03-28 17:16:12.000000000 +0800
@@ -37,7 +37,7 @@ extern struct serial_device eserial4_dev

 #endif

-#if defined(CONFIG_S3C2410)
+#if defined(CONFIG_S3C2410) ||  defined(CONFIG_S3C2440)
 extern struct serial_device s3c24xx_serial0_device;
 extern struct serial_device s3c24xx_serial1_device;
 extern struct serial_device s3c24xx_serial2_device;
```

在 cpu/arm920t/s3c24x0/speed.c 文件中必须根据 S3C2440 与 S3C2410 的不同来修改：

```
diff -aurNp u-boot-2009.11/cpu/arm920t/s3c24x0/speed.c u-boot-
2009.11_tekkaman/cpu/arm920t/s3c24x0/speed.c
--- u-boot-2009.11/cpu/arm920t/s3c24x0/speed.c          2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/cpu/arm920t/s3c24x0/speed.c   2010-03-28 17:16:12.000000000 +0800
@@ -30,13 +30,13 @@
  */

 #include <common.h>
-#if defined(CONFIG_S3C2400) || defined (CONFIG_S3C2410) || defined (CONFIG_TRAB)
+#if defined(CONFIG_S3C2400) || defined (CONFIG_S3C2410) || defined (CONFIG_TRAB) || defined
(CONFIG_S3C2440)

 #include <asm/io.h>

 #if defined(CONFIG_S3C2400)
 #include <s3c2400.h>
-#elif defined(CONFIG_S3C2410)
+#elif defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
 #include <s3c2410.h>
 #endif

@@ -68,7 +68,13 @@ static ulong get_PLLCLK(int pllreg)
        m = ((r & 0xFF000) >> 12) + 8;
        p = ((r & 0x003F0) >> 4) + 2;
        s = r & 0x3;
-
+//tekkaman
+#if defined(CONFIG_S3C2440)
+       if (pllreg == MPLL)
+       return ((CONFIG_SYS_CLK_FREQ * m * 2) /(p << s));
+       else if (pllreg == UPLL)
+#endif
+//tekkaman
```

```
            return (CONFIG_SYS_CLK_FREQ * m) / (p << s);
}

@@ -83,7 +89,21 @@ ulong get_HCLK(void)
{
            struct s3c24x0_clock_power *clk_power = s3c24x0_get_base_clock_power();

-           return (readl(&clk_power->CLKDIVN) & 2) ? get_FCLK() / 2 : get_FCLK();
+//         return (readl(&clk_power->CLKDIVN) & 2) ? get_FCLK() / 2 : get_FCLK();
+//tekkaman
+#if defined(CONFIG_S3C2440)
+      if (readl(&clk_power->CLKDIVN) & 0x6)
+                              {
+                              if ((readl(&clk_power->CLKDIVN) & 0x6)==2) return(get_FCLK()/2);
+                              if ((readl(&clk_power->CLKDIVN) & 0x6)==6) return((readl(&clk_power-
>CAMDIVN) & 0x100) ? get_FCLK()/6 : get_FCLK()/3);
+                              if ((readl(&clk_power->CLKDIVN) & 0x6)==4) return((readl(&clk_power-
>CAMDIVN) & 0x200) ? get_FCLK()/8 : get_FCLK()/4);
+                              return(get_FCLK());
+                              }
+      else      return(get_FCLK());
+#else
+      return((readl(&clk_power->CLKDIVN) & 0x2) ? get_FCLK()/2 : get_FCLK());
+#endif
+//tekkaman
}

/* return PCLK frequency */
@@ -102,4 +122,5 @@ ulong get_UCLK(void)

#endif /* defined(CONFIG_S3C2400) ||
            defined (CONFIG_S3C2410) ||
+           defined (CONFIG_S3C2440) ||
            defined (CONFIG_TRAB) */
```

include/s3c24x0.h 文件中主要放的是寄存器定义，根据两款芯片的不同在 Nand 、USB 和 SD 开接口等方面必须经过修改。

```
diff -aurNp u-boot-2009.11/include/s3c24x0.h u-boot-2009.11_tekkaman/include/s3c24x0.h
--- u-boot-2009.11/include/s3c24x0.h     2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/s3c24x0.h          2010-03-28 17:16:12.000000000 +0800
@@ -82,7 +82,7 @@ struct s3c24x0_interrupt {
      S3C24X0_REG32          PRIORITY;
      S3C24X0_REG32          INTPND;
      S3C24X0_REG32          INTOFFSET;
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
      S3C24X0_REG32          SUBSRCPND;
      S3C24X0_REG32          INTSUBMSK;
#endif
@@ -92,11 +92,11 @@ struct s3c24x0_interrupt {
/* DMAS (see manual chapter 8) */
struct s3c24x0_dma {
      S3C24X0_REG32          DISRC;
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
      S3C24X0_REG32          DISRCC;
#endif
      S3C24X0_REG32          DIDST;
```

```diff
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
	S3C24X0_REG32		DIDSTC;
 #endif
	S3C24X0_REG32		DCON;
@@ -107,7 +107,7 @@ struct s3c24x0_dma {
 #ifdef CONFIG_S3C2400
	S3C24X0_REG32		res[1];
 #endif
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
	S3C24X0_REG32		res[7];
 #endif
 };
@@ -126,6 +126,9 @@ struct s3c24x0_clock_power {
	S3C24X0_REG32		CLKCON;
	S3C24X0_REG32		CLKSLOW;
	S3C24X0_REG32		CLKDIVN;
+#if defined (CONFIG_S3C2440)
+	S3C24X0_REG32		CAMDIVN;
+#endif
 };


@@ -145,7 +148,7 @@ struct s3c24x0_lcd {
	S3C24X0_REG32		res[8];
	S3C24X0_REG32		DITHMODE;
	S3C24X0_REG32		TPAL;
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
	S3C24X0_REG32		LCDINTPND;
	S3C24X0_REG32		LCDSRCPND;
	S3C24X0_REG32		LCDINTMSK;
@@ -153,7 +156,7 @@ struct s3c24x0_lcd {
 #endif
 };

-
+#if defined(CONFIG_S3C2410)
 /* NAND FLASH (see S3C2410 manual chapter 6) */
 struct s3c2410_nand {
	S3C24X0_REG32		NFCONF;
@@ -163,7 +166,28 @@ struct s3c2410_nand {
	S3C24X0_REG32		NFSTAT;
	S3C24X0_REG32		NFECC;
 };
-
+#endif
+#if defined (CONFIG_S3C2440)
+/* NAND FLASH (see S3C2440 manual chapter 6) */
+struct s3c2410_nand {
+	S3C24X0_REG32		NFCONF;
+	S3C24X0_REG32		NFCONT;
+	S3C24X0_REG32		NFCMD;
+	S3C24X0_REG32		NFADDR;
+	S3C24X0_REG32		NFDATA;
+	S3C24X0_REG32		NFMECCD0;
+	S3C24X0_REG32		NFMECCD1;
+	S3C24X0_REG32		NFSECCD;
+	S3C24X0_REG32		NFSTAT;
```

```
+        S3C24X0_REG32        NFESTAT0;
+        S3C24X0_REG32        NFESTAT1;
+        S3C24X0_REG32        NFMECC0;
+        S3C24X0_REG32        NFMECC1;
+        S3C24X0_REG32        NFSECC;
+        S3C24X0_REG32        NFSBLK;
+        S3C24X0_REG32        NFEBLK;
+};
+#endif

 /* UART (see manual chapter 11) */
 struct s3c24x0_uart {
@@ -316,8 +340,17 @@ struct s3c24x0_usb_device {
        S3C24X0_REG8        OUT_FIFO_CNT2_REG;
        S3C24X0_REG8        res16[3];
 #endif /* __BIG_ENDIAN */
+//        struct s3c24x0_usb_dev_fifos    fifo[5];
+//        struct s3c24x0_usb_dev_dmas  dma[5];
+
+        S3C24X0_REG32   res17[8];
        struct s3c24x0_usb_dev_fifos     fifo[5];
-        struct s3c24x0_usb_dev_dmas  dma[5];
+        S3C24X0_REG32   res18[11];
+        struct s3c24x0_usb_dev_dmas  ep1;
+        struct s3c24x0_usb_dev_dmas  ep2;
+        S3C24X0_REG8    res19[16];
+        struct s3c24x0_usb_dev_dmas  ep3;
+        struct s3c24x0_usb_dev_dmas  ep4;
 };


@@ -401,7 +434,7 @@ struct s3c24x0_gpio {
        S3C24X0_REG32        MISCCR;
        S3C24X0_REG32        EXTINT;
 #endif
-#ifdef CONFIG_S3C2410
+#if defined(CONFIG_S3C2410) || defined (CONFIG_S3C2440)
        S3C24X0_REG32        GPACON;
        S3C24X0_REG32        GPADAT;
        S3C24X0_REG32        res1[2];
@@ -450,6 +483,14 @@ struct s3c24x0_gpio {
        S3C24X0_REG32        GSTATUS2;
        S3C24X0_REG32        GSTATUS3;
        S3C24X0_REG32        GSTATUS4;
+#if defined (CONFIG_S3C2440)
+        S3C24X0_REG32        res9[3];
+        S3C24X0_REG32        MSLCON;
+        S3C24X0_REG32        GPJCON;
+        S3C24X0_REG32        GPJDAT;
+        S3C24X0_REG32        GPJUP;
+
+#endif
 #endif
 };


@@ -643,6 +684,8 @@ struct s3c2410_sdi {
        S3C24X0_REG32        SDIDCNT;
        S3C24X0_REG32        SDIDSTA;
        S3C24X0_REG32        SDIFSTA;
+#if defined(CONFIG_S3C2410)
```

```
+#if 0
 #ifdef __BIG_ENDIAN
        S3C24X0_REG8         res[3];
        S3C24X0_REG8         SDIDAT;
@@ -650,7 +693,14 @@ struct s3c2410_sdi {
        S3C24X0_REG8         SDIDAT;
        S3C24X0_REG8         res[3];
 #endif
+#endif
+       S3C24X0_REG32        SDIDAT;
+       S3C24X0_REG32        SDIIMSK;
+#elif defined(CONFIG_S3C2440)
        S3C24X0_REG32        SDIIMSK;
+       S3C24X0_REG32        SDIDAT;
+#endif
+
 };

 #endif /*__S3C24X0_H__*/
```

6.5.2 修改配置文件 include/configs/mini2440.h

最后，还有修改配置文件，使得前面修改的很多功能编译进来。

这里主要做了以下修改：

（1）去除了 CS8900 网卡的定义，添加了 DM9000。

（2）使能了 JFFS2、FAT 文件系统。

（3）使能了 USB、SD 卡功能。

（5）使能了 I2C、EEPROM 功能。

（6）使能了 LCD 功能，以及 BMP 图片显示和字符 console 的功能。

（7）去除了 AMD 的 Nor Flash 芯片的定义，增加 SST Nor Flash 芯片定义。

```
--- u-boot-2009.11/include/configs/sbc2410x.h     2009-12-16 06:20:54.000000000 +0800
+++ u-boot-2009.11_tekkaman/include/configs/mini2440.h        2010-03-28 17:16:12.000000000 +0800
@@ -44,9 +44,12 @@
 * (easy to change)
 */
 #define CONFIG_ARM920T          1       /* This is an ARM920T Core      */
-#define      CONFIG_S3C2410              1       /* in a SAMSUNG S3C2410 SoC    */
-#define CONFIG_SBC2410X          1       /* on a friendly-arm SBC-2410X Board  */
-
+//#define      CONFIG_S3C2410              1       /* in a SAMSUNG S3C2410 SoC    */
+//#define CONFIG_SBC2410X          1       /* on a friendly-arm SBC-2410X Board  */
+#define      CONFIG_S3C2440              1       /* in a SAMSUNG S3C2440 SoC    */
+#define CONFIG_MINI2440          1       /* on a friendly-arm MINI2440 Board  */
+#define CONFIG_MINI2440_LED       1
+#define CONFIG_S3C2410_NAND_SKIP_BAD1
 /* input clock of PLL */
 #define CONFIG_SYS_CLK_FREQ       12000000/* the SBC2410X has 12MHz input clock */

@@ -63,11 +66,21 @@
 /*
 * Hardware drivers
```

```
 */
+#if 0
 #define CONFIG_NET_MULTI
 #define CONFIG_CS8900                   /* we have a CS8900 on-board */
 #define CONFIG_CS8900_BASE      0x19000300
 #define CONFIG_CS8900_BUS16     /* the Linux driver does accesses as shorts */
-
+#endif
+#define CONFIG_NET_MULTI            1
+#define CONFIG_NET_RETRY_COUNT            20
+#define CONFIG_DRIVER_DM9000         1
+#define CONFIG_DM9000_BASE             0x20000300
+#define DM9000_IO                CONFIG_DM9000_BASE
+#define DM9000_DATA                  (CONFIG_DM9000_BASE+4)
+#define CONFIG_DM9000_USE_16BIT            1
+#define CONFIG_DM9000_NO_SROM        1
+#undef CONFIG_DM9000_DEBUG
 /*
  * select serial console configuration
  */
@@ -104,19 +117,37 @@
 #define CONFIG_CMD_DATE
 #define CONFIG_CMD_DHCP
 #define CONFIG_CMD_ELF
+//#define CONFIG_MTD_DEVICE
+//#define CONFIG_CMD_MTDPARTS
 #define CONFIG_CMD_PING
-
-
-#define CONFIG_BOOTDELAY3
-#define CONFIG_BOOTARGS          "console=ttySAC0 root=/dev/nfs " \
-           "nfsroot=192.168.0.1:/friendly-arm/rootfs_netserv " \
-           "ip=192.168.0.69:192.168.0.1:192.168.0.1:255.255.255.0:debian:eth0:off"
-#define CONFIG_ETHADDR        08:00:3e:26:0a:5b
+#define CONFIG_CMD_NAND
+#define CONFIG_CMD_REGINFO
+#define CONFIG_CMD_FAT
+/* FAT support*/
+//#define CONFIG_CMD_EXT2
+
+#define CONFIG_CMD_JFFS2
+/* JFFS2 Support*/
+#define CONFIG_CMD_USB
+/* USB Support*/
+
+#define CONFIG_BOOTDELAY         1
+#define CONFIG_BOOTARGS          "noinitrd root=/dev/nfs rw
nfsroot=192.168.0.1:/home/tekkaman/working/nfs/rootfs ip=192.168.0.2:192.168.0.1::255.255.255.0
console=ttySAC0,115200 init=/linuxrc mem=64M"
+#define CONFIG_ETHADDR           08:08:11:18:12:27
 #define CONFIG_NETMASK      255.255.255.0
-#define CONFIG_IPADDR            192.168.0.69
+#define CONFIG_IPADDR            192.168.0.2
 #define CONFIG_SERVERIP          192.168.0.1
+#define CONFIG_GATEWAYIP192.168.0.1
+#define CONFIG_OVERWRITE_ETHADDR_ONCE
+
 /*#define CONFIG_BOOTFILE  "elinos-lart" */
-#define CONFIG_BOOTCOMMAND       "dhcp; bootm"
+#define CONFIG_BOOTCOMMAND      "nfs 0x30008000
```

```
192.168.0.1:/home/tekkaman/working/nfs/zImage.img;bootm"
+#define         CONFIG_EXTRA_ENV_SETTINGS                                    \
+        "tekkaman=bmp d 70000\0 "                          \
+        "stdin=serial\0"                              \
+        "stdout=serial\0"                             \
+        "stderr=serial\0"                             \
+        ""

 #if defined(CONFIG_CMD_KGDB)
 #define CONFIG_KGDB_BAUDRATE   115200            /* speed to run kgdb serial port */
 @@ -128,7 +159,7 @@
  * Miscellaneous configurable options
  */
 #defineCONFIG_SYS_LONGHELP                            /* undef to save memory            */
-#define         CONFIG_SYS_PROMPT              "[ ~ljh@GDLC ]# "      /* Monitor Command Prompt
         */
+#define         CONFIG_SYS_PROMPT              "[u-boot@MINI2440]# " /* Monitor Command Prompt
         */
 #defineCONFIG_SYS_CBSIZE          256               /* Console I/O Buffer Size       */
 #defineCONFIG_SYS_PBSIZE (CONFIG_SYS_CBSIZE+sizeof(CONFIG_SYS_PROMPT)+16) /* Print Buffer
Size */
 #defineCONFIG_SYS_MAXARGS             16                /* max number of command args        */
 @@ -137,7 +168,8 @@
 #define CONFIG_SYS_MEMTEST_START       0x30000000     /* memtest works on     */
 #define CONFIG_SYS_MEMTEST_END            0x33F00000     /* 63 MB in DRAM       */

-#define         CONFIG_SYS_LOAD_ADDR          0x33000000     /* default load address  */
+
+#define         CONFIG_SYS_LOAD_ADDR          0x30008000     /* default load address  */

 #defineCONFIG_SYS_HZ                     1000

 @@ -171,7 +203,8 @@
  */
 /* #define CONFIG_AMD_LV400       1        /\* uncomment this if you have a LV400 flash *\/ */

-#define CONFIG_AMD_LV800 1        /* uncomment this if you have a LV800 flash */
+//#define CONFIG_AMD_LV800         1        /* uncomment this if you have a LV800 flash */
+#define CONFIG_SST_VF1601         1        /* uncomment this if you have a Am29LV160DB flash */

 #define CONFIG_SYS_MAX_FLASH_BANKS   1        /* max number of memory banks */

 @@ -191,15 +224,62 @@
 #define CONFIG_SYS_FLASH_ERASE_TOUT (5*CONFIG_SYS_HZ) /* Timeout for Flash Erase */
 #define CONFIG_SYS_FLASH_WRITE_TOUT  (5*CONFIG_SYS_HZ) /* Timeout for Flash Write */

-#define         CONFIG_ENV_IS_IN_FLASH   1
-#define CONFIG_ENV_SIZE          0x10000          /* Total Size of Environment Sector */
+
+//#if 0
+#define CONFIG_CMD_EEPROM
+#define CONFIG_CMD_I2C
+
+#define CONFIG_DRIVER_S3C24X0_I2C    1   /* we use the buildin I2C controller */
+#define CONFIG_HARD_I2C    1      /* I2C with hardware support */
+
+#define CONFIG_SYS_I2C_SPEED       100000    /* I2C speed and slave address */
+#define CONFIG_SYS_I2C_SLAVE        0x7F
+
+#define CONFIG_SYS_I2C_EEPROM_ADDR       0x50   /* EEPROM at24c08        */
```

```
+#define CONFIG_SYS_I2C_EEPROM_ADDR_LEN    1    /* Bytes of address        */
+/* mask of address bits that overflow into the "EEPROM chip address"   */
+#define CONFIG_SYS_I2C_EEPROM_ADDR_OVERFLOW    0x07
+#define CONFIG_SYS_EEPROM_PAGE_WRITE_BITS 4    /* The Catalyst CAT24WC08 has    */
+            /* 16 byte page write mode using*/
+            /* last 4 bits of the address    */
+#define CONFIG_SYS_EEPROM_PAGE_WRITE_DELAY_MS    10   /* and takes up to 10 msec */
+#define CONFIG_SYS_EEPROM_PAGE_WRITE_ENABLE
+
+//#define CONFIG_ENV_IS_IN_EEPROM    1    /* use EEPROM for environment vars */
+//#define CONFIG_ENV_OFFSET      0x000   /* environment starts at offset 0 */
+//#define CONFIG_ENV_SIZE        0x400   /* 1KB  */
+
+//#else
+#define CONFIG_ENV_IS_IN_NAND 1
+//#define CONFIG_ENV_IS_IN_FLASH 1
+#define CONFIG_ENV_OFFSET 0X60000
+#define CONFIG_ENV_SIZE                 0x20000        /* Total Size of Environment Sector */
+//#endif
+
+/* == LENGTH_UBOOT*/
+#ifdef CONFIG_SST_VF1601
+#define PHYS_FLASH_SIZE            0x00200000 /* 2MB */
+#define CONFIG_SYS_MAX_FLASH_SECT    (32)      /* max number of sectors on one chip */
+#define CONFIG_ENV_ADDR          (CONFIG_SYS_FLASH_BASE + CONFIG_ENV_OFFSET) /* addr of
environment */
+#endif
+

 /*----------------------------------------------------------------------
  * NAND flash settings
  */
 #if defined(CONFIG_CMD_NAND)
 #define CONFIG_NAND_S3C2410
+#define CONFIG_SYS_NAND_BASE 0x4E000000
 #define CONFIG_SYS_MAX_NAND_DEVICE   1        /* Max number of NAND devices          */
+#define SECTORSIZE 512
+#define SECTORSIZE_2K 2048
+#define NAND_SECTOR_SIZE SECTORSIZE
+#define NAND_SECTOR_SIZE_2K SECTORSIZE_2K
+#define NAND_BLOCK_MASK 511
+#define NAND_BLOCK_MASK_2K 2047
+#define NAND_MAX_CHIPS 1
+#define CONFIG_MTD_NAND_VERIFY_WRITE
+#define CONFIG_SYS_64BIT_VSPRINTF                /* needed for nand_util.c */
 #endif  /* CONFIG_CMD_NAND */

 #define CONFIG_SETUP_MEMORY_TAGS
@@ -211,10 +291,83 @@

 #define CONFIG_CMDLINE_EDITING

-#ifdef CONFIG_CMDLINE_EDITING
-#undef CONFIG_AUTO_COMPLETE
-#else
+//#ifdef CONFIG_CMDLINE_EDITING
+//#undef CONFIG_AUTO_COMPLETE
+//#else
 #define CONFIG_AUTO_COMPLETE
+//#endif
```

```
+
+
+#if 1
+#define CONFIG_USB_OHCI
+#define CONFIG_USB_STORAGE
+//#define CONFIG_KEYBOARD
+//#define CONFIG_USB_KEYBOARD
+#define CONFIG_DOS_PARTITION
+#define CONFIG_SYS_DEVICE_DEREGISTER
+#define CONFIG_SUPPORT_VFAT
+#define LITTLEENDIAN
+#endif
+
+#define CONFIG_JFFS2_NAND 1
+//#undef CONFIG_JFFS2_CMDLINE
+#define CONFIG_JFFS2_DEV "nand0"
+#define CONFIG_JFFS2_PART_SIZE  0x480000
+#define CONFIG_JFFS2_PART_OFFSET       0x80000
+
+#define CONFIG_JFFS2_CMDLINE 1
+#define MTDIDS_DEFAULT "nand0=nandflash0"
+#define MTDPARTS_DEFAULT "mtdparts=nandflash0:384k(bootloader)," \
+                                   "128k(params)," \
+                                   "5m(kernel)," \
+                                   "-(root)"
+
+
+#define ENABLE_CMD_LOADB_X     1
+#define ENABLE_CMD_NAND_YAFFS          1
+#define ENABLE_CMD_NAND_YAFFS_SKIPFB        1
+//#define CFG_NAND_YAFFS1_NEW_OOB_LAYOUT  1
+
+#if 1
+#define CONFIG_CMD_BMP
+#define CONFIG_VIDEO
+#define CONFIG_VIDEO_S3C2410
+#define CONFIG_VIDEO_LOGO
+#define VIDEO_FB_16BPP_PIXEL_SWAP
+
+#define CONFIG_VIDEO_SW_CURSOR
+#define CONFIG_VIDEO_BMP_LOGO
+//#define CONFIG_CONSOLE_EXTRA_INFO
+//#define CONFIG_CONSOLE_CURSOR
+//#define CONFIG_CONSOLE_TIME
+#define CONFIG_CFB_CONSOLE
+#define CONFIG_SYS_CONSOLE_IS_IN_ENV
+//#define CFG_CONSOLE_INFO_QUIET
+//#define VIDEO_FB_LITTLE_ENDIAN
+#define CONFIG_SPLASH_SCREEN
+#define CONFIG_SYS_VIDEO_LOGO_MAX_SIZE       (240*320+1024+100) /* 100 = slack */
+#define CONFIG_VIDEO_BMP_GZIP
+#define CONFIG_CMD_UNZIP
+#define LCD_VIDEO_ADDR       0x33d00000
+
+/*for PC-keyboard*/
+#define VIDEO_KBD_INIT_FCT    0
+#define VIDEO_TSTC_FCT        serial_tstc
+#define VIDEO_GETC_FCT         serial_getc
+
+#endif
```

```
+
+/*for SD Card*/
+#define CONFIG_CMD_MMC
+#define CONFIG_MMC              1
+#define CONFIG_MMC_S3C   1        /* Enabling the MMC driver */
+#define CFG_MMC_BASE            0xff000000
+
+
+#if 0
+#define CONFIG_YAFFS2
+//#undef CONFIG_YAFFS_YAFFS2
+#undef CONFIG_YAFFS_NO_YAFFS1
 #endif

 #endif  /* __CONFIG_H */
```

所有的修改到此结束。

6.6 重新编译并测试

```
make distclean
make mini2440_config
Configuring for mini2440 board...
make
```

编译完成后按照前面讲的烧写方法烧到板子上，重启。

当听到蜂鸣器的响声，LCD 屏亮起并显示 DENX 的 logo，串口中传出：

```
U-Boot 2009.11 ( 4 月 04 2010 - 12:09:25)

 modified by tekkamanninja (tekkamanninja@163.com)
 Love Linux forever!!

I2C:   ready
DRAM:  64 MB
Flash:  2 MB
NAND:  128 MiB
Video: 240x320x16 20kHz 62Hz
In:    serial
Out:   serial
Err:   serial
Net:   dm9000
U-Boot 2009.11 ( 4 月 04 2010 - 12:09:25)
modified by tekkamanninja
(tekkamanninja@163.com)
Love Linux forever!!
Hit any key to stop autoboot:  0
[u-boot@MINI2440]#
```

说明你基本成功了！！！

第7章　下载最新的源代码

tekkaman ninja   port   2010-03   u-boot
github

如果在移植中出了什么问题，可以直接下载 Tekkaman Ninja的源码做参考。下载方法如

图：