Mini2440 に U-boot を移植マニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

更新日 2013/08/27

・修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2013/08/27

目次

第 1 章　BootLoader のコンセプトと機能

1.1　組み込み Linux ソフトウェア構造とディストリビューション

組み込み Linux システムのソフトウェアは下記の部分に分ける：

1)　ロードプログラム：内部 ROM のファーム起動コードと BootLoader を含む。

内部ファーム ROM はメーカーがチップ生産時のファームウェアで、BootLoader をロードする機能である。

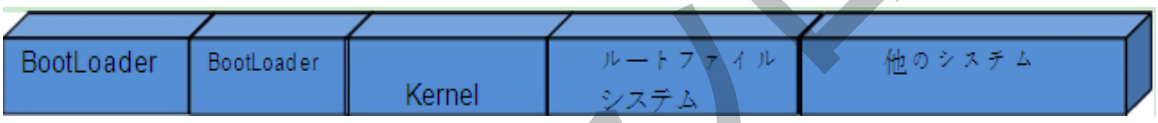
一部のチップ は複雑で、例えば Omap3 は flash 中コードがない時の起動方法は複数あり：USB、UART またはイーサネット等、S3C24x0 は Norboot と Nandboot の 2 種類を含まれる。

2)　Linux kernel と drivers。

3)　ファイルシステム：ルートファイルシステムと Flash に 構築する メモリーデバイスのファイルシステム（EXT4、UBI、CRAMFS 等）を含まれる。ファイルシステムは管理システムの各種設定ファイル及びシステムでユーザープログラムを実行する環境及プラットフォームを提供する。

4)　アプリプログラム。ユーザーカスタマイズアプリプログラム、ファイルシステムに保存する。

Flash メモリー、汎用仕組みは下記の通り：



多くの場合には上記の仕組みになりますが、あるルートファイルシステム、例えば initramfs で、他の部分はカーネルイメージに圧縮され；または、一部は Bootloader パラメータエリアがないなど。

1.2　組み込み Linux で BootLoader の必要性

Linux カーネル起動時、カーネルイメージはメインメモリの指定位置に保存する以外、CPU がある程度の条件を満たす必要がある：

1.　CPU　レジスタの設定：	R0＝0； R1＝Machine　ID(即 Machine　Type　Number、 linux/arch/arm/tools/mach-types に定義する)； R2＝カーネル起動パラメータは RAM の開始のベースアドレス
2.　CPU　モード：	割り込み禁止（IRQs と FIQs)； CPU　は SVC モードに必ず設定される；
3.　Cache　と　MMU　の設定：	MMU　クローズ； コマンド　Cache　オーペン/クローズ可； データ　Cache　クローズ；

CPU 起動の時、メモリーコントローラーも初期化しないため、メインメモリーでプログラム実行や Linux カーネル環境を実行できない。CPU 及び他の外部デバイスを初期化するため、 Linux カーネルはシステムメインメモリーで実行するため、システムは Linux カーネル起動の必要条件を満足するには、カーネル実行前のブートプログラムが必要とする（Boot Loader)。

BootLoader は Linux だけでなく、殆どの OS のデバイスが必要とする。そして PC の BOIS は Boot Loader の一部であり（初期のブート機能で、外部メモリーの BootLoader もある)、Linux PC にとって、Boot Loader = BIOS + GRUB/LILO。

1.3 Boot Loader の機能と選択

従って：BootLoader は即ち OS カーネル起動する前実行するサブプログラム。このプログラムを介し、ハードウェアデバイス初期化出来、システムのソフトハードウェア環境を適応の環境に設定し、最後に他のデバイス（Flash、イーサネット、UART）でカーネルイメージとメインメモリーにロードし、入口アドレスへジャンプする。

そして、 BootLoader はハードウェアを直接アクセスするため、ハードウェアに依存され、ロードする OS により、異なる選択がある、組み込み業界でも同じ状況。 S3C24x0 に対し、Linux をブートするには、一般は韓国の mizi 社の vivi または DENX ソフトウェアプロジェクトの Das U-boot を採用する；Win CE ブートの場合は Eboot。 StrongARM 構築の LART 開発には、 Jan-Derk Bakker と Erik Mouw 発表の Blob を採用(Boot Loader Object)。 eCos システムの場合は Redhat 社の Redboot を使われる。

全ての組み込みアプリケーションに適用する BootLoader は基本的に不可能であるため、同じ Boot Loader コードはより多い OS をサポートできるようにし、移植性を向上するのは可能である。U-boot はマルチプラットフォーム OS をサポートする良い例である。これも U-boot の優位性で、S3C2440 開発段階で U-boot を理解できれば、他のプラットフォームへの移植は簡単とする。そして U-boot コードの仕込みは改善され、新機能の追加も簡単となる。

第 2 章 U-boot について

2.1 U-boot の始まり

U-Boot は Das U-Boot の略称、Universal Boot Loader、GPL 条項従うオーペンソースコード項目である。最初ドイツの DENX ソフトウェアプロジェクトセンターの Wolfgang Denk は 8xxROM と FADSROM のソースコードに基づき PPCBoot プロジェクトを作成した、次はプロセッサーのサポートを追加する。その後、Sysgo Gmbh は PPCBoot を ARM プラットフォームに移植し、ARMBoot プロジェクトを作成した。最終的に PPCBoot プロジェクトと ARMBoot プロジェクトに基づき、U-Boot プロジェクトを作った。2002 年 12 月 17 日で最初のバージョン U-Boot-0.2.0 は発表され、 PPCBoot と ARMBoot サポートも同時に終了とする。

現在、U-Boot は汎用の BootLoader として、PowerPC、ARM、X86 、MIPS、NIOS、XScale などの百種類以上の開発ボードに移植され、柔軟性、汎用性が高いのオーペンソース BootLoader である。U-Boot は DENX の WolfgangDenk がサポートする。

2.2　U-boot の開発状況とリソース

最初の U-boot のバージョン晩後は X.Y.Z で表示される、 0.2.0～1.3.4。その後は年月日を追加した、2008.11 ～ 2010.3 まで平均3ヶ月毎新バージョンを開発した。毎回コードの構造と定義は修正と改善され、合理性や規範性を上がり、機能を強大させるに対し、移植の難易度逆に下がり、修正点は少なくなる。

U-boot はメインバージョンの他、 U-boot の Git コードレジストリには各種の CPU 構築のサブバージョンがある、一定の編集後、メインバージョンに追加出来る。

次は U-boot ソースコードのネットワークリソース

公式サイト	
ドイツ DENX ソフトウェアプロジェクトセンター	http://www.denx.de/
U-boot 公式サイト	http://www.denx.de/wiki/U-Boot/WebHome
U-boot 公式ソースコード FTP ダウンロード	ftp://ftp.denx.de/pub/u-boot/
U-boot 公式 Git コードレジストリ	http://git.denx.de/?p=u-boot.git
S3C2440 の編集について	
Openmoko モバイルの U-boot ソースコード Git	http://git.openmoko.org/?p=u-boot.git;a=shortlog;h=refs/heads/stable
buserror の U-boot ソースコード Git（mini2440 用）	http://repo.or.cz/w/u-boot-openmoko/mini2440.git
Tekkaman Ninja の U-boot ソースコード Git（mini2440 用）	http://github.com/tekkamanninja

第 3 章　開発環境構築

3.1　クロスコンパイルツールチェーンのインストール

mini2440 に U-boot をコンパイルする時、クロスコンパイルツールチェーンを使用する。付属のクロスコンパイルツール（gcc バージョン 4.3.2)、または rosstool-0.43、 crosstool-ng を使用し自行コンパイル出来る。

ツールを使用しクロスコンパイルツールチェーンコンパイルするには、下記の文章を参照できる。

crosstool0.43 で ARM－Linux クロスコンパイル環境構築

crosstool-ng で Linux クロスコンパイル環境構築（S3C2440（armv4t））

クロスコンパイルツールチェーンをコンパイル後、環境変数の PATH にコンパイルツールのパスを追加する（即ち arm-*-linux-*-gcc のパス)、こうして、コンパイル時システムはコンパイラのコマンドを識別出来る。 Ubuntu 下の変更方法は下記の通り：

vi ~/.profile、

最後に：PATH=`〈クロスコンパイルツールのパス〉:$PATH` を追加する。

3.2　ネットワークサービスの設定

U-boot を使用する時にホストの TFTP と NFS の 2 つのネットワークサービスを利用する、開発前で設定する。次は Ubuntu 下で apt-get を使用し インストールする：

3.2.1 TFTP サービスをインストールと設定

TFTP サービスをインストールと設定の手順は下記の通り：
(1) tftp-hpa、tftpd-hpa と openbsd-inetd プログラムインストール；
(2) 設定ファイル/etc/inetd.conf 編集；
(3) 設定ファイルのパスに従い、tftp ディレクトリ作成、ディレクトリ権限変更；
(4) tftp サービス再起動；
(5) ローカル伝送テスト。
インストールと設定手順のスクリプト：

```
#!/bin/sh
TFTPDIR=<設定する tftp ディレクトリパス>

        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 <設定する tftp パス>
        sudo vi /etc/inetd.conf


#tftp ディレクトリ作成，権限を変更：
        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

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

3.2.2 NFS サービスをインストールと設定

NFS サービスのインストールと設定の手順は下記の通り：

（1）NFS カーネルサービスをインストール；

（2）portmap サービス再設定、/etc/hosts.deny と/etc/hosts.allow 設定ファイル変更、portmap サービス再起動；

（3）NFS サービスの設定ファイル/etc/exports 編集、サービスディレクトリを追加と設定、設定を再インポート；

（4）NFS サービス再起動、ロード出来るディレクトリを検索；

（5）ローカルマウントをテスト。

インストールと設定手順のスクリプト：

```
#!/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)
        #上記の IP フォーマットを注意する、以前は 192.168.1.*で、現在は IP/マスク番号形式となる。
        #詳細は man マニュアルに参照： man exports
        sudo exportfs -r

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

3.3　シリアルプログラムのインストールと設定

U-boot を使用する時に、シリアルで開発ボードと通信のため、シリアル端末でプログラムが必要とする。Linux でよく使用するシリアル端末：minicom と C-kermit のインストール及び設定を説明する（Ubuntu で apt-get インストール）。

3.3.1 C-kermit のインストール設定

Linux ではシリアルを介し、ファイルを開発ボードに伝送する。

（1）ckermit プログラムをインストール；

（2）ckermit 設定ファイル~/.kermrc を編集。

```
#!/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
#USB→シリアルな場合、/dev/ttyUSB0 デバイスと同じ；ネイティブのハードウエアシリアルポートな場合、
/dev/ttyS0 のデバイスノートと同じ。
#使用するシリアルにより、デバイスノートは変更する場合がある，ls /dev/tty*コマンドで検索出来る。
        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 のインストールと設定

minicom はLinux システムでよく使うシリアルターミナルツール、クイックインストール設定手順は下記の通り：

(1) minicom プログラムをインストール；

(2) minicom -s コマンドで設定ファイル~/.minirc.dfl を作成する。

```
#!/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
```

コマンド minicom -s 実行、設定インタフェースに入る：

```
        +----[設定配置]----+
        | ファイル名とパス    |
        | ファイル伝送プロトコル |
```

```
|シリアルポート設定 |
|モデムとダイヤルアップ |
|スクリーンとキーボード |
|設定を dfl に保存|
|設定を保存.. |
|終了 |
|Minicom終了 |
+----------+
```

1.矢印キー選択゛シリアル設定゛：

```
+-----------------------------------------------------------+
|A- シリアルデバイス: /dev/ttyUSB0                          |
|B- ロックファイル位置: /var/lock                           |
|C- プログラムユール:                                       |
|D- プログラムリユール:                                     |
|E - Bps/Par/Bits : 115200 8N1                              |
|F – ハードウエアデータフロー制御: なし                     |
|G – ソフトウエアデータフロー制御: なし                     |
| 変更オプション？                                          |
+-----------------------------------------------------------+
```

対応の文字キーを設定し、例えばノート変更は A、カーソルを゛シリアルデバイス゛オプションに移動、値を変更する。 E を選択する場合：

```
+-------------[パラメータ]-------------+
|                                      |
|  現在: 115200 8N1                    |
|速度        パラメータ      データ    |
|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                |
|                                      |
|ストップビット                        |
|W: 1         Q: 8-N-1                 |
|X: 2         R: 7-E-1                 |
|                                      |
|                                      |
|Mini|終了、,<Enter>終了?|             |
+--------------------------------------+
```

E と Q 選択、設定完了後、゛設定゛メニューに入り、゛モデムとダイヤルアップ゛、下記の項目的データをクリア：

```
A - 文字列初期化 .....
B - 文字列リセット .....
K – 文字列停止 .....
```

完了後、゛設定゛→゛ dfl に保存゛、設定を~/.minirc.dfl に保存、゛終了゛。全て完了後、シリアルで開発ボードと接続できる。

また GUI 画面を搭載するシリアルターミナル：gtkterm、GUI 設定簡単が、ファイルを伝送機能はない。

第 4 章　U-boot の使用とプログラミング

U-boot の開発と移植、mini2440 を例とする。

開発ボードに U-boot をプログラミングする（mini2440 用の U-boot-2009.11 の bin ファイルを直接プログラミングする。）mini2440 の NAND または NOR Flash の開始アドレスにプログラミングすればよい。

4.1 U-boot を mini2440 開発ボードにプログラミング

4.2 汎用 U-boot コマンド

U-boot の発展により、そのコマンドラインモードは Linux の shell と類似する、TekkamanNinja からコンパイルした U-boot-2009.11 のコマンドラインモードでは゛Tab゛ボタンのコマンド補完とコマンドの歴史記録機能をサポート。コマンドの前の英数字を入力するだけで、必要なコマンドを選択出来る、U-boot のバージョン番号を確認コマンドは゛version゛、そして他のコマンドに頭文字は゛v゛のコマンドがないため、゛v゛だけを入力出来る。記録機能も下記の通り：

```
[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
```

4.2.1 ヘルプ

コマンド：help または ？

機能： U-boot の全てのコマンドを表示。

```
[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
```

特定コマンドの詳しいヘルプ内容、以下のコマンドを使ってください。
help 〈検索するコマンド〉
？ 〈検索するコマンド〉、
h 〈検索するコマンド略称〉。
bmp コマンドを例とする：

```
[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 環境変数と関連コマンド

Shell と類似、U-Boot にも環境変数（environment variables - ENV)、U-boot デフォルトの環境変数は下記の通り：

環境変数	説明
bootdelay	自動実行（bootcmd のコマンド）の待ち時間(s)
baudrate	シリアルコンソールのボーレート
netmask	イーサネットのネットワークマスク
ethaddr	イーサネットの MAC アドレス
bootfile	デフォルトダウンロードのファイル名
bootargs	Linux カーネルへ伝送する起動パラメータ
bootcmd	自動起動時に実行するコマンド
serverip	ファイルサーバーの IP アドレス
ipaddr	ローカル IP アドレス
stdin	標準入力デバイス、シリアル
stdout	標準出力、シリアル/ LCD (VGA)
stderr	標準エラー、シリアル/ LCD (VGA)

現在 U-boot の ENV 値を検索し、 printenv コマンドを使用する：

```
[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
```

上記の ENV と゛tekkaman゛の ENV が表示されない。環境変数が設定しない場合、プリントアウトされない、ENV をカスタマイズ出来る、コマンド${ENV}で呼び出す。ENV も消去出来る。設定 ENV のコマンドは setenv、フォーマットは下記の通り：

setenv name value

第 1 個パラメータ name は環境変数の名称。

第 2 個パラメータ value は設定の値、第 2 個パラメータが無い場合、環境変数を消去する。

例：゛tekkaman゛パラメータ消去、再設定、最後のコマンド文字列で呼び出し。

```
[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!
```

設定または ENV 変更する時、メモリーに保存される。ENV のソリッド･ステート･メモリに保存したい場合、saveenv コマンドを使用する。

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

起動時、U-boot が` Warning - bad CRC, using default environment` をプリントアウトする場合、U-boot は ENV のソリッド･ステート･メモリで有効の ENV を検出出来ず、デフォルトの ENV を使用する。U-boot ENV を保存するソリッド･ステート･メモリのドライバが正常すると、 コマンド saveenv を実行し、システムの ENV を全部ソリッド･ステート･メモリに書き込み、再起動は警告がでない。

ENV はマルチソリッド･ステート･メモリをサポートする、 mini2440 では Nor Flash、Nand Flash または EEPROM をサポートする。include/configs の設定ファイルの定義に依存する。例：

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 ：メモリーのオフセットアドレス；
CONFIG_ENV_SIZE ：ENV を保存するパーティションのサイズ。
CONFIG_ENV_OFFSET と CONFIG_ENV_SIZE の設定、他のパーティションを書き換えないことを注意する。

4.2.3 シリアル伝送コマンド

コマンド：
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)
機能：異なるプロトコルを使ってシリアルで Host から伝送したファイルを取得。

フォーマット：
Load? [off] [baud]
第 1 個パラメータ off：ダウンロードする SDRAM のアドレス。指定しない場合はデフォルト設定を使用する：
CONFIG_SYS_LOAD_ADDR
第 2 個パラメータはボーレート、デフォルト値： 115200.

windows 上のハイパーターミナルはこのプロトコルでファイル伝送、 ubuntu では kermit プロトコルのみを使用する。C-kermit でファイルを mini2440 に伝送する。

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

U-boot の kermit 伝送プロトコルを起動し、Ctrl + ¥　→　c、 C-kermit のコマンドラインモードに入り、コマンド：send 〈ファイルパス〉、確認 (Enter)。

```
[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/desktop/) C-Kermit>send /home/tekkaman/development/share/zImage.img
```

C-kermit 伝送開始、伝送画面を表示し、伝送のプロセスも表示する。

```
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.
```

転送完了後、 c を入力 、U-boot のシリアル画面に戻る。

```
[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/ Desktop /) C-Kermit>send /home/tekkaman/development/share/zImage.img
(/home/tekkaman/ Desktop /) 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 ネットワークコマンド

U-boot のNIC ドライバをインストールし、ネットワークを介しファイルを開発ボードに伝送出来る。シリアルより速度が速い。クロスケーブルまたはネットワークケーブルを使用し開発ボードと PC を接続、ネットワークを設定とファイアウォールをクローズ。

ネットワークが正常かをテスト、開発ボードで ping コマンドを実行する:

```
[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
```

失敗の場合：

```
[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
```

失敗の原因は下記の通り：

1、U-boot ネットワークドライバ；

2、U-boot ネットワークプロトコル遅延設定；

3、ネットワークパラメータ設定、IP 設定等、Host と Target など。Host では IPv6 をクローズ。Wireshark で分析も出来る。

ネットワークに問題がないと、下記のコマンドで tftp サービスディレクトリまたは nfs サービスディレクトリからファイルを SDRAM にダウンロード出来る。

コマンド：

dhcp ： DHCP/TFTP プロトコルを使用しファイル取得

rarpboot ： RARP/TFTP プロトコルを使用しファイル取得

nfs ： NFS プロトコルを使用しファイル取得

tftpboot ： TFTP プロトコルを使用しファイル取得

bootp ： BOOTP/TFTP プロトコルを使用しファイル取得

フォーマットは：コマンド [ダウンロード先 SDRAM アドレス] [[ホスト IP:]ファイル名]

注：

dhcp、rarpboot または bootp を使用するには、ルーターまたは Host がこれらのプロトコルとサービスをサポートする必要となる。

[ダウンロード先 SDRAM アドレス]を入力しないと、システムはコンパイル時デフォルト定義の CONFIG_SYS_LOAD_ADDR を使用する。

tftpboot と nfs コマンドは[ホスト IP:]を定義しないと、ENV 中のデフォルト serverip を使用する。他のコマンドでは、[ホスト IP:]を定義する必要がある、でないと動的 IP サービスのホスト IP を使用する。

使用例で分析する、赤字に注意：

```
[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 操作コマンド

汎用 Nand Flash コマンドは下記の通り：

コマンド	機能

nand info	使用可能の Nand Flash
nand device [dev]	使用中の Nand Flash 表示または設定
nand read addr off size	Nand Flash 読み取りコマンド、Nand の off オフセットアドレス、 size バイトのデータを SDRAM の addr アドレスに読み取る。
nand write addr off size	Nand Flash プログラミングコマンド、 SDRAM の addr アドレスからの size バイトのデータを Nand の off オフセットアドレスにプログラミング。
nand write[.yaffs[1]] addr off size	yaffs イメージをプログラミング専用のコマンド、.yaffs1 for512+16 NAND
nand erase [clean] [off size]	Nand Flash 消しコマンド、Nand Flash の off オフセットアドレスからの size バイトのデータを消去
nand bad	Nand Flash のバッドブロック表示
nand dump[.oob] off	Nand Flash の中のデータ（16 進数）表示
nand scrub	全ブロックの Nand Flash のデータを消す、OOB も含む。そしてソフトウェアバッドブロック標識を消去。
nand markbad off	Nand の off オフセットアドレスのブロックをバッドブロックと標識する

使用例：

```
[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 コマンドも１つの Nand Flash 読み取りコマンドであり、 Nand Flash の offset オフセットアドレスのカーネルイメージを SDRAM の loadAddr 位置に読み取る。そして自動的にカーネルイメージ（mkimage 処理後）の終了の事を読み取る、従って読み取りのサイズを表示しない。

フォーマット：nboot　loadAddr　dev　offset

使用例：

```
[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 メモリー/レジスタ操作コマンド

nm　　メモリー値変更(指定アドレス)
フォーマット：　nm [.b, .w, .l] address

mm　　メモリー値変更(アドレス自動的＋1)
フォーマット：　 mm [.b, .w, .l] address

md　　メモリー値表示
フォーマット：　 md [.b, .w, .l] address [# of objects]

mw　　指定のデータでメモリー充填
フォーマット：　 mw [.b, .w, .l] address value [count]

cp　　メモリーのコピー（メモリーと Nor Flash 間のデータコピーを含む）
フォーマット：cp [.b, .w, .l] source target count

メモリー値を検索/変更のコマンド、SDRAM とレジスタ値を検索/変更。
[.b, .w, .l]は検索と変更フォーマット：bit、word、long
使用例：

```
[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   ................
```

LED 関連の GPIO レジスタのデータレジスタ値を変更すると、LED の点灯を制御出来る。
LED1 点滅の使用例（消灯→点灯）：チップデータマニュアルと mini2440 の回路図を参照）

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

4.2.7 Nor Flash コマンド

Nor Flash のコマンドは Nor Flash にデータプログラミングに使用する。
flinfo　Flash メモリーの情報プリントアウト、全ての Sector をリストする。
flinfo　N 指定 Flash メモリーの N Block の情報をプリントアウト。(複数ブロック Nor Flash の時に使用する)

```
[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
```

(RO)付きのは Sector が書き込み保護される。

Nor Flash の読み取りインタフェースは SDRAM と同じ、Nor Flash の読み取りも md コマンドを使用する：

```
[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   ................
```

Nor Flash のプログラミングタイミングと SDRAM の書き込みと違って、Nor Flash プログラミングは mm 等のコマンドで消去出来ない、従って cp コマンドでメモリーを Nor Flash にコピーし、プロテクトを解除し、消去する必要がある。コマンドは下記の通り：

protect ：Flash 書き込み保護、有効/無効設定出来る。

フォーマット：

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 個パラメータ on 書き込み保護オン；off 書き込み保護オフ
第 2 、3 パラメータは Flash 書き込み保護範囲設定
start end は開始アドレスと終了アドレス範囲定義、start は消去ブロックの開始アドレス；end は消去ブロックの終了アドレス。
例：消去 Sector 2 と Sector 3 エリアのコマンドは erase 20000 3ffff 。
start +end は開始アドレスと操作バイト数の定義範囲の方法はよく使われる。start は消去ブロックの開始アドレス；end は消去のバイト数。
例：消去 Sector 2 と Sector 3 エリアのコマンドは erase 20000 +20000
N:SF[-SL]はグループとセクター、N は Flash の Block 番号、SF は消去する開始 Sector 番号、SL は消去する終了 Sector 番号。
例： Block1 の Sector 2 と Sector 3 セクターを消去するコマンド： erase 1:2-3。
bank N は全 Block 消去、Block の番号が N の全 Flash を消去。
all は全ての Flash の消去。

注：Nor Flash を消去する最小単位は Sector、即ち 0x10000 バイト、定義するサイズは＜1 Sector または Sector 範囲を超える場合、定義した Sector は全て消去される。

erase ： Flash を消去 するコマンド
フォーマット：
erase start end
erase start +end
erase N:SF[-SL]
erase bank N
erase all
パラメータは消去される Flash を指定する範囲、書き込み保護と同じ設定する。
使用例は mini2440 の Nor Flash の Sector 16 書き込み保護オン/オフし、データ消去、最後開始の 20 バイトを Sector 16 にコピーする。

```
[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 操作コマンド

コマンド	機能
usb reset	USB コントローラーを初期化
usb stop [f]	USB コントローラーをクローズ
usb tree	接続された USB デバイスツリー
usb info [dev]	USB デバイス[dev]の情報を表示
usb storage	接続された USB ストレージデバイスを表示
usb dev [dev]	現在 USB ストレージデバイス表示と設定

usb part [dev]	USB ストレージデバイス[dev]のパーティション情報を表示
usb read addr blk# cnt	USB ストレージデバイスデータを読み取む

4G の kingstonU（ブートローダー）を mini2440 に挿し込み、ヘッダーの 512 バイトを読み取り（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.
```

全てのコマンドを使用する前、USB デバイスを挿し込み、次：usb reset、USB コントローラー初期化、デバイス情報取得。

4.2.9 SD カード(MMC)コマンド

```
[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 カードのコマンドは初期化とデバイス情報表示、読み取り/書き込み、ファイルシステムコマンドで実現できる。

使い方は USB と類似で、全てのコマンドを使用する前、 SD カードを挿し込み、mmc init、MMC コントローラー初期化、デバイス情報取得。

mini2440 に 1GB SD カードを挿し込み：

```
[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 ファイルシステムコマンド

fatinfo：ファイルシステムの関連情報表示
フォーマット：fatinfo <interface> <dev[:part]>
Interface：インタフェース、例 usb、mmc；
dev：デバイスコードナンバー、例 0、1……；
part：ストレージデバイス中のパーティション、例 1、2……。

fatload： FAT32 ファイルシステムからバイナリファイルを SDRAM に読み取る。
フォーマット：fatload <interface> <dev[:part]> <addr> <filename> [bytes]
Interface、dev と part 上記と同じ；
addr： SDRAM に書き込む アドレス；
filename：ストレージデバイス中のファイル名；
bytes：ストレージデバイスから読み取ったファイルのサイズ設定、指定しなくても良い；
fatls： FAT32 ファイルシステム中ディレクトリ内のファイルをプリントアウト。
フォーマット：fatls <interface> <dev[:part]> [directory]
Interface、dev と part 上記と同じ；
directoryr：検索するディレクトリ、指定しなくても良い、デフォルトは/。

上記のコマンドは USB メモリまたは SD カードと同時に使用する。モバイルメモリの FAT32 パーティションを読み取る。

使用例：

```
[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 システムブートコマンド

boot と bootd は ENV` bootcmd` で指定されるコマンドを実行する。
bootm コマンドは SDRAM 中、U-boot の mkimage ツールで処理されたカーネルイメージを起動する。
フォーマット：bootm [addr [arg ...]]
addr ：カーネルイメージのある SDRAM 中のアドレス
Linux カーネルを起動する時、'arg' の initrd のアドレス。
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 他のコマンド

run　- ENV 定義のコマンドスクリプトを実行する。
使用例：

```
[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　- CPU 再起動
他のコマンド、`help`説明を検索出来る。

4.3　ダウンロードとプログラミング

U-boot でイメージファイルをボードの Flash にプログラミング手順は下記の通り：
(1) ネットワーク、シリアル、USB メモリ USB メモリ、SD カード等を通じて、ファイルを SDRAM に伝送する；
(2) Nand Flash または Nor Flash 関連の読み取り/書き込みコマンドを使用し、 SDRAM 中のデータを Flash にプログラミングする。

SD カードと USB メモリ USB メモリから U-boot を更新するには、SD カードと USB メモリ中の FAT32 ファイルシステムと u-boot.bin ファイルが必要とされる。

4.3.1 SD カードで Nand　Flash プログラミング：

```
[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 USB メモリで Nor　Flash プログラミング：

```
[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 TFTP サービスで Nand Flash プログラミング：

```
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 NFS サービスで Nand Flash プログラミング：

```
[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 カーネルブート

カーネルのブート手順：

（1） U-boot の mkimage ツールでカーネルイメージ zImage を作成する。

（2）ネットワーク、シリアル、USB メモリ、SD カード等でカーネルイメージを SDRAM の指定位置に伝送（0x30008000）

（3）｀bootm｀等カーネルブートコマンドでカーネル起動。

U-boot の mkimage ツールでカーネルイメージ zImage を作成する理由は？

bootm コマンドでカーネルをブートする時、bootm は１つ 64 バイトのファイルヘッダーを読み取り、カーネルイメージの CPU 構造、OS、メモリー中のロード位置、メモリー中入口ポイントの位置、イメージ名などの情報を取得する。その後、bootm は OS に起動環境を設定出来、カーネルイメージの入口ポイントへジャンプする。そして mkimage はファイルヘッダーを追加する専用ツールとなる。詳細は U-boot 中 bootm のソースコードと mkimage のソースコードを参照。

mkimage ツールの使用：

パラメータ説明：

-A CPU の

CPU アーキテクチャを指定、指定可能値は：alpha、arm 、x86、ia64、mips、mips64、ppc 、s390、sh、sparc 、sparc64、m68k 等

-O OS タイプ指定、指定可能値は：openbsd、netbsd、freebsd、4_4bsd、linux、svr4、esix、solaris、irix、sco、dell、ncr、lynxos、vxworks、psos、qnx、u-boot、rtems、artos

-T イメージタイプ指定、指定可能値は：standalone、kernel、ramdisk、multi、firmware、script、filesystem

-C イメージ圧縮タイプ指定、指定可能値は：

none 圧縮しない(デフォルト、zImage は既に bzip2 で処理された)

gzip の圧縮方式

bzip2 の圧縮方式

-a イメージはメモリー中のロードアドレス、イメージをメモリーにダウンロードする時、 mkimage でイメージ作成する時、パラメータが指定するアドレスにダウンロードする

-e イメージ実行の入口ポイントアドレス指定、-a パラメータ指定の値＋ 0x40（mkimage 追加の 0x40 個バイトのヘッダー）

-n イメージ名指定

-d 制作イメージ作成のソース元ファイルを指定

カーネルイメージ作成のコマンド例：

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

4.4.1 SD カードでカーネルブート：

SD カード中 FAT32 ファイルシステム、作成済みのカーネルイメージファイル。

```
[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 TFTP サービスでカーネルブート

```
[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 NFS サービスでカーネルブート：

```
[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)
```

4.4.4 Nand Flash で カーネルブート：

作成済みのカーネルイメージファイルを Nand Flash の指定位置にプログラミング（カーネルパーティションテーブルで決められる）。その後、起動時 Nand Flash の読み取りコマンドでカーネルイメージファイルをメモリーの指定位置に読み取り（カーネルイメージ作成時の-a パラメータで決められる）、次は bootm コマンドでカーネルブートする。

カーネルイメージファイルのプログラミング：

```
[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
```

カーネルブート:

```
[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　ソースコード分析

今回移植に使用する U-boot-2009.11。
ソースコードディレクトリ構造、とコードの実行順番でソースコードを分析する。

5.1 U-boot　ソースコード全体フレーム枠

ソースコード解凍後、ファイルとフォルダは下記の通り：

cpu	プロセッサ関連のファイル。各サブディレクトリに cpu.c と interrupt.c、start.S、u-boot.lds などを含む。 cpu.c　CPU 初期化、設定コマンド Cache とデータ Cache 等 interrupt.c 設定システムの各種割り込みと異常 start.S は U-boot 起動時実行の一番目ファイルで最初のシステム初期化とコードのロケーションとシステムスタックを行われる。このファイルは U-boot の第二段階の C プログラムに基礎を築く。 u-boot.lds リンクスクリプトファイル、コードの最後の組み立てである
board	サポートする開発ボードの関連ファイル、 SDRAM 初期化コード、Flash ドライバ、ボード初期化ファイルなどを含む。 その中の config.mk ファイルは TEXT_BASE を定義する、即ちコード

	がメモリーの開始アドレスとなり、非常に重要。
common	プロセッサ アーキテクチャ関連しないの汎用コード、U-boot のコマンド解析コード/common/command.c、全てのコマンドの上層コード cmd_*.c、U-boot 環境変数処理コード env_*.c、等は本ディレクトリ下に保存する
drivers	外部チップのドライバ、NIC、USB、シリアル、LCD、Nand Flash など
disk fs net	CPU をサポートするサブシステム： デイスクドライバのパーティション処理コード ファイルシステム：FAT、JFFS2、EXT2 等 ネットワークプロトコル：NFS、TFTP、RARP、DHCP など
include	ヘッダーファイル、各 CPU のレジスタ定義、ファイルシステム、ネットワークなどを含む configs サブディレクトリのファイルはターゲットボード関連の設定ヘッダーファイル
doc	U-Boot の説明ファイル、設定ファイル変更の時に使用する
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	プロセッサアーキテクチャ関連の初期化ファイル board.c ファイルでは殆どの U-boot の構築は第二段階コード入口関数と関連初期化関数保存する。
api examples	外部拡張アプリプログラムの API と使用例
nand_spl onenand_ipl post	特別の構築必要な起動コードと自己検査プログラムコード
libfdt	平坦化されたデバイスツリー(flattened device trees)のライブラリファイル
tools	コンパイル S-Record または U-Boot イメージ等の関連ツール、bootm ブート作成のカーネルイメージファイルツール mkimage ソースコード
Makefile MAKEALL config.mk rules.mk mkconfig	コンパイルプロセス制御する Makefile ファイルとルールファイル
CHANGELOG CHANGELOG-before-U-Boot-1.1.5 COPYING CREDITS MAINTAINERS README	説明ファイル、バージョン、著作権説明など

赤字は移植する時の重要ファイル/フォルダである。

5.2 U-boot コード実行プロセス（ S3C24x0 例）

リンクスクリプトファイル u-boot.lds でコードの開始アドレス：

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

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

プログラムの入口ポイントは_start、ロケーションは cpu/arm920t/start.S（即 u-boot 起動の第一段階）。start.S を分析する。（データマニュアルのソースコードを参照）

```
#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
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:
```

//¥include ディレクトリで他のヘッダーファイルを含む

//場所 ¥include¥linux ディレクトリ

u-boot のメイン入口、 start_code へジャンプ

ジャンプベクトルとチップの体系構造 to

ldr は第二個操作数（例：_undefined_instruction）指向のアドレスデータを PC へ伝送

.word は 1 つの 4 バイトのスペースを定義し ndefined_instruction はアドレス、 即後ろの標識の全てのオフセットアドレスデータ

16 バイトとアラインメントし、0xdeadbeef を入れる、Magic number である。

```
        .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
#if               defined(CONFIG_AT91RM9200DK) || defined(CONFIG_AT91
RM9200EK)
        /*
         * 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 b
ase addresses */
#  define CLKDIVN  0x14800014              /* clock divisor register
 */
#else
#  define pWTCON  0x53000000
#  define INTMSK    0x4A000008              /* Interupt-Controller ba
se addresses */
#  define INTSUBMSK               0x4A00001C
#  define CLKDIVN  0x4C000014              /* clock divisor register
*/
```

前のと同じ、1 つの 4 バイトのスペースを作成、アドレスを保存する

コード実行する！！
システムを SVC に入る
（Administrator モード）
AT91RM9200 用

システムクロックのレジスタアドレス定義

```
# endif

        ldr             r0, =pWTCON                                     クローズウォッチドッグ
        mov           r1, #0x0
        str             r1, [r0]

        /*                                                              全ての割り込みクローズ
         * 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]                                        クロックの分周比設定
# 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 */

        /*                                                              cpu_init_crit にシャンプー、システムの初期化関数で、board/*/lowlevel_init.S の lowlevel_init 関数を呼び出す。システムパスの初期化、ストレージのビットワイド、速度、リフレッシュなどのパラメータ設定。初期化完了後、Nor Flash、SDRAM がシステムに識別され使用される。下記のコードリダイレクトはこの関数に寄る
         * 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                                            コードリダイレクト、先ず自体がメモリーの位置を検索：ある場合、直接スタック初期化コード stack_setup へジャンプ。存在しない場合自体を自動的に Nor Flash からメモリーへコピーする。
          /* 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]
*/                                                                      自体コピーループ
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 */

        /* Set up the stack

                                                                        */
stack_setup:
        ldr             r0, _TEXT_BASE      /* upper 128 KiB: relocate         スタック初期化コード（第二段階の C 言語のための事前準備）
        d uboot    */
        sub             r0, r0, #CONFIG_SYS_MALLOC_LEN    /* malloc are
        a               */
        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 */
…...
```

BSS セグメントクリア（第二段階の C 言語のための事前準備）
BSS セグメント（bss segment）プログラム中未初期化のグローバル変数のブロックメモリーエリアである。BSS は英語の Block Started by Symbol の略称。BSS セグメントは静的メモリー割り当て。コンパイルの時、コンパイラはスペースを事前に割り当て、値は 0、スペース節約のため、bin または ELF ファイルはスペースを消耗しない。コンパイラは_bss_start と_bss_end の値を計算する（定義ではなく）

第二段階の C 言語コード入口 _start_armboot ジャンプ（メモリーにリダイレクト）

cpu_init_crit システム初期化関数動作 CP15 プロセッサ、

board/*/lowlevel_init.S 中の lowlevel_init 関数を呼び出し、システムパスの初期化、接続ストレージのビットワイド、速度、リフレッシュなどのパラメータ。初期化後、SDRAM はシステムに識別され、使用出来る。
次のコードは省略される。

次は lib_arm/board.c 中の第二段階入口関数 start_armboot ：

```
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;

        for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc

      _ptr) {
                if ((*init_fnc_ptr)() != 0) {
                        hang ();
                }
        }

        /* armboot_start is defined in the board-specific linker scri

      pt */
        mem_malloc_init (_armboot_start - CONFIG_SYS_MALLOC_LEN,
                         CONFIG_SYS_MALLOC_LEN);

#ifndef CONFIG_SYS_NO_FLASH
        /* configure available FLASH banks */
        display_flash_config (flash_init ());
#endif /* CONFIG_SYS_NO_FLASH */

#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) {
#               ifndef PAGE_SIZE
#                 define PAGE_SIZE 4096
```

gd_t と bd_t データ構造はとても重要
1 つのグローバルデータを保存するエリアを作成、アドレスポインタを gd に付けられる。
グローバルデータのエリアクリア。 gd->bd（ポインタ）値を割り当て（ gd の前）クリア
gd->flags 割り当て、リダイレクト（メモリー中）
monitor_flash_len は u-boot コード長。

初期化ループ：
init_sequence 初期化関数集の関数ポインタデータグループ
その中の任意関数失敗すると、無限ループに入る。（重要）

スタックスペース初期化、クリア。

Nor Flash 関連パラメータ初期化 、サイズ表示。

VFD ストレージエリア初期化（LCD 関連表示）

LCD メモリー初期化

#endif
/*
* reserve memory for LCD display (always full pages)
*/
/* bss_end is defined in the board-specific linker scri
pt */
addr = (_bss_end + (PAGE_SIZE - 1)) & ~(PAGE_SIZE - 1);
lcd_setmem (addr);
gd->fb_base = addr;
}
#endif /* CONFIG_LCD */	

#if defined(CONFIG_CMD_NAND) puts (" NAND: "); nand_init(); /* go init the NAND * / #endif	Nand Flash コントローラー初期化、容量も表示する。

#if defined(CONFIG_CMD_ONENAND)
onenand_init();
#endif
	OneNand 初期化

#ifdef CONFIG_HAS_DATAFLASH
AT91F_DataflashInit();
dataflash_print_info();
#endif
	DataFlash 初期化

/* initialize environment */ env_relocate ();	環境変数初期化、 メモリーにないものはデフォルト値を使用しプリントアウトする：" *** Warning - bad CRC, using default environment"。
#ifdef CONFIG_VFD /* must do this after the framebuffer is allocated */ drv_vfd_init(); #endif /* CONFIG_VFD */	VFD (LCD 関連表示) 初期化

#ifdef CONFIG_SERIAL_MULTI
serial_initialize();
#endif
	シリアル初期化

/* IP Address */ gd->bd->bi_ip_addr = getenv_IPaddr ("ipaddr"); stdio_init (); /* get the devices list going. */ jumptable_init ();	環境変数から IP アドレス取得、標準入出力デバイス初期化：シリアル、LCD、键盘など 初期化グローバルデータ表中のジャンプ表 gd->jt。 ジャンプ表は 1 つの関数ポインタデータグループで、u-boot 中基本関数ライブラリを定義する、gd->jt はこの関数ポインタデータグループの開始ポインタ。
#if defined(CONFIG_API) /* Initialize API */ api_init (); #endif	API 初期化、U-boot で" アプリプログラム" 編集に使用する
console_init_r (); /* fully init console as a device */	console 初期化、プラットフォームと関係なく、標準出力を vga に設定すると、LCD に文字が表示する。
#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 ();	プラットフォーム関連の他プラットフォーム初期化

#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
	割り込み有効（よく使用しない、多数のプラットフォームで関数は空き）
	
	TI チップ中の内蔵 MAC 初期化（プラットフォーム関連）
	
	NIC チップ初期化（プラットフォーム関連）
	
	bootfile パラメータ
	
	ボード初期化（一部ある）
	
	SD カード/MMC コントローラー初期化
	
	MII 関連初期化
	
	NIC 初期化
	
	メインループに入る、bootdelay と bootcmd 読み取り、 bootdelay 時

main_loop (); } /* NOTREACHED - no way out of command loop except booting */ }	间内ボタンを押し、コマンドライブラリに入る、タイムアウトは bootcmd コマンドに入る。

赤字は重要。

第 6 章 U-boot を mini2440 に移植

U-Boot は S3C2440 をサポートしないため、移植は S3C2410 のファイルに基づき行う。従って、移植は S3C2440 と S3C2410、及び S3C2410 と mini2440 開発ボードの外部デバイスの違いに対し、適当の変更を行い、新機能も追加する。

移植する前、 S3C2440 と S3C2410 の特性、違いも理解する必要がある。移植プロセスではチップと関連する、特に Nandboot の原理と Norboot の内部 ram マッチング原理など。

S3C2440 と S3C2410 の違いは 2440 のメイン周波数は高い、インタフェースでは、カメラインタフェースと AC‘97 オーディオインタフェースを追加し；レジスタでは、新しいモジュールレジスタの他、 NAND FlASH コントローラーのレジスタ、チップクロック周波数制御レジスタも変更がある、他のレジスタはほぼ同じ。

下記の変更は大部分がパッチの形式で移植する：

```
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
```

开发板设定オプション説明:

arm	CPU の構築(ARCH)

arm920t	CPU のタイプ(CPU)、cpu/arm920t サブディレクトリ対応
tekkamanninja	開発者/またはディーラー(vender)、 board/tekkamanninja ディレクトリ対応
mini2440	開発ボードのモデル(BOARD)、board/tekkamanninja/mini2440 ディレクトリ対応
s3c24x0	チップ上のシステム(SOC)定義

6.1.2 /board でmini2440 ディレクトリとファイルを作成

/board ディレクトリで開発ボード mini2440 のディレクトリを作成し、s3c2410x のファイルをここにコピーする、適当な変更を行う。目的： s3c2410x に基づき、移植の難易度を下げる。

ボードの vender に tekkamanninja を設定した 、開発ボード mini2440 ディレクトリは/board サブディレクトリ中の tekkamanninja ディレクトリに作成する必要がある 、そうしないと、コンパイルは失敗する。

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

mini2440 ディレクトリの Makefile ファイル変更（後も内容を変更する）：

```
@@ -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 include/configs/で開発ボード設定ファイルを作成

sbc2410x と mini2440 が一番類似するため、sbc2410x の設定に基づき変更する。

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

6.1.4 コンパイル環境をテスト

U-boot ソースコードのルートディレクトリ：

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

可能なエラー：

(1) 設定エラー

```
make mini2440_config
Makefile:????: ***      区切り記号漏れ 。 停止。
```

U-boot のルートディレクトリの Makefile の｀ @$(MKCONFIG) $(@:_config=) arm arm920t mini2440 tekkamanninja s3c24x0 ｀ 前で｀ Tab｀ ボタンを追加、Makefile のルール：全てのコマンドは｀ Tab｀ で開始する。

(2) コンパイル時に下記のエラーが出る（コンパイラの問題、無視）：

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

修正：

```
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
 # ==========================================================================
```

テストが正常に終了後、コンパイル環境と基本の開発ボードのコードの作成は成功である。現在コンパイルのは S3C2410 に基づき、次はコードの実行プロセスで mini2440 用に変更する。

6.2 第一段階:起動コード

/cpu/arm920t/start.S に入る

6.2.1 AT9200 用の LED ジャンプをクローズ

```
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 CPU 周波数初期化設定変更

2410 と 2440 異なるのは、PLL の初期化パラメータ、データマニュアルで検索出来る。先ず周波数を 405MHz に設定する。割り込みマスクコードの修正も含む。

```
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 lowlevel_init.S ファイル変更

mini2440 用のストレージ設定（パス接続の Nor Flash と SDRAM)、lowlevel_init.S ファイルを変更する。接続する Nor Flash のビット数と関連する。SDRAM のパラメータはチップマニュアルで検索出来る。64MB のメモリーを 128MB に変更する例もある、そのパラメータはここで変更できる、詳細は MINI2440: Auto probe for SDRAM size を参照 。

64MB メモリーのパラメータ変更：

```
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 +0
800
@@ -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:
```

lowlevel_init.S には 1 つの bug がある、直接 OpenJTAG を使用してメモリーにダウンロードできない、直接実行できないため下記のように修正：

```
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 コードリダイレクト部分変更

Tekkaman Ninja はバージョン 2009.08 から起動時の SDRAM 検索機能（OpenJTAG でロード）、及びチップ Norboot/Nandboot でコードリダイレクトのモデルを決定する判断機能追加した、従ってコンパイルした bin ファイルは同時に Nand Flash と Nor flash をプログラミングでき、そして OpenJTAG にロード/デバッグ出来る。

```
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
+   strbr2, [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 totarget 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:
```

上記の追加コードに 1 つのジャンプを追加した：bl nand_read_ll 、ジャンプは追加する C 言語ファイル（board/tekkamanninja/mini2440/nand_read.c）中の関数、元は vivi のコードを使用し、openmoko で変更した後、マルチ Nand Flash チップをサポートできる。従って、幾つのチップ ID を追加、mini2440 の Nand Flash をサポート。コードは下記の通り：

```
/*
 * 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)
#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)
{
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 a
        ddr)
        {
                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)
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;
}
```

ファイル追加後、Makefile にこのファイルのコンパイルを追加する。

```
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 LED の点灯動作

コードのプロセスと Debug 段階を表示する。コードは第二段階コード start_armboot 関数にジャンプする前、LED が点灯。

```
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

 /*
```

ここまで、起動の第一段階は変更完了。しかし U-boot-1.3.3 以後で、これら bin ファイル前の 4K コードは自動的に後ろに変更したため、起動失敗となる。従って、手動でリンクを使用する.lds ファイルへ変更する、ここでのコードは bin ファイルの一番前に置く：

```
diff -aurNp u-boot-2009.11/cpu/arm920t/u-boot.lds u-boot-2009.11_tekkaman/cpu/arm920t/u-bo
ot.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 第二段階:初期化コード変更

第二段階コード lib_arm/board.c 中の start_armboot 関数を実行、システムの初期化を開始する。

6.3.1 lib_arm/board.c ファイル変更

ファイルの変更内容は AT9200 用のコードをクローズ、LED 点灯コードを追加、console 初期化後、コマンドラインを入る前に、LED（3、4）を点灯する(第二個の LED の点灯コードは board_init 関数中)、情報プリントアウト（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 ();
```

lib_arm/board.c 中の start_armboot 関数は複数の初期化関数を呼び出した、これらの関数は各ファイルに存在する、後で変更する。

6.3.2 board/tekkamanninja/mini2440/mini2440.c ファイル変更

本ファイルはボード初期化機能である、変更は： LCD 初期化関数、GPIO 設定変更（開発ボードの外部デバイス接続と関連する、LCD と LED など）、LED の点灯（第二個）、使用済みの Nand コントローラー初期化コードをシールド、NIC チップ（DM9000）の初期化関数の追加など。

```
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
```

次は各サブシステムの初期化と機能コードの変更。

6.4　第三段階：ターゲットボード外部デバイスドライバ修正

6.4.1　Nand Flash 関連コードの変更

U-boot 起動の第一段階で Nand Flash コントローラーを初期化した。第二段階で start_armboot 関数はまた Nand Flash コントローラーを初期化した。原因は第二段階と第一段階のコードは基本的に独立して、第一段階のコードはリロケーション機能で、第二段階は U-boot の実行プロセス、従って、以前の初期化も繰り返し実行される。例えば Nand Flash コントローラー、CPU 周波数の初期化など。

S3C2440 と S3C2410 の間の最大の違いは：S3C2410 の Nand Flash コントローラーは 512B+16B の Nand Flash しかサポート出来ない； S3C2440 はまた 2KB+64B の大容量 Nand Flash をサポート出来る。従って、Nand Flash コントローラーのレジスタと制御プロセス、ドライバコードも変更する必要がある。

詳細はチップデータマニュアルに参照、Nand Flash ドライバコードの変更は下記の通り：

```
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 17:16:12.00000

```
0000
+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 Yaffs(2)イメージプログラミング機能追加

Nand Flash ドライバ変更後、次は関連の Yaffs(2)イメージプログラミングコードを変更する。Linux では Yaffs(2)をデータ保存のファイルシステム、或いはルートファイルシステムとして使用する。BootLoader 下で Yaffs(2) イメージファイルをプログラミングするのは必要とする。

Yaffs(2)イメージプログラミングのサポートはプログラミング段階で、データを書き込むと同時に、イメージファイル中の oob データを Nand Flash の Spare エリアに書き込む。これは Yaffs ファイルシステム原理と Nand Flash の構造と関連する、詳細は関連資料に参照。

変更する 4 つのファイルのパッチ：

```
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 +0
800
@@ -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 +0
800
@@ -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/mt
```

```
d/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 Nor Flash 書き込む機能のコード変更

S3C2440 と S3C2410 は Nor Flash との接続は同じであるが、S3C2410 が AMD の Nor Flash チップ、mini2440 は SST の Nor Flash を使用する。この 2 つのチップは書き込み時に使用するブロックのサイズ、タイミングとコマンドコードの差異で、チップのデータマニュアルに基づき変更する必要がある。詳細はデータマニュアルに参照する：

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	Bus Cycles (Notes 2–5) First Addr	First Data	Second Addr	Second Data	Third Addr	Third Data	Fourth Addr	Fourth Data	Fifth Addr	Fifth Data	Sixth Addr	Sixth 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							
Autoselect (Note 8) Device ID, Top Boot Block	Word	4	555	AA	2AA	55	555	90	X01	22C4				
	Byte		AAA		555		AAA		X02	C4				
Autoselect (Note 8) Device ID, Bottom Boot Block	Word	4	555	AA	2AA	55	555	90	X01	2249				
	Byte		AAA		555		AAA		X02	49				
Autoselect (Note 8) 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										

上記の部分以外、SST39VF1601 の SECTOR 毎 のサイズは同じ、Am29LV160 では開始のブロックは小さい。
変更する部分は board/tekkamanninja/mini2440/flash.c

変更例：

```
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;
}

+#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_PROAM;
+//     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 ネットワーク関連コード変更

以前の U-boot はネットワークの遅延設定に合わせないため、変更する必要がある。現在の U-boot バージョンの ネットワーク部分では DM9000 のドライバと NFS の TIMEOUT パラメータを修正する：

DM9000 のドライバで一部のコードをシールド：

```
diff -aurNp u-boot-2009.11/drivers/net/dm9000x.c u-boot-2009.11_tekkaman/drivers/net/dm900
0x.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 " );
```

NFS では遅延を追加、or" *** ERROR: Cannot mount" のエラーが発生する。

```
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 シリアル Xmodem 伝送プロトコル追加（選択可）

シリアルでデータをメモリーに伝送する時、Xmodem プロトコルを使用する。

```
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 LCD 表示機能追加

LCD のサポートはOpenmoko のコード移植を参照する。Openmoko の GTA2 はS3C2442 の CPUを使用する。LCD コントローラーは同じ。 GTA2 は U-boot の場合 LCD で文字を表示出来る、そしてソフトウェア層分けの U-boot にとっては、ドライバを移植し、初期化パラメータを簡単修正すれば、LCD で console を表示出来る。

そして、LCDがプログラムで識別出来ない場合、Nand Flash 設定でパラメータをインポートし、識別する必要がある。(64MB Nand Flash は NEC のボード、他のはTPOのを使用する)。

本機能の移植は５つのファイルを変更する (drivers/video/Makefile、board/tekkamanninja/mini2440/mini2440.c ファイル)、/drivers/video/で１つのドライバファイル s3c2410_fb.c を追加する。

```
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 +0
800
@@ -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/Ma
kefile
--- 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 +0
```

```
800
@@ -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 +0
800
@@ -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) 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;
//    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;
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 SD カード（MMC）読み取り機能追加

SD カードのサポートは buserror の Git コードレジストリのソースコードを参照し mini2440 に移植する。使用するコードも Openmoko の GTA2 ソースコード。GTA2 は U-boot で SD カードを使用しシステムをアップデート出来る。SD カードドライバコードを移植/修正する。

変更ファイルは 5 つある、3 つのドライバコードファイルを追加する。

```
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:
+   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 -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 -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 +0
        800
        @@ -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 -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
```

追加の 3 個ドライバコードファイル：

/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>
#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 */
}

        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
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;
}
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;
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;
}
/* 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,

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` );

/* 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);
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
#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)
#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 第四段階：設定ファイル修正

6.5.1 CONFIG_S3C2440 条件定義を追加

S3C2440 の多数のコードは S3C2410 に基づくため、従って、条件コンパイルの中で CONFIG_S3C2410 がある所に CONFIG_S3C2440 を追加する必要がある、そうしないとコンパイル出来ない。簡単な方法はコードで全ての CONFIG_S3C2410 を検索し、状況により変更する。一部の所は CONFIG_S3C2440 だけでなく、2 つのチップのディストリビューションにより変更する。U-boot-2009.11 の変更例は下記の通り：

```
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 -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 -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 +0
800
@@ -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 -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 +0
800
@@ -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/s
3c24x0_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/s
3c24x0_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 +0
800
@@ -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/hos
t/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 +0
800
@@ -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;
```

cpu/arm920t/s3c24x0/speed.c ファイルで、 S3C2440 と S3C2410 の差異に基づき変更する：

```
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 +0
800
@@ -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) || defin
ed
(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 ファイルはレジスタの定義を保存する。2 つのチップの違いは Nand 、USB と SD インタフェース等。

```
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;
-#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
+#else
+       S3C24X0_REG32         SDIDAT;
+#endif
+#else
+       S3C24X0_REG32         SDIDAT;
+#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 設定ファイル include/configs/mini2440.h 変更

変更：

(1) CS8900 NIC 定義削除、DM9000 追加。

(2) JFFS2、FAT ファイルシステム有効。

(3) USB、SD カード機能有効。

(5) I2C、EEPROM 機能有効。

(6) LCD 機能、BMP 表示と文字 console の機能有効。

(7) AMD の Nor Flash チップの定義、SST Nor Flash チップ定義削除。

```
--- 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 +0
800
@@ -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 p=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 */
```

6.6 再コンパイル/テスト

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

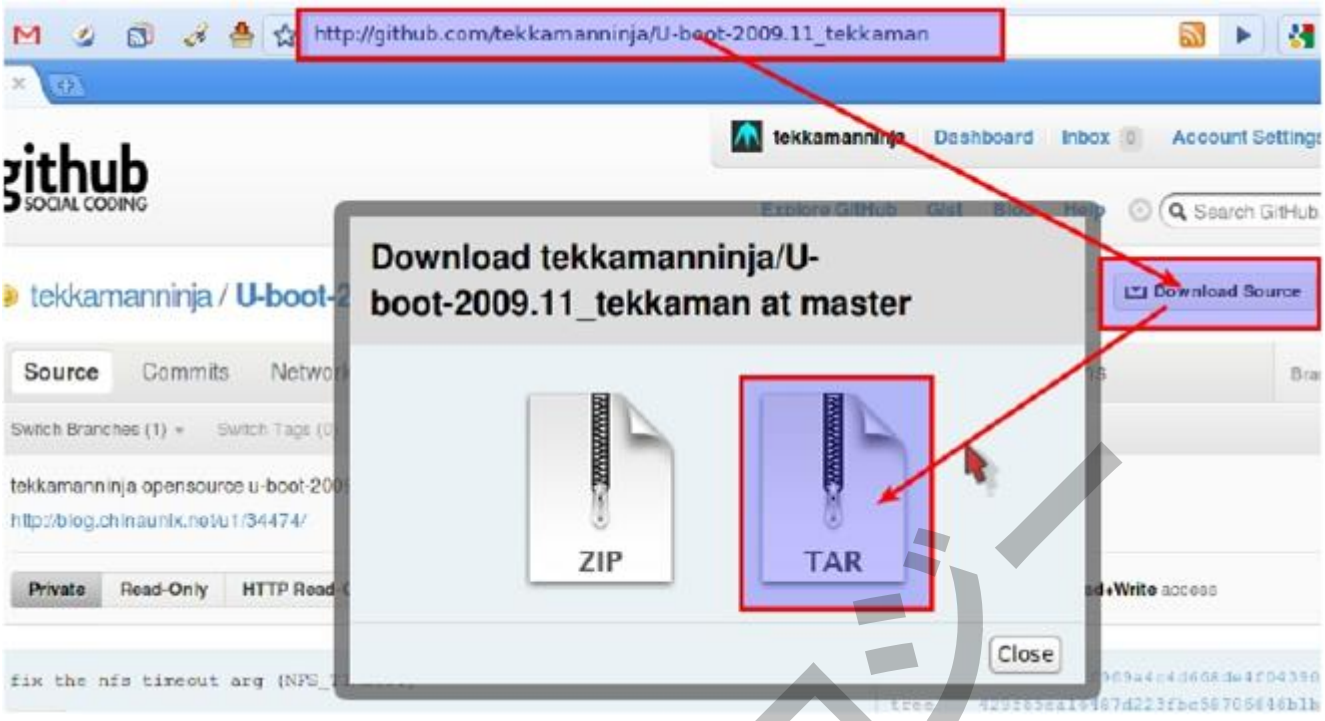
コンパイル完了後、ボードにプログラミング、再起動。
ブザーが鳴らすと、LCD 起動、DENX の logo を表示、シリアルから伝送：

```
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]#
```

ここまで、移植は成功する！！！

第 7 章 最新のソースコードダウンロード

移植に問題がある場合、Tekkaman Ninja のソースコードに参考出来る。ソースコードのダウンロード先：


http://github.com/tekkamanninja/U-boot-2009.11_tekkaman
github
SOCIAL CODING
tekkamanninja
Dashboard
Inbox
Account Settings
Search GitHub
tekkamanninja / U-boot-2
Download Source
Source
Commits
Switch Branches (1)
tekkamanninja opensource u-boot-200
http://blog.chinaunix.net/u1/34474/
Private
Read-Only
Download tekkamanninja/U-boot-2009.11_tekkaman at master
ZIP
TAR
Close
fix the nfs timeout arg (NFS_T