Software Manual

Version 1.0.15

2007 年 10 月 19 日

株式会社アットマークテクノ
http://www.atmark-techno.com/

Armadillo公式サイト
http://armadillo.atmark-techno.com/

目次

表目次

図目次

1.はじめに

1.1. マニュアルについて

本マニュアルは、Armadillo-9 を使用する上で必要な情報のうち、以下の点について記載されています。

- フラッシュメモリの書き換え
- 基本的な使い方
- カーネルとユーザーランドのビルド
- アプリケーション開発

Armadillo-9 の機能を最大限に引き出すために、ご活用いただければ幸いです。

1.2. フォントについて

このマニュアルでは以下のようにフォントを使っています。

表 1-1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列

1.3. コマンド入力例の表記について

このマニュアルに記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「/」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「~」で表わします。

表 1-2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC 上の特権ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo9 /]#	Armadillo-9 上の特権ユーザで実行
[armadillo9 /]$	Armadillo-9 上の一般ユーザで実行

コマンド中で、変更の可能性のあるものや、環境により異なるものに関しては以下のように表記します。適時読み替えて入力してください。

表 1-3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号
[user]	一般ユーザ名
[group]	グループ名

1.4. 謝辞

Armadillo-9 で使用しているソフトウェアは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によってなりたっています。この場を借りて感謝の意を示したいと思います。

1.5. ソフトウェアに関する注意事項

本製品に含まれるソフトウェア(付属のドキュメント等も含みます)は、現状のまま(AS IS)提供されるものであり、特定の目的に適合することや、その信頼性、正確性を保証するものではありません。また、本製品の使用による結果についてもなんら保証するものではありません。

1.6. 保証に関する注意事項

- 製品保証範囲について

付属品（ソフトウェアを含みます）を使用し、取扱説明書、各注意事項に基づく正常なご使用に限り有効です。万一正常なご使用のもと製品が故障した場合は、初期不良保証期間内であれば新品交換をさせていただきます。

- 保証対象外になる場合

次のような場合の故障・損傷は、保証期間内であっても保証対象外になります。

1. 取扱説明書に記載されている使用方法、または注意に反したお取り扱いによる場合
2. 改造や部品交換に起因する場合。または正規のものではない機器を接続したことによる場合
3. お客様のお手元に届いた後の輸送、移動時の落下など、お取り扱いの不備による場合
4. 火災、地震、水害、落雷、その他の天災、公害や異常電圧による場合
5. AC アダプタ、専用ケーブルなどの付属品について、同梱のものを使用していない場合
6. 修理依頼の際に購入時の付属品がすべて揃っていない場合

- 免責事項

弊社に故意または重大な過失があった場合を除き、製品の使用および、故障、修理によって発生するいかなる損害についても、弊社は一切の責任を負わないものとします。

本製品の初期不良保証期間は商品到着後２週間です。本製品をご購入されましたらお手数でも必ず動作確認をおこなってからご使用ください。本製品に対して注意事項を守らずに発生した故障につきましては保証対象外となります。

2.作業の前に

2.1.準備するもの

Armadillo-9 を使用する前に、次のものを準備してください。

- 作業用 PC
  Linux もしくは Windows が動作し、1 ポート以上のシリアルポートを持つ PC です。

- シリアルクロスケーブル
  D-Sub9 ピン（メスーメス）の「クロス接続用」ケーブルです。

- 付属 CD-ROM（以降、付属 CD）
  Armadillo-9 に関する各種マニュアルやソースコードが収納されています。

- シリアルコンソールソフト
  minicom や Tera Term などのシリアルコンソールソフトです。（Linux 用のソフトは付属 CD の「tools」ディレクトリにあります。）作業用 PC にインストールしてください。

2.2.接続方法

下の図を参照して、シリアルクロスケーブル、電源ケーブル、そして LAN ケーブルを Armadillo-9 に接続してください。

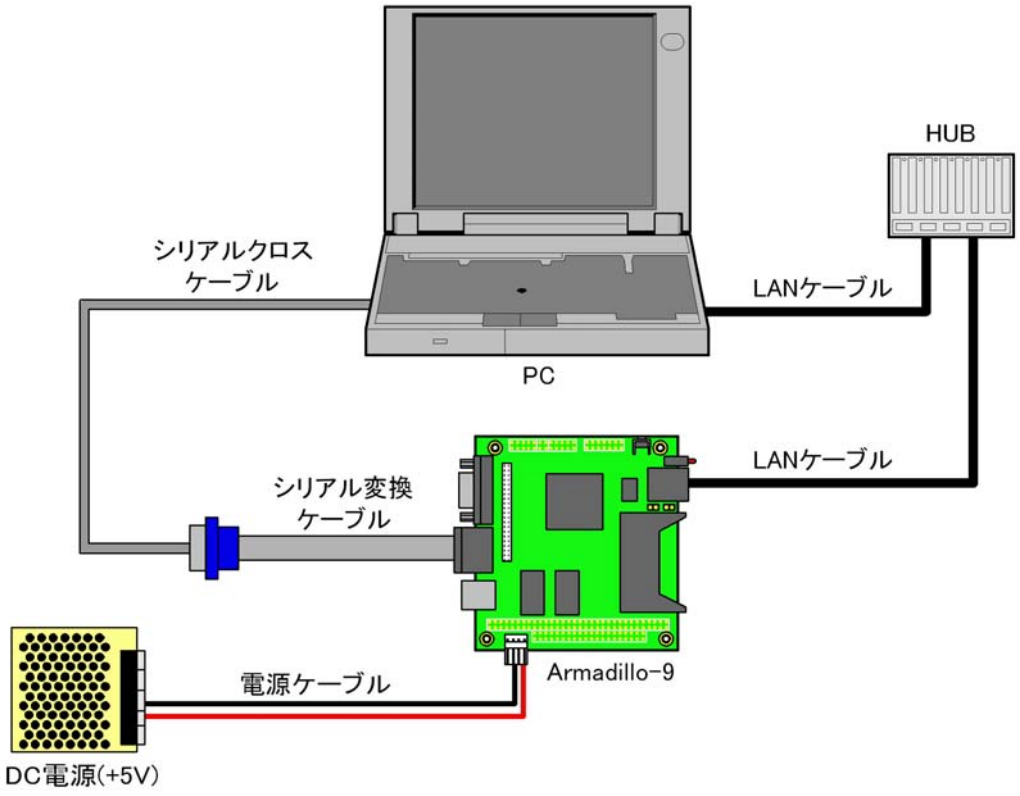

図 2-1 Armadillo-9 接続例

2.3. ジャンパピンの設定について

Armadillo-9 ではジャンパの設定を変えることで、ブート時の動作を変更することができます。以下の表に設定と動作の関連を記載します。

表 2-1 ジャンパの設定とブート時の動作

JP1	JP2	ブート時の動作
オープン	オープン	**オンボードフラッシュメモリ内の Linux カーネルを起動**
オープン	ショート	「12.1.Armadillo-9 で起動可能なCompact Flashの作成」で作成されたIDEドライブ、またはCompact Flashが接続されている場合(※1) **IDE ドライブまたは Compact Flash 内の Linux カーネルを起動**(※2) 上記以外の場合 **Hermit コマンドプロンプトを起動**
ショート	—	**EP9315 オンチップブート ROM を起動**(※3)

※1 ブートローダーHermit v1.3-armadillo9-3 から、IDE ドライブ内カーネルの起動に対応しました。
※2 カーネルの検出は、IDE ドライブ→Compact Flash の順です。
※3 ブートローダーの復旧などに使用します。

TIPS

ジャンパのオープン、ショートとは

・オープン　：　ジャンパピンにジャンパソケットを挿さない状態
・ショート　：　ジャンパピンにジャンパソケットを挿した状態

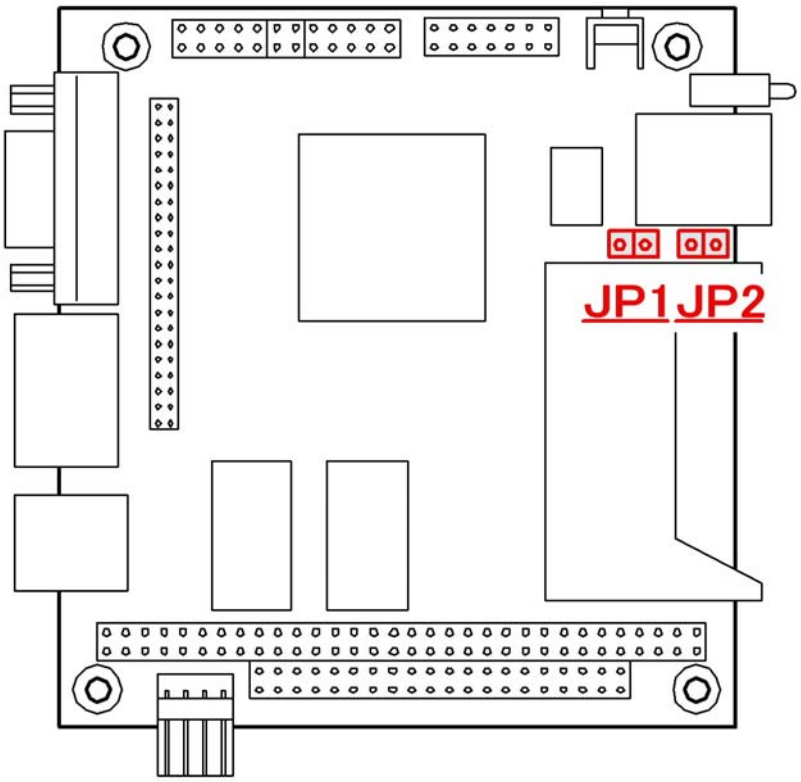

図 2-2 ジャンパの位置

3.開発環境の準備

開発ボードのソフトウェア開発には、Debian/GNU Linux系のOS環境[1]（Debian etchを標準とします）が必要です。作業用PCがWindowsの場合、仮想的なLinux環境を構築する必要があります。

Windows上にLinux環境を構築する方法として、「VMware」を推奨しています。VMwareを使用する場合は、開発に必要なソフトウェアがインストールされた状態のOSイメージ「ATDE（Atmark Techno Development Environment）」[2]を提供しています。

Windows 上に Linux 環境を構築する手順についてのドキュメントは以下のとおりです。詳しくは、こちらを参照してください。

- ATDE Install Guide
- coLinux Guide

ATDE をお使いになる場合は、本章で新たにインストールする必要はありません。

3.1.クロス開発環境パッケージのインストール

付属CDのcross-dev/debディレクトリにクロス開発環境パッケージが用意されています。サポートしている開発環境は、表 3-1のとおりです。通常は、armクロス開発環境をインストールしてください。cross-dev/deb/クロスターゲットディレクトリ以下のパッケージをすべてインストールしてください。インストールは必ず特権ユーザで行ってください。図 3-1のようにコマンドを実行します。

表 3-1 開発環境一覧

クロスターゲット	説明
arm	通常の ARM クロス開発環境です。

```
[PC ~]# dpkg –i *.deb
```

図 3-1 インストールコマンド

ご使用の開発環境に既に同一のターゲット用クロス開発環境がインストールされている場合、新しいクロス開発環境をインストールする前に必ずアンインストールするようにしてください。

[1] debian 系以外の Linux でも開発はできますが、本書記載事項すべてが全く同じように動作するわけではありません。各作業はお使いの Linux 環境に合わせた形で自己責任のもと行ってください。

[2] Armadillo-9 の開発環境としては、ATDE v2.0 以降を推奨しています。

3.2. atmark-dist のビルドに必要なパッケージ

atmark-distをビルドするためには、表 3-2に示すパッケージを作業用PCにインストールされている必要があります。作業用PCの環境に合わせて適切にインストールしてください。

表 3-2 atmark-dist のビルドに必要なパッケージ一覧

パッケージ名	バージョン	備考
genext2fs	1.3-7.1-cvs20050225	付属 CD の tool ディレクトリに収録されています
file	4.12-1 以降	
sed	4.1.2-8 以降	
perl	5.8.4-8 以降	
bison	1.875d 以降	
flex	2.5.31 以降	
libncurses5-dev	5.4-4 以降	

現在インストールされているバージョンを表示するには、図 3-2のようにパッケージ名を指定して実行してください。

```
[PC ~]# dpkg -l    file
                   パッケージ名
```

図 3-2 インストール情報表示コマンド

3.3. クロス開発用ライブラリパッケージの作成方法

アプリケーション開発を行なう際に、付属 CD には収録されていないライブラリパッケージが必要になることがあります。ここでは、ARM のクロス開発用ライブラリパッケージの作成方法を紹介します。

まず、作成したいクロス開発用パッケージの元となるライブラリパッケージを取得します。元となるパッケージは、ARM 用のパッケージです。例えば、**libjpeg6b** の場合「libjpeg6b_x.x-x_arm.deb」というパッケージになります。
次のコマンドで、取得したライブラリパッケージをクロス開発用に変換します。

```
[PC ~]$ dpkg-cross --build --arch arm libjpeg6b_[version]_arm.deb
[PC ~]$ ls
libjpeg6b-arm-cross_[version]_all.deb libjpeg6b_[version]_arm.deb
```

図 3-3 クロス開発用ライブラリパッケージの作成

Debian etch 以外の Linux 環境で dpkg-cross を行った場合、インストール可能なパッケージを生成できない場合があります。

4.使用方法

この章では Armadillo-9 の基本的な使用方法の説明を行ないます。

4.1.起動の前に

Armadillo-9 と作業用 PC をシリアルケーブルで接続し、シリアルコンソールソフトを起動します。次のように通信設定を行なってください。

表 4-1 シリアル通信設定

項目	設定
転送レート	115,200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

4.2.起動

JP1、JP2 をオープンに設定して電源を接続すると、Armadillo-9 が起動します。正常に起動した場合、次のようなログが出力されます。

```
Uncompressing  kernel.....................................................................
..............done.
Uncompressing
ramdisk.....................................................................................
.....................................................................................
....done.
Doing console=ttyAM0,115200
Doing mtdparts=armadillo9-nor:0x10000(bootloader)ro,0x200000(kernel),0x5e0000(userland),-(config)
Linux version 2.6.12.3-a9-10 (build@debian) (gcc version 3.4.4 20050314 (prerelease) (Debian 3.4.3-13))
#1 Fri Sep 14 22:00:29 JST 2007
CPU: ARM920Tid(wb) [41129200] revision 0 (ARMv4T)
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 64, 32 byte lines, 8 sets
CPU0: D cache: 16384 bytes, associativity 64, 32 byte lines, 8 sets
Machine: Armadillo-9
ATAG_INITRD is deprecated; please update your bootloader.
Memory policy: ECC disabled, Data cache writeback
Built 1 zonelists
Kernel command line: console=ttyAM0,115200
mtdparts=armadillo9-nor:0x10000(bootloader)ro,0x200000(kernel),0x5e0000(userland),-(config)
PID hash table entries: 512 (order: 9, 8192 bytes)
Console: colour dummy device 80x30
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
Memory: 32MB 32MB = 64MB total
Memory: 55536KB available (2369K code, 575K data, 104K init)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
checking if image is initramfs...it isn't (bad gzip magic numbers); looks like an initrd
Freeing initrd memory: 6144K
NET: Registered protocol family 16
SCSI subsystem initialized
usbcore: registered new driver usbfs
usbcore: registered new driver hub
NetWinder Floating Point Emulator V0.97 (double precision)
devfs: 2004-01-31 Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x0
Console: switching to colour frame buffer device 80x30
fb0: EP93xx frame buffer at 640x480x16
ttyAM0 at MMIO 0x808c0000 (irq = 52) is a EP93XX
ttyAM1 at MMIO 0x808d0000 (irq = 54) is a EP93XX
ttyAM2 at MMIO 0x808e0000 (irq = 55) is a EP93XX
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 16384K size 1024 blocksize
loop: loaded (max 8 devices)
i2c /dev entries driver
i2c-armadillo9: i2c Armadillo-9 driver, (C) 2004-2005 Atmark Techno, Inc.
```

```
i2c-at24cxx: i2c at24cxx eeprom driver, (C) 2003-2005 Atmark Techno, Inc.
i2c-s3531a: Device Type [S-353x0A]
i2c-s3531a: i2c S-3531A/S-353X0A driver, (C) 2001-2005 Atmark Techno, Inc.
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
No card in slot: PFDR=000000ff
armadillo9-nor: Found 1 x16 devices at 0x0 in 16-bit bank
 Amd/Fujitsu Extended Query Table at 0x0040
armadillo9-nor: CFI does not contain boot bank location. Assuming top.
number of CFI chips: 1
cfi_cmdset_0002: Disabling erase-suspend-program due to code brokenness.
4 cmdlinepart partitions found on MTD device armadillo9-nor
parse_mtd_partitions:4
Creating 4 MTD partitions on "armadillo9-nor":
0x00000000-0x00010000 : "bootloader"
0x00010000-0x00210000 : "kernel"
0x00210000-0x007f0000 : "userland"
0x007f0000-0x00800000 : "config"
ep93xxusb ep93xxusb.0: EP93xx OHCI
ep93xxusb ep93xxusb.0: new USB bus registered, assigned bus number 1
ep93xxusb ep93xxusb.0: irq 56, io base 0xff020000
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 3 ports detected
usbcore: registered new driver audio
drivers/usb/class/audio.c: v1.0.0:USB Audio Class driver
Initializing USB Mass Storage driver...
usbcore: registered new driver usb-storage
USB Mass Storage support registered.
usbcore: registered new driver usbhid
drivers/usb/input/hid-core.c: v2.01:USB HID core driver
usbcore: registered new driver usbserial
drivers/usb/serial/usb-serial.c: USB Serial support registered for Generic
usbcore: registered new driver usbserial_generic
drivers/usb/serial/usb-serial.c: USB Serial Driver core v2.0
mice: PS/2 mouse device common for all mice
NET: Registered protocol family 2
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP established hash table entries: 4096 (order: 3, 32768 bytes)
TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
ip_tables: (C) 2000-2002 Netfilter core team
NET: Registered protocol family 1
NET: Registered protocol family 17
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 6144KiB [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 104K
init started:  BusyBox v1.00 (2007.09.14-13:02+0000) multi-call binary
Starting fsck for root filesystem.
fsck 1.25 (20-Sep-2001)
ext2fs_check_if_mount: No such file or directory while determining whether /dev/ram0 is mounted.
/dev/ram0: clean, 666/1024 files, 4245/6144 blocks
Checking root filesystem:                                   done
Remounting root rw:                                         done
Mounting proc:                                              done
Mounting usbfs:                                             done
```

```
Cleaning up system:                                 done
Running local start scripts.
Changing file permissions:                          done
Configure /home/ftp:                                done
Starting syslogd:                                   done
Starting klogd:                                     done
Starting basic firewall:                            done
Loading /etc/config:                                done
Configuring network interfaces:                     done
Starting thttpd:                                    done
Starting inetd:                                     done
Setting hostname:                                   done

atmark-dist v1.11.0 (AtmarkTechno/Armadillo-9)
Linux 2.6.12.3-a9-10 [armv4tl arch]

a9-0 login:
```

図 4-1 起動ログ

デフォルトのユーザーランドでは、ログインプロンプトはシリアルポート ttyAM0(CON1)に加え、VGAにも表示されます。

※ VGA 側からログインを行なうには、USB キーボードを接続する必要があります。

※ VGAをLinuxカーネルブートログが出力される標準コンソールに設定する方法は、「6.5.Linuxブートオプションの設定」を参照してください。

ログインユーザは、次の 2 種類が用意されています。

表 4-2 コンソールログイン時のユーザ名とパスワード

ユーザ名	パスワード	権限
root	root	特権ユーザ
guest	(なし)	一般ユーザ

4.3. ディレクトリ構成

ディレクトリ構成は次のようになっています。

表 4-3 ディレクトリ構成の一覧

ディレクトリ名	説　明
/bin	アプリケーション用
/dev	デバイスノード用
/etc	システム設定用
/etc/network	ネットワーク設定用
/lib	共有ライブラリ用
/mnt	マウントポイント用
/proc	プロセス情報用
/root	root ホームディレクトリ
/sbin	システム管理コマンド用
/usr	ユーザ共有情報用
/home	ユーザホームディレクトリ
/home/ftp/pub	ftp データ送受信用
/tmp	テンポラリ保存用
/var	変更データ用

4.4. 終了

電源を切断することで Armadillo-9 を終了させます。

ただし IDE ドライブや Compact Flash がマウントされている場合は、電源切断前にアンマウントするか、halt コマンドを実行してシステムを停止させてから電源を切断してください。これを行なわない場合は、IDE ドライブや Compact Flash のデータが破損する恐れがあります。

4.5.ネットワーク設定

Armadillo-9 内の「/etc/network/interfaces」ファイルを編集することで、ネットワークの設定を変更することができます。

4.5.1. 固定 IP アドレスで使用する場合

固定 IP アドレスを指定する場合の設定例を次に示します。

表 4-4 ネットワーク設定詳細

項目	設定値
IP アドレス	192.168.10.10
ネットマスク	255.255.255.0
ブロードキャストアドレス	192.168.10.255
デフォルトゲートウェイ	192.168.10.1

```
# /etc/network/interfaces -  configuration file for ifup(8), ifdown(8)

auto lo eth0

iface lo inet loopback

iface eth0 inet static
        address 192.168.10.10
        netmask 255.255.255.0
        network 192.168.10.0
        broadcast 192.168.10.255
        gateway 192.168.10.1
```

図 4-2 ネットワーク設定例(固定 IP アドレス時)

ゲートウェイを使用しない場合、gateway に 0.0.0.0 を指定してください。

```
        gateway 0.0.0.0
```

図 4-3 ネットワーク設定例(ゲートウェイの無効化)

4.5.2. DNS サーバの設定

DNS サーバを設定する場合、/etc/config/resolv.conf を変更します。

```
nameserver 192.168.10.1
```

図 4-4 DNS サーバの設定

変更は即座に適用されます。

4.5.3. DHCP を使用する場合

DHCP を利用して IP アドレスを取得する場合の設定例を次に示します。

```
# /etc/network/interfaces -  configuration file for ifup(8), ifdown(8)

auto lo eth0

iface lo inet loopback

iface eth0 inet dhcp
```

図 4-5 ネットワーク設定例(DHCP 使用時)

4.5.4. ネットワーク設定の有効化

ネットワーク設定の変更後、新しい設定でネットワーク接続を行なうには、ifup コマンドを実行します。

既にネットワークに接続済みの場合、一旦今のネットワーク接続を閉じる必要があります。ifdown コマンドで終了します。

```
[armadillo9 /]# ifdown eth0
```

図 4-6 ネットワーク接続の終了

```
[armadillo9 /]# ifup eth0
```

図 4-7 ネットワーク接続の開始

4.5.5. ネットワーク設定をフラッシュメモリに保存する

Armadillo-9 のデフォルトのネットワーク設定は、DHCP となっています。これを、固定 IP アドレスにしてフラッシュメモリに保存する方法を説明します。

まず、ネットワーク設定ファイルを任意の内容に書き換えます。

```
[armadillo9 /etc/config]# vi interfaces
//--- 編集ファイル
# /etc/network/interfaces -  configuration file for ifup(8), ifdown(8)

auto lo eth0

iface lo inet loopback

iface eth0 inet static
        address 192.168.10.10
        netmask 255.255.255.0
        network 192.168.10.0
        broadcast 192.168.10.255
        gateway 192.168.10.1
//--- ファイル終端
[armadillo9 /etc/config]# vi resolv.conf
//--- 編集ファイル
nameserver 192.168.10.1
//--- ファイル終端
[armadillo9 /etc/config]#
```

次に、フラッシュに書き込みます。

```
[armadillo9 /etc/config]# flatfsd -s
```

これで書き換えたネットワーク設定がフラッシュメモリに書き込まれ、次回以降の起動時に反映されます。

4.6. telnet ログイン

次のユーザ名／パスワードで telnet ログインが可能です。root でのログインは行なえません。root 権限が必要な作業を行なう場合、guest でログイン後に「su」コマンドで root 権限を取得してください。

表 4-5 telnet ログイン時のユーザ名とパスワード

ユーザ名	パスワード
guest	なし

4.7. ファイル転送

ftp によるファイル転送が可能です。次のユーザ／パスワードでログインしてください。ホームディレクトリは「/home/ftp」です。「/home/ftp/pub」ディレクトリに移動することでデータの書き込みが可能になります。

表 4-6 ftp のユーザ名とパスワード

ユーザ名	パスワード
ftp	なし

4.8. Web サーバ

thttpdという小さなHTTPサーバが起動しており、WebブラウザからArmadillo-9 をブラウズすることが出来ます。データディレクトリは「/home/www-data」です。URLは「http://(Armadillo-9 のIPアドレス)/」になります。(例 http://192.168.10.10/)

5.フラッシュメモリの書き換え方法

フラッシュメモリの内容を書き換えることで、Armadillo-9 の機能を変更することができます。この章ではフラッシュメモリの書き換え方法を説明します。

注意

何らかの原因により「書き換えイメージの転送」に失敗した場合、Armadillo-9 が正常に起動しなくなる場合があります。書き換えの最中は次の点に注意してください。

- Armadillo-9 の電源を切らない。
- Armadillo-9 と開発用 PC を接続しているシリアルケーブルを外さない。

5.1.ダウンローダのインストール

作業用 PC に「ダウンローダ(hermit)」をインストールします。ダウンローダは Armadillo-9 のフラッシュメモリの書き換えに使用します。

1) Linux の場合

付属 CD の downloader/deb ディレクトリよりパッケージファイルを用意し、インストールします。必ず**特権ユーザ**で行ってください。

図 5-1 展開処理コマンド入力例

```
[PC ~]# dpkg -i hermit-at_[version]_i386.deb
```

2) Windows の場合

付属 CD より「Hermit-At WIN32（downloader/win32/hermit-at-win_xxxxxxxx.zip)」を任意のフォルダに展開します。

5.2. リージョン指定について

フラッシュメモリの書き込み先アドレスをリージョン名で指定することができます。リージョン名には3種類あります。それぞれに書き込むべきイメージと合わせて以下で説明します。

- bootloader
  ブートローダーと呼ばれる、電源投入後、最初に実行されるソフトウェアのイメージを格納する領域です。ブートローダーは、シリアル経由でフラッシュメモリを書き換える機能や、OS を起動する機能などを持ちます。

- kernel
  Linux のカーネルイメージを格納する領域です。この領域に格納されたカーネルはブートローダーによって起動されます。

- userland
  各アプリケーションを含むシステムイメージを格納する領域です。telnet、ftp、Web サーバなどのアプリケーションや各種設定ファイル、ユーザーデータなどが格納されます。

付属 CD の image ディレクトリには、各リージョン向けのイメージファイルが格納されています。

表 5-1 各リージョン用のイメージファイル名

リージョン	ファイル名
bootloader	loader-armadillo9-x.bin
kernel	linux-2.x.x-a9-x.bin.gz
userland	romfs-YYYYMMDD.img.gz

フラッシュメモリのメモリマップは「8.メモリマップについて」を参照してください。

5.3. 書き換え手順

以下の手順でフラッシュメモリの書き換えを行ないます。

5.3.1. ジャンパピンの設定

Armadillo-9 に電源を投入する前に、ジャンパピンを次のように設定します。

- JP1：オープン
- JP2：ショート

また、IDE ドライブ、及び Compact Flash は接続しないでください。

詳しいジャンパピンの設定については、「2.3.ジャンパピンの設定について」を参照してください。

5.3.2. 書き換えイメージの転送

はじめに Armadillo-9 に電源を投入します。
以降の手順は、作業用 PC の OS によって異なります。

1) Linux の場合

Linux が動作する作業用 PC でターミナルを起動し、カーネルイメージファイルとリージョンを指定して hermit コマンドを入力します。
下の図ではファイル名にカーネルイメージ（linux.bin.gz）を指定しています。リージョンの指定には、bootloader、kernel、userland のいずれかを指定してください。

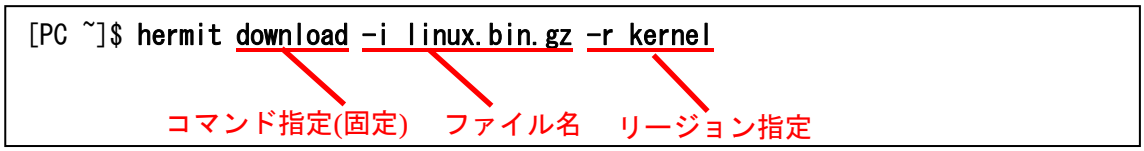

図 5-2 コマンド入力例

作業用 PC で使用するシリアルポートが「ttyS0」以外の場合、オプション「--port “ポート名”」を追加してください。

TIPS

ブートローダー領域(リージョン:bootloader / アドレス:0x60000000-0x6000ffff)を書き換える際は、「--force-locked」を追加する必要があります。これを指定しない場合、警告を表示しブートローダー領域への書き込みを行ないません。

注意

ブートローダー領域に誤ったイメージを書き込んでしまった場合、オンボードフラッシュメモリからの起動ができなくなります。この場合は「6.4.1.ブートローダーを出荷状態に戻す」を参照してブートローダーを復旧してください。

書き換え終了後、JP2 をオープンに設定して Armadillo-9 を再起動すると、新たに書き込んだイメージで起動されます。

2) Windows の場合

「5.1.ダウンローダのインストール」にてファイルを展開したフォルダにある、「Hermit-At WIN32（hermit.exe)」を起動します。

「Download」ボタンをクリックすると 図 5-3が表示されます。

"Serial Port" には、Armadillo-9 と接続しているシリアルポートを設定してください。
"Image" には、書き込みを行ないたいイメージファイルを指定します。ファイルダイアログによる指定も可能です。
"Region" には、書き込むリージョンまたは、アドレスを指定します。

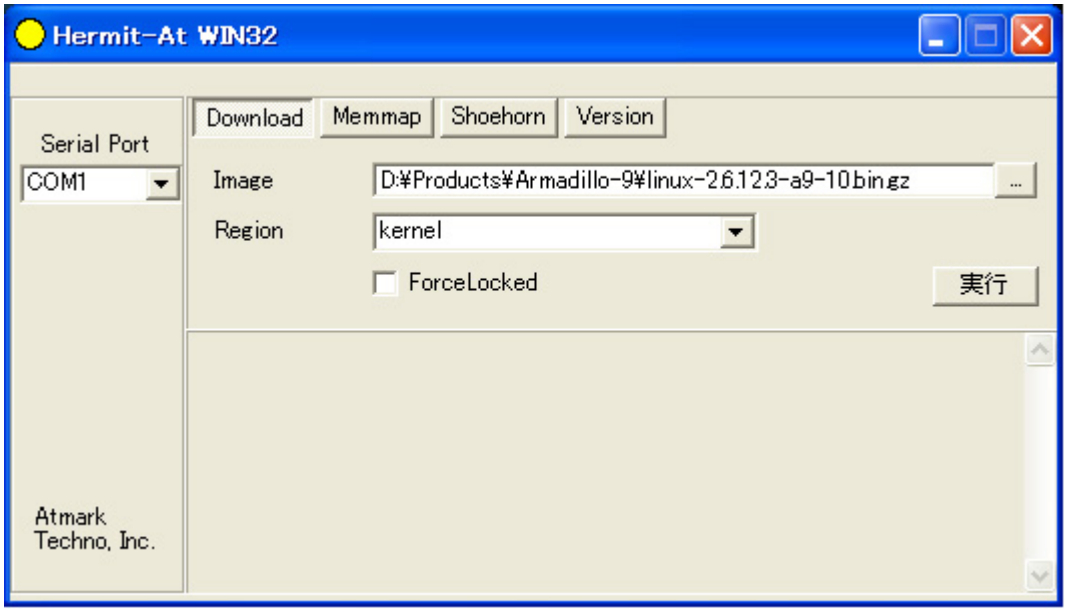

図 5-3 Download 画面

「実行」ボタンをクリックすると、フラッシュメモリの書き換えが開始されます。書き換え中は、進捗状況が 図 5-4のように表示されます。ダイアログは、書き換えが終了すると自動的にクローズされます。

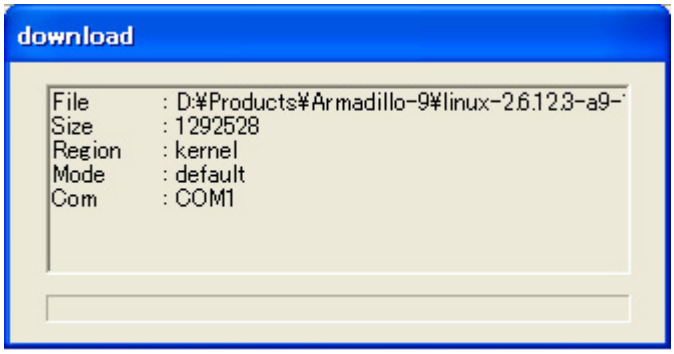

図 5-4 書き換え進捗ダイアログ

TIPS

ブートローダー領域(リージョン:bootloader / アドレス:0x60000000-0x6000ffff)を書き換える際は、「Force locked ボタン」を選択する必要があります。これを選択しない場合、警告を表示しブートローダー領域への書き込みを行ないません。

注意

ブートローダー領域に誤ったイメージを書き込んでしまった場合、オンボードフラッシュメモリからの起動ができなくなります。この場合は「6.4.1.ブートローダーを出荷状態に戻す」を参照してブートローダーを復旧してください。

書き換え終了後、JP2 をオープンに設定して Armadillo-9 を再起動すると、新たに書き込んだイメージで起動します。

5.4. netflash を使ってフラッシュメモリの書き換えをする

フラッシュメモリの内容を書き換える方法として、ユーザアプリケーションの netflash を使用することも可能です。ここでは、netflash を使用してフラッシュメモリを書き換える方法を説明します。

注意

何らかの原因により「フラッシュメモリの書き換え」に失敗した場合、Armadillo-9 が正常に起動しなくなる場合があります。書き換えの最中は Armadillo-9 の電源を切らないように注意してください。

netflash は、HTTP や FTP サーバからファイルを取得し、フラッシュメモリへ書き込みます。はじめに、HTTP や FTP サーバにイメージファイルを置いておく必要があります。

Armadillo-9 上での kernel イメージを変更するコマンド例です。

```
[armadillo9 ~]# netflash  -k   -n  -r /dev/flash/kernel
                          オプション     リージョン指定
        http://download.atmark-techno.com/armadillo-9/image/linux-[version].bin.gz
                                  ファイル名

※通常は１行のコマンドとなります。
```

図 5-5 netflash コマンド例

オプションの “-r /dev/flash/kernel” でリージョンを指定しています。リージョンの指定は下記表を参照してください。

カーネル	/dev/flash/kernel
ユーザーランド	/dev/flash/userland

netflash のヘルプは以下のコマンドで参照することができます。

```
[armadillo9 ~]# netflash -h
```

図 5-6 netflash ヘルプコマンド

6. ブートローダー

この章では、Armadillo-9 のブートローダーに関して説明します。

6.1. パッケージの準備

付属 CD の hermit ディレクトリから以下のパッケージを、作業用 PC にコピーします。

表 6-1 ブートローダー関連のパッケージ一覧

パッケージ名	説明
hermit-at-x.x.x	Armadillo9 ブートプログラムと協調動作するダウンローダ（Armadillo-9 ブートプログラム自体も含む）
shoehorn-at-x.x.x	CPU オンチップブート ROM と協調動作するダウンローダ

パッケージのインストール方法については「3.1クロス開発環境パッケージのインストール」を参照してください。

6.2. ブートローダーの種類

Armadillo-9 で用意されているブートローダーを以下に記載します。

表 6-2 ブートローダー 一覧

ブートローダー名	説明
loader-armadillo9	出荷時にフラッシュメモリに書き込まれている標準ブートローダー コンソールに COM1 を使用
loader-armadillo9-ttyAM1	コンソールに COM2 を使用するブートローダー
loader-armadillo9-notty	コンソールを使用しないブートローダー

6.3. ブートローダーの作成

付属 CD には、各ブートローダーが用意されていますが、ソースからビルドしてオリジナルのブートローダーを作成することができます。

6.3.1. ソースコードの準備

付属 CD の source ディレクトリから、hermit-at-x.x.x-source.tar.gz を作業用 PC にコピーし、解凍します。

```
[PC ~]$ tar zxf hermit-at-[version]-source.tar.gz
```

6.3.2. ビルド

解凍してできたディレクトリへ移動し、make コマンドを入力します。

```
[PC ~]$ cd hermit-at-[version]
[PC ~]$ make TARGET=armadillo9
```

make が完了後、hermit-at-x.x.x/src/target/armadillo9 のディレクトリにブートローダーのイメージファイルが作成されます。

6.4. CPU オンチップブート ROM

loader-armadillo9-notty が書き込まれている Armadillo-9 のブートローダーを書き換えるときや､不正なブートローダーを書き込んでしまいArmadillo-9がブートできなくなってしまった場合の対処方法について説明します。

Armadillo-9 は、CPU オンチップブート ROM を実装しています。この ROM に格納されているソフトウェアを使用して、ブートローダーを出荷状態に戻すことができます。以下にその手順を説明します。

6.4.1. ブートローダーを出荷状態に戻す

1) Linux の場合

① Armadillo-9 の電源が**切断**されていることを確認し、Armadillo-9 の COM1 と、作業用 PC のシリアルポートをクロス(リバース)シリアルケーブルで接続します。

② Armadillo-9 のジャンパ JP1 をショートに設定します。

③ 作業用 PC で shoehorn を起動します。

```
[PC ~]$ shoehorn --boot --terminal --initrd /dev/null
         --kernel /usr/lib/hermit/loader-armadillo9-boot.bin
         --loader /usr/lib/shoehorn/shoehorn-armadillo9.bin
         --initfile /usr/lib/shoehorn/shoehorn-armadillo9.init
         --postfile /usr/lib/shoehorn/shoehorn-armadillo9.post
```

図 6-1 shoehorn コマンド例

※ 上記は、作業用 PC のシリアルポート”/dev/ttyS0”に Armadillo-9 を接続した場合の例です。
他のシリアルポートに接続した場合は、shoehorn コマンドのオプションに
--port [シリアルポート名]
を追加してください。

※ コマンドは１行で入力します

④ Armadillo-9 に電源を接続する。

※ すぐにメッセージ表示が開始されます。正常に表示されない場合は、Armadillo-9 の電源を切断し、シリアルケーブルの接続や Armadillo-9 のジャンパ(JP1)設定を確認してください。

⑤ “ hermit > ” と表示されたら、Ctrl＋C をキー入力します。

以上で作業用PCからhermitを使用してArmadillo-9 へブートローダーをダウンロードする準備が整います。**ジャンパの設定変更や電源の切断をしない**で、「5.フラッシュメモリの書き換え方法」を参照しブートローダーを書き換えてください。

2) Windows の場合

① Armadillo-9 の電源が切断されていることを確認し、Armadillo-9 の COM1 と、作業用 PC のシリアルポートをクロス(リバース)シリアルケーブルで接続します。
② Armadillo-9 のジャンパ JP1 をショートに設定します。
③ 作業用 PC で「Hermit-At WIN32」を起動します。
④ 「Shoehorn」ボタンを押します。

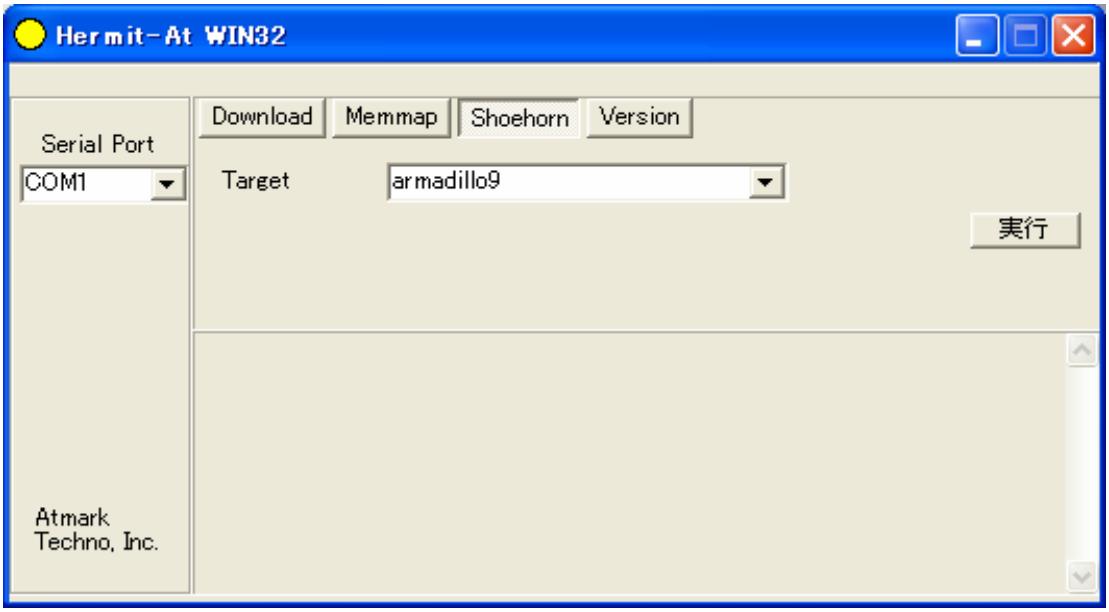

図 6-2 shoehorn ブート時

⑤ "Target" に armadillo9 を指定します。
⑥ 「実行」ボタンをクリックすると 図 6-3が表示されます。

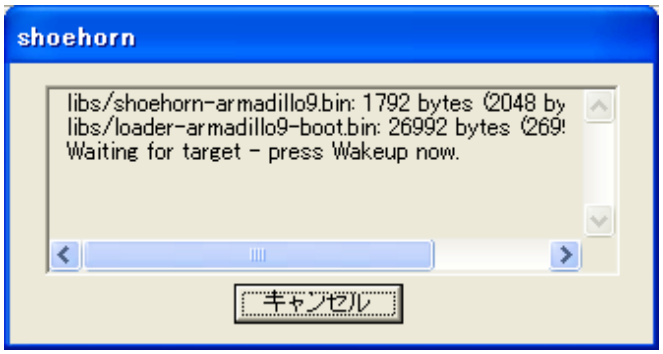

図 6-3 shoehorn ダイアログ

⑦ Armadillo-9 に電源を接続します。

すぐにメッセージ表示が開始されます。正常に表示されない場合は、Armadillo-9 の電源を切断し、シリアルケーブルの接続や Armadillo-9 のジャンパ(JP1)設定を確認してください。

以上で作業用PCからhermitを使用してArmadillo-9 へブートローダーをダウンロードする準備が整います。**ジャンパの設定変更や電源の切断をしない**で、「5.フラッシュメモリの書き換え方法」を参照しブートローダーを書き換えてください。

6.5. Linux ブートオプションの設定

Armadillo-9 では、自動起動する Linux のブートオプションを設定することができます。設定はフラッシュメモリ上に保存され、次回の Linux 起動時から使用されます。

Linux ブートオプションの設定は、Hermit コマンドプロンプトから行ないます。

TIPS

設定する Linux ブートオプションを決定するためには、使用する Linux カーネルについての知識が必要です。オプションの内容と効果については、Linux カーネルについての文献や、ソースファイル付属ドキュメントを参照してください。

6.5.1. Hermit コマンドプロンプトの起動

① シリアルコンソールソフトの起動

Armadillo-9 と作業用 PC をシリアルケーブルで接続し、シリアルコンソールソフトを起動します。次のように通信設定を行なってください。

表 6-3 シリアル通信設定

項目	設定
転送レート	115,200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

② ジャンパピンの設定

Armadillo-9 に電源を投入する前に、ジャンパピンを次のように設定します。

- JP1：オープン
- JP2：ショート

また、IDE ドライブ、及び Compact Flash は接続しないでください。

詳しいジャンパピンの設定については、「2.3.ジャンパピンの設定について」を参照してください。

③ Armadillo-9 の起動

Armadillo-9 に電源を投入すると、Hermit コマンドプロンプトが表示されます。

```
Hermit v1.3-armadillo9-1 compiled at 06:45:05, Dec 18 2004
hermit>
```

6.5.2. Linux ブートオプションの設定

Linux ブートオプションを設定するには、Hermit コマンドプロンプトから setenv コマンドを使用します。setenv に続けて、設定したい Linux ブートオプションを入力します。

```
hermit> setenv console=ttyAM0,115200 root=/dev/hda1 noinitrd
```

注意

Linux ブートオプションが未設定(デフォルト)の場合、ブートローダーは Linux の起動時に自動的にオプション「console=ttyAM0,115200」を使用してシリアルポート（COM1）をコンソールにしますが、setenv により任意のブートオプションを設定した場合は、このオプションは自動使用されません。

setenv した場合でもシリアルコンソールを使用する場合、オプションに「console=ttyAM0,115200」を含めてください。

設定したブートオプションを使用して Linux を起動するには、一旦 Armadillo-9 の電源を切断し、適切なジャンパ設定を行なってから再度電源を入れ直してください。

6.5.3. 設定されている Linux ブートオプションの確認

現在設定されている Linux ブートオプションを表示して確認するには、setenv コマンドをパラメータなしで入力します。

```
hermit> setenv
1: console=ttyAM0,115200
2: root=/dev/hda1
3: noinitrd
```

6.5.4. Linux ブートオプションを初期化する

現在設定されている Linux ブートオプションをクリアし、デフォルトの状態に初期化するには、clearenv コマンドを入力します。

```
hermit> clearenv
```

6.5.5. Linux ブートオプションの例

Linux ブートオプションの設定例を紹介します。

ex.1) シリアルコンソールを使用し、IDE ディスクドライブの第 1 パーティションをルートデバイスとする場合
(ジャンパピンは JP1,JP2 ともオープンとして、Linux カーネルはオンボードフラッシュメモリ内のものを使用)

```
hermit> setenv console=ttyAM0,115200 root=/dev/hda1 noinitrd
```

ex.2) コンソールとして VGA を使用する場合
(Debian/GNU Linux で X-Window System を使用する際に推奨)

```
hermit> setenv video
```

TIPS
VGA コンソールに入力を行なうには、USB キーボードを接続する必要があります。

7.atmark-dist でイメージを作成

この章では、atmark-dist を使用して、カーネル／ユーザーランドのイメージを作成する方法を説明します。atmark-dist に関する詳しい使用方法は、「atmark-dist Developers Guide」を参照してください。

注意

atmark-dist を使用した開発作業では、基本ライブラリ・アプリケーションやシステム設定ファイルの作成・配置を行ないます。各ファイルは atmark-dist ディレクトリ配下で作成・配置作業を行ないますが、作業ミスにより誤って作業用 PC 自体の OS を破壊しないために、すべての作業は root ユーザではなく一般ユーザで行なってください。

7.1. ソースコードアーカイブの展開

付属 CD の dist ディレクトリに atmark-dist.tar.gz というファイル名のソースコードアーカイブがあります。このファイルを任意のディレクトリに展開します。ここでは、ユーザのホームディレクトリ(~/)に展開することとします。

```
[PC ~]$ tar zxvf atmark-dist.tar.gz
```

次に Linux カーネルソースコードを展開し、atmark-dist ディレクトリ内に linux-2.6.x という名前でシンボリックリンクを作成します。付属 CD の source ディレクトリに linux-[version].tar.gz という名前でカーネルソースコードがあります。

```
[PC ~]$ tar zxvf linux-[version].tar.gz
 :
 :
[PC ~]$ cd atmark-dist
[PC ~/atmark-dist]$ ln - s ../linux-[version] ./linux-2.6.x
```

注意

- linux-2.6.x の「x」はそのまま記述してください

7.2.設定

ターゲットボード用の dist をコンフィギュレーションします。以下の例のようにコマンドを入力し、コンフィギュレーションを開始します。

```
[PC ~/atmark-dist]$ make config
```

続いて、使用するボードのベンダー名を聞かれます。「**AtmarkTechno**」と入力してください。

```
[PC ~/atmark-dist]$ make config
config/mkconfig > config.in
#
# No defaults found
#
*
* Vendor/Product Selection
*
*
* Select the Vendor you wish to target
*
Vendor (3com, ADI, Akizuki, Apple, Arcturus, Arnewsh, AtmarkTechno, Atmel, Avnet,
Cirrus, Cogent, Conexant, Cwlinux, CyberGuard, Cytek, Exys, Feith, Future, GDB,
Hitachi, Imt, Insight, Intel, KendinMicrel, LEOX, Mecel, Midas, Motorola, NEC,
NetSilicon, Netburner, Nintendo, OPENcores, Promise, SNEHA, SSV, SWARM, Samsung,
SecureEdge, Signal, SnapGear, Soekris, Sony, StrawberryLinux, TI, TeleIP, Triscend,
Via, Weiss, Xilinx, senTec) [SnapGear] (NEW) AtmarkTechno
```

次にボード名を聞かれます。「**Armadillo-9**」と入力してください。

```
*
* Select the Product you wish to target
*
AtmarkTechno Products (Armadillo-9, SUZAKU) [Armadillo-9] (NEW) Armadillo-9
```

使用するCライブラリを指定します。使用するボードによってサポートされているライブラリは異なります。Armadillo-9では、「**None**」を選択します。

```
*
* Kernel/Library/Defaults Selection
*
*
* Kernel is linux-2.6.x
*
Libc Version (None, glibc, uC-libc, uClibc) [uClibc] (NEW) None
```

デフォルトの設定にするかどうか質問されます。「**y**」(Yes)を選択してください。

```
Default all settings (lose changes) (CONFIG_DEFAULTS_OVERRIDE) [N/y/?] (NEW) y
```

最後の３つの質問は「**n**」(No)と答えてください。

```
Customize Kernel Settings (CONFIG_DEFAULTS_KERNEL) [N/y/?] n
Customize Vendor/User Settings (CONFIG_DEFAULTS_VENDOR) [N/y/?] n
Update Default Vendor Settings (CONFIG_DEFAULTS_VENDOR_UPDATE) [N/y/?] n
```

質問事項が終わるとビルドシステムの設定を行ないます。すべての設定が終わるとプロンプトに戻ります。

7.3. ビルド

実際にビルドするには以下のコマンドを入力してください。

```
[PC ~/atmark-dist]$ make all
```

dist のバージョンによっては、make の途中で一時停止し、未設定項目の問合せが表示される場合があります。通常はデフォルト設定のままで構いませんので、このような場合はそのままリターンキーを入力して進めてください。

ビルドが終了すると、atmark-dist/images ディレクトリに、カーネルイメージであるlinux.bin.gzとユーザーランドイメージであるromfs.imgが作成されます。作成したイメージをArmadillo-9 に書き込む方法は「5.フラッシュメモリの書き換え方法」を参照してください。

8.メモリマップについて

表 8-1 メモリマップ(フラッシュメモリ)

アドレス	リージョン	サイズ	説明
0x60000000 0x6000ffff	bootloader	64KB	Hermit ブートローダー 「loader-armadillo9.bin」のイメージ
0x60010000 0x6017ffff	kernel	約 1.44MB	Linux カーネル 「linux.bin.gz」のイメージ (非圧縮イメージ、gz 圧縮イメージに対応)
0x60180000 0x607effff	userland	約 6.44MB	ユーザーランド 「romfs.img」のイメージ (非圧縮イメージ、gz 圧縮イメージに対応)
0x607f0000 0x607fffff	config	64KB	コンフィグ領域

※ kernel とユーザーランドのみ、linux の起動前に RAM へ展開・コピーされる

表 8-2 メモリマップ(RAM)

アドレス	内容	ファイル システム	説明
0xc0018000	kernel	—	linux 起動前に フラッシュメモリから展開・コピー
0xc0800000	userland	EXT2	linux の起動前に フラッシュメモリから展開・コピー

表 8-3 メモリマップ(PC/104)

Linux 論理アドレス	物理アドレス	説明
0xf2000000 0xf200ffff	0x12000000 0x1200ffff	PC/104 I/O Space (8bit)
0xf3000000 0xf3ffffff	0x13000000 0x13ffffff	PC/104 Memory Space (8bit)
0xf6000000 0xf600ffff	0x22000000 0x2200ffff	PC/104 I/O Space (16bit)
0xf7000000 0xf7ffffff	0x23000000 0x23ffffff	PC/104 Memory Space (16bit)

9.割り込み(IRQ)について

表 9-1 割り込み(IRQ)一覧表

Linuxでの IRQ番号	名称	インタラプト ソース	説明
2	IRQ_COMMRX	VIC #2	COMMRX
3	IRQ_COMMTX	VIC #3	COMMTX
4	IRQ_TIMER1	VIC #4	TIMER1
5	IRQ_TIMER2	VIC #5	TIMER2
6	IRQ_AAC	VIC #6	AAC
7	IRQ_DMAM2P0	VIC #7	DMAM2P0
8	IRQ_DMAM2P1	VIC #8	DMAM2P1
9	IRQ_DMAM2P2	VIC #9	DMAM2P2
10	IRQ_DMAM2P3	VIC #10	DMAM2P3
11	IRQ_DMAM2P4	VIC #11	DMAM2P4
12	IRQ_DMAM2P5	VIC #12	DMAM2P5
13	IRQ_DMAM2P6	VIC #13	DMAM2P6
14	IRQ_DMAM2P7	VIC #14	DMAM2P7
15	IRQ_DMAM2P8	VIC #15	DMAM2P8
16	IRQ_DMAM2P9	VIC #16	DMAM2P9
17	IRQ_DMAM2M0	VIC #17	DMAM2M0
18	IRQ_DMAM2M1	VIC #18	DMAM2M1
19	IRQ_GPIO0	VIC #19	GPIO0
20	IRQ_GPIO1	VIC #20	GPIO1
21	IRQ_GPIO2	VIC #21	GPIO2
22	IRQ_GPIO3	VIC #22	GPIO3
23	IRQ_UARTRX1	VIC #23	UARTRX1
24	IRQ_UARTTX1	VIC #24	UARTTX1
25	IRQ_UARTRX2	VIC #25	UARTRX2
26	IRQ_UARTTX2	VIC #26	UARTTX2
27	IRQ_UARTRX3	VIC #27	UARTRX3
28	IRQ_UARTTX3	VIC #28	UARTTX3
29	IRQ_KEY	VIC #29	KEY
30	IRQ_TOUCH	VIC #30	TOUCH
32	IRQ_EXT0	VIC #32	EXT0
33	IRQ_EXT1	VIC #33	EXT1
34	IRQ_EXT2	VIC #34	EXT2
35	IRQ_64HZ	VIC #35	64HZ
36	IRQ_WEIN	VIC #36	WEIN
37	IRQ_RTC	VIC #37	RTC
38	IRQ_IRDA	VIC #38	IRDA
39	IRQ_MAC	VIC #39	MAC
40	IRQ_EXT3 (IRQ_EIDE)	VIC #40	EXT3 (EIDE)
41	IRQ_PROG	VIC #41	PROG
42	IRQ_1HZ	VIC #42	1HZ
43	IRQ_VSYNC	VIC #43	VSYNC
44	IRQ_VIDEOFIFO	VIC #44	VIDEOFIFO

45	IRQ_SSPRX	VIC #45	SSPRX
46	IRQ_SSPTX	VIC #46	SSPTX
47	IRQ_GPIO4	VIC #47	GPIO4
48	IRQ_GPIO5	VIC #48	GPIO5
49	IRQ_GPIO6	VIC #49	GPIO6
50	IRQ_GPIO7	VIC #50	GPIO7
51	IRQ_TIMER	VIC #51	TIMER3
52	IRQ_UART1	VIC #52	UART1
53	IRQ_SSP	VIC #53	SSP
54	IRQ_UART2	VIC #54	UART2
55	IRQ_UART3	VIC #55	UART3
56	IRQ_USH	VIC #56	USH
57	IRQ_PME	VIC #57	PME
58	IRQ_DSP	VIC #58	DSP
59	IRQ_GPIO	VIC #59	GPIO
60	IRQ_SAI	VIC #60	SAI
64	IRQ_ISA3	PC/104 #3	
65	IRQ_ISA4	PC/104 #4	
66	IRQ_ISA5	PC/104 #5	
67	IRQ_ISA6	PC/104 #6	
68	IRQ_ISA7	PC/104 #7	
69	IRQ_ISA9	PC/104 #9	
70	IRQ_ISA10	PC/104 #10	
71	IRQ_ISA11	PC/104 #11	
72	IRQ_ISA12	PC/104 #12	
73	IRQ_ISA14	PC/104 #14	
74	IRQ_ISA15	PC/104 #15	

表 9-2 PC/104 IRQ サポート関数

用途	Linux の IRQ 番号から PC/104 の IRQ 番号へ変換
関数定義	static __inline__ unsigned int **convirq_to_isa** (unsigned int irq);

使用例	・Linux の IRQ 番号 IRQ_ISA3 を PC/104 の IRQ 番号に変換
	const unsigned linux_irq = IRQ_ISA3;
	unsigned int isa_irq;
	isa_irq = convirq_to_isa(linux_irq);

用途	PC/104 の IRQ 番号から Linux の IRQ 番号へ変換
関数定義	static __inline__ unsigned int **convirq_from_isa** (unsigned int irq);
使用例	・PC/104 の IRQ 番号 3 を Linux の IRQ 番号に変換 const unsigned int isa_irq = 3; unsigned linux_irq; linux_irq = convirq_from_isa(isa_irq);

※(カーネルソース)/include/asm-arm/arch-ep93xx/irqs.h 内で定義

10. VGA デバイスドライバ仕様

VGA 出力はフレームバッファドライバが用意されており、コンソール画面として使用することができます。

初期状態では VGA サイズ(解像度:640x480)の 16 ビットカラー設定となっていますが、SVGA サイズ(800x600)及び XGA サイズ(1024x768)や 8/24 ビットカラーにも対応しています。

ここでは、この設定の変更方法について説明します。

10.1. デフォルト設定の変更

デフォルト設定の変更には、カーネルのリコンパイルが必要となります。
まず、コンフィギュレーションします。

```
[PC ~/atmark-dist]$ make menuconfig
```

メニューが表示されるので、

```
Kernel/Library/Defaults Selection --->
  --- Kernel is linux-2.6.x
  (None) Libc Version
  [ ] Default all settings
  [*] Customize Kernel Settings                    ここを選択する
  [ ] Customize Vendor/User Settings
  [ ] Update Default Vendor Settings
```

とします。続いて Kernel Configuration のメニューが表示されるので、

```
Device Drivers  --->
  Graphics support  --->
    <*> Support for frame buffer devices  --->
      <*> EP93xx frame buffer support  --->
        EP93xx frame buffer display (CRT display)
        EP93xx frame buffer resolution (VGA(60Hz))     解像度の上限
        EP93xx frame buffer depth (16bpp true color)   カラー設定
```

上記の項目を変更した後、コンフィギュレーションを終了させます。

注意

Armadillo-9 ではすべての設定をサポートしているわけではありません。
hardware manual の「5.13 CON12 (VGA コネクタ)」を参照してください。

続いて、ビルドします。

```
[PC ~/atmark-dist]$ make all
```

ビルドしてできたカーネルイメージ（linux.bin.gz）を Armadillo-9 へ書き込み、VGA のデフォルトの設定は完了です。

10.2. 解像度・色深度の変更

デフォルトの解像度・色深度以外で VGA を動作させるときは、Linux ブートオプションに設定を追加するだけで変更ができます。

「6.5.Linuxブートオプションの設定」を参考にhermitを起動させます。
ブートオプションに“ video=ep93xxfb:mode=??? ”を追加します。“ ??? ”には、表からモード名を挿入してください。

表 10-1 解像度一覧

モード名	解像度
CRT-640x480	640x480 60Hz
CRT-640x480@75	640x480 75Hz
CRT-800x600	800x600 60Hz
CRT-800x600@75	800x600 75Hz
CRT-1024x768	1024x768 60Hz
CRT-1024x768@75	1024x768 75Hz

表 10-2 色深度一覧

モード名	解像度
8bpp	8 ビットカラー
16bpp	16 ビットカラー
24bpp	24 ビットカラー
32bpp	32 ビットカラー

設定例です。

```
hermit> setenv video=ep93xxfb:CRT-800x600,8bpp
                解像度のオプション
```

11. その他のデバイスドライバ仕様

11.1. GPIO ポート

GPIO ポートに対応するデバイスノードのパラメータは、以下の通りです。

表 11-1 GPIO ノード

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/xxx)	デバイス名
キャラクタデバイス	10	185	gpio	EP93XX_GPIO_PADR EP93XX_GPIO_PBDR EP93XX_GPIO_PCDR EP93XX_GPIO_PDDR EP93XX_GPIO_PEDR EP93XX_GPIO_PFDR EP93XX_GPIO_PGDR EP93XX_GPIO_PHDR EP93XX_GPIO_PADDR EP93XX_GPIO_PBDDR EP93XX_GPIO_PCDDR EP93XX_GPIO_PDDDR EP93XX_GPIO_PEDDR EP93XX_GPIO_PFDDR EP93XX_GPIO_PGDDR EP93XX_GPIO_PHDDR

ioctl を使用してアクセスすることにより、EP9315 の GPIO レジスタを直接操作することができます。第 3 引数には、(カーネルソース)/include/linux/ep93xx_gpio.h に定義されている構造体「struct ep93xx_gpio_ioctl_data」と各マクロを使用します。

レジスタの詳細については、Cirrus Logic 社 EP9315 User's Guide の「Chapter 28 GPIO Interface」を参照してください。

CON4/5 の各ピンとレジスタの対応は下記のようになります。

ピン名	ポート	アドレス
CON4 ピン 3	PortA:4	PADR/PADDR:0x00000010
CON4 ピン 4	PortA:5	PADR/PADDR:0x00000020
CON4 ピン 5	PortA:6	PADR/PADDR:0x00000040
CON4 ピン 6	PortA:7	PADR/PADDR:0x00000080
CON4 ピン 7	PortB:0	PBDR/PBDDR:0x00000001
CON4 ピン 8	PortB:1	PBDR/PBDDR:0x00000002
CON4 ピン 9	PortB:2	PBDR/PBDDR:0x00000004
CON4 ピン 10	PortB:3	PBDR/PBDDR:0x00000008
CON5 ピン 1	PortD:4	PDDR/PDDDR:0x00000010
CON5 ピン 2	PortD:5	PDDR/PDDDR:0x00000020
CON5 ピン 3	PortD:6	PDDR/PDDDR:0x00000040
CON5 ピン 4	PortD:7	PDDR/PDDDR:0x00000080

例 11-1 ep93xx_gpio.h の構造体とマクロ定義

```
        ↓データ構造体(実体定義し ioctl 第 3 引数にポインタ指定)
struct ep93xx_gpio_ioctl_data {
        __u32 device;       ←レジスタ指定マクロを代入
        __u32 mask;         ←取得・操作する bit 位置を指定
        __u32 data;         ←読込/書込データ用変数
};

#define EP93XX_GPIO_IOCTL_BASE      'N'
        ↓コマンド指定マクロ(ioctl 第 2 引数に使用)
#define EP93XX_GPIO_IN      _IOWR(EP93XX_GPIO_IOCTL_BASE, 0, struct ep93xx_gpio_ioctl_data)
#define EP93XX_GPIO_OUT     _IOW (EP93XX_GPIO_IOCTL_BASE, 1, struct ep93xx_gpio_ioctl_data)

enum {    ↓レジスタ指定マクロ
  EP93XX_GPIO_PADR = 0,
  EP93XX_GPIO_PBDR,
  EP93XX_GPIO_PCDR,
  EP93XX_GPIO_PDDR,
  EP93XX_GPIO_PADDR,
  EP93XX_GPIO_PBDDR,
  EP93XX_GPIO_PCDDR,
  EP93XX_GPIO_PDDDR,
  EP93XX_GPIO_PEDR,
  EP93XX_GPIO_PEDDR,
  EP93XX_GPIO_PFDR,
  EP93XX_GPIO_PFDDR,
  EP93XX_GPIO_PGDR,
  EP93XX_GPIO_PGDDR,
  EP93XX_GPIO_PHDR,
  EP93XX_GPIO_PHDDR,
  EP93XX_GPIO_NUM,
};
```

例 11-2 GPIO 操作のサンプル Makefile

```
ROOTDIR=../atmark-dist     ←環境に合わせ修正が必要

ROMFSDIR  = $(ROOTDIR)/romfs
ROMFSINST = $(ROOTDIR)/tools/romfs-inst.sh

UCLINUX_BUILD_USER = 1
include $(ROOTDIR)/.config
include $(ROOTDIR)/config.arch

TARGET = gpio_sample
OBJS   = sample.o

CFLAGS += -I$(ROOTDIR)/$(CONFIG_LINUXDIR)/include

all: $(TARGET)

$(TARGET): $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
        -rm -f *.o *.elf *.gdb $(TARGET) *~

romfs:
        $(ROMFSINST) /bin/$(TARGET)

%.o: %.c
        $(CC) -c $(CFLAGS) -o $@ $<
```

例 11-3 GPIO 操作のサンプルプログラム(sample.c)

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>

#include <asm/types.h>
#include <linux/ep93xx_gpio.h>

int main(int argc, char **argv){
        int fd;
        struct ep93xx_gpio_ioctl_data d;

        // GPIO を読み書き可能でオープン
        fd = open("/dev/gpio", O_RDWR);
        if(fd < 0) {
                fprintf(stderr, "Open error.¥n");
                return -1;
        }

        // Port B[0]を入力、[1]を出力に変更
        d.device = EP93XX_GPIO_PBDDR;
        d.mask   = 0x00000003;
        d.data   = 0x00000002;
        ioctl(fd, EP93XX_GPIO_OUT, &d);

        // Port B[0]の入力値を表示
        d.device = EP93XX_GPIO_PBDR;
        d.mask   = 0x00000001;
        ioctl(fd, EP93XX_GPIO_IN, &d);
        printf("Port B[0]: %d¥n", (d.data & d.mask));

        // Port B[1]に High を出力
        d.device = EP93XX_GPIO_PBDR;
        d.mask   = 0x00000002;
        d.data   = 0x00000002;
        ioctl(fd, EP93XX_GPIO_OUT, &d);

        close(fd);

        return 0;
}
```

11.2. リアルタイムクロック

リアルタイムクロックに対応するデバイスノードのパラメータは、以下の通りです。

表 11-2 リアルタイムクロックノード

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/xxx)	デバイス名
キャラクタデバイス	10	135	rtc	リアルタイムクロック

11.2.1. リアルタイムクロックの設定

Armadillo-9 は、カレンダ時計（Real Time Clock）が実装されているため、電源を OFF／ON した場合でも日付と時刻が正しく表示されます。詳細については、hardware manual の「6.4.カレンダ時計」を参照してください。

RTC に日時を設定するためには、まずシステムクロックを設定します。その後に、ハードウェアクロック（RTC）をシステムクロックと一致させる手順となります。

●システムクロックを date で設定する

```
[armadillo9 ~]# date                                  現在のシステムクロックを表示
[armadillo9 ~]# date 012304562000.00                  システムクロックの設定
                        ↓設定値詳細
     01    23    04    56    2000    .00
     月    日    時    分    年      秒

[armadillo9 ~]# date
※一応 date コマンドでシステムクロックが正しく設定されているか確認する
```

●システムクロックを msntp で設定する

msntp は、SNTP プロトコルを使用してタイムサーバから時刻を取得し、システムクロックを設定するアプリケーションです。

```
[armadillo9 ~]# msntp   -r   time.server.com
                             タイムサーバアドレス
[armadillo9 ~]# date
※一応 date コマンドでシステムクロックが正しく設定されているか確認する
```

●RTC を設定する

```
[armadillo9 ~]# hwclock                               ハードウェアクロックを表示
[armadillo9 ~]# hwclock   --systohc                   ハードウェアクロックを設定
[armadillo9 ~]# hwclock
※一応 hwclock コマンドでハードウェアクロックが正しく設定されたか確認する
```

11.3. オンボードフラッシュメモリ

オンボードフラッシュメモリは、Memory Technology Device(MTD)としてリージョン単位で扱われます。オンボードフラッシュメモリのリージョンについては、「8.メモリマップについて」を参照してください。

各リージョンに対応するデバイスノードのパラメータは、以下の通りです。

表 11-3 MTD ノード

タイプ	メジャー番号	マイナー番号	ノード名 (/dev/xxx)	デバイス名
キャラクタデバイス	90	0	mtd0	bootloader
		1	mtdr0	bootloader (read only)
		2	mtd1	kernel
		3	mtdr1	kernel (read only)
		4	mtd2	userland
		5	mtdr2	userland (read only)
		6	mtd3	config
		7	mtdr3	config (read only)
ブロックデバイス	31	0	mtdblock0	bootloader
		1	mtdblock1	kernel
		2	mtdblock2	userland
		3	mtdblock3	config

11.4. USB ホスト

EP9315 は、OHCI 互換の USB ホスト機能を持っています。いくつかのデバイスについては初期状態のカーネルでドライバを有効化しており、接続するだけで使用できるようになっています。

11.4.1. USB Audio

USB オーディオ機器をサポートします。/dev/dsp(キャラクタデバイス、メジャー番号:14、マイナー番号:3)などから、一般的なサウンドデバイスとして扱うことができます。

11.4.2. USB Storage

USB メモリやディスクドライブ、メモリカードリーダなどをサポートします。Linux からは一般的な SCSI 機器と同様に認識され、/dev/sda(ブロックデバイス、メジャー番号:8、マイナー番号:0)や/dev/sda1(ブロックデバイス、メジャー番号:8、マイナー番号:1)などから扱うことができます。

11.4.3. USB Human Interface Device (HID)

USB キーボードやマウスなど、各種入力機器をサポートします。特に USB キーボードについては、VGA 出力との組み合わせで/dev/tty0 としてコンソール入力に使用できます。

11.5. IDE と Compact Flash

IDE に接続されたディスクドライブは、/dev/hda(ブロックデバイス、メジャー番号:3、マイナー番号:0)や/dev/hda1(ブロックデバイス、メジャー番号:3、マイナー番号:1)などから扱うことができます。

Compact Flash ソケットに挿入したストレージデバイスは、/dev/hdc (ブロックデバイス、メジャー番号:22、マイナー番号:0)や/dev/hdc1(ブロックデバイス、メジャー番号:22、マイナー番号:1)などから扱うことができます。初期状態のカーネルによる Compact Flash 認識の場合、Compact Flash を Armadillo-9 動作中に挿入したり、抜いたりすることはできません。活線挿抜を行ないたい場合、カーネルによる Compact Flash の認識を使用せず、PCMCIA-CS を使用してください。

注意

カーネル内蔵 Compact Flash ドライバは、カーネルコンフィギュレーションの「Device Drivers」→「ATA/ATAPI/MFM/RLL support」→「EP93xx PCMCIA IDE support」により有効化されています。このドライバは PCMCIA-CS と競合するので、PCMCIA-CS を使用する場合は「EP93xx PCMCIA IDE Support」を無効化したカーネルと組み合わせてください。

12. Compact Flash システム構築

12.1. Armadillo-9 で起動可能な Compact Flash の作成

Armadillo-9 は、Compact Flash に搭載した Linux システムから起動することができます。ここでは起動可能な Compact Flash を Armadillo-9 で作成する手順を説明します。

① Armadillo-9 の起動
Armadillo-9 に Compact Flash を挿入し、JP1、JP2 をオープンに設定してオンボードフラッシュメモリ内の Linux を起動します。

② パーティションの設定
fdisk を使用して Compact Flash に Armadillo-9 で起動可能なパーティションを作成します。起動パーティションは、パーティションタイプを 83(Linux)に設定する必要があります。

```
[armadillo9 ~]# fdisk /dev/hdc
 hdc: hdc1
Command (m for help):
```

TIPS
fdisk コマンドの例を示します。
- d コマンドで既存のパーティションを削除
- n コマンドでパーティションを作成
- t コマンドでパーティションタイプを 83(Linux)に設定
- w コマンドで設定を書き込み、fdisk を終了

③ パーティションの初期化
作成したパーティションを EXT2 ファイルシステムで初期化します。Armadillo-9 の起動パーティションは、mke2fs による初期化の際に必ず「-O none」オプションを指定する必要があります。

```
[armadillo9 ~]# mke2fs -O none /dev/hdc1
```

④ Compact Flash のマウント
Compact Flash を/mnt にマウントします。

```
[armadillo9 ~]# mount /dev/hdc1 /mnt
```

⑤ Compact Flash 上へのシステム構築
/mnt にマウントされた Compact Flash に、ルートファイルシステムを構築します。
詳しくは、「12.2.Compact Flashにルートファイルシステムを構築する」を参照してください。

⑥ Linux カーネルのコピー
Armadillo-9 を Compact Flash から起動する場合、カーネルは Compact Flash 内の/boot ディレクトリ内に非圧縮イメージ「Image」、または gz 圧縮イメージ「Image.gz」として保存しておく必要があります。/boot ディレクトリが存在しない場合は作成してください。

```
[armadillo9 ~]# mkdir /mnt/boot
```

任意のカーネルイメージをこのファイル名になるようコピーしてください。

```
armadillo9 ~]# cp linux.bin.gz /mnt/boot/Image.gz
```

※あらかじめカレントディレクトリ内にカーネルイメージlinux.bin.gzを作成しておいた場合の例です。

⑦ Compact Flash のアンマウント
Compact Flash への書き込み作業完了後、アンマウントします。

```
[armadillo9 ~]# umount /mnt
```

これで、Compact Flashへのシステム構築は完了です。Armadillo-9 を終了し、Compact Flash内のシステムから起動するよう「2.3.ジャンパピンの設定について」を参照して設定し、Armadillo-9 を再起動してください。

12.2. Compact Flash にルートファイルシステムを構築する

Compact Flash にルートファイルシステムを構築する方法として以下の内容を紹介します。

・Debian/GNU Linuxのルートファイルシステムを構築する場合
・atmark-distで作成されるルートファイルシステムを構築する場合

また、どちらの場合についても「12.1.Armadillo-9 で起動可能なCompact Flashの作成」の「④ Compact Flashのマウント」までの作業は完了しているものとします。

Compact Flashにルートファイルシステムを構築後、LinuxカーネルをCompact Flashにコピーする必要があります。「12.1.Armadillo-9 で起動可能なCompact Flashの作成」の「⑥.Linuxカーネルのコピー」を参照し、Linuxカーネルを適切にコピーしてください。

作業上、Armadillo-9 の/home/ftp/pub には特定のファイルを保存する場合があります。以下のようにRAM ファイルシステムをマウントし、書き込み権限を与えて置いてください。

```
[armadillo9 ~]# mount -t ramfs none /home/ftp/pub
[armadillo9 ~]# chmod  777 /home/ftp/pub
```

12.2.1. Debian/GNU Linux のルートファイルシステムを構築する場合

Compact Flash に Debian/GNU Linux のルートファイルシステムを構築します。Debian イメージは、付属 CD の debian ディレクトリに arm-sarge1.tar.gz～arm-sarge5.tar.gz として分割されたファイルとして用意されています。このファイルを Compact Flash へ展開する手順を説明します。

TIPS

Debian/GNU Linux のイメージを使用するには、Compact Flash の空き容量が 300MB 以上必要です。

a. ftp によるファイル転送

PC から、arm-sarge1.tar.gz を ftp で転送します。

```
[PC ~]$ ftp 192.168.10.10 ←Armadillo-9 の IP アドレス
Password:
ftp> cd pub
ftp> bin
ftp> put arm-sarge1.tar.gz
```

※IP アドレスは環境に合わせて読みかえてください。

b. Compact Flash への展開

圧縮ファイル arm-sarge1.tar.gz を Compact Flash に展開します。展開が完了したら、RAM ファイルシステム内の圧縮ファイルを削除します。

```
[armadillo9 ~]# gzip -cd /home/ftp/pub/arm-sarge1.tar.gz | (cd /mnt; tar xf -)
[armadillo9 ~]# rm /home/ftp/pub/arm-sarge1.tar.gz
```

c. 全分割ファイルの転送・展開

残りの arm-sarge2.tar.gz～arm-sarge5.tar.gz についても、⑤-b～⑤-c を繰り返します。

12.2.2. atmark-dist で作成されるルートファイルシステムを構築する場合

Compact Flash に atmark-dist で作成されるルートファイルシステムを構築します。ここでは、ユーザーランドイメージファイル（romfs.img.gz）から Compact Flash にルートファイルシステムを構築する手順を説明します。

a. romfs.img.gz を展開し romfs.img を作成する

```
[PC ~]$ gzip -dc romfs.img.gz > romfs.img
[PC ~]$ ls
romfs.img    romfs.img.gz
```

b. romfs.img をマウントする
この作業は、root ユーザで行なってください。

```
[PC ~]$ su
[PC ~]# mount -t ext2 -o loop romfs.img /mnt
```

c. ルートファイルシステムのアーカイブを作成する
一般ユーザがアーカイブファイルを扱えるようにファイルの所有者も変更します。

```
[PC ~]# (cd /mnt; tar czvf - * ) > romfs-image.tar.gz
[PC ~]# chown [user]:[group] romfs-image.tar.gz
[PC ~]# umount /mnt
[PC ~]# exit
```

d. ftp によるファイル転送
PC から、romfs-image.tar.gz を ftp で転送します。

```
[PC ~]$ ftp 192.168.10.10 ←Armadillo-9 の IP アドレス
Password:
ftp> cd pub
ftp> bin
ftp> put romfs-image.tar.gz
```

※IP アドレスは環境に合わせて読みかえてください。

e. Compact Flash への展開
圧縮ファイル romfs-image.tar.gz を Compact Flash に展開します。

```
[armadillo9 ~]# gzip -cd /home/ftp/pub/romfs-image.tar.gz | (cd /mnt; tar xf -)
[armadillo9 ~]# rm /home/ftp/pub/romfs-image.tar.gz
```

13. PCMCIA-CS 対応版ユーザーランド

13.1. カーネルとユーザーランドイメージ

PCMCIA-CS は、Compact Flash の活線挿抜や、ストレージ以外の Compact Flash カード(I/O デバイスカード)のサポートを可能にするカードサービスパッケージです。Armadillo-9 では、PCMCIA-CS を含んだユーザーランドと、組み合わせで使用するカーネルを用意しています。

・romfs-pcmcia.img.gz　PCMCIA-CS 対応ユーザーランド
・linux-pcmcia.bin.gz　PCMCIA-CS 対応ユーザーランドと同時に使用可能な Linux カーネル

注意

PCMCIA-CS に対応しない通常のカーネルは、PCMCIA-CS と競合するドライバを含みますので使用しないでください。

それぞれオンボードフラッシュメモリに書き込んで使用してください。フラッシュメモリへのイメージの書き換え方は、「5.フラッシュメモリの書き換え方法」を参照してください。

13.2. PCMCIA-CS の有効化

PCMCIA-CS を有効化するには、以下のコマンドを入力してください。

```
[armadillo9 ~]# /etc/rc.d/rc.pcmcia start
```

Compact Flash を挿入するとカードが認識され、サポートされているデバイスと判別できた場合は自動的にドライバがロードされます。
※PCMCIA-CS の対応リストに含まれないデバイスの場合、自動で認識できないことがあります。設定ファイルを書き換えることにより認識するようになる場合もありますが、詳しくは PCMCIA-CS 添付のドキュメントなどをご覧ください。

13.3. PCMCIA-CS 対応版にのみ含まれるその他のパッケージ

PCMCIA-CS 対応版ユーザーランドは、サポートパッケージとして以下を含みます。

・linux-wlan-ng　Prism2 チップを搭載した無線 LAN カード用ドライバ
(いくつかの Compact Flash 型無線 LAN カードをサポートします)
・wireless-tools　無線 LAN コントロールツール群

詳しくは、各パッケージのソース付属ドキュメントなどを参照してください。

改訂履歴

Version	年月日	改訂内容
1.0.0	2004/12/18	・初版発行
1.0.1	2004/12/28	・uClinux-dist を使用した作業について、一般ユーザで行なうよう追記 ・「注意」や「Tips」について、アイコン表記に統一
1.0.2	2005/2/11	・IDE ドライブをルートデバイスとするオプションに「noinitrd」が不足していた点を修正 ・「11.1 Armadillo-9 で起動可能な Compact Flash の作成」マウントポイント記載ミスを修正
1.0.3	2005/02/25	・GPIO にピン-レジスタ対応を追加
1.0.4	2005/03/14	・誤字の修正
1.0.6	2005/08/12	・章の構成を変更 ・distribution に関する箇所の修正／追加 ・VGA に関する箇所の修正／追加 ・「5.4.netflash を使ってフラッシュメモリの書き換えをする」を追加 ・「4.5.5.ネットワーク設定をフラッシュメモリに保存する」を追加 ・「12.Compact Flash システム構築」を修正／追加
1.0.7	2005/09/26	・「11.1.GPIO ポート」のサンプルソースを修正／追加
1.0.8	2005/10/18	・「14.1.1.coLinux のインストール」を coLinux 本体バージョンアップ対応
1.0.9	2006/01/30	・「12.2.2.e.Compact Flash への展開」の誤記を修正
1.0.10	2006/08/16	・「3.1 クロス開発環境パッケージのインストール」のパッケージ一覧に、追加されたライブラリパッケージを追記 ・「4.5 ネットワーク設定」について、atmark-dist-20060801 で変更された内容にあわせて修正 ・「5 フラッシュメモリの書き換え方法」について、hermit-at/shoehorn-at にあわせた記述内容に変更 ・誤字の修正
1.0.11	2006/10/20	・ドキュメントプロパティのタイトルと作成者を修正 ・「1.5 注意事項」を「1.5 ソフトウェアに関する注意事項」に修正 ・「1.6 保証に関する注意事項」を追加 ・「ユーザランド」を「ユーザーランド」に統一
1.0.12	2007.7.20	・「Flash メモリ」を「フラッシュメモリ」に統一 ・「7.1 ソースコードアーカイブの展開」のカーネルディレクトリへのシンボリックリンク作成に注意書きを追加 ・「3.1 クロス開発環境パッケージのインストール」へ rpm パッケージを使用した場合の注意点追記 ・「3.1 クロス開発環境パッケージのインストール」にパッケージの一括インストール方法を追加
1.0.13	2007.9.3	・ページヘッダ、フッタを追加
1.0.14	2007.9.14	・「1.6 保証に関する注意事項」の製品の保証方法を修正 ・「表 1 3 コマンド入力例での省略表記」を追加 ・コマンド入力例で、バージョン番号などの省略の表記方法を修正
1.0.15	2007.10.19	・Linux カーネル 2.6.x に対応 ・開発環境のバージョンアップに伴う記述の変更 ・「14.1. Windows 上に開発環境を構築する方法」を削除

Armadillo-9 Software Manual　　2007年10月19日　　version 1.0.15

株式会社アットマークテクノ
060-0035 札幌市中央区北5条東2丁目 AFTビル6F　　TEL:011-207-6550　FAX:011-207-6570