Armadillo-9
ソフトウェアマニュアル

AN010

Version 1.1.2-d308169
2009/08/03

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

Armadillo 開発者サイト [http://armadillo.atmark-techno.com]

Armadillo-9
ソフトウェアマニュアル

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

Version 1.1.2-d308169
2009/08/03

目次

図目次

表目次

1.はじめに

以降、本書では他の Armadillo シリーズにも共通する記述については、製品名を Armadillo と表記します。

1.1. 対象となる読者

- Armadillo のソフトウェアをカスタマイズされる方
- 外部ストレージにシステム構築される方

上記以外の方でも、本書を有効に利用していただけたら幸いです。

1.2. 本書の構成

本書は、Armadillo のソフトウェアをカスタマイズする上で必要となる情報について記載しています。

- 開発環境の構築方法
- フラッシュメモリの書き換え方法
- ビルド方法

1.3. 表記について

1.3.1. フォント

本書では以下のような意味でフォントを使いわけています。

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
text	編集する文字列や出力される文字列。またはコメント

1.3.2. コマンド入力例

本書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 root ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo /]#	Armadillo 上の root ユーザで実行
[armadillo /]$	Armadillo 上の一般ユーザで実行
hermit>	Armadillo 上の保守モードで実行

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

1.3.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

1.4. 謝辞

Armadillo 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表します。

1.5. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて

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. 商標について

Armadillo は株式会社アットマークテクノの登録商標です。その他の記載の商品名および会社名は、各社・各団体の商標または登録商標です。

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インターフェースを持つ PC です。
シリアルクロスケーブル	D-Sub9 ピン (メス - 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「`tool`」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、シリアル変換ケーブル、LAN ケーブル、AC アダプター変換ケーブル、AC アダプターを Armadillo-9 に接続してください。

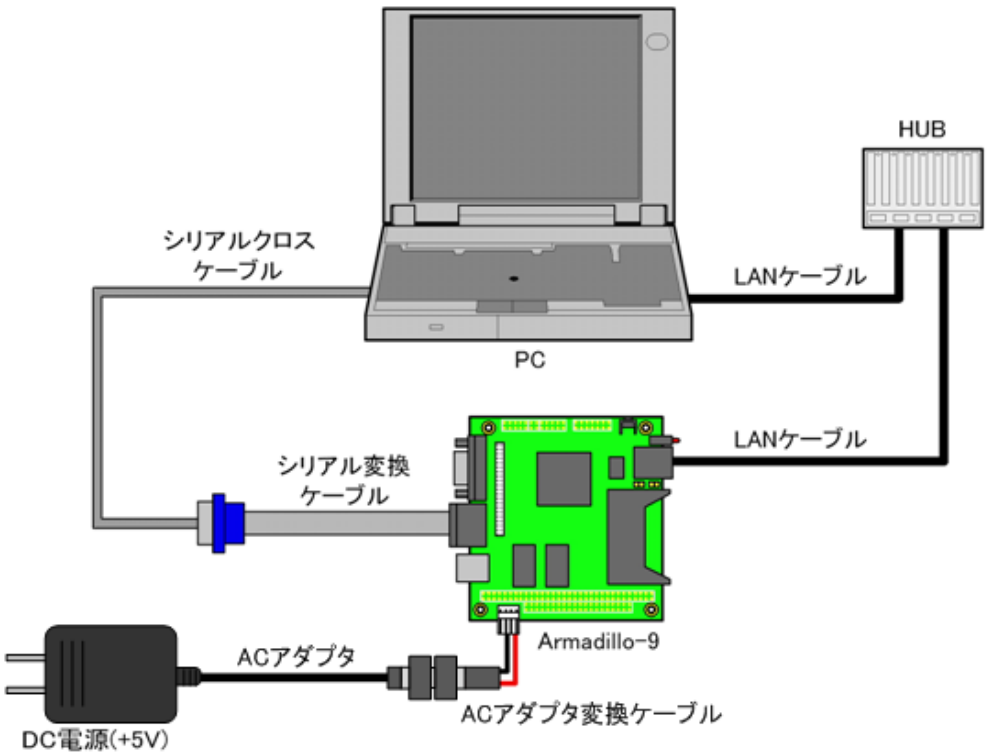

図 2.1. Armadillo-9 接続例

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	「12.2. コンパクトフラッシュの初期化」で作成された IDE ドライブ、またはコンパクトフラッシュが接続されている場合 [1] IDE ドライブまたはコンパクトフラッシュ内の 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→コンパクトフラッシュの順です。
[3] ブートローダーの復旧などに使用します。

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

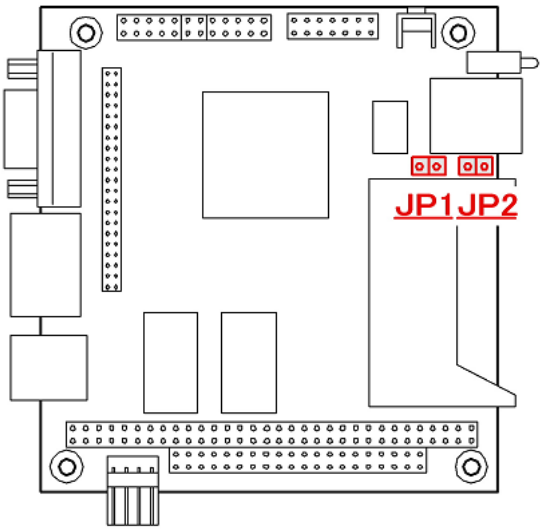

図 2.2. ジャンパの位置

3.開発環境の準備

Armadillo 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. 開発環境一覧」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 root ユーザで行ってください。「図 3.1. インストールコマンド」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 --install *.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 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. atmark-dist のビルドに必要なパッケージ一覧」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 cross-dev 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. インストール情報表示コマンド」のようにパッケージ名を指定して実行してください。

`--list` はパッケージ情報を表示する dpkg のオプションです。`file` にはバージョンを表示したいパッケージ名を指定します。

```
[PC ~]# dpkg --list file
```

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_[version]_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 の基本的な使用方法の説明を行います。

4.1. シリアル通信ソフトウェアの設定

シリアル通信ソフトウェアを起動し、シリアルの通信設定を、「表 4.1. シリアル通信設定」のように設定してください。

Armadillo-240 では、RS232C レベル変換アダプターを経由させる必要があります。

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インターフェース ttyAM0(CON1) に加え、VGA にも表示されます。[1]

4.3. コンソールログイン時のユーザ名とパスワード

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	root ユーザ
guest	(なし)	一般ユーザ

4.4. 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

4.5. 終了

電源を切断することで Armadillo を終了させます。

ただし IDE ドライブやコンパクトフラッシュ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。これを行わない場合は、IDE ドライブやコンパクトフラッシュのデータが破損する恐れがあります。

4.6. ネットワーク設定

Armadillo の「/etc/config/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。Armadillo-230 はネットワークインターフェイスを 2 つ搭載しているため、通常の

[1]VGA 側からログインを行うには、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. Linux ブートオプション」を参照してください。

eth0 に加え eth1 も存在します。USB のインターフェイスを持つ Armadillo で USB 対応 LAN アダプターを使用する場合も同じです。eth1 側を設定する場合、以降 eth0 の個所を eth1 に読み替えてください。また詳しい interfaces の書き方については、interfaces のマニュアルを参照してください。

4.6.1. 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. ネットワーク設定例

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/config/interfaces - configuration file for ifup(8), ifdown(8)auto lo eth0
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)

4.6.2. 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。

4.6.3. 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/config/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)

4.6.4. ネットワーク接続の開始と終了

ネットワーク接続を開始するには ifup を、ネットワーク接続を終了するには ifdown というコマンドを使用します。コマンドには開始または終了させたいインターフェイスを指定してください。

```
[armadillo /]# ifup eth0
```

図 4.6. ネットワーク接続の開始

```
[armadillo /]# ifdown eth0
```

図 4.7. ネットワーク接続の終了

4.6.5. ネットワーク設定をフラッシュメモリに保存する

ネットワーク設定に必要なファイルは、/etc/config/ディレクトリにあります。このディレクトリにあるファイルをフラッシュメモリに保存するには、flatfsd というコマンドを使います。オプション「-s」を指定し、Armadillo 上で flatfsd を実行してください。

```
[armadillo /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。

4.7. 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 でのログインは行えません。root 権限が必要な作業を行う場合、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

Armadillo-220/230/240 の Recover イメージ(出荷状態)の起動直後の状態では、telnet ログインをすることができません。telnet ログインをするには、/etc/inetd.conf を編集し、以下のコマンドを実行してください。

```
telnet  stream  tcp   nowait   root    /usr/sbin/telnetd      telnetd -l /bin/login
```

図 4.8. /etc/inetd ファイル編集例

```
[armadillo ~]# iptables --append INPUT --proto tcp --dport telnet --jump ACCEPT
```

図 4.9. ファイアウォールの設定コマンド入力例

```
[armadillo ~]# inetd
```

図 4.10. スーパーサーバ起動コマンド

4.8. 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

Armadillo-220/230/240 の Recover イメージ(出荷状態)の起動直後の状態では、ftp によるファイル転送をすることができません。ftp によるファイル転送をするには、 「図 4.10. スーパーサーバ起動コマンド」 を実行してください。

4.9. Web サーバ

thttpd という小さな HTTP サーバが起動しており、Web ブラウザを使って Armadillo にアクセスすることができます。データディレクトリは「/home/www-data」です。URL は「http://(Armadillo-240 の IP アドレス)/」になります。(例 http://192.168.10.10/)

4.10. ssh ログイン

次のユーザ名/パスワードで ssh ログインが可能です。root でのログインは行えません。root 権限が必要な作業を行う場合、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 4.7. ssh 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

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 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。

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 が正常に起動しなくなる場合があります。書き換えの際は次の点に注意してください。

- Armadillo の電源を切らない
- Armadillo と開発用 PC を接続しているシリアルケーブルと LAN 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 にダウンローダをインストールします。

ダウンローダの種類には、「表 5.1. ダウンローダ一覧」のようなものがあります。

表 5.1. ダウンローダ一覧

ダウンローダ	OS タイプ	説明
hermit-at	Linux	Linux 用の CUI アプリケーションです。
shoehorn-at	Linux	Linux 用の CUI アプリケーションです。
hermit-at-win	Windows	Windows 用の GUI アプリケーションです。

ATDE(Atmark Techno Development Environment)を利用する場合、ダウンローダパッケージはすでにインストールされているので、インストールする必要はありません。

5.1.1. 作業用 PC が 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 **root ユーザ**で行ってください。

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

図 5.1. ダウンローダのインストール（Linux）

5.1.2. 作業用 PC が Windows の場合

付属 CD の `downloader/win32/hermit-at-win_[version].zip` を任意のフォルダに展開します。

5.2. フラッシュメモリの書き込み領域について

フラッシュメモリの書き込み先頭アドレスは、領域（リージョン）名で指定することができます。書き込み領域毎に指定するイメージファイルは、「表 5.2. リージョン名と対応するイメージファイル 」のようになります。

表 5.2. リージョン名と対応するイメージファイル [1]

製品	領域名	ファイル名
Armadillo-210	bootloader	loader-armadillo2x0-*[version]*.bin
	kernel	linux-a210-*[version]*.bin.gz
	userland	romfs-a210-recover-*[version]*.img.gz romfs-a210-base-*[version]*.img.gz
Armadillo-220/230/240	bootloader	loader-armadillo2x0-eth-*[version]*.bin
	kernel	linux-a2x0-*[version]*.bin.gz
	userland	romfs-a2x0-recover-*[version]*.img.gz romfs-a2x0-base-*[version]*.img.gz
Armadillo-9	bootloader	loader-armadillo9-*[version]*.bin
	kernel	linux-*[version]*.bin.gz
	userland	romfs-*[version]*.img.gz
Armadillo-300	ipl	ipl-a300.bin(※書き換え不可)
	bootloader	loader-armadillo-3x0-*[version]*.bin
	kernel	linux-a300-*[version]*.bin.gz
	userland	romfs-a300-*[version]*.img.gz
Armadillo-500	bootloader	loader-armadillo5x0-*[version]*.bin
	kernel	linux-a500-*[version]*.bin.gz
	userland	romfs-a500-*[version]*.img.gz
Armadillo-500 FX	bootloader	loader-armadillo5x0-fx-*[version]*.bin
	kernel	linux-a500-fx-*[version]*.bin.gz
	userland	romfs-a500-fx-*[version]*.img.gz

[1]「x」にはバージョン番号の任意の数値が入ります。

一部製品のユーザーランドには、Recover と Base という 2 種類のイメージファイルが用意されています。Recover イメージは、出荷状態でオンボードフラッシュメモリに書き込まれていて、各製品の特徴や性能を利用するアプリケーションが含まれています。Base イメージは、開発のベースとなるように、基本的なアプリケーションやツールのみが含まれています。

5.3. Hermit-At ダウンローダを使用してフラッシュメモリを書き換える

ここでは、Hermit-At ダウンローダを使用してフラッシュメモリを書き換える手順について説明します。「5.1. ダウンローダのインストール」でインストールした Hermit-At ダウンローダを使用します。これは、Armadillo のブートローダーと協調動作を行い、作業用 PC から Armadillo のフラッシュメモリを書き換えることができます。

5.3.1. 準備

「2.3. ジャンパピンの設定について」を参照し、Hermit-At を起動してください。

Armadillo と接続している作業用 PC のシリアルインターフェースが他のアプリケーションで使用されていないことを確認します。使用されている場合は、該当アプリケーションを終了するなどしてシリアルインターフェースを開放してください。

5.3.2. 作業用 PC が Linux の場合

「図 5.2. ダウンロードコマンド」のようにコマンドを実行します。

`download` は hermit のサブコマンドの一つです。`--input-file` で指定されたファイルをターゲットボードに書き込む時に使用します。`--region` は書き込み対象の領域を指定するオプションです。下記の例では、「kernel 領域に linux.bin.gz を書き込む」という命令になります。

```
[PC ~]$ hermit download --input-file linux.bin.gz --region kernel
```

図 5.2. ダウンロードコマンド

シリアルインターフェースが ttyS0 以外の場合は、「図 5.3. ダウンロードコマンド（ポート指定） 」のように`--port` オプションを使用してポートを指定してください。

```
[PC ~]$ hermit download --input-file linux.bin.gz --region kernel --port ttyS1
```

図 5.3. ダウンロードコマンド（ポート指定） [1]

bootloader リージョンは、誤って書き換えることがないように簡易プロテクトされています。書き換える場合は、「図 5.4. ダウンロードコマンド（アンプロテクト）[1]」のように`--force-locked` オプションを使用して、プロテクトの解除をしてください。

```
[PC ~]$ hermit download --input-file loader-armadillo5x0-fx.bin --region
bootloader --force-locked
```

図 5.4. ダウンロードコマンド（アンプロテクト） [1]

[1] コマンドは 1 行で入力します。

bootloader リージョン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「5.5. 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。

5.3.3. 作業用 PC が Windows の場合

hermit-at-win.exe を実行します。「図 5.5. Hermit-At：Download ウィンドウ」が表示されます。

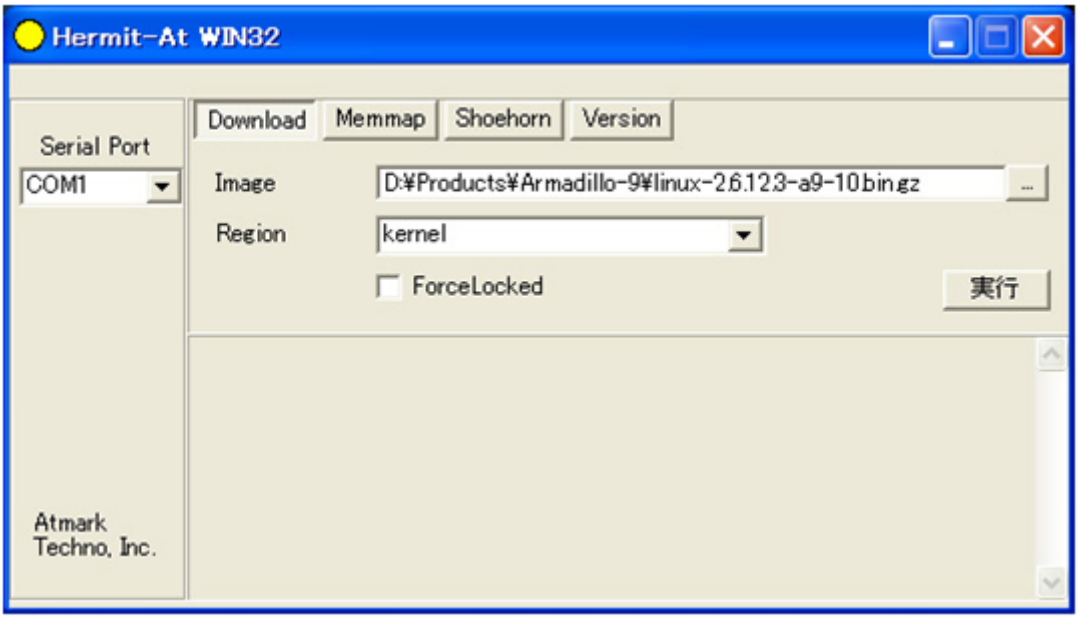

図 5.5. Hermit-At：Download ウィンドウ

Armadillo と接続されているシリアルインターフェースを「Serial Port」に指定してください。ドロップダウンリストに表示されない場合は、直接ポートを入力してください。

Image には書き込むファイルを指定してください。Region には書き込み対象のリージョンを指定してください。all や bootloader リージョンを指定する場合は、Force Locked をチェックしてください。

すべて設定してから実行ボタンをクリックします。「図 5.6. Hermit-At：download ダイアログ」が表示されます。

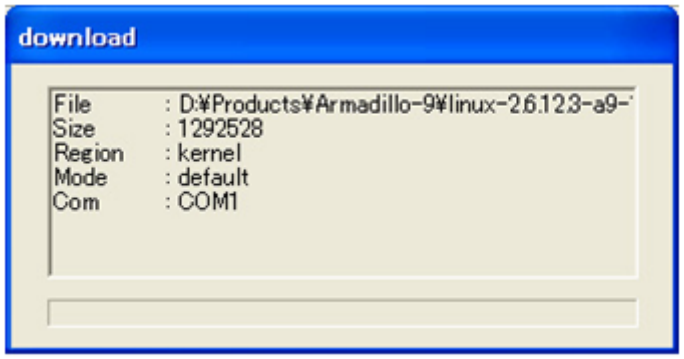

図 5.6. Hermit-At：download ダイアログ

ダウンロードの設定と進捗状況が表示されます。ダウンロードが完了するとダイアログはクローズされます。

5.4. netflash を使用してフラッシュメモリを書き換える

Linux アプリケーションの netflash を使用してフラッシュメモリを書き換えることができます。netflash は、所属するネットワークにある HTTP サーバーや FTP サーバーが公開しているファイルをダウンロードしてフラッシュメモリを書き換えることができます。

Armadillo にログインし、「図 5.7. netflash コマンド例」のようにコマンドを実行します。

```
[armadillo ~]# netflash -k -n -u -r /dev/flash/kernel [URL]
```

図 5.7. netflash コマンド例

オプションの"-r [デバイスファイル名]"で書き込み対象のリージョンを指定しています。「表 5.3. リージョンとデバイスファイルの対応」を参照してください。その他のオプションについては、netflash -h で詳細を確認する事ができます。

表 5.3. リージョンとデバイスファイルの対応

リージョン	デバイスファイル
カーネル	/dev/flash/kernel
ユーザランド	/dev/flash/userland

5.5. ブートローダーを出荷状態に戻す

`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。

5.5.1. ブートローダーの種類

Armadillo には複数のブートローダーが用意されています。ブートローダーの一覧は、「7.3. ブートローダーイメージのビルド」を参照してください。

5.5.2. 準備

Armadillo の電源が**切断**されていることを確認し、Armadillo のジャンパ JP1 をショートに設定してください。

Armadillo と接続している作業用 PC のシリアルインターフェースが他のアプリケーションで使用されていないことを確認します。使用されている場合は、該当アプリケーションを終了するなどしてシリアルインターフェースを開放してください。

5.5.3. 作業用 PC が Linux の場合

「図 5.8. shoehorn コマンド例」のようにコマンド[2]を実行してから、Armadillo の電源を入れてください。

```
[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 5.8. shoehorn コマンド例

上記は、作業用 PC のシリアルインターフェース"/dev/ttyS0"に Armadillo を接続した場合の例です。他のシリアルインターフェースに接続した場合は、shoehorn コマンドのオプションに

--port [シリアルインターフェース名]

を追加してください。

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、Armadillo の電源を切断し、シリアルケーブルの接続や Armadillo のジャンパ設定を再度確認してください。

この状態で、**ジャンパの設定変更や電源の切断をしないで** Ctrl-C を押して Shoehorn を終了してから、「5.3. Hermit-At ダウンローダを使用してフラッシュメモリを書き換える」を参照してブートローダの書き込みを行ってください。

5.5.4. 作業用 PC が Windows の場合

`hermit-at-win.exe` を実行し Shoehorn ボタンをクリックすると、「図 5.9. Hermit-At：Shoehorn ウィンドウ」が表示されます。

[2] 書面の都合上折り返して表記しています。通常は 1 行のコマンドとなります。

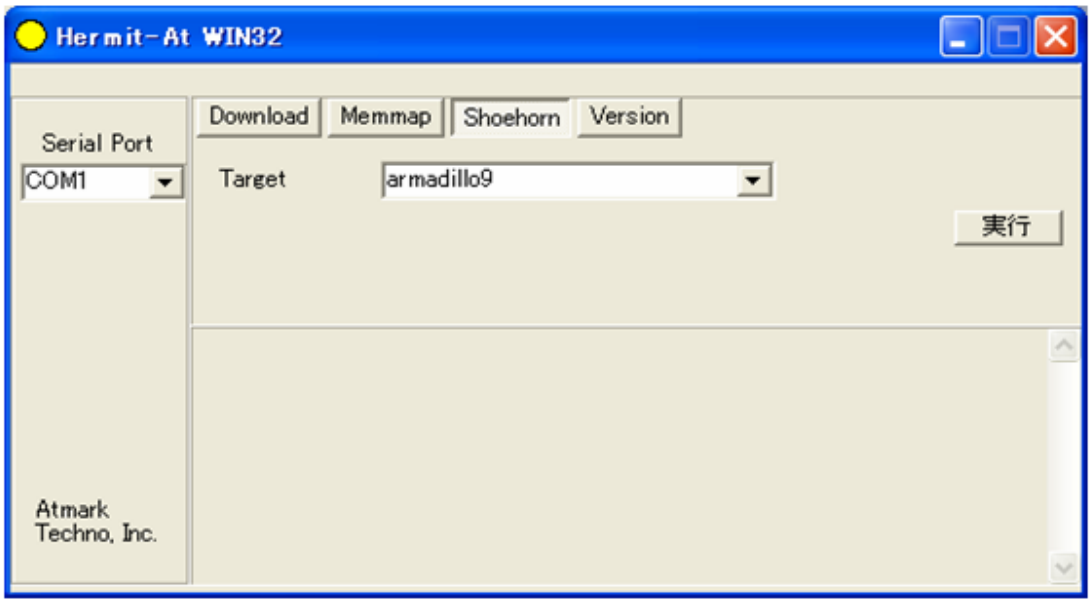

図 5.9. Hermit-At：Shoehorn ウィンドウ

Target に armadillo9 を選択して実行ボタンをクリックします。

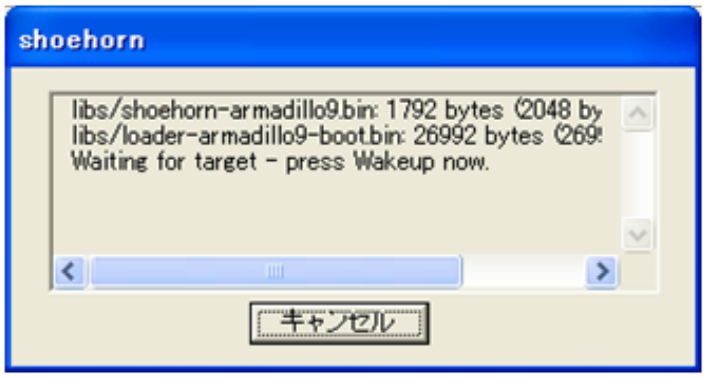

図 5.10. Hermit-At：shoehorn ダイアログ

ダイアログの表示後、Armadillo の電源を入れてください。

すぐにダイアログにメッセージが表示されます。正常に表示されない場合、Armadillo の電源を切断し、シリアルケーブルの接続や Armadillo のジャンパ設定を再度確認してください。

ダウンロードするための準備が完了するとダイアログは自動的にクローズされます。

この状態で、**ジャンパの設定変更や電源の切断をしないで**「5.3. Hermit-At ダウンローダを使用してフラッシュメモリを書き換える」を参照してブートローダの書き込みを行ってください。

6.Linux ブートオプション

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います。

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。

6.1. 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ってください。

   表 6.1. 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

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、及びコンパクトフラッシュ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>
```

6.2. 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
```

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インターフェース 1 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ってから再度電源を入れ直してください。

6.3. 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
```

6.4. 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. 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
```

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

7.ビルド

この章では、ソースコードからデフォルトイメージを作成する手順を説明します。以下の例では、作業ディレクトリとしてホームディレクトリ（~/）を使用していきます。

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・配置を行います。各ファイルは作業ディレクトリ配下で作成・配置作業を行いますが、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**で行ってください。

7.1. カーネルイメージとユーザーランドイメージのビルド

ここでは、付属 CD に収録されているデフォルトイメージを作成してみます。開発環境を構築してない場合は、「3. 開発環境の準備」を参照して作業用 PC に開発環境を構築してください。

7.1.1. ソースコードの準備

付属 CD の source/dist にある atmark-dist.tar.gz と source/kernel にある linux.tar.gz を作業ディレクトリに展開します。展開後、atmark-dist にカーネルソースを登録します。「図 7.1. ソースコード準備」のように作業してください。

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

図 7.1. ソースコード準備

7.1.2. コンフィグレーション

ターゲットボード用の dist をコンフィグレーションします。以下の例のようにコマンドを入力し、コンフィグ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
```

次にプロダクト名を聞かれます。「表 7.1. プロダクト名一覧」から、使用する製品に対応するプロダクト名を入力してください。

表 7.1. プロダクト名一覧

製品	プロダクト名	備考
Armadillo-210	Armadillo-210.Base	
	Armadillo-210.Recover	出荷時イメージ
Armadillo-220	Armadillo-220.Base	
	Armadillo-220.Recover	出荷時イメージ
Armadillo-230	Armadillo-230.Base	
	Armadillo-230.Recover	出荷時イメージ
Armadillo-240	Armadillo-240.Base	
	Armadillo-240.Recover	出荷時イメージ
Armadillo-9	Armadillo-9	出荷時イメージ
	Armadillo-9.PCMCIA	
Armadillo-300	Armadillo-300	出荷時イメージ
Armadillo-500	Armadillo-500	出荷時イメージ
Armadillo-500 FX	Armadillo-500-FX.dev	出荷時イメージ

以下は、Armadillo-210.Base の例です。

```
*
* Select the Product you wish to target
*
AtmarkTechno Products (Armadillo-210.Base, Armadillo-210.Recover,
Armadillo-220.Base, Armadillo-220.Recover, Armadillo-230.Base,
Armadillo-230.Recover, Armadillo-240.Base, Armadillo-240.Recover, Armadillo-300,
Armadillo-500, Armadillo-500-FX.dev, Armadillo-9, Armadillo-9.PCMCIA, SUZAKU-
V.SZ310, SUZAKU-V.SZ310-SIL, SUZAKU-V.SZ410, SUZAKU-V.SZ410-SIL)
[Armadillo-210.Base] (NEW) Armadillo-210.Base
```

ビルドする開発環境を聞かれます。「default」と入力してください。

```
*
* Kernel/Library/Defaults Selection
*
```

```
*
* Kernel is linux-2.6.x
*
Cross-dev (default, arm-vfp, arm, armnommu, common, h8300, host, i386, i960,
m68knommu, microblaze, mips, powerpc, sh) [default] (NEW) default
```

使用する C ライブラリを指定します。「None」を選択してください。

```
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。

7.1.3. ビルド

ビルドするには、atmark-dist ディレクトリで「図 7.2. ビルド」のようにコマンドを実行します。ビルドが完了すると、`atmark-dist/images` ディレクトリに `linux.bin.gz` と `romfs.img.gz` が作成されます。

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


[PC ~/atmark-dist]$ ls images
linux.bin  linux.bin.gz  romfs.img  romfs.img.gz
```

図 7.2. ビルド

7.2. ユーザーランドイメージをカスタマイズする

自作のアプリケーションを`/bin` に追加したユーザーランドイメージの作成方法について説明します。ここでは、「7.1. カーネルイメージとユーザーランドイメージのビルド」が完了している前提で説明します。

自作アプリケーションは、`~/sample/hello` にある仮定とします。

```
[PC ~/atmark-dist]$ cp ~/sample/hello romfs/bin/
[PC ~/atmark-dist]$ make image


[PC ~/atmark-dist]$ ls images
linux.bin  linux.bin.gz  romfs.img  romfs.img.gz
```

図 7.3. ユーザーランドイメージのカスタマイズ

できた romfs.img 及び romfs.img.gz の/bin には、hello がインストールされています。

7.3. ブートローダーイメージのビルド

7.3.1. ソースコードの準備

付属 CD の source/bootloader にある hermit-at-*[version]*-source.tar.gz を作業ディレクトリに展開します。「図 7.4. ソースコード展開例」のように作業してください。

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

図 7.4. ソースコード展開例

7.3.2. ビルド

ビルドオプションに TARGET と PROFILE を指定します。製品毎にパラメータが異なりますので、「表 7.2. ビルドオプション一覧」を参照してください。

また、生成されるイメージファイル名は loader-*[TARGET]*-*[PROFILE]*.bin(PROFILE が未指定の場合は loader-*[TARGET]*.bin)になります。

表 7.2. ビルドオプション一覧

製品	TARGET	PROFILE	説明
Armadillo-210 Armadillo-220 Armadillo-230 Armadillo-240	armadillo2x0	指定なし	hermit コンソールにシリアルインターフェース 1 を使用。
		eth	出荷時イメージ。 hermit コンソールにシリアルインターフェース 1 を使用。 tftp によるフラッシュメモリ書き換えが可能。
		ttyAM1	hermit コンソールにシリアルインターフェース 2 を使用。
		notty	hermit コンソールにシリアルインターフェースを使用しない。
		boot	Shoehorn-At で使用。
		boot-eth	Shoehorn-At で使用。 LAN 経由でのフラッシュメモリ書き換えが可能。

製品	TARGET	PROFILE	説明
Armadillo-9	armadillo9	指定なし	出荷時イメージ。 hermit コンソールにシリアルインターフェース 1 を使用。
		eth	hermit コンソールにシリアルインターフェース 1 を使用。 tftp によるフラッシュメモリ書き換えが可能。
		ttyAM1	hermit コンソールにシリアルインターフェース 2 を使用。
		notty	hermit コンソールにシリアルインターフェースを使用しない。
		boot	Shoehorn-At で使用。
		boot-eth	Shoehorn-At で使用。 LAN 経由でのフラッシュメモリ書き換えが可能。
Armadillo-300	armadillo3x0	指定なし	hermit コンソールにシリアルインターフェース 2 を使用。
		eth	出荷時イメージ。 hermit コンソールにシリアルインターフェース 2 を使用。 tftp によるフラッシュメモリ書き換えが可能。
		ttyAM1	hermit コンソールにシリアルインターフェース 1 を使用。
		notty	hermit コンソールにシリアルインターフェースを使用しない。
		boot	Shoehorn-At で使用。
		boot-eth	Shoehorn-At で使用。 LAN 経由でのフラッシュメモリ書き換えが可能。
Armadillo-500 Armadillo-500 FX	armadillo5x0	指定なし	Armadillo-500 開発ボード用のイメージ。
		fx	Armadillo-500 FX 液晶モデル用のイメージ。
		boot	Shoehorn-At で使用。
		zero	Armadillo-500 CPU モジュール単体用のイメージ。

例えば、Armadillo-210(PROFILE=指定なし)の場合「図 7.5. ビルド例 1」のように実行します。

```
[PC ~]$ cd hermit-at-[version]
[PC ~/hermit-at]$ make TARGET=armadillo2x0


[PC ~/hermit-at]$ ls src/target/armadillo2x0/*.bin
loader-armadillo2x0.bin
```

図 7.5. ビルド例 1

同様に、Armadillo-500 FX の場合「図 7.6. ビルド例 2」のように実行します。

```
[PC ~]$ cd hermit-at-[version]
[PC ~/hermit-at]$ make TARGET=armadillo5x0 PROFILE=fx


[PC ~/hermit-at]$ ls src/target/armadillo5x0/*.bin
loader-armadillo5x0-fx.bin
```

図 7.6. ビルド例 2

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)[1]

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

[1]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	UARROWX1
24	IRQ_UARTTX1	VIC #24	UARTTX1
25	IRQ_UARTRX2	VIC #25	UARROWX2
26	IRQ_UARTTX2	VIC #26	UARTTX2
27	IRQ_UARTRX3	VIC #27	UARROWX3
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

Linux での IRQ 番号	名称	インタラプトソース	説明
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 [1]

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

用途	Linux の IRQ 番号から PC/104 の IRQ 番号へ変換
	const unsigned int isa_irq = 3; unsigned linux_irq; linux_irq = convirq_from_isa(isa_irq);

[1](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。

現在のソフトウェアでは、デバイスが提供する設定の全てに対応していません。また、Armadillo-9 の VGA 出力は、VESA などの規格化されているタイミングを完全に満しているわけではありません。そのため、許容範囲の狭いモニタでは同期ずれが起こる場合があります。

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。

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. Linux ブートオプション」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
800x600 75Hz	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

以下は 800x600 60Hz, 8 ビットカラーの設定例です。

```
hermit> setenv video=ep93xxfb:CRT-800x600,8bpp
```

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

ピン名	ポート	アドレス
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

```
   ↓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,
};
```

図 11.1. ep93xx_gpio.h の構造体とマクロ定義

```
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 $@ $<
```

図 11.2. GPIO 操作のサンプル Makefile

```
#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;
}
```

図 11.3. GPIO 操作のサンプルプログラム(sample.c)

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、ハードウェアマニュアルの「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

  date コマンドの引数で渡す時刻のフォーマットは[MMDDhhmmCCYY.ss]となります。以下の例では、2000 年 1 月 23 日 4 時 56 分 00 秒に設定しています。

```
[armadillo ~]# date 現在のシステムクロックを表示
[armadillo ~]# date 012304562000.00
[armadillo ~]# 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。

  以下は、SNTP サーバー time.server.com から時刻を取得する例です。

```
[armadillo ~]# msntp -r time.server.com
[armadillo ~]# 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

```
[armadillo ~]# hwclock ハードウェアクロックを表示
[armadillo ~]# hwclock --systohc ハードウェアクロックを設定
[armadillo ~]# 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)

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
		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 とコンパクトフラッシュ

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。

コンパクトフラッシュ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コンパクトフラッシュ認識の場合、コンパクトフラッシュを 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。活線挿抜を行いたい場合、カーネルによるコンパクトフラッシュの認識を使用せず、PCMCIA-CS を使用してください。

カーネル内蔵コンパクトフラッシュドライバは、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.コンパクトフラッシュシステム構築

12.1. コンパクトフラッシュシステム例

Armadillo では、コンパクトフラッシュに Linux システムを構築することができます。この章では、起動可能なコンパクトフラッシュシステムの構築手順について説明します。

ブートローダがカーネルイメージを読み込むことができるファイルシステムは、EXT2 ファイルシステムとなっています。

この章では、「表 12.1. コンパクトフラッシュシステム例」のようなコンパクトフラッシュシステムを例に、構築手順を説明します。

表 12.1. コンパクトフラッシュシステム例

パーティション[1]	タイプ	容量	説明
/dev/hda1	ext2	32MB	起動パーティション。 カーネルイメージを配置する領域です。
/dev/hda2	ext3	-	ルートファイルシステムを配置する領域です。

[1]Armadillo-9 の場合、パーティション名は/dev/hdc1,/dev/hdc2 となります。以降、適宜読み替えてください。

12.2. コンパクトフラッシュの初期化

ここでは、コンパクトフラッシュをフォーマットし、パーティション 1 に EXT2 ファイルシステムを、パーティション 2 に EXT3 ファイルシステムを作成するところまでの手順を説明します。

作業の前に、ジャンパピンを以下のように設定してください。

表 12.2. コンパクトフラッシュ初期化時のジャンパピン設定

製品	ジャンパピン設定
Armadillo-9	JP1:オープン JP2:オープン
Armadillo-300	JP1:1-2

12.2.1. ディスクフォーマット

「図 12.1. ディスク初期化方法」のように、ディスクをフォーマットします。

```
[armadillo ~]# fdisk /dev/hda
The number of cylinders for this disk is set to 1324.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): d
No partition is defined yet!

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1324, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1324, default 1324): +32M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (85-1324, default 85):
Using default value 85
Last cylinder or +size or +sizeM or +sizeK (85-1324, default 1324):
Using default value 1324

Command (m for help): p

Disk /dev/hda: 512 MB, 512483328 bytes
12 heads, 63 sectors/track, 1324 cylinders
Units = cylinders of 756 * 512 = 387072 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1               1          84       31720+  83  Linux
/dev/hda2              85        1324      468720   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
 hda: hda1 hda2
 hda: hda1 hda2
Syncing disks.
```

図 12.1. ディスク初期化方法

12.2.2. ファイルシステムの作成

「図 12.2. ファイルシステムの構築」のように初期化したディスクのパーティションにファイルシステムを作成します。

mke2fs で起動パーティション（カーネルイメージを配置するパーティション）に EXT2 ファイルシステムを作成する場合は、必ず「**-O none**」オプションを指定する必要があります。

```
[armadillo ~]# mke2fs -O none /dev/hda1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
7936 inodes, 31720 blocks
1586 blocks (5%) reserved for the super user
First data block=1
4 block groups
8192 blocks per group, 8192 fragments per group
1984 inodes per group
Superblock backups stored on blocks:
        8193, 16385, 24577

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
[armadillo ~]# mke2fs -j /dev/hda2
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
117392 inodes, 468720 blocks
23436 blocks (5%) reserved for the super user
First data block=1
58 block groups
8192 blocks per group, 8192 fragments per group
2024 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
```

図 12.2. ファイルシステムの構築

12.3. カーネルイメージを配置する

コンパクトフラッシュシステムから起動する場合は、起動パーティションの/boot ディレクトリにカーネルイメージを配置する必要があります。対応しているカーネルイメージは、非圧縮カーネルイメージ（Image linux.bin）または、圧縮イメージ（Image.gz linux.bin.gz）のどちらかになります。

ここで説明する例では、カーネルイメージの取得に wget コマンドを使用します。wget コマンドで指定する URL は製品によって異なりますので、以下の表を参照し適宜読み替えてください。

表 12.3. カーネルイメージのダウンロード先 URL

製品	URL
Armadillo-9	http://download.atmark-techno.com/armadillo-9/image/linux-*[version]*.bin.gz
Armadillo-300	http://download.atmark-techno.com/armadillo-300/image/linux-a300-*[version]*.bin.gz
Armadillo-500	http://download.atmark-techno.com/armadillo-500/image/linux-a500-*[version]*.bin.gz

以下に Armadillo-500 での配置例を示します。

```
[armadillo ~]# mount /dev/hda1 /mnt
[armadillo ~]# mkdir /mnt/boot
[armadillo ~]# cd /mnt/boot
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-500/image/linux-
a500-[version].bin.gz
Connecting to download.atmark-techno.com [210.191.215.172]:80
linux-a500-[version].bin.gz 100% |**********************|  **** KB    00:00 ETA
[armadillo ~]# mv linux-a500-[version].bin.gz /mnt/boot/Image.gz
[armadillo ~]# sync
[armadillo ~]# umount /mnt
```

図 12.3. カーネルイメージの配置

12.4. ルートファイルシステムの構築

ここでは、コンパクトフラッシュにルートファイルシステムを構築する手順について説明します。

12.4.1. Debian GNU/Linux を構築する

Debian を構築する場合、付属 CD の debian ディレクトリ以下のアーカイブを使用するか、弊社ダウンロードサイトからアーカイブを取得します。これは、純粋な Debian でインストールされるファイルを分割してアーカイブ化したものとなります。これらをファイルシステム上に展開することでルートファイルシステムを構築することができます。

ルートファイルシステムに Debian を構築する場合は、パーティションの空き容量が最低でも 256MB 必要です。

ここで説明する例では、debian アーカイブの取得に wget コマンドを使用します。wget コマンドで指定する URL は製品によって異なりますので、以下の表を参照し適宜読み替えてください。

表 12.4. debian アーカイブのダウンロード先 URL

製品	URL
Armadillo-9	http://download.atmark-techno.com/armadillo-9/debian/debian-etch-a9-#.tgz
Armadillo-300	http://download.atmark-techno.com/armadillo-300/debian/debian-etch-a300-#.tgz
Armadillo-500	http://download.atmark-techno.com/armadillo-500/debian/debian-etch-arm#.tgz

以下に Armadillo-500 での構築例を示します。

```
[armadillo ~]# mount /dev/hda2 /mnt
[armadillo ~]# mount -t ramfs ramfs /tmp
[armadillo ~]# cd /tmp

[LOOP]  debian-etch-arm#.tgzの#の部分を 1~5 まで繰り返します。

[armadillo /tmp]# wget http://download.atmark-techno.com/armadillo-500/debian/
debian-etch-arm#.tgz
Connecting to download.atmark-techno.com [210.191.215.172]:80
debian-etch-#.tgz    100% |************************|  **** KB    00:00 ETA
[armadillo /tmp]# gzip -cd debian-etch-arm#.tgz | (cd /mnt; tar xf -)
[armadillo /tmp]# sync
[armadillo /tmp]# rm -f debian-etch-arm#.tgz

[LOOP] に戻る

[armadillo /tmp]# umount /mnt
```

図 12.4. Debian アーカイブの構築例

12.4.2. atmark-dist イメージから構築する

atmark-dist で作成されるシステムイメージをコンパクトフラッシュのルートファイルシステムとして構築する方法を説明します。Debian を構築する場合に比べ、ディスク容量の少ないコンパクトフラッシュへシステムを構築することができます。

ここで説明する例では、atmark-dist イメージの取得に wget コマンドを使用します。wget コマンドで指定する URL は製品によって異なりますので、以下の表を参照し適宜読み替えてください。

表 12.5. atmark-dist イメージのダウンロード先 URL

製品	URL
Armadillo-9	http://download.atmark-techno.com/armadillo-9/image/romfs-*[version]*.img.gz
Armadillo-300	http://download.atmark-techno.com/armadillo-300/image/romfs-a300-*[version]*.img.gz
Armadillo-500	http://download.atmark-techno.com/armadillo-500/image/romfs-a500-*[version]*.img.gz

以下に Armadillo-500 での構築例を示します。

```
[armadillo ~]# mount -t ramfs ramfs /tmp
[armadillo ~]# cd /tmp
[armadillo /tmp]# wget http://download.atmark-techno.com/armadillo-500/images/
romfs-a500-[version].img.gz
Connecting to download.atmark-techno.com [210.191.215.172]:80
romfs-a500-1.00.img.gz 100% |**********************|  **** KB    00:00 ETA
[armadillo /tmp]# gzip -dc romfs-a500-[version].img.gz > romfs.img
[armadillo /tmp]# mount /dev/hda2 /mnt
[armadillo /tmp]# mkdir romfs
[armadillo /tmp]# mount -o loop romfs.img romfs
[armadillo /tmp]# (cd romfs/; tar cf - *) | (cd /mnt; tar xf -)
[armadillo /tmp]# sync
[armadillo /tmp]# umount romfs
[armadillo /tmp]# umount /mnt
```

図 12.5. romfs.img.gz からの作成例

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、コンパクトフラッシュの活線挿抜や、ストレージ以外のコンパクトフラッシュカード(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

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。

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

コンパクトフラッシュ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 (いくつかのコンパクトフラッシュ型無線 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

バージョン	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	・lDE 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. GPlO ポート」マウントポイント記載ミスを修正
1.0.3	2005/02/25	・GPlO にピン-レジスタ対応を追加
1.0.3	2005/02/25	・GPlO 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 を使用してフラッシュメモリを書き換える」を追加 ・「4.6.5. 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. コンパクトフラッシュシステム構築」を修正／追加
1.0.7	2005/09/26	・「11.1. GPlO 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	・「12.3. カーネルイメージを配置する」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. 終了」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. ソフトウェア使用に関しての注意事項」に修正 ・「1.5. 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 ・「7.1.1. ソースコードの準備」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	・「1.5. 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

1.0.16	2007.12.14	・「7.1.2. コンフィグレーション」について、atmark-dist-20071112 で変更された内容にあわせて修正 ・「12.4. ルートファイルシステムの構築」の Debian イメージファイル名を変更
1.0.17	2008.3.14	・「10.1. デフォルト設定の変更」の注記を「10. VGA デバイスドライバ仕様」に移動し、VGA のタイミングが規格に一致していないことを追記
1.0.18	2008.5.30	・「2.2. 接続方法」で、接続する機材とその接続例の図を修正
1.0.19	2008.9.26	・CD-ROM ディレクトリ構成変更に伴う修正 ・タイトルを英語表記からカタカナ表記に
1.1.0	2009/03/19	・「1. はじめに」、「3. 開発環境の準備」、「4. 使用方法」、「5. フラッシュメモリの書き換え方法」、「6. Linux ブートオプション」、「7. ビルド」、「12. コンパクトフラッシュシステム構築」 構成変更 ・誤記、表記ゆれ修正
1.1.1	2009/07/17	・「11. その他のデバイスドライバ仕様」コマンド例の説明を追記 ・「5. フラッシュメモリの書き換え方法」にコマンド例の説明を追記 ・「10. VGA デバイスドライバ仕様」の表記を変更 ・date による時刻の設定コマンドを追加 ・表記ゆれ修正 ・本文のレイアウト統一 ・「図 7.1. ソースコード準備」の誤記を修正
1.1.2	2009/07/29	・製品保証に関する記載を http://www.atmark-techno.com/support/warranty-policy に移動(2009/08/03 適用)

Armadillo-9 ソフトウェアマニュアル
Version 1.1.2-d308169
2009/08/03

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