

Armadillo-240

Software Manual

Version 1.00

2006年 6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

1. はじめに.....	1
1.1. マニュアルについて.....	1
1.2. フォントについて.....	1
1.3. コマンド入力例の表記について.....	1
1.4. 謝辞.....	2
1.5. 注意事項.....	2
2. 作業の前に.....	3
2.1. 準備するもの.....	3
2.2. 接続方法.....	3
2.3. ジャンパピンの設定について.....	4
3. 開発環境の準備.....	5
3.1. クロス開発環境パッケージのインストール.....	5
3.2. atmark-dist のビルドに必要なパッケージ.....	6
3.3. クロス開発用パッケージの作成方法.....	7
4. 使用方法.....	8
4.1. 起動の前に.....	8
4.2. 起動.....	9
4.3. ディレクトリ構成.....	12
4.4. 終了.....	12
4.5. ネットワーク設定.....	13
4.5.1. 固定 IP アドレスで使用する場合.....	13
4.5.2. DNS サーバの設定.....	13
4.5.3. DHCP を使用する場合.....	14
4.5.4. ネットワーク設定の有効化.....	14
4.5.5. ネットワーク設定を Flash メモリに保存する.....	15
4.6. telnet ログイン.....	16
4.7. ファイル転送.....	16
4.8. Web サーバ.....	16
4.9. ssh ログイン.....	16
5. Flash メモリの書き換え方法.....	17
5.1. ダウンローダのインストール.....	17
5.2. リージョン指定について.....	18
5.3. 書き換え手順.....	19
5.3.1. ジャンパピンの設定.....	19
5.3.2. 書き換えイメージの転送.....	19
5.4. netflash を使って Flash メモリを書き換える.....	22
6. ブートローダー.....	23
6.1. パッケージの準備.....	23
6.2. ブートローダーの種類.....	23
6.3. ブートローダーの作成.....	24
6.3.1. ソースコードの準備.....	24
6.3.2. ビルド.....	24
6.4. CPU オンチップブート ROM.....	25
6.4.1. ブートローダーを出荷状態に戻す.....	25
6.5. Linux ブートオプション.....	27
6.5.1. Hermit コマンドプロンプトの起動.....	27
6.5.2. Linux ブートオプションの設定.....	28
6.5.3. 設定されている Linux ブートオプションの確認.....	28

6.5.4.	Linux ブートオプションを初期化する	28
6.5.5.	Linux ブートオプションの例	29
7.	atmark-dist でイメージを作成	30
7.1.	ソースコードアーカイブの展開	30
7.2.	設定	31
7.3.	ビルド	33
8.	メモリマップについて	34
9.	デバイスドライバ仕様	35
9.1.	GPIO ポート	35
9.2.	LED	36
9.3.	オンボード Flash メモリ/NAND Flash メモリ(Optional)	37
9.4.	USB ホスト	37
9.4.1.	USB Storage	37
9.4.2.	USB Human Interface Device (HID)	37
9.5.	VGA	38
9.5.1.	デフォルト設定の変更	38
9.5.2.	解像度・色深度の変更	39
10.	Appendix	40
10.1.	Windows 上に開発環境を構築する方法	40
10.1.1.	coLinux のインストール	40
10.1.2.	環境構築用ファイルの準備	40
10.1.3.	coLinux の実行	40
10.1.4.	ネットワークの設定	41
10.1.5.	coLinux ユーザの作成	42
10.1.6.	Windows-coLinux 間のファイル共有	42
10.1.7.	クロス開発環境の導入	42
10.1.8.	特殊な場合の Windows ネットワーク設定方法	43
10.1.9.	coLinux のネットワーク設定方法	44

表目次

表 1-1	使用しているフォント	1
表 1-2	表示プロンプトと実行環境の関係	1
表 2-1	ジャンパの設定とブート時の動作	4
表 3-1	クロス開発環境パッケージ一覧	5
表 3-2	atmark-dist のビルドに必要なパッケージ一覧	6
表 4-1	シリアル通信設定	8
表 4-2	コンソールログイン時のユーザ名とパスワード	11
表 4-3	ディレクトリ構成の一覧	12
表 4-4	ネットワーク設定詳細	13
表 4-5	telnet ログイン時のユーザ名とパスワード	16
表 4-6	ftp のユーザ名とパスワード	16
表 4-7	ssh ログイン時のユーザ名とパスワード	16
表 5-1	各リージョン用のイメージファイル名	18
表 6-1	ブートローダー関連のパッケージ一覧	23
表 6-2	ブートローダー 一覧	23
表 6-3	シリアル通信設定	27
表 8-1	メモリマップ(Flash メモリ)	34
表 8-2	メモリマップ(RAM)	34
表 9-1	GPIO ノード	35
表 9-2	GPIO 操作コマンド	35
表 9-3	LED ノード	36
表 9-4	LED 操作コマンド	36
表 9-5	MTD ノード	37
表 9-6	解像度一覧	39
表 9-7	色深度一覧	39
表 10-1	ネットワーク設定	45

目次

図 2-1 Armadillo-240 接続例	3
図 2-2 ジャンパの位置	4
図 3-1 開発用パッケージの展開例	6
図 3-2 複数パッケージの展開例	6
図 3-3 クロス開発用パッケージの作成(deb)	7
図 3-4 クロス開発用パッケージの作成(rpm、tgz)	7
図 4-1 起動ログ	11
図 4-2 ネットワーク設定例(固定 IP アドレス時)	13
図 4-3 ネットワーク設定例(ゲートウェイの無効化)	13
図 4-4 DNS サーバの設定	13
図 4-5 ネットワーク設定例(DHCP 使用時)	14
図 4-6 ネットワーク接続の終了	14
図 4-7 ネットワーク接続の開始	14
図 5-1 展開処理コマンド入力例	17
図 5-2 コマンド入力例	19
図 5-3 Download 画面	20
図 5-4 書き換え進捗ダイアログ	20
図 5-5 netflash コマンド例	22
図 5-6 netflash ヘルプコマンド	22
図 6-1 shoehorn コマンド例	25
図 6-2 Shoehorn 画面	26
図 6-3 shoehorn ダイアログ	26

1.はじめに

1.1. マニュアルについて

本マニュアルは、Armadillo-240 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。

- Flash メモリの書き換え方法
- 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

また、本マニュアルはベースイメージのデフォルト設定を前提に記載されています。リカバリーイメージについては、「Armadillo-240 Startup Guide」を参照してください。

Armadillo-240 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
[a240 /]#	Armadillo-240 上の特権ユーザで実行
[a240 /]\$	Armadillo-240 上の一般ユーザで実行

1.4. 謝辞

Armadillo-240 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成り立っています。この場を借りて感謝の意を示します。

1.5. 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。

2.作業の前に

2.1.準備するもの

Armadillo-240 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 - 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-240 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、RS232C レベル変換アダプタ、AC アダプタ、VGA モニタ、そして LAN ケーブルを Armadillo-240 に接続してください。

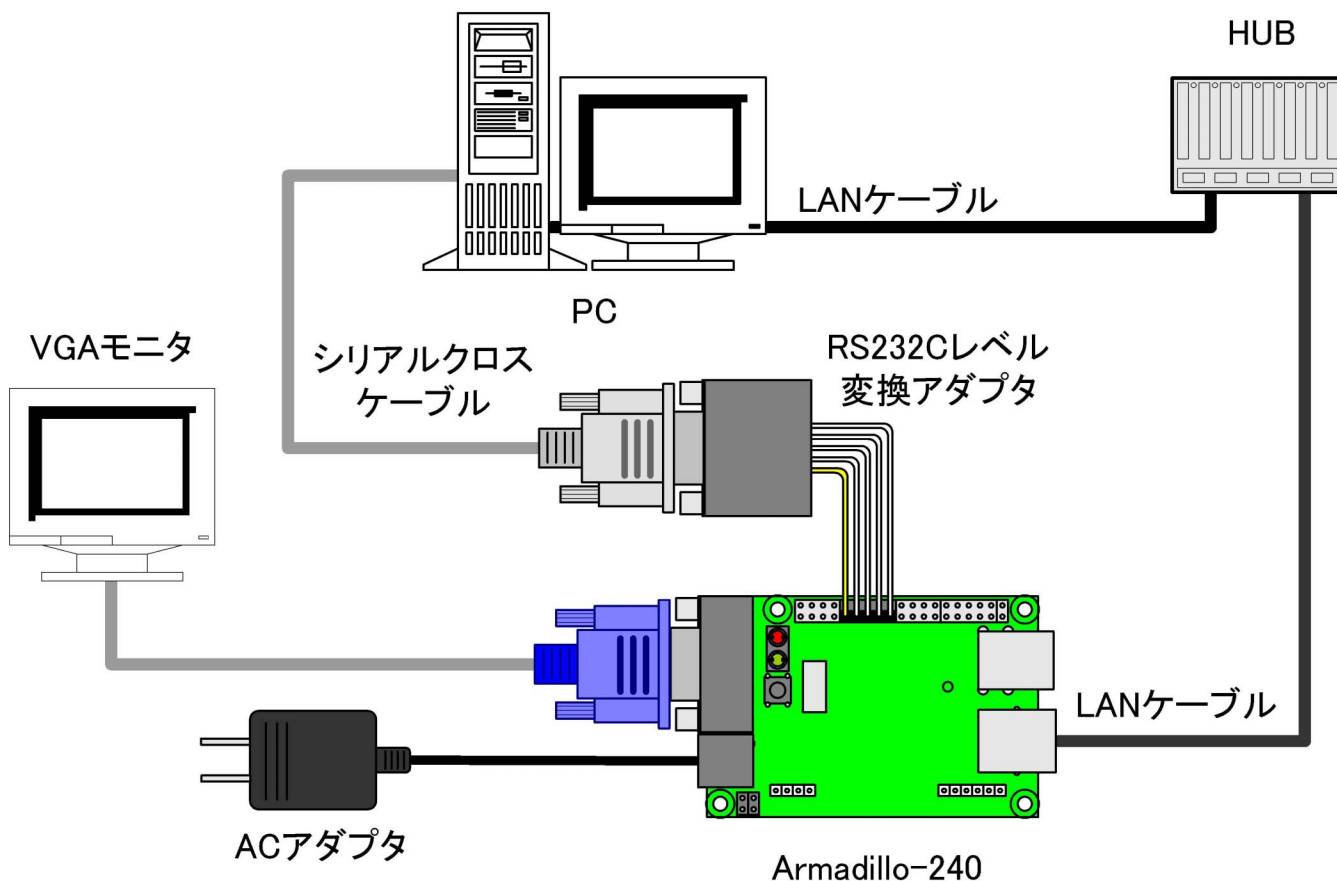


図 2-1 Armadillo-240 接続例

2.3. ジャンパピンの設定について

Armadillo-240 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	Linux カーネルを起動
オープン	ショート	Hermit コマンドプロンプトを起動
ショート	-	CPU オンチップブート ROM を起動(1)

1 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

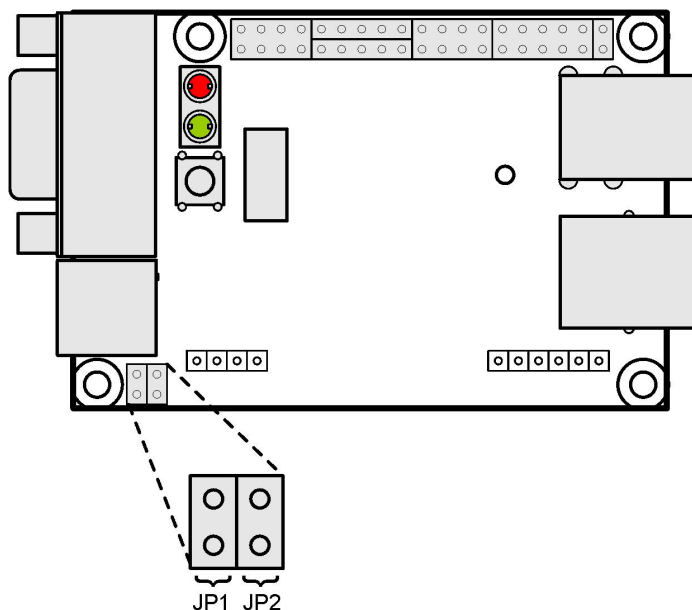


図 2-2 ジャンパの位置

3. 開発環境の準備

作業用の PC 上で Armadillo-240 のソフトウェアをクロス開発することができます。

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 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るので、これらを全てインストールします。インストールは必ず root 権限で行ってください。以下のパッケージが用意されています。

表 3-1 クロス開発環境パッケージ一覧

パッケージ名	バージョン	説明
binutils-arm-linux	2.15-6	The GNU Binary utilities
cpp-3.4-arm-linux	3.4.3-13	The GNU C preprocessor
g++-3.4-arm-linux	3.4.3-13	The GNU C++ compiler
gcc-3.4-arm-linux	3.4.3-13	The GNU C compiler
libc6-arm-cross	2.3.2.ds1-22	GNU C Library: Shared libraries and Timezone data
libc6-dev-arm-cross	2.3.2.ds1-22	GNU C Library: Development Libraries and Header Files
libc6-pic-arm-cross	2.3.2.ds1-22	GNU C Library: PIC archive library
libc6-prof-arm-cross	2.3.2.ds1-22	GNU C Library: Profiling Libraries
libdb1-compat-arm-cross	2.1.3-7	The Berkeley database routines
libgcc1-arm-cross	3.4.3-13	GCC support library
libstdc++6-0-arm-cross	3.4.3-13	The GNU Standard C++ Library v3
libstdc++6-0-dbg-arm-cross	3.4.3-13	The GNU Standard C++ Library v3 (debugging files)
libstdc++6-0-dev-arm-cross	3.4.3-13	The GNU Standard C++ Library v3 (development files)
libstdc++6-0-pic-arm-cross	3.4.3-13	The GNU Standard C++ Library v3 (shared library subset kit)
linux-kernel-headers-arm-cross	2.5.999-test7-bk-17	Linux Kernel Headers for development
libdaemon0-arm-cross	0.7-1	lightweight C library for daemons
libdaemon0-dev-arm-cross	0.7-1	lightweight C library for daemons
libexpat1-arm-cross	1.95.8-3	XML parsing C library – runtime library
libexpat1-dev-arm-cross	1.95.8-3	XML parsing C library – development kit
libncurses5-arm-cross	5.4-9	Shared libraries for terminal handling
libncurses5-dev-arm-cross	5.4-9	Developer's libraries and docs for ncurses
libnet0-arm-cross	1.0.2a-7	Library for the construction and handling of network packets (obsolete)
libnet0-dev-arm-cross	1.0.2a-7	Development files for libnet0 (obsolete)
libpcap0.8-arm-cross	0.8.3-5	System interface for user-level packet capture
libpcap0.8-dev-arm-cross	0.8.3-5	Development library and header files for libpcap 0.8
libssl0.9.7-arm-cross	0.9.7g-1	SSL shared libraries
libssl-dev-arm-cross	0.9.7g-1	SSL development libraries, header files and documentation
zlib1g-arm-cross	1.2.3-3	compression library - runtime
zlib1g-dev-arm-cross	1.2.3-3	compression library - development

パッケージファイルは deb(Debian 系ディストリビューション向け)、rpm(Red Hat 系ディストリビューション向け)、tgz(インストーラ非使用)が用意されています。お使いの OS にあわせて、いずれか1つを選択してご利用ください。

```
[PC ~]# dpkg -i binutils-arm-linux_2.14.90.0.7-8_i386.deb    deb パッケージを使用する場合
[PC ~]# rpm -i binutils-arm-linux-2.14.90.0.7-8.i386.rpm    rpm パッケージを使用する場合
[PC ~]# tar zxf binutils-arm-linux-2.14.90.0.7.tgz -C /      tgz を使用する場合
```

図 3-1 開発用パッケージの展開例

インストール時に依存関係でエラーになる場合は、次のようにしてください。

```
[PC ~]# dpkg -i xxx.deb yyy.deb    複数の deb パッケージを一度にインストールする場合
```

図 3-2 複数パッケージの展開例

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、作業用 PC に表 3-2に記されているパッケージが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	説明
genext2fs	1.3-7.1-cvs20050225	ext2 filesystem generator for embedded systems
file	4.12-1 以降	Determines file type using "magic" numbers
sed	4.1.2-8 以降	The GNU sed stream editor
perl	5.8.4-8 以降	Larry Wall's Practical Extraction and Report Language

genext2fs のパッケージファイルは付属 CD の tools ディレクトリに用意されています。

3.3. クロス開発用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、作成したいクロスパッケージの元となる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、libncurses5 の場合「libncurses5_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、取得したパッケージをクロス開発用に変換します。

```
[PC ~]$ dpkg-cross --build --arch arm libncurses5_x.x-x_arm.deb
[PC ~]$ ls
libncurses5-arm-cross_x.x-x_all.deb libncurses5_x.x-x_arm.deb
```

図 3-3 クロス開発用パッケージの作成(deb)

「libncurses5-arm-cross_x.x-x_all.deb」というクロスパッケージが作成されます。これは deb パッケージです。必要に応じて rpm パッケージや tgz を作成すると良いでしょう。rpm と tgz の作成方法を以下に示します。

```
[PC ~]# alien -r -k libncurses5-arm-cross_x.x-x_all.deb      rpm パッケージを作成
[PC ~]# alien -t -k libncurses5-arm-cross_x.x-x_all.deb    tgz を作成
[PC ~]$ ls
libncurses5-arm-cross_x.x-x_all.deb libncurses5_x.x-x_arm.deb
libncurses5-arm-cross-x.x-x.noarch.rpm libncurses5-arm-cross_x.x.tgz
```

図 3-4 クロス開発用パッケージの作成(rpm、tgz)

4. 使用方法

この章では Armadillo-240 の基本的な使用方法の説明を行いません。

4.1. 起動の前に

Armadillo-240 のシリアルポート 1(CON3)と作業用 PC を RS232C レベル変換アダプタを經由して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、Linux が起動します。正常に起動した場合、シリアルポート 1 に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=armadillo2x0-nor:0x10000(boot loader)ro,0x170000(kernel),0x670000(user land),-(config)
Linux version 2.6.12.3-a9-5 (atmark@pc-nsx) (gcc version 3.4.4 20050314 (prerelease) (Debian 3.4.3-13)) #1 Fri Jun 16 18:43:41 JST 2006
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-240
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=armadillo2x0-nor:0x10000(boot loader)ro,0x170000(kernel),0x670000(us
er land),-(config)
PID hash table entries: 512 (order: 9, 8192 bytes)
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: 54768KB available (2670K code, 609K data, 96K 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: 6591K
NET: Registered protocol family 16
SCSI subsystem initialized
usbcore: registered new driver usbfs
usbcore: registered new iver hub
Bluetooth: Core ver 2.7
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
NetWinder Floating Point Emulator V0.97 (double precision)
JFFS2 version 2.2. (NAND) (C) 2001-2003 Red Hat, Inc.
Initializing Cryptographic API
fb0: EP93xx frame buffer at 800x600x24
gpio: Armadillo-2x0 GPIO driver, (C) 2005-2006 Atmark Techno, Inc.
led: Armadillo-2x0 LED driver, (C) 2005-2006 Atmark Techno, Inc.
sw: Armadillo-2x0 Takt Switch driver, (C) 2006 Atmark Techno, Inc.
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.
armadillo2x0-nor: Found 1 x16 devices at 0x0 in 16-bit bank
  Amd/Fujitsu Extended Query Table at 0x0040
armadillo2x0-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 armadillo2x0-nor
parse_mtd_partitions:4
Creating 4 MTD partitions on "armadillo2x0-nor":
0x00000000-0x00010000 : "bootloader"
0x00010000-0x00180000 : "kernel"
0x00180000-0x007f0000 : "userland"
0x007f0000-0x00800000 : "config"
No NAND device found!!!
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
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
pegasus: v0.6.12 (2005/01/13), Pegasus/Pegasus II USB Ethernet driver
usbcore: registered new driver pegasus
zd1211 - http://zd1211.ath.cx/
Based on www.zydax.com.tw driver version 2.0.0.0
usbcore: registered new driver zd1211
Bluetooth: HCI USB driver ver 2.8
usbcore: registered new driver hci_usb
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)
IPv4 over IPv4 tunneling driver
ip_tables: (C) 2000-2002 Netfilter core team
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 10
Disabled Privacy Extensions on device c02ee504(lo)
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
Bluetooth: L2CAP ver 2.7
Bluetooth: L2CAP socket layer initialized
Bluetooth: RFCOMM ver 1.5
Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM TTY layer initialized
SCTP: Hash tables configured (established 2048 bind 4096)
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 6591KiB [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 96K
init started: BusyBox v1.00 (2006.06.16-11:12+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, 614/1024 files, 5354/6591 blocks
Checking root filesystem: done
Remounting root rw: done
Mounting proc: done
Mounting usbfs: done
Mounting sysfs: done
Loading /etc/config: done
Setting hostname: done
Cleaning up system: done
Running local start scripts.
Changing file permissions: done
Starting syslogd: done
Starting klogd: done
Starting basic firewall: done

```

```

Loading /etc/config:           done
Configuring network interfaces: done
Starting inetd:                done
Starting sshd:                 done
Starting thttpd:               done

atmark-dist v1.5.1 (AtmarkTechno/Armadillo-240.Base)
Linux 2.6.12.3-a9-5 [armv4tl arch]

a240-0 login:
    
```

図 4-1 起動ログ

ベースイメージ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 1(CON3)とシリアルポート 2(CON4)に表示されます。

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/config	flatfsd 向け設定用
/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
/home/www-data	WEB サーバホームディレクトリ用
/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-240 を終了させます。

4.5. ネットワーク設定

Armadillo-240 内の「/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。

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/config/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/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)

4.5.4. 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。

```
[a240 /]# ifdown eth0
```

図 4-6 ネットワーク接続の終了

```
[a240 /]# 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 Flash メモリに保存する

Armadillo-240 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 Flash 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。

```
[a240 /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
//--- ファイル終了
[a240 /etc/config]# vi resolv.conf
//--- 編集ファイル
nameserver 192.168.10.1
//--- ファイル終了
[a240 /etc/config]#
```

次に、フラッシュに書き込みます。

```
[a240 /etc/config]# flatfsd -s
```

これで書き換えたネットワーク設定が Flash 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

tthttpd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-240 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-240 の IP アドレス)/」になります。(例 http://192.168.10.10/)

4.9.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 4-7 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. Flash メモリの書き換え方法

Flash メモリの内容を書き換えることで、Armadillo-240 の機能を変更することができます。この章では Flash 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-240 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-240 の電源を切らない。
- Armadillo-240 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-240 の Flash メモリの書き換えに使用します。

1) Linux の場合

付属 CD 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 権限で行ってください。パッケージファイルは deb(Debian 系ディストリビューション向け)、rpm(Red Hat 系ディストリビューション向け)、tgz(インストーラ非使用)が用意されています。お使いの OS にあわせて、いずれか 1 つを選択してご利用ください。

```
[PC ~]# dpkg -i hermit-at_1.0.4_i386.deb    deb パッケージを使用する場合
[PC ~]# rpm -i hermit-at_1.0.4.rpm        rpm パッケージを使用する場合
[PC ~]# tar zxf hermit-at-1.0.4.tgz -C /   tgz を使用する場合
```

図 5-1 展開処理コマンド入力例

2) Windows の場合

付属 CD より「Hermit-At WIN32 (downloader/win32/hermit-at-win_XXXXXXXXX.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

Flash 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イメージとあ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、最初に行わ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 Flash 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 `images` 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-armadillo2x0.bin
kernel	linux-a240-x.xx.bin.gz
userland	romfs-a240-recover-x.xx.img.gz romfs-a240-base-x.xx.img.gz

Flash 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 Flash 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-240 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

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、作業用 PC と Armadillo-240 のシリアルポート 1(CON3)をシリアルケーブルで接続し、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、警告が表示されブートローダー領域への書き込みは実行されません。



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 Flash 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-240 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-240 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。

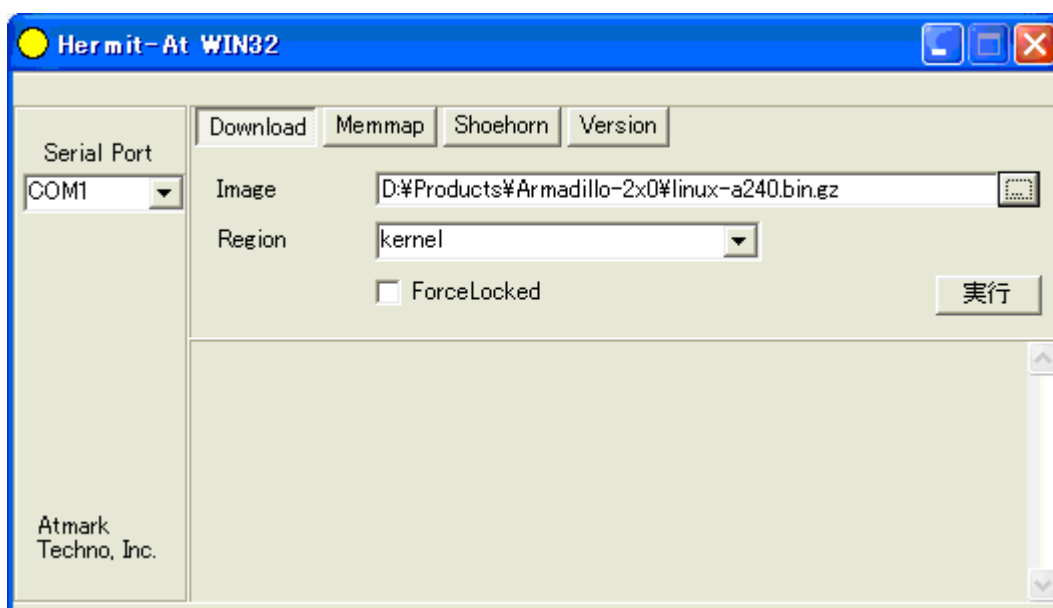


図 5-3 Download 画面

「実行」ボタンをクリックすると、Flash 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。

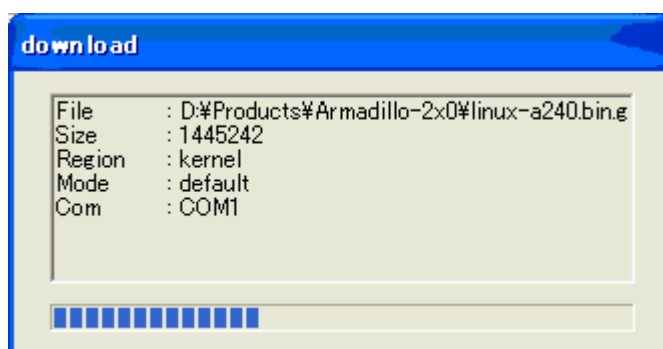


図 5-4 書き換え進捗ダイアログ



TIPS

ブートローダー領域(リージョン:boot loader / アドレス:0x60000000-0x6000ffff)を書き換える際は、「ForceLocked」をチェックする必要があります。これを選択しない場合、警告が表示されブートローダー領域への書き込みは実行されません。



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 Flash 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-240 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 Flash メモリを書き換える

Flash 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 Flash 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「Flash メモリ書き換え」に失敗した場合、Armadillo-240 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-240 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、Flash 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-240 上での kernel イメージを変更するコマンド例です。

```
[a240 ~]# 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-2x0/images/linux-a240-1.00.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。

```
[a240 ~]# 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-240 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 downloader 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	Armadillo-240 ブートプログラムと協調動作するダウンローダ (Armadillo-240 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-240 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-armadillo2x0	出荷時に Flash メモリに書き込まれている標準ブートローダー hermit コンソールにシリアルポート 1 を使用
loader-armadillo2x0-ttyAM1	hermit コンソールにシリアルポート 2 を使用するブートローダー
loader-armadillo2x0-notty	hermit 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/bootloader` ディレクトリから、`hermit-at-x.x.x.tar.gz` を作業用 PC にコピーし、展開します。

```
[PC ~]$ tar xzf hermit-at-x.x.x.tar.gz
```

6.3.2. ビルド

展開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-x.x.x  
[PC ~]$ make TARGET=armadillo2x0
```

`make` が完了後、`hermit-at-x.x.x/src/target/armadillo2x0` 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-armadillo-2x0-notty が書き込まれている Armadillo-240 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-240 がブートできなくなってしまう場合の対処方法について説明します。

Armadillo-240 の CPU にはオンチップブート ROM が搭載されており、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

Armadillo-240 の電源が切断されていることを確認し、Armadillo-240 のシリアルポート 1(CON3)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。Armadillo-240 のジャンパ JP1 をショートに設定します。作業用 PC で shoehorn を起動します。

```
[PC ~]$ shoehorn --boot --terminal --initrd /dev/null
--kernel /usr/lib/hermit/loader-armadillo-2x0-boot.bin
--loader /usr/lib/shoehorn/shoehorn-armadillo2x0.bin
--initfile /usr/lib/shoehorn/shoehorn-armadillo2x0.init
--postfile /usr/lib/shoehorn/shoehorn-armadillo2x0.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-240 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

Armadillo-240 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-240 の電源を切断し、シリアルケーブルの接続や Armadillo-240 のジャンパ(JP1)設定を確認してください。

“ 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-240 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.Flash 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

Armadillo-240 の電源が切断されていることを確認し、Armadillo-240 のシリアルポート 1(CON3)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。
 Armadillo-240 のジャンパ JP1 をショートに設定します。
 作業用 PC で「Hermit-At WIN32」を起動します。
 「Shoehorn」ボタンをクリックします。

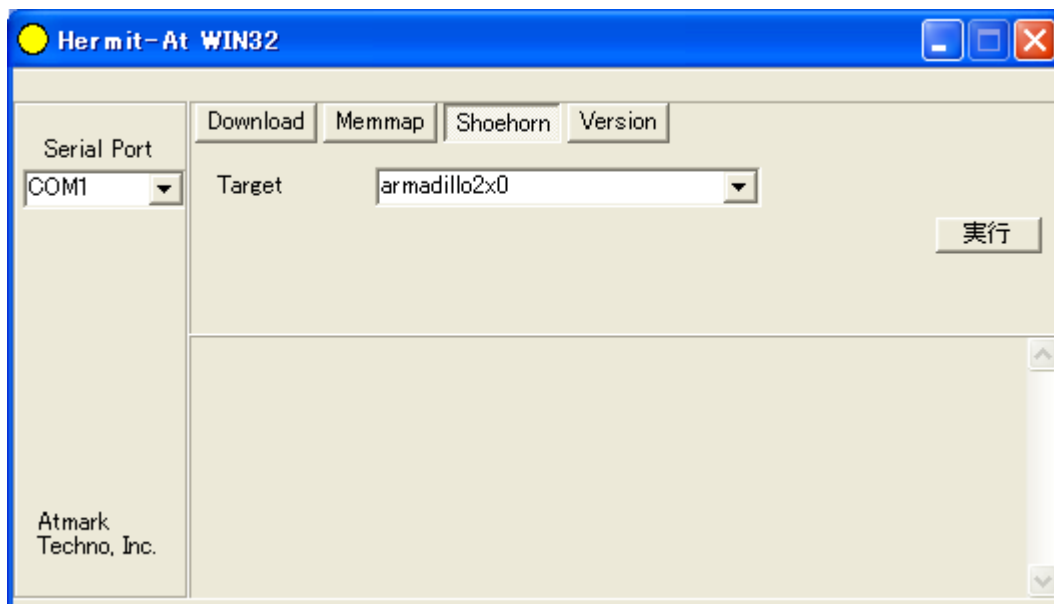


図 6-2 Shoehorn 画面

"Target" に armadillo2x0 を指定します。
 「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

Armadillo-240 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-240 の電源を切断し、シリアルケーブルの接続や Armadillo-240 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-240 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.Flash 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

Armadillo-240 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 Flash 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

シリアルコンソールソフトの起動

Armadillo-240 のシリアルポート 1(CON3)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

ジャンパピンの設定

Armadillo-240 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

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。

Armadillo-240 の起動

Armadillo-240 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-At v1.0.4 (armadillo240) compiled at 00:00:00, Jun 1 2006
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
```



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 1 (ttyAM0) 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-240 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
```

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
```



注意

ブートローダーを書き換えた場合、Linux ブートオプションの領域が壊れてしまい正常に起動しない場合があります。この場合、一度 `clearenv` を実行し、Linux ブートオプション領域を初期化する必要があります。

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、Linux 起動ログをシリアルポート 1 (ttyAM0) に表示させる場合

```
hermit> setenv console=ttyAM0,115200
```

ex.2) Linux 起動ログを表示させない場合

```
hermit> setenv console=null
```

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・配置を行いません。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 source/dist ディレクトリに atmark-dist-YYYYMMDD.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/kernel ディレクトリに linux-2.6.x-a9-x.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-2.6.x-a9-x.tar.gz
:
:
[PC ~]$ cd atmark-dist
[PC ~/atmark-dist]$ ln -s ../linux-2.6.x-a9-x ./linux-2.6.x
```

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
#
# Using defaults found in .config
#
*
* 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-240.Base」と入力してください。

```
*
* Select the Product you wish to target
*
AtmarkTechno Products (Armadillo, Armadillo-240.Base, Armadillo-240.Recover,
Armadillo-9, Armadillo-9.PCMCIA, Armadillo-J.Base, Armadillo-J.Jffs2,
Armadillo-J.Recover, SUZAKU, SUZAKU-UQ-XUP) [Armadillo] (NEW) Armadillo-240.Base
```

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-240では、「None」を選択します。

```
*
* Kernel/Library/Defaults Selection
*
*
* Kernel is linux-2.4.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いません。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 dep 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.gz` が作成されます。作成したイメージを `Armadillo-240` に書き込む方法は「5.Flash 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(Flash 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-armadillo-2x0.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(.gz)」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 Flash メモリから展開・コピー
0xc4800000	userland	EXT2	linux の起動前に Flash メモリから展開・コピー

9. デバイスドライバ仕様

9.1. 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 9-1 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)
キャラクタデバイス	10	185	gpio

ioctl を使用してアクセスすることにより、Armadillo-240 の GPIO を直接操作することができます。

第 1 引数には、デバイスファイルのファイルディスクリプタを指定します。

第 2 引数には、GPIO を操作するためのコマンドを指定します。

表 9-2 GPIO 操作コマンド

コマンド	説明	第 3 引数の Type
PARAM_SET	第 3 引数で指定する内容で GPIO の状態を設定します	struct gpio_param
PARAM_GET	第 3 引数で指定する内容で GPIO の状態を取得します	struct gpio_param
INTERRUPT_WAIT	第 3 引数で指定する内容で GPIO の割込みが発生するまで WAIT します	struct wait_param

第 3 引数には、(カーネルソース)/include/asm-arm/arch-ep93xx/armadillo2x0_gpio.h に定義されている構造体「struct gpio_param」と「struct wait_param」を使用します。「struct gpio_param」は単方向リストになっているので、複数の GPIO を一度に制御する場合は next メンバを使用してください。また、リストの最後の next メンバには"0(NULL)"を指定してください。

GPIO デバイスドライバの詳細な使用方法については、サンプルの GPIO 制御アプリケーション (atmark-dist/vendors/AtmarkTechno/Armadillo-2x0.Common/gpiood)のソースコードを参考にしてください。

9.2.LED

LED 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 9-3 LED ノード

タイプ	メジャー 番号	マイナー 番号	ノード名 (/dev/xxx)
キャラクタ デバイス	10	215	led

ioctl を使用してアクセスすることにより、Armadillo-240 の LED を直接操作することができます。

第 1 引数には、デバイスファイルのファイルディスクリプタを指定します。

第 2 引数には、LED を操作するためのコマンドを指定します。

表 9-4 LED 操作コマンド

コマンド	説明	第 3 引数の Type
LED_RED_ON	LED(赤)を点灯します	なし
LED_RED_OFF	LED(赤)を消灯します	なし
LED_RED_STATUS	LED(赤)の状態を取得します	状態を保存するバッファ(最小 1 Byte)
LED_GREEN_ON	LED(緑)を点灯します	なし
LED_GREEN_OFF	LED(緑)を消灯します	なし
LED_GREEN_STATUS	LED(緑)の状態を取得します	状態を保存するバッファ(最小 1 Byte)

LED デバイスドライバの詳細な使用方法については、サンプルの LED 制御アプリケーション (atmark-dist/vendors/AtmarkTechno/Armadillo-2x0.Common/ledctrl) のソースコードを参考にしてください。

9.3. オンボード Flash メモリ/NAND Flash メモリ(Optional)

オンボード Flash 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 Flash 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。

また、オプション品の NAND Flash メモリ(受注生産品)についても、オンボード Flash メモリに続く形でリージョンで扱われます。

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 9-5 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	boot loader
		1	mtdr0	boot loader (read only)
		2	mtd1	kernel
		3	mtdr1	kernel (read only)
		4	mtd2	user land
		5	mtdr2	user land (read only)
		6	mtd3	config
		7	mtdr3	config (read only)
		8	mtd4	NAND Flash(接続時のみ)
		9	mtdr4	NAND Flash(接続時のみ/read only)
ブロックデバイス	31	0	mtdblock0	boot loader
		1	mtdblock1	kernel
		2	mtdblock2	user land
		3	mtdblock3	config
		4	mtdblock4	NAND Flash(接続時のみ)

9.4. USB ホスト

EP9307 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。

9.4.1. USB Storage

USB メモリやディスクドライブ、メモリカードリーダー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。

9.4.2. 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。

9.5. VGA

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 SVGA サイズ(解像度:800x600)の 24 ビットカラー設定となっていますが、VGA サイズ(640x480)及び XGA サイズ(1024x768)や 8/16 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。

9.5.1. 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-2.4.x
(None) Libc Version
[ ] Default all settings
[*] Customize Kernel Settings
[ ] Customize Vendor/User Settings
[ ] Update Default Vendor Settings
```

ここを選択する

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 --->
[*] EP93xx frame buffer support
    EP93xx frame buffer display (CRT display) --->
    EP93xx frame buffer resolution (SVGA(60Hz)) ---> デフォルトの解像度
    EP93xx frame buffer depth (24bpp 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 dep all
```

ビルドしてできたカーネルイメージ (linux.bin.gz) を Armadillo-240 へ書き込み、VGA のデフォルトの設定は完了です。

9.5.2. 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」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 9-6 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 9-7 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
```

10. Appendix

10.1. Windows 上に開発環境を構築する方法

Linux 環境を実現する coLinux(<http://www.colinux.org/>)を利用することで、Windows 上にクロス開発環境を構築することができます。対応している OS は WindowsXP、Windows2000 です。

10.1.1. coLinux のインストール

- 1) 付属 CD の colinux ディレクトリにある coLinux-0.6.2.exe を実行する
- 2) インストール先フォルダには c:¥coLinux を指定し、それ以外はデフォルトの設定のままでインストール作業を行なう



TIPS

インストール先に他のディレクトリを指定する場合は、次の手順で用意する default.colinux.xml を編集し、ディレクトリ名を適切に変更する必要があります。

10.1.2. 環境構築用ファイルの準備

付属 CD の colinux ディレクトリから以下のファイルを用意し、coLinux のインストールフォルダ(c:¥coLinux)に展開します

- root_fs.zip (ルートファイルシステム)
- swap_device_256M.zip (swap ファイルシステム)
- home_fs_2G.zip (/home にマウントされるファイルシステム)
- default.colinux.xml.zip (デバイス情報の設定ファイル)



TIPS

swap_device_..., home_fs_... のファイル名の数値は展開後のファイルサイズです。他のサイズのファイルも用意していますので、必要と思われるサイズのファイルを展開してください。展開ソフトによっては展開に失敗する場合があります。WindowsXP の標準機能で正常に展開できることを確認してあります。

10.1.3. coLinux の実行

- 1) DOS プロンプトを起動し、インストールフォルダ(c:¥coLinux)に移動する
- 2) 「colinux-daemon.exe -c default.colinux.xml」とコマンド入力する
- 3) 起動ログの表示後「colinux login:」と表示されたら「root」でログインする

10.1.4. ネットワークの設定

coLinux では Windows とは別の IP アドレスを持ち、Windows を介してネットワークにアクセスするため、Windows のネットワーク設定の変更が必要となります。

設定方法には「ルーター接続」「ブリッジ接続」がありますが、ここでは「ルーター接続」の方法を説明します。

(WindowsXP の場合)

- 1) コントロールパネルから「ネットワーク接続」を開く
- 2) 外部に接続しているネットワークを右クリックして「プロパティ」を開く
- 3) 「詳細設定」タブを開き、インターネット接続の共有を有効にする

(Windows2000 の場合)

- 1) コントロールパネルから「ネットワークとダイヤルアップ接続」を開く
- 2) 外部に接続しているネットワークを右クリックして「プロパティ」を開く
- 3) 「共有」タブを開き、インターネット接続の共有を有効にする

次に、ネットワークの設定を有効にするためのコマンドを coLinux 上で実行します。

例 10-1 ネットワークの設定コマンド

```
colinux:~# /etc/init.d/networking restart
Reconfiguring network interfaces: done.
colinux:~#
```



「ルーター接続」では 192.168.0.0/24 のネットワークアドレスが自動的に使用されるため、外部接続用のネットワークアドレスが同じ 192.168.0.0/24 の場合、設定に失敗します。この場合は外部接続用のネットワークアドレスを変更してください。

外部接続用のネットワークアドレスを変更できない場合は「10.1.8.特殊な場合の Windows ネットワーク設定方法」を参照してください。

10.1.5. coLinux ユーザの作成

coLinux の画面で以下のようにコマンドを入力し作業用ユーザを作成します。適宜パスワードなどを設定してください。

例 10-2 ユーザ「somebody」を作業用ユーザとして追加する場合

```
colinux:~# adduser somebody
Adding user somebody...
Adding new group somebody (1000).
Adding new user somebody (1000) with group somebody.
Creating home directory /home/somebody.
Copying files from /etc/skel
Enter new UNIX password:
```

10.1.6. Windows-coLinux 間のファイル共有

Windows の共有フォルダを利用して、coLinux と Windows 間でファイルを交換する方法です。coLinux の画面で以下のように smbmount コマンドを実行して、共有フォルダのパスワードを入力してください。

例 10-3 Windows の IP アドレス: 192.168.0.100、共有フォルダ名: shared の場合

```
colinux:~# mkdir /mnt/smb
colinux:~# smbmount //192.168.0.100/shared /mnt/smb
212: session request to 192.168.0.100 failed (Called name not present)
212: session request to 192 failed (Called name not present)
Password:
```

ユーザ名が Windows 側と異なる場合は、ユーザ名をコマンドのオプションで指定します。詳しくは「man smbmount」を実行してヘルプを参照してください。

以後、windows の共有フォルダ「shared」と coLinux のディレクトリ「/mnt/smb」のデータは同じものになります。

10.1.7. クロス開発環境の導入

「3.開発環境の準備」を参照して、クロス開発環境を coLinux 上に導入してください。環境の構築に必要なファイルは、前項で使用した共有フォルダを通じて coLinux からアクセスできます。

これで Windows 上で開発を行なうことができます。以降の説明は特殊なケースです。

10.1.8. 特殊な場合の Windows ネットワーク設定方法

外部接続用のネットワークアドレスが 192.168.0.0/24 の場合のネットワーク設定方法です。

(WindowsXP の場合)

「ブリッジ接続」を利用する方法です。

- 1) コントロールパネルから「ネットワーク接続」を開く
- 2) 外部に接続しているネットワークと「TAP-Win32 adapter」というデバイス名のネットワークの二つを選択状態にする
- 3) メニューの「詳細設定」から「ブリッジ接続」を選択する

(Windows2000 の場合)

Windows2000 では 192.168.0.0/24 以外のネットワークアドレスをプライベートネットワークで使用する方法です。ここでは 192.168.1.0/24 を使用します。

- 1) コントロールパネルから「ネットワークとダイヤルアップ接続」を開く
- 2) 外部に接続しているネットワークを右クリックして「無効」にする
- 3) 外部に接続しているネットワークを右クリックして「プロパティ」を開く
- 4) 「全般」タブの「インターネットプロトコル(TCP/IP)」を選択し「プロパティ」ボタンを押す
- 5) 「次の IP アドレスを使う」を選択して 192.168.100.100 を設定する
- 6) 「共有」タブを開き、インターネット接続の共有を有効にする
- 7) 「TAP-Win32 adapter」というデバイス名のネットワークを右クリックして「プロパティ」を開く
- 8) 「全般」タブの「インターネットプロトコル(TCP/IP)」を選択し「プロパティ」ボタンを押す
- 9) 「次の IP アドレスを使う」を選択して 192.168.1.1 を設定する
- 10) 外部に接続しているネットワークを右クリックして「プロパティ」を開く
- 11) 「全般」タブの「インターネットプロトコル(TCP/IP)」を選択し「プロパティ」ボタンを押す
- 12) IP アドレスの設定を元の設定に戻す
- 13) 外部に接続しているネットワークを右クリックして「有効」にする

10.1.9. coLinux のネットワーク設定方法

インストール状態では DHCP が使用されますが、DHCP サーバが動作していない環境等では固定で IP アドレスを設定する必要があります。

ネットワーク設定は `ifconfig` コマンドで表示することができます。

例 10-4 ifconfig コマンドの実行

```
colinux:~# ifconfig
eth0    Link encap:Ethernet  HWaddr XX:XX:XX:XX:XX:XX
        inet addr:192.168.0.151  Bcast:192.168.0.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:189 errors:0 dropped:0 overruns:0 frame:0
        TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:24472 (23.8 KiB)  TX bytes:9776 (9.5 KiB)
        Interrupt:2

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

colinux:~#
```

eth0 デバイスの IP アドレスが表示されない場合は、固定で IP アドレスを設定する必要があります。設定すべき IP アドレスですが、「ルーター接続」の場合は「TAP-Win32 adapter」のネットワークに合わせ、「ブリッジ接続」の場合は外部ネットワークに合わせます。ここでは、以下の表の内容に設定を変更する方法を説明します。

表 10-1 ネットワーク設定

項目	設定
IP アドレス	192.168.1.100
ネットマスク	255.255.255.0
ゲートウェイ	192.168.1.1
DNS サーバ	192.168.1.1

- 1) coLinux 上で/etc/network/interfaces を以下のように編集する

例 10-5 /etc/network/interfaces ファイル編集例

```
auto lo eth0
iface lo inet loopback
iface eth0 inet static
address 192.168.1.100
gateway 192.168.1.1
netmask 255.255.255.0
```

- 2) coLinux 上で/etc/resolv.conf を以下のように編集する

例 10-6 /etc/resole.conf ファイル編集例

```
nameserver 192.168.1.1
```

- 3) 以下のコマンドを実行し、編集した内容でネットワーク設定を更新する

例 10-7 ネットワークの再設定コマンド

```
colinux:~# /etc/init.d/networking restart
Reconfiguring network interfaces: done.
colinux:~#
```

改訂履歴

Ver	年月日	改訂内容
1.00	2006.6.19	・初版発行

Armadillo-240 Software Manual

2006 年 6 月 19 日 version 1.00

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
