Armadillo-200 シリーズ

220/230/240

Software Manual

Version 2.0.1

2006 年 9 月 7 日

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

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

目次

表目次

図目次

1.はじめに

1.1.マニュアルについて

このマニュアルは Armadillo-220 と Armadillo-230、Armadillo-240 の 3 製品のソフトウェア開発について記載されています。特にシリーズ内製品の指定がない場合は Armadillo を代名詞として使用します。シリーズ全体を表わすときは「Armadillo-200 シリーズ」と呼びます。

表 1-1 製品の呼び名

呼び名	説明
Armadillo	とくに指定がない場合の代名詞として
Armadillo-220	Armadilo-220 固有の場合
Armadillo-230	Armadilo-220 固有の場合
Armadillo-240	Armadilo-220 固有の場合
Armadillo-200 シリーズ	シリーズ製品全体を表わす場合

また、このマニュアルはソフトウェア開発者向けに書かれています。Armadillo-200 シリーズを使い、組み込み機器を開発するときに必要となる項目を記載しています。

- 基本的な使い方
- 開発環境の準備
- コンパイル方法
- データの書き換え方
- アプリケーション開発

初期状態の Armadillo-200 シリーズには個々の特徴を活かしたサンプルアプリケーションが組込まれています。サンプルアプリケーションの使い方は、別紙「Startup Guide」に記載してありますので、そちらをご覧ください。サンプルアプリケーションのソースコードは CD-ROM または Armadillo Official Site から入手可能です。ぜひ開発の参考にしてください。

Armadillo-200 シリーズ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 1-2 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 1-3 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
[a2x0 /]#	Armadillo 上の特権ユーザで実行
[a2x0 /]$	Armadillo 上の一般ユーザで実行

1.4. 謝辞

Armadillo-200 シリーズ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 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（及び、Armadillo-240 では RS232C レベル変換アダプタ）
  D-Sub9 ピン（メスーメス）の「クロス接続用」ケーブルです。RS232C **レベル変換アダプタをコネクタ基板に接続する際は、黄色のケーブルが 1 ピン側になるよう接続してください。**

- 付属 CD-ROM（以降、付属 CD）
  Armadillo-200 シリーズ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(と 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 に接続してください。

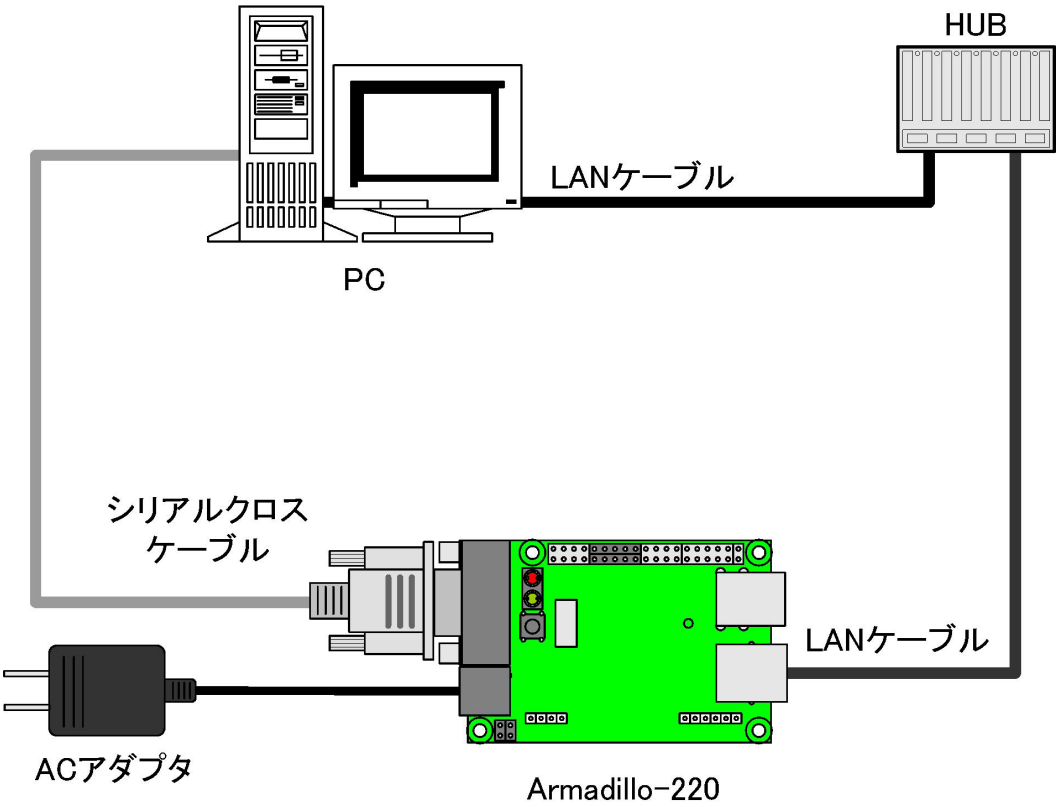

図 2-1 Armadillo-220 接続例

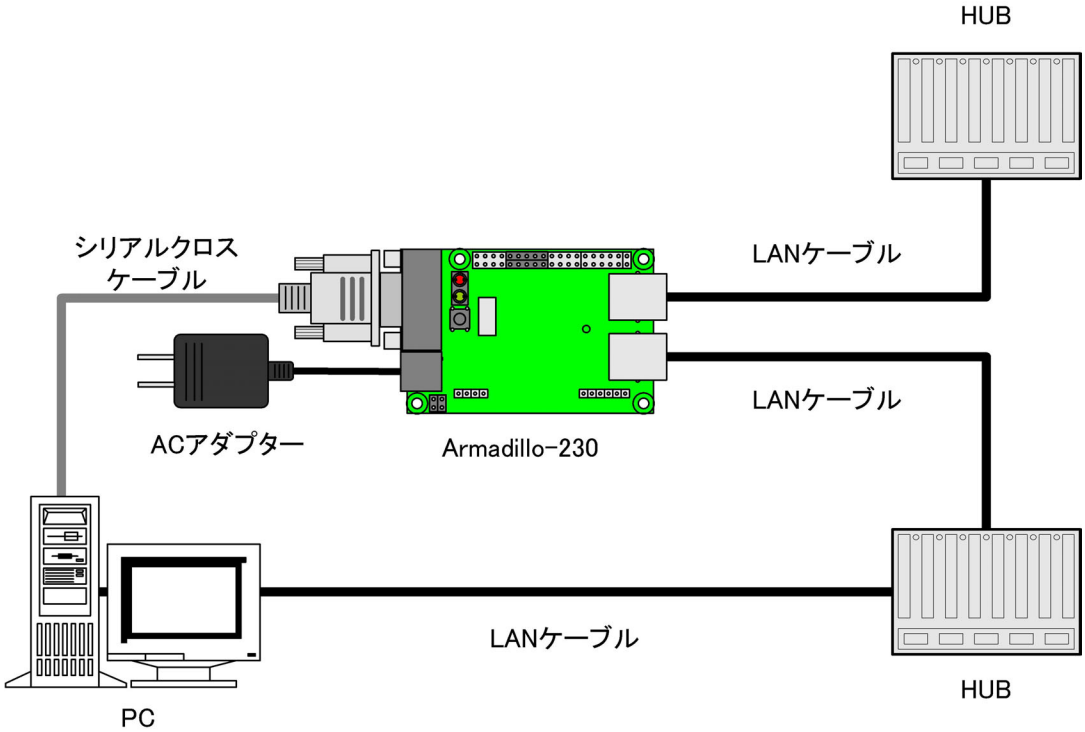

図 2-2 Armadillo-230 接続例

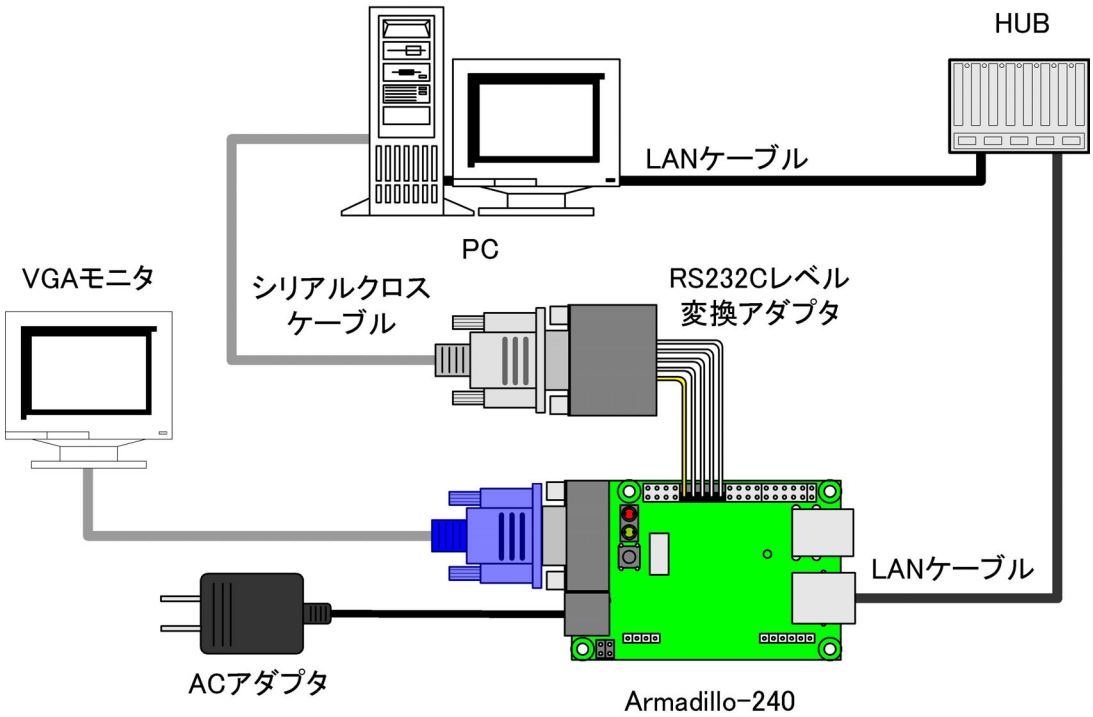

図 2-3 Armadillo-240 接続例

2.3. ジャンパピンの設定について

Armadillo-200 シリーズ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。Hermit や CPU オンチップブート ROM の使用については、「5 Flash メモリの書き換え方法」や「6.4CPU オンチップブート ROM」で説明します。

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

TIPS

ジャンパのオープンまたはショートとは、ジャンパピンにジャンパソケットを

- オープン: 挿さない
- ショート: 挿す

状態を表わします。

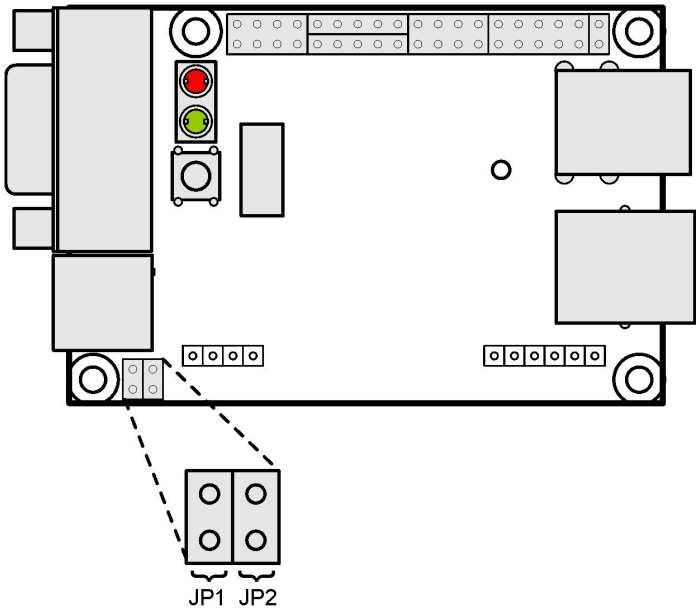

図 2-4 ジャンパの位置

3.開発環境の準備

作業用の PC 上で Armadillo-200 シリーズ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 lthe 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
libpcre3	4.5-1.2sarge1	Perl 5 Compatible Regular Expression Library - runtime files
libpcre3-dev	4.5-1.2sarge1	Perl 5 Compatible Regular Expression Library - development files
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 ファイルシステムの生成ツール[1]
file	4.12-1 以降	マジック番号を使ってファイルの種類を調べる
sed	4.1.2-8 以降	The GNU sed ストリームエディタ
perl	5.8.4-8 以降	Perl スクリプト言語のインタープリタ

[1] 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. クロス開発用ライブラリパッケージの作成方法

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

まず、作成したいクロス開発用パッケージの元となるライブラリパッケージを取得します。元となるパッケージは、ARM 用のパッケージです。例えば、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 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 クロス開発用ライブラリパッケージの作成(deb)

「libncurses5-arm-cross_x.x-x_all.deb」という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 クロス開発用ライブラリパッケージの作成(rpm、tgz)

4.使用方法

この章では Armadillo-200 シリーズの基本的な使用方法の説明を行ないます。

4.1.起動の前に

Armadillo のシリアルポート 1 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。

注意

Armadillo-240 では、RS232C レベル変換アダプタ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、Linux が起動します。正常に起動した場合、シリアルポート 1 に起動ログが出力されます。以下は、Armadillo-240 における例です。

```
Uncompressing  kernel.....................................................................done.
Uncompressing ramdisk.....................................................................done.
Doing console=ttyAM0,115200
Doing mtdparts=armadillo2x0-nor:0x10000(bootloader)ro,0x170000(kernel),0x670000(userland),-(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 1
8:43:41 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(bootloader)ro,0x170000(kernel),0x670000(us
erland),-(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.zydas.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
Stating 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(Armadillo-240 の例)

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 とシリアルポート 2 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

Armadillo 内の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

Armadillo を終了するには、電源を切断します。

4.5. 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 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。

4.5.1. 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

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。

```
[a2x0 /]# ifup eth0
```

図 4-6 ネットワーク接続の開始

```
[a2x0 /]# ifdown eth0
```

図 4-7 ネットワーク接続の終了

4.5.5. ネットワーク設定を Flash 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 Flash 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。

```
[a2x0 /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. ネットワークブリッジの設定

複数のネットワークインターフェイスを持つ Armadillo では、ネットワークブリッジ機能を利用することができます。設定にはブリッジユーティリティ「brctl」コマンドを利用します。

ネットワークブリッジインターフェイスには、eth0 などのインターフェイスと同じように IP 番号を割り当てることができます。IP アドレスを割り当てることで、他の LAN インターフェイスと同じように使用することができます。IP 番号の設定については「4.5 ネットワーク設定」を参照してください。

この章では 2 つの LAN インターフェイスを持っている Armadillo-230 を例として使用します。

4.6.1. ネットワークブリッジ設定の準備

始めに、Armadillo-230 でネットワーク設定がすでに有効になっている場合、「4.5.4ネットワーク」を参考にして無効にしてください。次に、各ネットワークインターフェイスの保持する IP アドレスを完全に解放した上で有効化するために、以下のようにコマンド入力してください。

```
[a230 ~]# ifconfig eth0 0.0.0.0
[a230 ~]# ifconfig eth1 0.0.0.0
```

図 4-8 ブリッジに追加するインターフェイスの有効化

4.6.2. ブリッジ作成

ブリッジを実現するため、brctl を利用して論理的なブリッジインターフェイスを作成します。続けて、このブリッジインターフェイスに 2 つのネットワークインターフェイスを追加します。

```
[a230 ~]# brctl addbr br0
[a230 ~]# brctl addif br0 eth0
device eth0 entered promiscuous mode
[a230 ~]# brctl addif br0 eth1
device eth1 entered promiscuous mode
```

図 4-9 ブリッジの作成

4.6.3. ブリッジの有効化

ブリッジを有効にする方法は、通常のネットワークインターフェイスと同様です。以下のコマンドを入力すると、ブリッジが動作し始めます。

```
[a230 ~]# ifconfig br0 0.0.0.0
br0: port 2(eth1) entering learning state
br0: port 1(eth0) entering learning state
br0: port 2(eth1) entering forwarding state
br0: port 1(eth0) entering forwarding state
```

図 4-10 ブリッジの有効化

4.6.4. ブリッジの廃棄

ブリッジの使用をやめる場合、まずブリッジインターフェイスを無効化するコマンドを入力します。

```
[a230 ~]# ifconfig br0 down
br0: port 2(eth1) entering disabled state
br0: port 1(eth0) entering disabled state
```

図 4-11 ブリッジの無効化

次に、ブリッジを完全に廃棄するために、追加されているネットワークインターフェイスを外し、最後にブリッジインターフェイスを削除します。

```
[a230 ~]# brctl delif br0 eth0
[a230 ~]# brctl delif br0 eth1
[a230 ~]# brctl delbr br0
```

図 4-12 ブリッジの廃棄

4.6.5. ブリッジのスクリプト例

brctl コマンドを利用すると、ブリッジの状態表示や STP というブリッジプロトコルの設定ができます。Armadillo-230 では、こうした機能を利用するためのサンプルスクリプト「/etc/init.d/bridges」が用意されています。

このスクリプトを利用することで、簡単にブリッジ設定や STP の設定を行うことができます。以下は、このスクリプトを利用してブリッジを有効化する場合の例です。

```
[a230 ~]# /etc/init.d/bridges create
Creating bridge:
device eth0 entered promiscuous mode
device eth1 entered promiscuous mode
Upping bridge (8sec):
br0: port 2(eth1) entering listening state
br0: port 1(eth0) entering listening state
br0: port 2(eth1) entering learning state
br0: port 1(eth0) entering learning state
br0: topology change detected, propagating
br0: port 2(eth1) entering forwarding state
br0: topology change detected, propagating
br0: port 1(eth0) entering forwarding state
```

図 4-13 ブリッジのスクリプト例

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

4.8. 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

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 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 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 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_xxxxxxxx.zip)」を任意のフォルダに展開します。

5.2. リージョン指定について

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、最初に実行されるソフトウェアのイメージを格納する領域です。ブートローダーは、シリアル経由で 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[1]

リージョン	ファイル名
bootloader	loader-armadillo2x0-eth-vx.x.x.bin
kernel	linux-a2x0-x.xx.bin.gz
userland	romfs-a2x0-recover-x.xx.img.gz romfs-a2x0-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。

[1] x にはバージョン番号の任意の数値が入ります

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 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 のシリアルポート 1 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。

```
[PC ~]$ hermit download -i linux.bin.gz -r kernel
```

コマンド指定(固定)　ファイル名　リージョン指定

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 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 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。

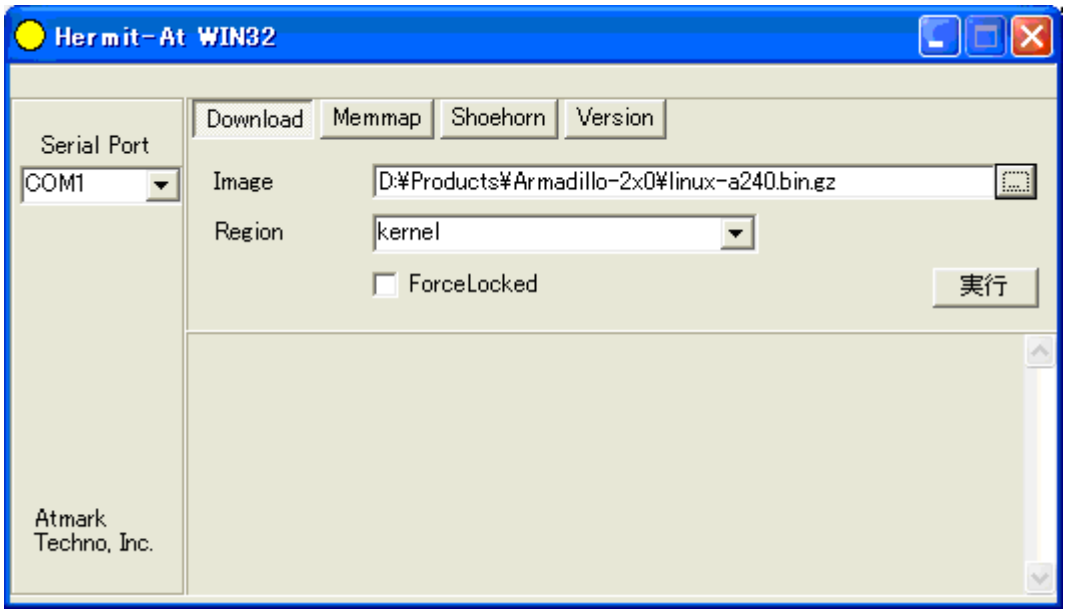

図 5-3 Download 画面 (Armadillo-240 の例)

「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。

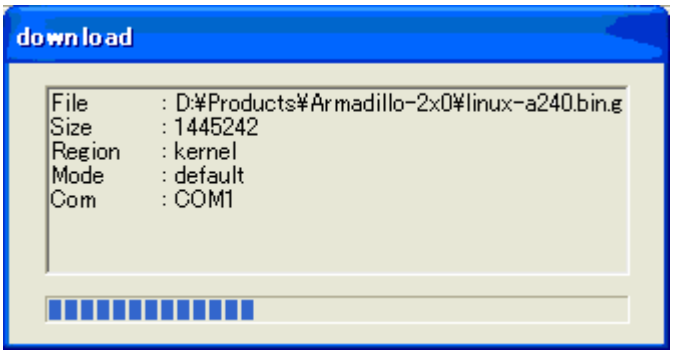

図 5-4 書き換え進捗ダイアログ(Armadillo-240 の例)

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、「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 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、Armadillo 上で netflash というアプリケーション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 メモリの書き換え」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。書き換えの最中は Armadillo 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 上での kernel イメージを変更するコマンド例です。

```
[a2x0 ~]# 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(Armadillo-240 の場合)

※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。

```
[a2x0 ~]# 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-200 シリーズ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 ブートプログラムと協調動作するダウンローダ （Armadillo 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-200 シリーズ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
loader-armadillo2x0-eth	ネットワーク通信が可能なブートローダー

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 zxf 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 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 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-200 シリーズの 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

① Armadillo の電源が**切断**されていることを確認し、Armadillo-200 シリーズのシリアルポート 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。

② Armadillo のジャンパ JP1 をショートに設定します。

③ 作業用 PC で shoehorn を起動します。

```
[PC ~]$ shoehorn --boot --terminal --initrd /dev/null
        --kernel /usr/lib/hermit/loader-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 を接続した場合の例です。他のシリアルポートに接続した場合は、shoehorn コマンドのオプションに
--port [シリアルポート名]
を追加してください。

※ コマンドは１行で入力します

④ Armadillo 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 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(JP1)設定を確認してください。

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

以上で作業用 PC から hermit を使用して Armadillo 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

① Armadillo の電源が切断されていることを確認し、Armadillo-200 シリーズのシリアルポート1(と、作業用 PC のシリアルポートをクロス(リバース)シリアルケーブルで接続します。

② Armadillo のジャンパ JP1 をショートに設定します。

③ 作業用 PC で「Hermit-At WIN32」を起動します。

④ 「Shoehorn」ボタンをクリックします。

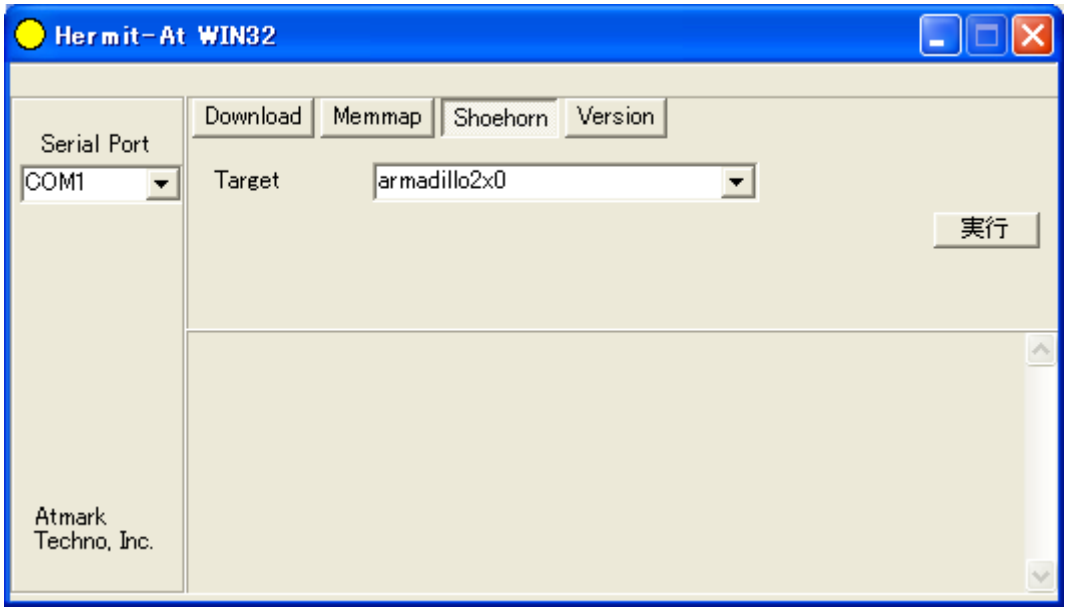

図 6-2 Shoehorn 画面

⑤ "Target" に armadillo2x0 を指定します。

⑥ 「実行」ボタンをクリックすると図 6-3が表示されます。

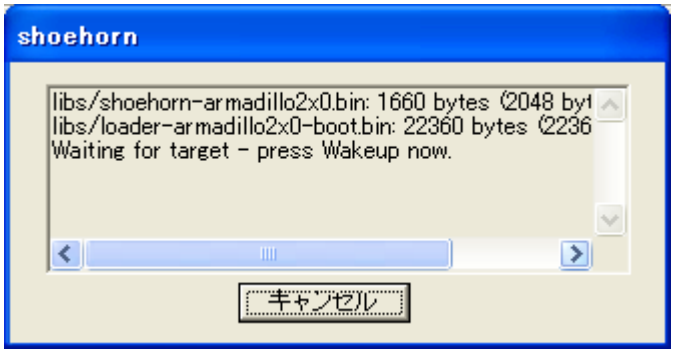

図 6-3 shoehorn ダイアログ

⑦ Armadillo 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 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(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 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-200 シリーズでは、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

① シリアルコンソールソフトの起動
Armadillo のシリアルポート 1(と作業用 PC をシリアルケーブルで接続し、シリアルコンソールソフトを起動します。次のように通信設定を行なってください。

表 6-3 シリアル通信設定

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

② ジャンパピンの設定
Armadillo 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。

③ Armadillo の起動
Armadillo 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 (armadillo2x0) 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 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 でイメージファイルを作成する

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ファイルを作成する方法を説明します。atmark-dist に関する詳しい使用方法は、「atmark-dist Developers Guide」を参照してください。

注意

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

7.1. ソースコードアーカイブの展開

付属 CD の 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-220 では「**Armadillo-220.Base**」、Armadillo-230 では「**Armadillo-230.Base**」、Armadillo-240 では「**Armadillo-240.Base**」と入力してください。

```
*
* Select the Product you wish to target
*
AtmarkTechno Products (Armadillo, Armadillo-220.Base, Armadillo-220.Recover,
Armadillo-230.Base, Armadillo-230.Recover, 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-2x0.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-200 シリーズでは、「**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ないます。すべての設定が終わるとプロンプトに戻ります。

7.3. ビルド

実際にビルドするには以下のコマンドを入力してください。

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

dist のバージョンによっては、make の途中で一時停止し、未設定項目の問合せが表示される場合があります。通常はデフォルト設定のままで構いませんので、このような場合はそのまま Enter 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 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/gpiod)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-200 シリーズの 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)の点灯状態を取得します	状態を保存するバッファ(最小１Byte)
LED_RED_BLINKON	LED(赤)を点滅を開始します	なし
LED_RED_BLINKOFF	LED(赤)を点滅を停止します	なし
LED_RED_BLINKSTATUS	LED(赤)の点滅状態を取得します	状態を保存するバッファ(最小１Byte)
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)の点灯状態を取得します	状態を保存するバッファ(最小１Byte)
LED_GREEN_BLINKON	LED(緑)を点滅を開始します	なし
LED_GREEN_BLINKOFF	LED(緑)を点滅を停止します	なし
LED_GREEN_BLINKSTATUS	LED(緑)の点滅状態を取得します	状態を保存するバッファ(最小１Byte)

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(オプション)

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	bootloader
		1	mtdr0	bootloader (read only)
		2	mtd1	kernel
		3	mtdr1	kernel (read only)
		4	mtd2	userland
		5	mtdr2	userland (read only)
		6	mtd3	config
		7	mtdr3	config (read only)
		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	bootloader
		1	mtdblock1	kernel
		2	mtdblock2	userland
		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 (Armadillo-240 のみ)

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                    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 --->
[*]   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 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」などがありますが、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アルアップ接続」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アルアップ接続」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
2.00	2006.8.17	・Armadillo-220 Softoware Manual v1.01 と Armadillo-240 Softoware Manual v1.01 をベースに統一し、Armadillo-230 の記述を加え新規作成
2.01	2006.9.5	・「4.6ネットワークブリッジの設定」を追加 ・「9.2 LED」仕様に点滅状態制御についての記述を追加

Armadillo-200 シリーズ　Software Manual　2006 年 9 月 5 日　version 2.01

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