# AN010
# Software Manual

Version 1.0.1

2004 年 12 月 28 日


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


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

# 目次

## 表目次



## 図目次

# 1.はじめに

## 1.1.マニュアルについて

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

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

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

## 1.2.フォントについて

このマニュアルでは以下のようにフォントを使っています。

**表 1-1 使用しているフォント**

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

## 1.3.コマンド入力例の表記について

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

**表 1-2 表示プロンプトと実行環境の関係**

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

## 1.4. 謝辞

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

## 1.5. 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-9 を使用する前に、次のものを準備してください。

- 作業用 PC
  Linux もしくは Windows が動作し、1 ポート以上のシリアルポートを持つ PC です。
- シリアルクロスケーブル
  D-Sub9 ピン（メス－メス）の「クロス接続用」ケーブルです。
- 付属 CD-ROM
  Armadillo-9 に関する各種マニュアルやソースコードが収納されています。
- シリアルコンソールソフト
  minicom や Tera Term などのシリアルコンソールソフトです。（Linux 用のソフトは付属 CD の「tools」ディレクトリにあります。）作業用 PC にインストールしてください。

## 2.2.接続方法

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

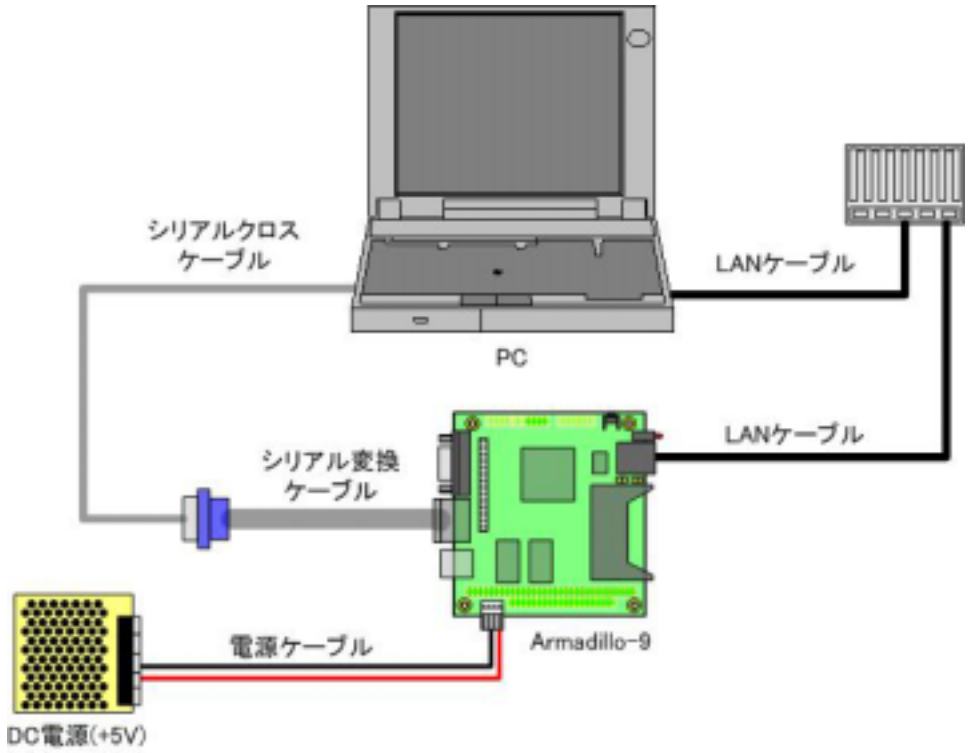



**図 2-1 Armadillo-9 接続例**

## 2.3. ジャンパピンの設定について

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

**表 2-1 ジャンパの設定とブート時の動作**

| JP1 | JP2 | Compact Flash ソケット状態 | ブート時の動作 |
|---|---|---|---|
| OFF | OFF | — | オンボード Flash メモリ内の Linux カーネルを起動 |
| OFF | ON | 挿入されている Compact Flash に Linux カーネルがブート可能な状態で適切に保存されている | Compact Flash 内の Linux カーネルを起動 |
| | | 挿入されている Compact Flash にブート可能な Linux カーネルがない、または Compact Flash 未挿入 | Hermit コマンドプロンプトを起動 |
| ON | — | — | EP9315 オンチップブート ROM を起動<br>※Shoehorn を使用したブートローダー復旧などに使用 |

※ 現在は、IDE 接続ディスクドライブ内の Linux カーネルの起動には対応していません。次期リリースバージョンのブートローダで対応予定です。

※ IDE 接続ディスクドライブを Linux のルートデバイスとして使用したい場合、オンボード Flash メモリ内カーネルと Hermit コマンド「setenv」によるルートデバイス指定「root=/dev/hda1」を組み合わせることで実現できます。

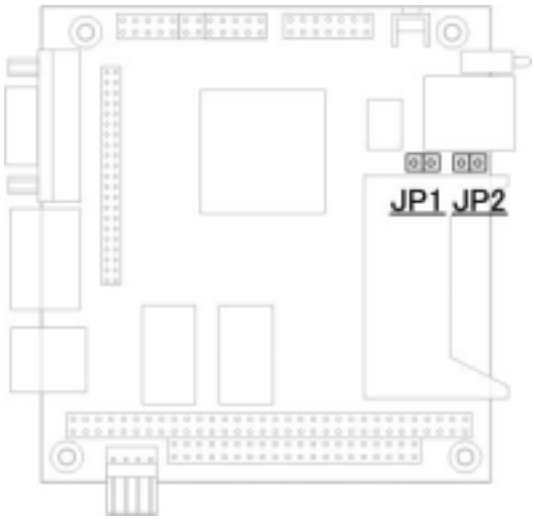



**図 2-2 ジャンパの位置**

# 3.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-9 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-9 が正常に起動しなくなる場合があります。書き換えの最中は次の点に注意してください。

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

## 3.1.ダウンローダのインストール

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

1) Linux の場合

付属の CD−ROM(以降、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_1.3-armadillo9-1_i386.deb ←deb パッケージを使用する場合

[PC ~]# rpm -i hermit-1.3_armadillo9-1.rpm       ←rpm パッケージを使用する場合

[PC ~]# tar zxf hermit-1.3-armadillo9.tgz -C /   ←tgz を使用する場合
```

**図 3-1 展開処理コマンド入力例**

2) Windows の場合

付属 CD より「Hermit host for Win32 (**tools/hermit-1.3_win32.zip**)」を任意のフォルダに展開します。

## 3.2. 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べきイメージと合わせて以下で説明します。

- bootloader
  ブートローダーと呼ばれる、電源投入後、最初に実行されるソフトウェアのイメージを格納する領域です。ブートローダーは、シリアルダウンローダを使用した Flash メモリの書き換えを行なったり、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 3-1 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-a9-1.img |
| kernel | linux.bin.gz |
| userland | romfs-a9-1.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。

## 3.3. 書き換え手順

以下の手順で Flash メモリの書き換えを行います。

### 3.3.1. ジャンパピンの設定

Armadillo-9 の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、Compact Flash ソケットには何も挿入しないでください。

ジャンパピン設定の詳細については、「2.3.ジャンパピンの設定について」を参照してください。

### 3.3.2. 書き換えイメージの転送

はじめに Armadillo-9 に電源を投入します。

1) Linux の場合

Linux が動作する作業用 PC で terminal を起動し、カーネルイメージ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 イメージに **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 3-3 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**

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

**注意**

ブートローダ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を復旧してください。

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

2) Windows の場合
「3.1ダウンローダのインストール」にてファイルを展開したディレクトリにある、「Hermit host for Win32」を起動します。

各パラメータは以下の表を参考に設定してください。

**表 3-2 Hermit for WIN32 版使用時のパラメータ例**

| 項目名 | 説明 | 設定値 |
|---|---|---|
| Port | Armadillo-9 と接続しているポート名 | COM1 (COM1 利用時) |
| Input file | 書き込みを行うファイル名 | ファイル名(ダイアログ選択可) |
| Region | 書き込むリージョンの指定 | bootloader、kernel、userland |

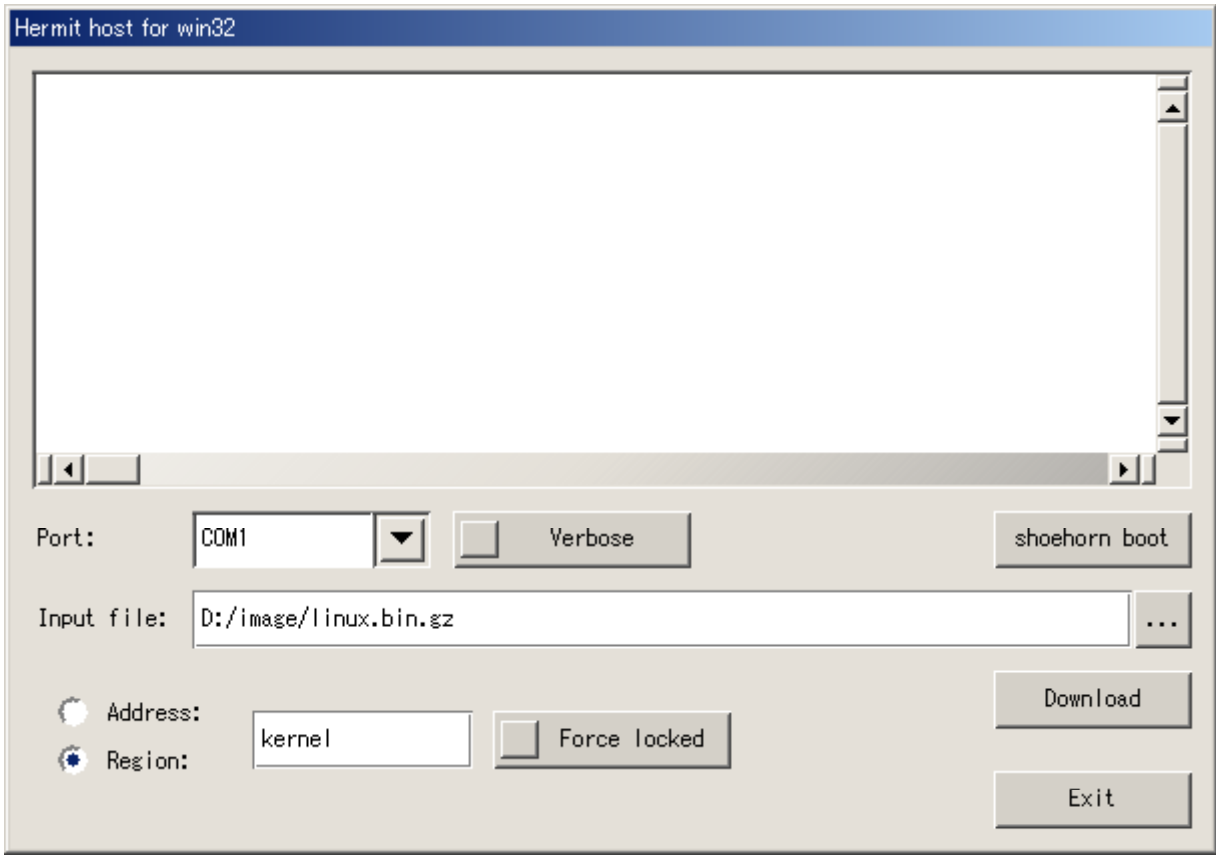



**図 3-4 書き換えイメージの転送例**

**TIPS**

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

**注意**

ブートローダ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を復旧してください。

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

## 3.4. Linux ブートオプションの設定

Armadillo-9 では、自動起動する Linux のブートオプションを設定することができます。設定は 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。

### 3.4.1. Hermit コマンドプロンプトの起動

① シリアルコンソールソフトの起動

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

**表 3-4-1 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 |

② ジャンパピンの設定

Armadillo-9 の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、Compact Flash ソケットには何も挿入しないでください。

ジャンパピン設定の詳細については、「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>
```

### 3.4.2. 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
```

**注意**

Linux ブートオプションが未設定の場合、ブートローダは Linux の起動時に自動的にオプション「console=ttyAM0,115200」を使用して、シリアルポートをコンソールにします。ブートオプション未設定状態と同じシリアルポートをコンソールとして使用する場合、setenv のパラメータに必ず「console=ttyAM0,115200」を含めてください。

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

### 3.4.3. 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
```

### 3.4.4. 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
```

### 3.4.5. 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 Flash 内のものを使用)

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

ex.2) コンソールとして VGA を使用する場合

```
hermit> setenv video
```

**TIPS**

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

# 4.使用方法

この章では Armadillo-9 の基本的な使用方法の説明を行います。

## 4.1.起動の前に

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

**表 4-1 シリアル通信設定**

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

## 4.2. 起動

JP2 を開放して電源を投入すると、Armadillo-9 が起動します。正常に起動した場合、次ようなログが出力されます。

```
Uncompressing  kernel.....................................................done.
Copying       ramdisk.done.
Doing console=ttyAM0,115200
Doing mtdparts=armadillo9-nor:0x10000(bootloader)ro,0x170000(kernel),0x670000(userland),-(config)
Linux version 2.4.27-a9-1 (atmark@build) (gcc version 3.4.1 (Debian 3.4.1-4sarge1)) #2 Sat Dec 18 11:57:29 JST 2004
CPU: Arm920Tid(wb) revision 0
Machine: Armadillo-9
alloc_bootmem_low
memtable_init
On node 0 totalpages: 16384
zone(0): 24576 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: console=ttyAM0,115200 mtdparts=armadillo9-nor:0x10000(bootloader)ro,0x170000(kernel),0x670000(user
land),-(config)
Console: colour dummy device 80x30
Calibrating delay loop... 99.73 BogoMIPS
Memory: 32MB 32MB = 64MB total
Memory: 55880KB available (1827K code, 351K data, 68K init)
Dentry cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
Journalled Block Device driver loaded
i2c-core.o: i2c core module version 2.6.1 (20010830)
i2c-algo-bit.o: i2c bit algorithm module
i2c-armadillo9: i2c Armadillo9 driver, (C) 2004 Atmark Techno, Inc.
i2c-s3531a: i2c S-3531A driver, (C) 2001-2004 Atmark Techno, Inc.
i2c-at24cxx: i2c at24cxx eeprom driver, (C) 2004 Atmark Techno, Inc.
Console: switching to colour frame buffer device 80x60
ttyAM0 at MMIO 0xff8c0000 (irq = 52) is a AMBA
ttyAM1 at MMIO 0xff8d0000 (irq = 54) is a AMBA
ttyAM2 at MMIO 0xff8e0000 (irq = 55) is a AMBA
ep93xx_eth() version: ep93xx_eth.c: V1.0 09/04/2003 Cirrus Logic
RAMDISK driver initialized: 16 RAM disks of 12288K size 1024 blocksize
loop: loaded (max 8 devices)
Uniform Multi-Platform E-IDE driver Revision: 7.00beta4-2.4
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
No card in slot: PFDR=000000ff
SCSI subsystem driver Revision: 1.00
ARMADILL09-NOR:0x00800000 at 0x60000000
 Amd/Fujitsu Extended Query Table v1.3 at 0x0040
number of CFI chips: 1
cfi_cmdset_0002: Disabling fast programming due to code brokenness.
ARMADILL09-NOR:using command line partition definition
Creating 4 MTD partitions on "NOR flash on ARMADILL09":
0x00000000-0x00010000 : "bootloader"
0x00010000-0x00180000 : "kernel"
0x00180000-0x007f0000 : "userland"
0x007f0000-0x00800000 : "config"
usb.c: registered new driver usbdevfs
```

```
usb.c: registered new driver hub
host/usb-ohci.c: USB OHCI at membase 0xff020000, IRQ 56
host/usb-ohci.c: usb-, ep93xx
usb.c: new USB bus registered, assigned bus number 1
hub.c: USB hub found
hub.c: 3 ports detected
usb.c: registered new driver hid
hid-core.c: v1.8.1 Andreas Gal, Vojtech Pavlik <vojtech@suse.cz>
hid-core.c: USB HID support drivers
usb.c: registered new driver audio
audio.c: v1.0.0:USB Audio Class driver
Initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
USB Mass Storage support registered.
mice: PS/2 mouse device common for all mice
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 4096 bind 8192)
ip_tables: (C) 2000-2002 Netfilter core team
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.97 (double precision)
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 6592 blocks [1 disk] into ram disk... done.
Freeing initrd memory: 6656K
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 68K
init started:  BusyBox v1.00 (2004.12.01-11:04+0000) multi-call binary
Mounting proc:                                             done
Starting fsck for root filesystem.
mount: /etc/mtab: Read-only file system
fsck 1.25 (20-Sep-2001)
/dev/ram0: clean, 596/824 files, 5830/6592 blocks
Checking root filesystem:                                  done
Remounting root rw:                                        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
Configuring network interfaces...                          done
Starting thttpd:                                           done
Starting inetd:                                            done

Armadillo Linux V2.0.0
Linux 2.4.27-a9-1 [armv4l arch]

armadillo9 login:
```

**図 4-1 起動ログ**

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

※ VGA 側からログインを行なうには、USB キーボードを接続する必要があります。
※ VGA を Linux カーネルブートログが出力される標準コンソールとする場合、「3.4Linux ブートオプションの設定」を参照してください。

ログインユーザは、次の 2 種類が用意されています。

**表 4-2コンソールログイン時のユーザ名とパスワード**

| | パスワード | 権限 |
|---|---|---|
| root | root | スーパーユーザ |
| guest | (なし) | 一般ユーザ |

## 4.3. ディレクトリ構成

ディレクトリ構成は次のようになっています。

**表 4-3 ディレクトリ構成の一覧**

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

## 4.4. 終了

電源を切断することで Armadillo-9 を終了させます。

ただしコンパクトフラッシュがマウントされている場合、データを破損する場合があります。この場合、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。

## 4.5. ネットワーク設定

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

### 4.5.1. 固定 IP アドレスで使用する場合

固定 IP アドレスを指定する場合の設定例を次に示します。

**表 4-4 ネットワーク設定詳細**

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

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

auto lo eth0

iface lo inet loopback

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

**図 4-2 ネットワーク設定例(固定 IP アドレス時)**

ゲートウェイを使用しない場合、gateway 指定行全体を削除するか、コメントアウトしてください。

```
#       gateway 192.168.10.1 ←先頭に#を付加することでコメントアウトされる
```

**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. DHCP を使用する場合

DHCP を利用して IP アドレスを取得する場合の設定例を次に示します。

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

auto lo eth0

iface lo inet loopback

iface eth0 inet dhcp
```

**図 4-4 ネットワーク設定例(DHCP 使用時)**

### 4.5.3. 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、networking スクリプトを実行します。このスクリプトは、環境によって/etc/rc.d/rc.start または/etc/init.d のいずれかに存在します。

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。固定 IP 接続の場合は ifconfig を用いて、DHCP 接続の場合は dhcpcd コマンドの-k オプションで、それぞれ終了します。

```
[armadillo9 /]# ifconfig eth0 down ←固定 IP 接続のネットワークを閉じる

[armadillo9 /]# dhcpcd -k          ←DHCP 接続を終了する
```

**図 4-5 ネットワーク接続の終了**

```
[armadillo9 /]# /etc/init.d/networking
```

**図 4-6 ネットワーク接続の開始**

## 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 でのログインは行えません。root 権限が必要な作業を行なう場合、guest でログイン後に「**su**」コマンドで root 権限を取得してください。

**表 4-5 telnet ログイン時のユーザ名とパスワード**

| ユーザ名 | パスワード |
|---|---|
| guest | なし |

## 4.7. ファイル転送

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

**表 4-6 ftp のユーザ名とパスワード**

| ユーザ名 | パスワード |
|---|---|
| ftp | なし |

## 4.8. Web サーバ

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

# 5. 開発環境の準備

Linux 上で Armadillo-9 の開発を行なうことができます。

## 5.1. 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、これを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 5-1 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 | 2.14.90.0.7-8 | Binary utilities |
| cpp-arm-linux | 3.4.1-4sarge1 | The GNU C preprocessor |
| g++-arm-linux | 3.4.1-4sarge1 | The GNU C++ compiler |
| gcc-arm-linux | 3.4.1-4sarge1 | The GNU C compiler |
| libc6-arm-cross | 2.3.2.ds1-13 | GNU C Library: Shared libraries and Timezone data |
| libc6-dev-arm-cross | 2.3.2.ds1-13 | GNU C Library: Development Libraries and Header Files |
| libc6-pic-arm-cross | 2.3.2.ds1-13 | GNU C Library: PIC archive library |
| libc6-prof-arm-cross | 2.3.2.ds1-13 | GNU C Library: Profiling Libraries |
| libdb1-compat-arm-cross | 2.1.3-7 | The Berkeley database routines |
| libgcc1-arm-cross | 3.4.1-4sarge1 | GCC support library |
| libstdc++6-0-arm-cross | 3.4.1-4sarge1 | The GNU Standard C++ Library v3 |
| libstdc++6-0-dbg-arm-cross | 3.4.1-4sarge1 | The GNU Standard C++ Library v3 (debugging files) |
| libstdc++6-0-dev-arm-cross | 3.4.1-4sarge1 | The GNU Standard C++ Library v3 (development files) |
| libstdc++6-0-pic-arm-cross | 3.4.1-4sarge1 | The GNU Standard C++ Library v3 (shared library subset kit) |
| linux-kernel-headers-arm-cross | 2.5.999-test7-bk-16 | Linux Kernel Headers for 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 5-1 開発用ツールチェーンの展開例**

## 5.2. 環境変数の設定

開発ツールチェーンを使いやすくするために、開発ツールチェーンの実行ファイルが入っているディレクトリを PATH 環境変数に追加します。シェルによって設定方法が異なりますので、詳しくはお使いのシェルのマニュアルを参照してください。

ここでは bash の設定方法を例に取ります。

```
[PC ~]$ export PATH="$PATH:/usr/local/bin"
[PC ~]$ echo $PATH
/usr/bin:/bin:/usr/bin/X11:/usr/sbin:/sbin:/usr/local/bin
[PC ~]$
```

**図 5-2 環境変数「PATH」の設定(bash の場合)**

# 6.ブートローダー

この章では、Armadillo-9 のブートローダーに関して説明します。

## 6.1.パッケージの準備

付属 CD の hermit ディレクトリから以下のパッケージを、作業 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-1.3-armadillo9 | Armadillo9 ブートプログラムと協調動作するダウンローダ（Armadillo-9 ブートプログラム自体も含む） |
| shoehorn-3.4-armadillo9 | 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「5.1.クロス開発環境パッケージのインストール」を参照してください。

## 6.2.ブートローダーの種類

Armadillo-9 で用意されているブートローダーを以下に記載します。

**表 6-2 ブートローダー 一覧**

| ブートローダー名 | 説明 |
|---|---|
| loader-armadillo9 | 出荷時に Flash に書き込まれている標準ブートローダ。COM1 を使用。 |
| loader-armadillo9-ttyAM1 | COM2 を使用するブートローダー。 |
| loader-armadillo9-notty | コンソールを使用しないブートローダー。 |

## 6.3.ブートローダーの作成

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

### 6.3.1. ソースコードの準備

付属 CD の source ディレクトリから、hermit_1.3-armadillo9-1.tar.gz を作業用 PC にコピーし、解凍します。

```
[PC ~]$ tar zxf hermit_1.3-armadillo9-1.tar.gz
```

### 6.3.2. ビルド

解凍してできたディレクトリへ移動し、make コマンドを入力します。

```
[PC ~]$ cd hermit-1.3-armadillo9-1
[PC ~]$ make target=armadillo9
```

make が完了後、hermit-1.3-armadillo9-1/src/target/armadillo9 のディレクトリにブートローダーが作成されます。

## 6.4. CPU **オンチップブート** ROM

loader-armadillo9-notty が書き込まれている Armadillo-9 のブートローダーを書き換えるときや、不正なブートローダーを書込んでしまい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 のデータを RAM へ展開し、ブートローダーを出荷状態に戻すことができます。以下にその手順を説明します。

### 6.4.1. **ブートローダーを出荷状態に戻す**

1) Linux の場合
   ① Armadillo-9 の電源が Off であることを確認し、Armadillo-9 の COM1 と、作業 PC のシリアルポートをクロス(リバース)シリアルケーブルで接続します。
   ② Armadillo-9 のジャンパ JP1 をショートします。
   ③ 作業 PC で shoehorn を起動します。

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

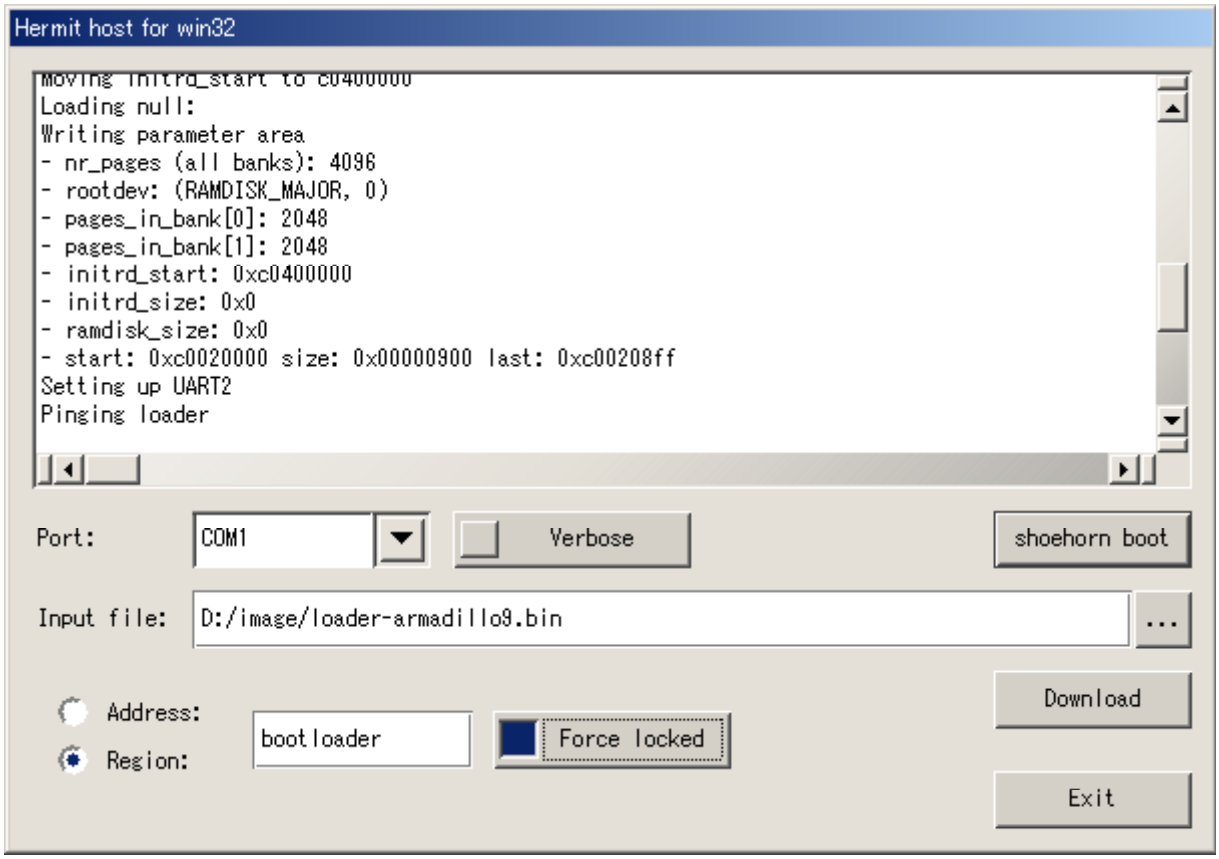
※ 上記は、作業 PC のシリアルポート"/dev/ttyS0"に Armadillo-9 を接続した場合の例です。
他のシリアルポートに接続した場合は、shoehorn コマンドのオプションに
--port [シリアルポート名]
を追加してください。
※ コマンドは１行で入力します

④ Armadillo-9 の電源を On にする
※ すぐにメッセージ表示が開始されます。正常に表示されない場合は、Armadillo-9 の電源を Off にし、シリアルケーブルの接続や Armadillo-9 のジャンパ(JP1)設定を確認してください。

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

以上で作業 PC から hermit を使用して Armadillo-9 へブートローダーをダウンロードする準備が整います。Flash メモリへのイメージの書き換え方は、「3.Flash メモリの書き換え方法」を参照してください。

2) Windows の場合
   ① Armadillo-9 の電源が Off であることを確認し、Armadillo-9 の COM1 と、作業 PC のシリアルポートをクロス(リバース)シリアルケーブルで接続します。
   ② Armadillo-9 のジャンパ JP1 をショートします。
   ③ 作業 PC で「Hermit host for Win32」を起動します。
   ④ 「shoehorn boot」ボタンを押します。




**TIPS**

この例は、作業 PC のシリアルポート"COM1"に Armadillo-9 を接続した場合の例です。他のシリアルポートに接続した場合は、「Port」の選択を変更してください。

   ⑤ Armadillo-9 の電源を On にする

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

以上で作業 PC から hermit を使用して Armadillo-9 へブートローダーをダウンロードする準備が整います。Flash メモリへのイメージの書き換え方は、「3.Flash メモリの書き換え方法」を参照してください。

# 7.uClinux-dist でイメージを作成

この章では、uClinux-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。uClinux-dist に関する詳しい使用方法は、「uClinux-dist Developers Guide」を参照してください。

**注意**

**uClinux-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 uClinux-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開発 PC 自体の OS を破壊しないために、すべての作業は root ユーザではなく一般ユーザで行なってください。**

## 7.1.ソースコードアーカイブの展開

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

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

**図 7-1 ソースコードの展開例**

次に Linux カーネルソースコードを展開し、uClinux-dist ディレクトリ内に linux-2.4.x という名前でシンボリックリンクを作成します。付属 CD の source ディレクトリに linux-2.4.27-a9-1.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-2.4.27-a9-1.tar.gz
 :
 :
[PC ~]$ cd uClinux-dist
[PC ~/uClinux-dist]$ ln -s ../linux-2.4.27-a9-1 ./linux-2.4.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 ~/uClinux-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 ~/uClinux-dist]$ make config
config/mkconfig > config.in
#
# No defaults found
#
*
* Vendor/Product Selection
*
*
* Select the Vendor you wish to target
*
Vendor (3com, ADI, Akizuki, Apple, Arcturus, Arnewsh, AtmarkTechno, Atmel, Avnet,
Cirrus, Cogent, Conexant, Cwlinux, CyberGuard, Cytek, Exys, Feith, Future, GDB,
Hitachi, Imt, Insight, Intel, KendinMicrel, LEOX, Mecel, Midas, Motorola, NEC,
NetSilicon, Netburner, Nintendo, OPENcores, Promise, SNEHA, SSV, SWARM, Samsung,
SecureEdge, Signal, SnapGear, Soekris, Sony, StrawberryLinux, TI, TeleIP,
Triscend, Via, Weiss, Xilinx, senTec) [SnapGear] (NEW) AtmarkTechno
```

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

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

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

```
*
* Kernel/Library/Defaults Selection
*
*
* Kernel is linux-2.4.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。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 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。

## 7.3. 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 ~/uClinux-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で構いませんので、こうした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、uClinux-dist/images ディレクトリに、カーネルイメージである linux.bin.gz とユーザーランドイメージである romfs.img が作成されます。作成したイメージを Armadillo-9 に書き込む方法は「3.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<br>0x6000ffff | bootloader | 64KB | Hermit ブートローダ<br>「loader-armadillo9.bin」のイメージ |
| 0x60010000<br>0x6017ffff | kernel | 約 1.44MB | Linux カーネル<br>「linux.bin.gz」のイメージ<br>(非圧縮イメージ、gz 圧縮イメージに対応) |
| 0x60180000<br>0x607effff | userland | 約 6.44MB | ユーザランド<br>「romfs.img」のイメージ<br>(非圧縮イメージ、gz 圧縮イメージに対応) |
| 0x607f0000<br>0x60800000 | 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<br>システム | 説明 |
|---|---|---|---|
| 0xc0018000 | kernel | — | linux 起動前に<br>Flash メモリから展開・コピー |
| 0xc0800000 | userland | EXT2 | linux の起動前に<br>Flash メモリから展開・コピー |

**表 8-3 メモリマップ(PC/104)**

| Linux<br>論理アドレス | 物理アドレス | 説明 |
|---|---|---|
| 0xf2000000<br>0xf200ffff | 0x12000000<br>0x1200ffff | PC/104 I/O Space (8bit) |
| 0xf3000000<br>0xf3ffffff | 0x13000000<br>0x13ffffff | PC/104 Memory Space (8bit) |
| 0xf6000000<br>0xf600ffff | 0x22000000<br>0x2200ffff | PC/104 I/O Space (16bit) |
| 0xf7000000<br>0xf7ffffff | 0x17000000<br>0x17ffffff | PC/104 Memory Space (16bit) |

# 9.割り込み(IRQ)について

**表 9-1 割り込み(IRQ)一覧表**

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

| | | | |
|---|---|---|---|
| 44 | IRQ_VIDEOFIFO | VIC #44 | VIDEOFIFO |
| 45 | IRQ_SSPRX | VIC #45 | SSPRX |
| 46 | IRQ_SSPTX | VIC #46 | SSPTX |
| 47 | IRQ_GPIO4 | VIC #47 | GPIO4 |
| 48 | IRQ_GPIO5 | VIC #48 | GPIO5 |
| 49 | IRQ_GPIO6 | VIC #49 | GPIO6 |
| 50 | IRQ_GPIO7 | VIC #50 | GPIO7 |
| 51 | IRQ_TIMER | VIC #51 | TIMER3 |
| 52 | IRQ_UART1 | VIC #52 | UART1 |
| 53 | IRQ_SSP | VIC #53 | SSP |
| 54 | IRQ_UART2 | VIC #54 | UART2 |
| 55 | IRQ_UART3 | VIC #55 | UART3 |
| 56 | IRQ_USH | VIC #56 | USH |
| 57 | IRQ_PME | VIC #57 | PME |
| 58 | IRQ_DSP | VIC #58 | DSP |
| 59 | IRQ_GPIO | VIC #59 | GPIO |
| 60 | IRQ_SAI | VIC #60 | SAI |
| 64 | IRQ_ISA3 | PC/104 #3 | |
| 65 | IRQ_ISA4 | PC/104 #4 | |
| 66 | IRQ_ISA5 | PC/104 #5 | |
| 67 | IRQ_ISA6 | PC/104 #6 | |
| 68 | IRQ_ISA7 | PC/104 #7 | |
| 69 | IRQ_ISA9 | PC/104 #9 | |
| 70 | IRQ_ISA10 | PC/104 #10 | |
| 71 | IRQ_ISA11 | PC/104 #11 | |
| 72 | IRQ_ISA12 | PC/104 #12 | |
| 73 | IRQ_ISA14 | PC/104 #14 | |
| 74 | IRQ_ISA15 | PC/104 #15 | |

**表 9-2 PC/104 IRQ サポート関数**

| 用途 | 関数定義 | 使用例 |
|---|---|---|
| Linux の IRQ 番号から PC/104 の IRQ 番号へ変換 | static __inline__ unsigned int **convirq_to_isa** (unsigned int irq); | **//Linux の IRQ 番号 IRQ_ISA3 を<br>//PC/104 の IRQ 番号に変換<br>const unsigned<br>linux_irq = IRQ_ISA3;<br>unsigned int isa_irq;<br>isa_irq =<br>convirq_to_isa(linux_irq);** |
| PC/104 の IRQ 番号から Linux の IRQ 番号へ変換 | static __inline__ unsigned int **convirq_from_isa** (unsigned int irq); | **//PC/104 の IRQ 番号 3 を<br>//Linux の IRQ 番号に変換<br>const unsigned int isa_irq = 3;<br>unsigned linux_irq;<br>linux_irq =<br>convirq_from_isa(isa_irq);** |

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

# 10. オリジナルデバイスドライバ仕様

## 10.1. 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 10-1 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<br>EP93XX_GPIO_PBDR<br>EP93XX_GPIO_PCDR<br>EP93XX_GPIO_PDDR<br>EP93XX_GPIO_PEDR<br>EP93XX_GPIO_PFDR<br>EP93XX_GPIO_PGDR<br>EP93XX_GPIO_PHDR<br>EP93XX_GPIO_PADDR<br>EP93XX_GPIO_PBDDR<br>EP93XX_GPIO_PCDDR<br>EP93XX_GPIO_PDDDR<br>EP93XX_GPIO_PEDDR<br>EP93XX_GPIO_PFDDR<br>EP93XX_GPIO_PGDDR<br>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 EP9351 User's Guide の Chapter 28 GPIO Interface を参照してください。

**例 10-1 ep93xx_gpio.h の構造体とマクロ定義**

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

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

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

**例 10-2 GPIO 操作のサンプルプログラム**

```
#include <fcntl.h>
#include <stdio.h>

#include "ep93xx_gpio.h"

int main (void)
{
        int fd;
        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 のピン 0 を入力、ピン 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 のピン 0 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 & 0x1);

        //Port B のピン 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;
}
```

## 10.2. 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 10-2 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 |

## 10.3. オンボード Flash メモリ

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 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。
Linux 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 10-3 MTD ノード**

| タイプ | メジャー番号 | マイナー番号 | ノード名 (/dev/xxx) | デバイス名 |
|---|---|---|---|---|
| キャラクタデバイス | 90 | 0 | mtd0 | bootloader |
| | | 1 | mtdr0 | bootloader (read only) |
| | | 2 | mtd1 | kernel |
| | | 3 | mtdr1 | kernel (read only) |
| | | 4 | mtd2 | usernald |
| | | 5 | mtdr2 | userland (read only) |
| | | 6 | mtd3 | config |
| | | 7 | mtdr4 | config (read only) |
| ブロックデバイス | 31 | 0 | mtdblock0 | bootloader |
| | | 1 | mtdblock1 | kernel |
| | | 2 | mtdblock2 | userland |
| | | 3 | mtdblock3 | config |

## 10.4. 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。

### 10.4.1. 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。

### 10.4.2. 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。

### 10.4.3. USB Human Interafce 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。

## 10.5. VGA 出力

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 ビットカラーにも対応しています。

**TIPS**

現在のカーネルでは、解像度やカラー設定を変更するにはリコンパイルが必要です。

## 10.6. IDE と Compact Flash

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

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


**注意**

カーネル内蔵 Compact Flash ドライバは、カーネルコンフィギュレーションの「ATA/ATAPI/MFM/RLL support」→「IDE, ATA and ATAPI Block devices」→「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。

# 11. Compact Flash システム構築

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

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

① Armadillo-9 の起動
Armadillo-9 に Compact Flash を挿入し、JP1:OFF JP2:OFF としてオンボード Flash メモリ内の Linux を起動します。

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

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

**TIPS**

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

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

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

**TIPS**

初期状態のオンボード Flash メモリ内 Linux システムには、mke2fs が用意されていません。この場合、
1) uClinux-dist を使用して mke2fs の追加(「Customize Vendor/User Settings」から「Filesystem Applications」→「mke2fs」)したユーザランドを再構築し、使用する
2) Compact Flash を使用可能な PC 上の Linux から作業する
などの方法を用いてください。
(次期リリースバージョンのユーザランドでは、初期状態で mke2fs を含むよう変更する予定です)

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

```
[armadillo9 ~]# mount /dev/hda1 /mnt
```

⑤ Compact Flash 上へのシステム構築
/mnt にマウントされた Compact Flash に、あらかじめ用意したシステムイメージからファイルをコピーするなどして、システムを構築します。

```
[armadillo9 ~]# (cd /tmp; tar cf - *) | (cd /mnt; tar xf -)
```

※あらかじめ/tmp 内に Compact Flash 向けシステムのディレクトリツリーを作成しておいた場合の例です。

⑥ Linux カーネルのコピー
Armadillo-9 を Compact Flash から起動する場合、カーネルは Compact Flash 内の/boot ディレクトリ内に非圧縮イメージ「Image」、または gz 圧縮イメージ「Image.gz」として保存しておく必要があります。任意のカーネルイメージをこのファイル名になるようコピーしてください。

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

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

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

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

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

## 11.2. Compact Flash への Debian/GNU Linux インストール

Compact Flash に Debian/GNU Linux システムをインストールします。Debian イメージは、debian/arm-sarge1.tar.gz～arm-sarge4.tar.gz として分割して用意されています。

Armadillo-9 で起動可能な Compact Flash の作成方法は「11.1Armadillo-9 で起動可能な Compact Flash の作成」と同様です。この中の「⑤Compact Flash 上へのシステム構築」の部分で、用意された Debian/GNU Linux を展開する手順を説明します。

**TIPS**

Debian/GNU Linux のイメージを使用するには、Compact Flash のインストールパーティションに 300MB 以上の空き容量が必要です。

⑤-a RAM ファイルシステムのマウント

/home/ftp/pub に RAM ファイルシステムをマウントし、書き込み権限を与えます。

```
[armadillo9 ~]# mount -t ramfs ramfs /home/ftp/pub
[armadillo9 ~]# chmod  777 /home/ftp/pub
```

⑤-b ftp によるファイル転送

PC から、arm-sarge1.tar.gz を ftp で転送します。

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

⑤-c Compact Flash への展開

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

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

⑤-d 全分割ファイルの転送・展開

残りの arm-sarge2.tar.gz～arm-sarge4.tar.gz についても、⑤-b～⑤-c を繰り返します。

以上ですべての Debian/GNU Linux のファイルが展開されたます。Linux カーネルについては Debian イメージに含まれていませんので、「11.1Armadillo-9 で起動可能な Compact Flash の作成」を参照して個別にコピーするのを忘れないようにしてください。

# 12. PCMCIA-CS 対応版ユーザランド

## 12.1. カーネルとユーザランドイメージ

PCMCIA-CS は、Compact Flash の活線挿抜や、ストレージ以外の Compact Flash カード(I/O デバイスカード)のサポートを可能にするカードサービスパッケージです。Armadillo-9 では、PCMCIA-CS を含んだユーザランドと、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 対応ユーザランド
・linux-pcmcia.bin.gz　PCMCIA-CS 対応ユーザランドと同時に使用可能な Linux カーネル

**注意**

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

それぞれオンボード Flash メモリに書き込んで使用してください。Flash メモリへのイメージの書き換え方は、「3.Flash メモリの書き換え方法」を参照してください。

## 12.2. PCMCIA-CS の有効化

PCMCIA-CS を有効化するには、以下のコマンドを入力してください。

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

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

## 12.3. PCMCIA-CS 対応版にのみ含まれるその他のパッケージ

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

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

詳しくは、各パッケージのソース付属ドキュメントなどを参照してください。

# 13. Appendix

## 13.1. 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 AT-9750LX 用の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。

### 13.1.1. coLinux のインストール

1) 付属 CD の colinux ディレクトリにある **coLinux-0.6.1.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。

### 13.1.2. 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**)に解凍します

- **root_fs.lzh** (ルートファイルシステム)
- **swap_device_256M.lzh** (swap ファイルシステム)
- **home_fs_2G.lzh** (**/home** にマウントされるファイルシステム)
- **default.colinux.xml.lzh** (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解凍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ますので、必要と思われるサイズのファイルを解凍してください。

解凍ソフトによっては解凍に失敗する場合があります。「LHA ユーティリティ 32 Ver1.46(http://www.lhut32.com/index.shtml)」にて正常に解凍できることを確認してあります。

### 13.1.3. 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:\colinx**)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

### 13.1.4. ネットワークの設定

coLinux は Windows と別の 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 13-1 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下さい。
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「13.1.8.特殊な場合の Windows ネットワーク設定方法」を参照して下さい。

### 13.1.5. 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下さい。

**例 13-2 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:
```

### 13.1.6. 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下さい。

**例 13-3 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下さい。
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。

### 13.1.7. クロス開発環境の導入

「5.開発環境の準備」を参照して、クロス開発環境を coLinux 上に導入して下さい。
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 から AT-9750LX の開発を行うことができます。以降の説明は特殊なケースです。

### 13.1.8. 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

### 13.1.9. 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 13-4 ifconfig コマンドの実行**

```
colinux:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:43:4F:4E:45:30
          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」場合「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 13-1 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 13-5 /etc/network/interfaces ファイル編集例**

```
auto lo eth0
iface lo inetloopback
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 13-6 /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 13-7 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

| Version | 年月日 | 改訂内容 |
|---|---|---|
| 1.0.0 | 2004.12.18 | ・初版発行 |
| 1.0.1 | 2004.12.28 | ・uClinux-dist を使用した作業について、一般ユーザで行なうよう追記<br>・「注意」や「Tips」について、アイコン表記に統一 |

Armadillo-9 Software Manual　2004年12月28日　version 1.0.1

**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　TEL011-207-6550　FAX011-207-6570