Armadillo-500
Development Board

ソフトウェアマニュアル

Version 1.0.1

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

本書は Armadillo-500 開発ボード（以降、開発ボードと表記）をカスタマイズするための手順書となります。出荷状態のソフトウェアの仕様に関しては「Armadillo-500 Development Board Startup Guide」を参照してください。また、atmark-dist の詳細については、「atmark-dist Developers Guide」を参照してください。

1.1. 対象となる読者

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

上記以外の方でも、本書を有効に利用していただけたら幸いです。

1.2. 本書の構成

本書では、開発ボードのソフトウェアをカスタマイズする上で必要となる情報について記載されています。

- 開発環境の構築方法
- フラッシュメモリの書き換え方法
- ビルド方法

1.3. 表記について

1.3.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1-1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列
text	編集する文字列や出力される文字列。またはコメント

1.3.2. コマンド入力例

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

表 1-2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC 上の特権ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo500 /]#	開発ボード上の特権ユーザで実行
[armadillo500 /]$	開発ボード上の一般ユーザで実行
hermit>	開発ボード上の保守モードで実行

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

表 1-3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.3.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

2.作業の前に

2.1. 見取り図

開発ボードの見取り図です。各インターフェースの配置場所等を確認してください。

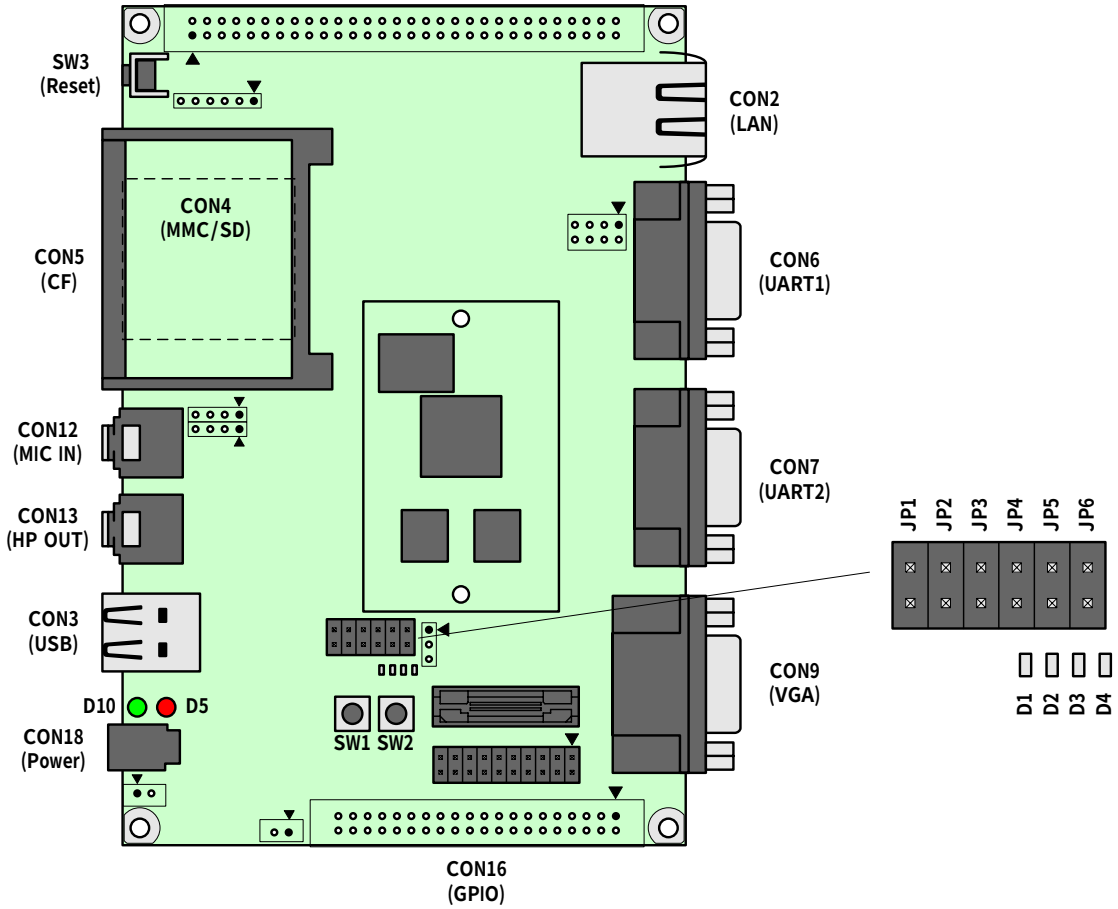

図 2-1 見取り図

2.2. 準備するもの

開発ボードを使用する前に、次のものを準備してください。

- 作業用 PC とシリアルクロスケーブル
  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 と D-Sub9 ピン（メス－メス）のクロス接続用ケーブルです。作業用 PC には debian 系 Linux OS が動作する環境が必要です。
- シリアル通信ソフトウェア [1]
  開発ボードを制御するために使用します。作業用 PC にインストールしてください。(Linux 用のソフトウェアは、付属 CD の tool ディレクトリに収録されています。)

[1] Linux では「minicom」、Windows では「Tera Term Pro」などです。

2.3. ジャンパピンについて

開発ボードのジャンパピンは、表 2-1のように割り当てられています。

表 2-1 ジャンパピンの割り当て

ジャンパ	割り当て	デフォルトソフトウェアでの使用状況
JP1	ユーザ設定	ブートローダのモード設定に使用
JP2	ユーザ設定	未使用
JP3	CPU 起動モード設定	—
JP4	CPU 起動モード設定	—
JP5	CPU 起動モード設定	—
JP6	CPU 起動モード設定	—

2.3.1. CPU 起動モード設定

JP3-6 の設定により、CPU 起動モードを切り替えることができます。起動モードには、フラッシュメモリブートモードと UART ブートモードがあります。

表 2-2 CPU 起動モード

JP3	JP4	JP5	JP6	モード
オープン	オープン	オープン	オープン	オンボードフラッシュメモリブート
ショート	ショート	オープン	ショート	UART ブート

2.3.2. オンボードフラッシュメモリブートモード

このモードでは、リセット時にオンボードフラッシュメモリ内のブートローダが最初に起動します。ブートローダは起動後JP1 の設定によって、表 2-3に示すモードへ移行します。

表 2-3 フラッシュメモリブートモード

JP1	モード	説明
オープン	オートブート	電源投入後、カーネルを自動的に起動します。
ショート	保守	起動後、保守モードプロンプトが表示されます。

2.3.3. UART ブートモード

このモードでは、CPUのInternal ROM機能のUARTブートが使用できます。UARTブート機能は、フラッシュメモリのブートローダが壊れた場合など、システム復旧のために使用します。詳しくは、「4.6.ブートローダを出荷状態に戻す」を参照してください。

2.4. シリアル通信ソフトウェアの設定

シリアル通信ソフトウェアを起動し、シリアルの通信設定を、表 2-4のように設定してください。

表 2-4 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

2.5. メモリマップ

デフォルトのフラッシュメモリのメモリマップを、表 2-5に示します。

表 2-5 メモリマップ(フラッシュメモリ)

物理 アドレス	リージョン名	サイズ	説明
0xa0000000 ｜ 0xa0ffffff	all	16MB	フラッシュメモリ全領域
0xa0000000 ｜ 0xa001ffff	bootloader	128KB	ブートローダ領域 「loader-a5x0.bin」のイメージ
0xa0020000 ｜ 0xa021ffff	kernel	2MB	カーネル領域 「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)
0xa0220000 ｜ 0xa0fdffff	userland	13.75MB	ユーザランド領域 「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)
0xa0fe0000 ｜ 0xa0ffffff	config	128KB	コンフィグ領域 flatfsd が使用する領域

3.開発環境の準備

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

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

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

- ATDE Install Guide
- coLinux Guide

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

3.1. クロス開発環境パッケージのインストール

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

表 3-1 開発環境一覧

クロスターゲット	説明
arm	通常の ARM クロス開発環境です。

```
[PC ~]# dpkg -i *.deb
```

図 3-1 インストールコマンド

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

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

[2] Armadillo-500 開発ボードの開発環境としては、ATDE v2.0 以降を推奨しています。

3.2. atmark-dist のビルドに必要なパッケージ

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

表 3-2 atmark-dist のビルドに必要なパッケージ一覧

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

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

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

図 3-2 インストール情報表示コマンド

3.3. クロス開発用ライブラリパッケージの作成方法

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

まず、作成したいクロス開発用パッケージの元となるライブラリパッケージを取得します。元となるパッケージは、ARM 用のパッケージです。例えば、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_[version]_arm.deb
[PC ~]$ ls
libncurses5-arm-cross_[version]_all.deb libncurses5_[version]_arm.deb
```

図 3-3 クロス開発用ライブラリパッケージの作成

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

4.フラッシュメモリの書き換え

開発ボードのフラッシュメモリを書き換えることで、ソフトウェアの機能を変更することができます。

何らかの原因により「フラッシュメモリの書き換え」に失敗した場合、ソフトウェアが正常に起動しなくなる場合があります。書き換え中は以下の点に注意してください。

- 開発ボードの電源を切断しない
- 開発ボードと作業用 PC を接続しているシリアルケーブルを外さない

4.1. ダウンローダのインストール

作業用 PC にダウンローダをインストールします。これらは、開発ボードのフラッシュメモリの書き換えに使用します。

ダウンローダの種類には、表 4-1のようなものがあります。

表 4-1 ダウンローダー覧

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

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

4.1.1. 作業用 PC が Linux の場合

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

```
[PC ~]# dpkg -i hermit-at_[version]_i386.deb
[PC ~]# dpkg -i shoehorn-at_[version]_i386.deb
#には、バージョン番号が入ります。最新のものをインストールしてください。
```

図 4-1 ダウンローダのインストール（Linux）

4.1.2. 作業用 PC が Windows の場合

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

4.2. フラッシュメモリの書き込み領域について

フラッシュメモリの書き込み先頭アドレスは、領域（リージョン）名で指定することができます。書き込み領域には 5 種類あり、それぞれに書き込むイメージファイルは、表 4-2のようになります。

表 4-2 リージョン名と対応するイメージファイル

領域名	ファイル名
all	※提供していません。
bootloader	loader-armadillo5x0.bin
kernel	linux.bin.gz
userland	romfs.img.gz
config	※提供していません。

4.3. Hermit-At Host を使用してフラッシュメモリを書き換える

ここでは、Hermit-At Hostを使用してフラッシュメモリを書き換える手順について説明します。「4.1.ダウンローダのインストール」でインストールしたHermit-At Hostを使用します。これは、開発ボードのブートローダと協調動作を行い、作業用PCから開発ボードのフラッシュメモリを書き換えることができます。

4.3.1. 準備

開発ボードの起動モードを、保守モードに変更します。JP1 をショートして再起動してください。
開発ボードのCON6[1]と接続されている作業用PCのシリアルポートが他のアプリケーションで使用されていないことを確認します。使用されている場合は、該当アプリケーションを終了するなどしてシリアルポートを開放してください。

4.3.2. 作業用 PC が Linux の場合

図 4-2のようにコマンドを実行します。

```
[PC ~]$ hermit download -i linux.bin.gz    -r kernel
                          ファイル名        リージョン指定
```

図 4-2 ダウンロードコマンド

シリアルポートがttyS0 以外の場合は、図 4-3のようにポートを指定してください。

```
[PC ~]$ hermit download -i linux.bin.gz -r kernel    --port ttyS1
                                                     シリアルポート指定
```

図 4-3 ダウンロードコマンド（ポート指定）

[1] デフォルトの場合。設定によってはCON7 の場合もあります。

bootloaderリージョンは、簡易プロテクトされているため、書き換える場合は、図 4-4のようにプロテクトの解除を指定します。

```
[PC ~]$ hermit download -i loader-armadillo5x0.bin -r bootloader --force-locked
                                                                  アンプロテクト
```

図 4-4 ダウンロードコマンド（アンプロテクト）

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、オンボードフラッシュメモリからの起動ができなくなります。この場合は「4.6.ブートローダを出荷状態に戻す」を参照してブートローダを復旧してください。

4.3.3. 作業用 PC が Windows の場合

hermit-at-win.exeを実行します。図 4-5の画面が表示されます。

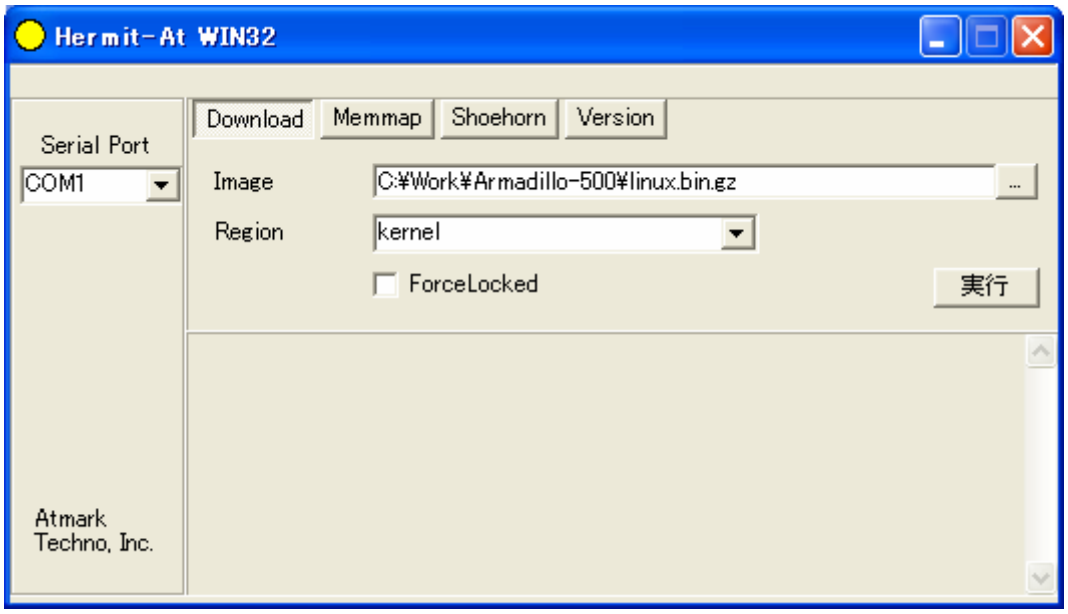

図 4-5 Hermit-At：Download ウィンドウ

開発ボードと接続されているシリアルポートを「Serial Port」に指定してください。ドロップダウンリストに表示されない場合は、直接ポートを入力してください。

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

すべて設定してから実行ボタンをクリックします。図 4-6の画面が表示されます。

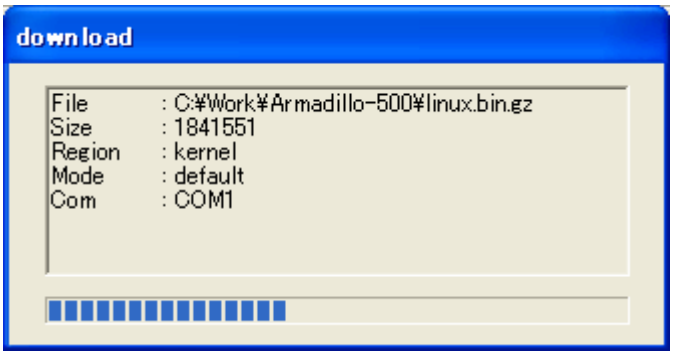

図 4-6 Hermit-At：download ダイアログ

ダウンロードの設定と進捗状況が表示されます。ダウンロードが完了すると自動的にクローズされます。

4.4. tftpdl を使用してフラッシュメモリを書き換える

開発ボードのブートローダ機能の tftpdl を使用してフラッシュメモリを書き換えることができます。この機能は、所属するネットワークにある TFTP サーバーが公開しているファイルをダウンロードしてフラッシュメモリを書き換えることができます。

開発ボードの起動モードを保守モードに変更します。JP1 をショートに設定して再起動してください。
作業用PCのシリアル通信ソフトウェアを使用して、コマンドを入力します。図 4-7のようにコマンドを入力してください。

```
hermit> tftpdl 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz
               ①              ②             ③
① Armadillo-500 が使用する IP アドレス
② TFTP サーバーの IP アドレス
③ リージョンとそこに書き込むサーバー上のファイルパス
```

図 4-7 tftpdl コマンド例

実行すると、図 4-8のようにログが出力されます。「**completed!!**」と表示されたら書き換えが終了します。

```
hermit> tftpdl 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz

Client: 192.168.10.10
Server: 192.168.10.1
Region(kernel): linux.bin.gz

initializing net-device...OK
Filename : linux.bin.gz
................................................................................
................................................................................
................................
Filesize : 1841551

programing: kernel
###############

completed!!
```

図 4-8 tftpdl ログ

4.5. netflash を使用してフラッシュメモリを書き換える

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

開発ボードにログインし、図 4-9のようにコマンドを実行します。

```
[armadillo500 ~]# netflash -k -n -u -r /dev/flash/kernel [URL]
                           ①          ②                 ③
① オプションを指定します。詳しくは netflash -h で確認できます。
② リージョンを指定します。
③ 書き込むファイルの URL を指定します。
```

図 4-9 netflash コマンド例

4.6. ブートローダを出荷状態に戻す

CPU の Internal ROM 機能の UART ブートモードを使用して、ブートローダを出荷状態に戻すことができます。

4.6.1. 準備

開発ボードのジャンパを、表 4-3のように設定してください。

表 4-3 UART ブートモードジャンパー設定

ジャンパ	設定
JP3	ショート
JP4	ショート
JP5	オープン
JP6	ショート

開発ボードの CON6 と接続されている作業用 PC のシリアルポートが他のアプリケーションで使用されていないのを確認します。使用されている場合は、シリアルポートを開放してください。

4.6.2. 作業用 PC が Linux の場合

図 4-10のようにコマンド[1]を実行してから、開発ボードを再起動してください。

```
[PC ~]$ shoehorn --boot --target armadillo5x0
                    --initrd /dev/null
                    --kernel /usr/lib/hermit/loader-armadillo5x0-boot.bin
                    --loader /usr/lib/shoehorn/shoehorn-armadillo5x0.bin
                    --initfile /usr/lib/shoehorn/shoehorn-armadillo5x0.init
                    --postfile /usr/lib/shoehorn/shoehorn-armadillo5x0.post
```

図 4-10 shoehorn コマンド例

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

実行すると、図 4-11のようにログが表示されます。

```
/usr/lib/shoehorn/shoehorn-armadillo5x0.bin: 1996 bytes (2048 bytes buffer)
/usr/lib/hermit/loader-armadillo5x0-boot.bin: 39772 bytes (39772 bytes buffer)
/dev/null: 0 bytes (0 bytes buffer)
Waiting for target - press Wakeup now.
Initializing target...
Writing SRAM loader...
Pinging loader
Initialising hardware:
- flushing cache/TLB
- Switching to 115200 baud
- Setting up DDR
Pinging loader
Detecting DRAM
- 32 bits wide
- start: 0x80000000 size: 0x04000000 last: 0x83ffffff
Total DRAM: 65536kB
Loading /usr/lib/hermit/loader-armadillo5x0-boot.bin:
- start: 0x83000000 size: 0x00009b5c last: 0x83009b5b
initrd_start is c0400000
Moving initrd_start to c0400000
Loading /dev/null:
- start: 0xc0400000 size: 0x00000000
Writing parameter area
- nr_pages (all banks): 4096
- rootdev: (RAMDISK_MAJOR, 0)
- pages_in_bank[0]: 2048
- pages_in_bank[1]: 2048
- initrd_start: 0xc0400000
- initrd_size: 0x0
- ramdisk_size: 0x0
- start: 0x80020000 size: 0x00000900 last: 0x800208ff
Pinging loader
Starting kernel at 0x83000000
```

図 4-11 shoehorn ログ

この状態で、「4.3.Hermit-At Hostを使用してフラッシュメモリを書き換える」を参照してブートローダの書き込みを行ってください。

4.6.3. 作業用 PC が Windows の場合

hermit-at-win.exeを実行しShoehornボタンをクリックすると、図 4-12のような画面が表示されます。

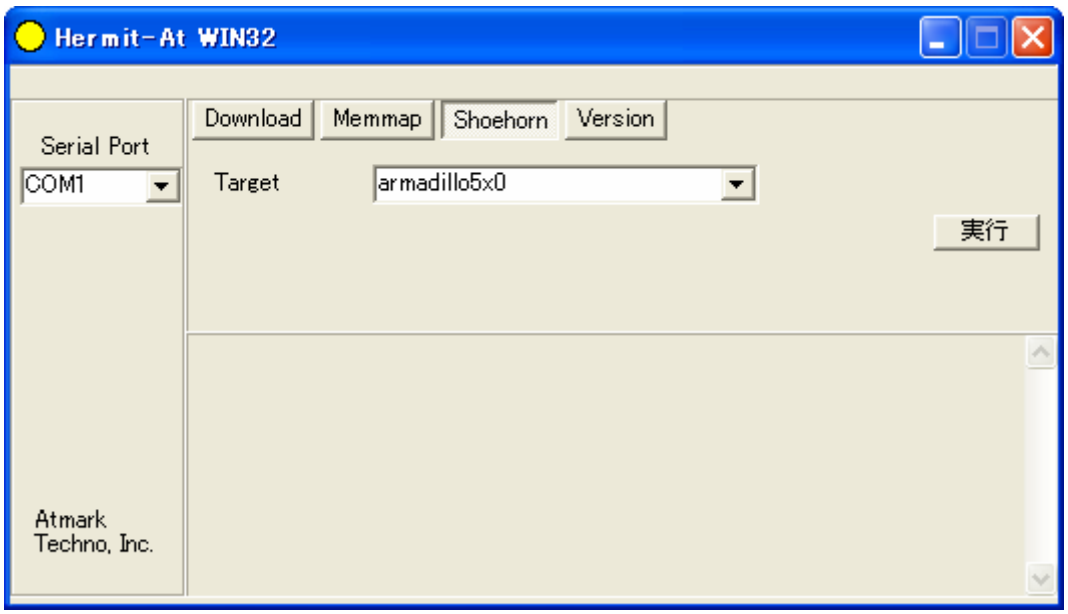

図 4-12 Hermit-At：Shoehorn ウィンドウ

Target に armadillo5x0 を選択して実行ボタンをクリックします。

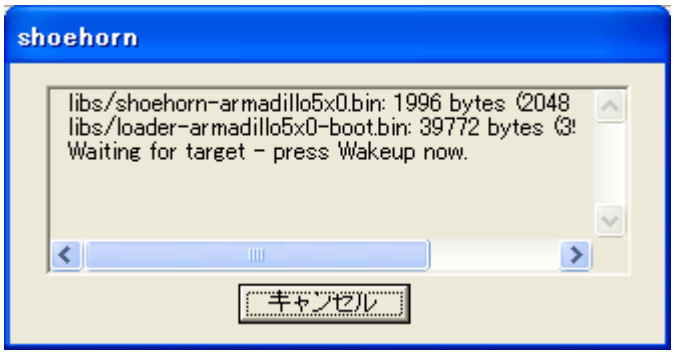

図 4-13 Hermit-At：shoehorn ダイアログ

ダイアログが表示されます。開発ボードを再起動してください。ダウンロードするための準備が完了すると自動的にクローズされます。

この状態で、「4.3.Hermit-At Hostを使用してフラッシュメモリを書き換える」を参照してブートローダの書き込みを行ってください。

5.ビルド

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

開発作業では、基本ライブラリ・アプリケーションやシステム設定ファイルの作成・配置を行ないます。各ファイルは作業ディレクトリ配下で作成・配置作業を行ないますが、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。

5.1. カーネルイメージとユーザーランドイメージのビルド

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

5.1.1. ソースコードの準備

付属CDのsource/distにあるatmark-dist.tar.gzとsource/kernelにあるlinux.tar.gzを作業ディレクトリに展開します。展開後、atmark-distにカーネルソースを登録します。図 5-1のように作業してください。

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

図 5-1 ソースコード準備

5.1.2. コンフィグレーション

開発ボード用にビルドシステムをコンフィグレーションします[1]。図 5-2のようにコマンドを実行してください。

```
[PC ~]$ cd atamrk-dist
[PC ~/atmark-dist]$ make config
config/mkconfig > config.in
*
* 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] AtmarkTechno
*
* Select the Product you wish to target
*
AtmarkTechno Products (Armadillo, Armadillo-210.Base, Armadillo-210.Recover,
Armadillo-220.Base, Armadillo-220.Recover, Armadillo-230.Base,
Armadillo-230.Recover, Armadillo-240.Base, Armadillo-240.Recover,
Armadillo-300, Armadillo-500, Armadillo-9, Armadillo-9.PCMCIA,
Armadillo-J.Base, Armadillo-J.Jffs2, Armadillo-J.Recover, SUZAKU,
SUZAKU-UQ-XUP) [Armadillo] Armadillo-500
*
* Kernel/Library/Defaults Selection
*
*
* Kernel is linux-2.6.x
*
Cross-dev (default, arm, arm-vfp) [default] default
Libc Version (None, glibc, uC-libc, uClibc) [uClibc] None
Default all settings (lose changes) (CONFIG_DEFAULTS_OVERRIDE) [N/y/?] y
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
       :
       :
[PC ~/atmark-dist]$
```

図 5-2 コンフィグレーション

[1] より詳しく知りたい場合は、「atmark-dist Developers Guide」を参照してください。

5.1.3. ビルド

ビルドするには、atmark-distディレクトリで 図 5-3のようにコマンドを実行します。ビルドが完了すると、atmark-dist/imagesディレクトリにlinux.bin.gzとromfs.img.gzが作成されます。

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

図 5-3 ビルド

5.2. ユーザーランドイメージをカスタマイズする

自作のアプリケーションを/binに追加したユーザーランドイメージの作成方法について説明します。ここでは、「5.1.カーネルイメージとユーザーランドイメージのビルド」が完了している前提で説明します。
自作アプリケーションは、~/sample/hello にある仮定とします。

```
[PC ~/atmark-dist]$ cp ../sample/hello romfs/bin/
[PC ~/atmark-dist]$ make image
       :
       :
[PC ~/atmark-dist]$ ls images
linux.bin  linux.bin.gz  romfs.img  romfs.img.gz
```

図 5-4 ユーザーランドイメージのカスタマイズ

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

5.3. ブートローダーイメージのビルド

5.3.1. ソースコードの準備

付属CDのsource/bootloaderにあるhermit-at-x.x.x-source.tar.gzを作業ディレクトリに展開します。します。図 5-5のように作業してください。

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

図 5-5ソースコード展開例

5.3.2. ビルド

ビルドオプションにTARGET=armadillo5x0 を指定してビルドします。図 5-6のように実行してください。

```
[PC ~]$ cd hermit-at-[version]
[PC ~/hermit-at]$ make TARGET=armadillo5x0
       :
       :
[PC ~/hermit-at]$ ls src/target/armadillo5x0/*.bin
loader-armadillo5x0.bin
```

図 5-6 ビルド

6.コンパクトフラッシュシステム構築

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

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

この章では、表 6-1のようなコンパクトフラッシュシステムを例に、構築手順を説明します。

表 6-1 コンパクトフラッシュシステム例

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

6.1. コンパクトフラッシュの初期化

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

6.1.1. ディスクフォーマット

図 6-1のように、ディスクをフォーマットします。

```
[armadillo500 ~]# fdisk /dev/hda
The number of cylinders for this disk is set to 1324.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

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

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

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

Command (m for help): p

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

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

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

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

図 6-1 ディスク初期化方法

6.1.2. ファイルシステムの作成

図 6-2のように初期化したディスクのパーティションにファイルシステムを作成します。

mke2fs で起動パーティション（カーネルイメージを配置するパーティション）に EXT2 ファイルシステムを作成する場合は、必ず「-0 none」オプションを指定する必要があります。

```
[armadillo500 ~]# mke2fs -O none /dev/hda1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
7936 inodes, 31720 blocks
1586 blocks (5%) reserved for the super user
First data block=1
4 block groups
8192 blocks per group, 8192 fragments per group
1984 inodes per group
Superblock backups stored on blocks:
       8193, 16385, 24577

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

This filesystem will be automatically checked every 36 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
[armadillo500 ~]# mke2fs -j /dev/hda2
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
117392 inodes, 468720 blocks
23436 blocks (5%) reserved for the super user
First data block=1
58 block groups
8192 blocks per group, 8192 fragments per group
2024 inodes per group
Superblock backups stored on blocks:
       8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

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

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

図 6-2 ファイルシステムの構築

6.2. カーネルイメージを配置する

コンパクトフラッシュシステムから起動する場合は、起動パーティション（/dev/hda1）の/bootディレクトリにカーネルイメージを配置する必要があります。対応しているカーネルイメージは、非圧縮カーネルイメージ（Image、linux.bin）または、圧縮イメージ（Image,gz、linux.bin.gz）のどちらかになります。図 6-3に配置例を示します。

```
[armadillo500 ~]# mount /dev/hda1 /mnt
[armadillo500 ~]# mkdir /mnt/boot
[armadillo500 ~]# wget http://download.atmark-techno.com/armadillo-500/
                                        images/linux-a500-[version].bin.gz
Connecting to download.atmark-techno.com [210.191.215.172]:80
linux-a500-[version].bin.gz 100% |**********************|  **** KB    00:00 ETA
[armadillo500 ~]# mv linux-a500-[version].bin.gz /mnt/boot/Image.gz
[armadillo500 ~]# sync
[armadillo500 ~]# umount /mnt
```

図 6-3 カーネルイメージの配置

6.3. ルートファイルシステムの構築

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

6.3.1. Debian GNU/Linux を構築する

Debianを構築する場合、付属CDのdebianディレクトリ以下のアーカイブを使用します。これは、純粋なDebianでインストールされるファイルを分割してアーカイブ化したものとなります。これらをファイルシステム上に展開することでルートファイルシステムを構築することができます。図 6-4に展開例を示します。

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

```
[armadillo500 ~]# mount /dev/hda2 /mnt
[armadillo500 ~]# mount -t ramfs ramfs /tmp
[armadillo500 ~]# cd /tmp

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

[armadillo500 /tmp]# wget http://download.atmark-techno.com/armadillo-500/
                                              debian/debian-etch-arm#.tgz
Connecting to download.atmark-techno.com [210.191.215.172]:80
debian-etch-#.tgz    100% |************************|  **** KB    00:00 ETA
[armadillo500 /tmp]# gzip -cd debian-etch-arm#.tgz | (cd /mnt; tar xf -)
[armadillo500 /tmp]# sync
[armadillo500 /tmp]# rm -f debian-etch-arm#.tgz

[LOOP] に戻る

[armadillo500 /tmp]# umount /mnt
```

図 6-4 Debian アーカイブの展開例

6.3.2. atmark-dist イメージから構築する

atmark-distで作成されるシステムイメージをコンパクトフラッシュのルートファイルシステムとして構築する方法を説明します。Debianを構築する場合に比べ、ディスク容量の少ないコンパクトフラッシュへシステムを構築することができます。ここでは、ユーザーランドイメージ（romfs.img.gz）から構築する手順を 図 6-5で示します。

```
[armadillo500 ~]# mount -t ramfs ramfs /tmp
[armadillo500 ~]# cd /tmp
[armadillo500 /tmp]# wget http://download.atmark-techno.com/armadillo-500/
                                        images/romfs-a500-[version].img.gz
Connecting to download.atmark-techno.com [210.191.215.172]:80
romfs-a500-1.00.img.gz 100% |**********************|  **** KB    00:00 ETA
[armadillo500 /tmp]# gzip -dc romfs-a500-[version].img.gz > romfs.img
[armadillo500 /tmp]# mount /dev/hda2 /mnt
[armadillo500 /tmp]# mkdir romfs
[armadillo500 /tmp]# mount -o loop romfs.img romfs
[armadillo500 /tmp]# (cd romfs/; tar cf - *) | (cd /mnt; tar xf -)
[armadillo500 /tmp]# sync
[armadillo500 /tmp]# umount romfs
[armadillo500 /tmp]# umount /mnt
```

図 6-5 romfs.img.gz からの作成例

6.4. コンパクトフラッシュシステムの起動

ジャンパにより起動モードを保守モードに設定し、再起動してください。
保守モードで立ち上げ、コンパクトフラッシュのカーネルイメージで起動するためには、図 6-6を実行してください。ルートファイルシステムの設定については、図 6-7を実行してください。

```
hermit> setbootdevice hda1
```

図 6-6 起動デバイスの指定

```
hermit> setenv console=ttymxc0 root=/dev/hda2 rootdelay=3 noinitrd
```

図 6-7 ルートファイルシステム指定例

6.5. 各種システム設定例

新しくシステムを構築した場合、システム起動時に WARNING が表示される場合があります。それらの WARNING を解決する方法を説明します。

6.5.1. Debian システム

6.5.1.1. modules ディレクトリの更新

- WARNING

```
modprobe: FATAL: Could not load /lib/modules/2.6.18/modules.dep:
                                                No such file or directory
```

図 6-8 WARNING：modules.dep

- 解決方法

システムにログインし、図 6-9のようにコマンドを実行します。

```
[debian ~]# mkdir -p /lib/modules/`uname -r`
[debian ~]# depmod
```

図 6-9 解決方法：modules.dep

6.5.2. atmark-dist システム

6.5.2.1. fstab の更新

- WARNING

```
fsck.ext2: Bad magic number in super-block while trying to open /dev/ram0
(null):
The superblock could not be read or does not describe a correct ext2
filesystem.  If the device is valid and it really contains an ext2
filesystem (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
    e2fsck -b 8193 <device>

WARNING: Error while checking root filesystem.
You can login as root now, the system will reboot after logout.

Give root password for system maintenance
(or type Control-D for normal startup):
```

図 6-10 WARNING：fstab

- 解決方法

システムにログインし、/etc/fstabを 図 6-11のように変更します。

```
[debian ~]# vi /etc/fstab

/dev/hda2            /              ext3   defaults      0      1
proc                 /proc          proc   defaults      0      0
usbfs                /proc/bus/usb  usbfs  defaults      0      0
sysfs                /sys           sysfs  defaults      0      0
```

図 6-11 解決方法：fstab

7.JTAG

この章では、JTAG デバッガを使用する際の注意点などを説明します。

7.1. ターゲットボードの初期化について

ETM を接続してデバッグする場合は、ETM で使用するポートをコンフィグレーションしなければなりません。「Armadillo-500 Development Board Hardware Manual」を参照して適切に設定してください。

7.2. Linux をデバッグする場合

JTAG を使用して Linux をデバッグする場合は、Linux 起動オプションを適切に設定しなければなりません。図 7-1のように設定します。

```
hermit> setenv    jtag=on
                  JTAG モード指定
```

図 7-1 JTAG モード指定

表 7-1 JTAG モード

JTAG モード	説明
on	JTAG でデバッグ可能にします。
etm8	ETM の TRACE 機能（8bit）を有効にします。 GPIO ポートはアクセスできなくなります。
etm16	ETM の TRACE 機能（16bit）を有効にします。 GPIO ポートと USB Host2 はアクセスできなくなります。

7.2.1. 設定例

```
hermit> setenv console=ttymxc0 jtag=on
```

図 7-2 JTAG モード指定例

Appendix A. Hermit-At について

Hermit-At とは、Atmark Techno 製品のブートローダに採用している高機能ダウンローダ/ブートローダです。フラッシュメモリの書き換えや、Linux カーネル起動オプションの設定等、様々な機能があります。ここでは、代表的な機能について説明します。

A.1. setenv と clearenv

Linux カーネル起動オプションを設定するコマンドです。setenv で設定されたパラメータは、Linux カーネル起動時に渡されます。clearenv を実行すると、設定がクリアされます。このパラメータは、フラッシュメモリに保存され再起動後も設定は有効となります。

```
構文：setenv [起動オプション]...
構文：clearenv
```

図 A-1 構文：setenv、clearenv

A.1.1. setenv/clearenv 使用例

```
hermit> setenv console=ttymxc0
hermit> setenv
1: console=ttymxc0

hermit> clearenv
```

図 A-2 使用例：setenv、clearenv

A.1.2. Linux 起動オプション

表 A-1 よく使用される Linux 起動オプション

オプション	説明
console	シリアルコンソールが使用するデバイスを指示します。
root	ルートファイルシステム関連の設定を指示します。
rootdelay	ルートファイルシステムをマウントする前に指定秒間待機します。
noinitrd	カーネルが起動した後に initrd データがどうなるのかを指示します。
nfsroot	NFS を使用する場合に、ルートファイルシステムの場所や NFS オプションを指示します。

A.2. frob

指定したアドレスのデータを読み込む、または、変更することができるモードに移行するコマンドです。

表 A-2 2nd ブートローダイメージの種類

frob コマンド	説明
peek [addr]	指定されたアドレスから 32bit のデータを読み出します。
peek8 [addr]	指定されたアドレスから 8bit のデータを読み出します。
peek16 [addr]	指定されたアドレスから 16bit のデータを読み出します。
poke [addr] [value]	指定されたアドレスに 32bit のデータを書き込みます。
poke8 [addr] [value]	指定されたアドレスに 8bit のデータを書き込みます。
poke16 [addr] [value]	指定されたアドレスに 16bit のデータを書き込みます。

A.3. memmap

フラッシュメモリのリージョン情報を表示するコマンドです。

構文：memmap

図 A-3 構文：memmap

A.3.1. 使用例

```
hermit> memmap
0xa0000000:0xa0ffffff FLA all bf:8K bl:4x32K/l,127x128K/l
0xa0000000:0xa001ffff FLA bootloader bf:8K bl:4x32K/l
0xa0020000:0xa021ffff FLA kernel bf:8K bl:16x128K
0xa0220000:0xa0fdffff FLA userland bf:8K bl:110x128K
0xa0fe0000:0xa0ffffff FLA config bf:8K bl:1x128K
0x80000000:0x83ffffff RAM dram-1
```

図 A-4 使用例：memmap

A.4. erase

フラッシュメモリの消去を行うコマンドです。

構文：erase [アドレス]

図 A-5 構文：erase

A.4.1. 使用例

コンフィグ領域を消去する場合は、図 A-6のようになります。

```
hermit> erase 0xa0fe0000
```

図 A-6 使用例：erase

A.5. tftpdl

TFTP プロトコルを使用して TFTP サーバーからファイルをダウンロードし、フラッシュメモリの書き換えを行うコマンドです。

```
構文：tftpdl [クライアント IP アドレス] [サーバーIP アドレス] [オプション]...
```

図 A-7 構文：tftpdl

表 A-3 tftpdl オプション

オプション	説明
--region=filepath	region に書き込むファイルを filepath で指定します。
--fake	実際にフラッシュメモリの書き込みを行わないモードになります。

A.5.1. 使用例

```
hermit> tftpdl 192.168.10.10 192.168.10.1 --kernel=linux.bin.gz

Client: 192.168.10.10
Server: 192.168.10.1
Region(kernel): linux.bin.gz

initializing net-device...OK
Filename : linux.bin.gz
................................................................................
................................................................................
................................
Filesize : 1841551

programing: kernel
###############

completed!!
```

図 A-8 使用例：tftpdl

改訂履歴

Version	年月日	改訂内容
1.0.0	2007.7.27	・初版発行
1.0.1	2007.9.14	・「表 1-3 コマンド入力例での省略表記」を追加 ・コマンド入力例で、バージョン番号などの省略の表記方法を修正 ・「表 3-2 atmark-distのビルドに必要なパッケージ一覧」にlibncurses5-devを追加 ・「図 6-4 Debianアーカイブの展開例」のアーカイブファイル名を変更

■ 困ったときは...

Armadillo 開発者サイトでは、本書に記載されていない情報や最新の技術情報や FAQ などが随時更新されています。困っときは、まず Armadillo 開発者サイトにアクセスしてみましょう！

Armadillo開発者サイト

http://armadillo.atmark-techno.com

■ 開発に関する質問は...

こんなことがしたいけど、実績が知りたい。などの技術的な質問がある方は、Armadillo メーリングリストを利用したらどうでしょうか？Armadillo メーリングリストには、多くの方々が購読されています。もしかしたら会員メンバーの中に、解決までの道標を提供してくれるかもしれません。

Armadilloメーリングリスト

http://armadillo.atmark-techno.com/mailinglists

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