Armadillo-500
開発ボード
ソフトウェアマニュアル

A5501/A5527

Version 1.1.1-8d87fa8
2009/07/17

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

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

Armadillo-500
開発ボード
ソフトウェアマニュアル

株式会社アットマークテクノ

060-0035 札幌市中央区北 5 条東 2 丁目 AFT ビル 6F
TEL 011-207-6550 FAX 011-207-6570

Version 1.1.1-8d87fa8
2009/07/17

目次

図目次

表目次

1.はじめに

このたびは Armadillo-500 開発セットをお求めいただき、ありがとうございます。

Armadillo-500 は、CPU Core に ARM1136JF-S を搭載した超小型・高性能 CPU モジュールです。情報表示機器やマルチメディア機器などのメインプロセッサとしてご利用頂くことが可能です。

Armadillo-500 開発ボード（以降、開発ボードと表記）は、Armadillo-500 と Armadillo-500 に搭載された機能を効率的に使用することができるように各種コントローラ及び、コネクタを実装した開発用ベースボード（以降、ベースボードと表記）の構成となります。

本書は Armadillo-500 開発ボード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 開発ボード スタートアップガイド」を参照してください。また、atmark-dist の詳細については、「atmark-dist 開発者ガイド」を参照してください。

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

1.1. 対象となる読者

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

上記以外の方でも、本書を有効に利用していただけたら幸いです。

1.2. 本書の構成

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

- 開発環境の構築方法
- フラッシュメモリの書き換え方法
- ビルド方法

1.3. 表記について

1.3.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1. 使用しているフォント

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

1.3.2. コマンド入力例

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

表 1.2. 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC 上の root ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo /]#	Armadillo 上の root ユーザで実行
[armadillo /]$	Armadillo 上の一般ユーザで実行
hermit>	Armadillo 上の保守モードで実行

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

表 1.3. コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.3.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

1.4. 謝辞

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

1.5. 保証に関する注意事項

製品保証範囲について　付属品(ソフトウェアを含みます)を使用し、取扱説明書、各注意事項に基づく正常なご使用に限り有効です。万一正常なご使用のもと製品が故障した場合は、初期不良保証期間内であれば新品交換をさせていただきます。

保証対象外になる場合

次のような場合の故障・損傷は、保証期間内であっても保証対象外になります。

1. 取扱説明書に記載されている使用方法、または注意に反したお取り扱いによる場合
2. 改造や部品交換に起因する場合。または正規のものではない機器を接続したことによる場合
3. お客様のお手元に届いた後の輸送、移動時の落下など、お取り扱いの不備による場合
4. 火災、地震、水害、落雷、その他の天災、公害や異常電圧による場合
5. AC アダプター、専用ケーブルなどの付属品について、同梱のものを使用していない場合
6. 修理依頼の際に購入時の付属品がすべて揃っていない場合

免責事項

弊社に故意または重大な過失があった場合を除き、製品の使用および、故障、修理によって発生するいかなる損害についても、弊社は一切の責任を負わないものとします。

本製品は購入時の初期不良以外の保証を行っておりません。保証期間は商品到着後２週間です。本製品をご購入されましたらお手数でも必ず動作確認を行ってからご使用ください。本製品に対して注意事項を守らずに発生した故障につきましては保証対象外となります。

1.6. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて

本製品に含まれるソフトウェア(付属のドキュメント等も含みます)は、現状のまま(AS IS)提供されるものであり、特定の目的に適合することや、その信頼性、正確性を保証するものではありません。また、本製品の使用による結果についてもなんら保証するものではありません。

1.7. 商標について

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

2.作業の前に

2.1. 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。

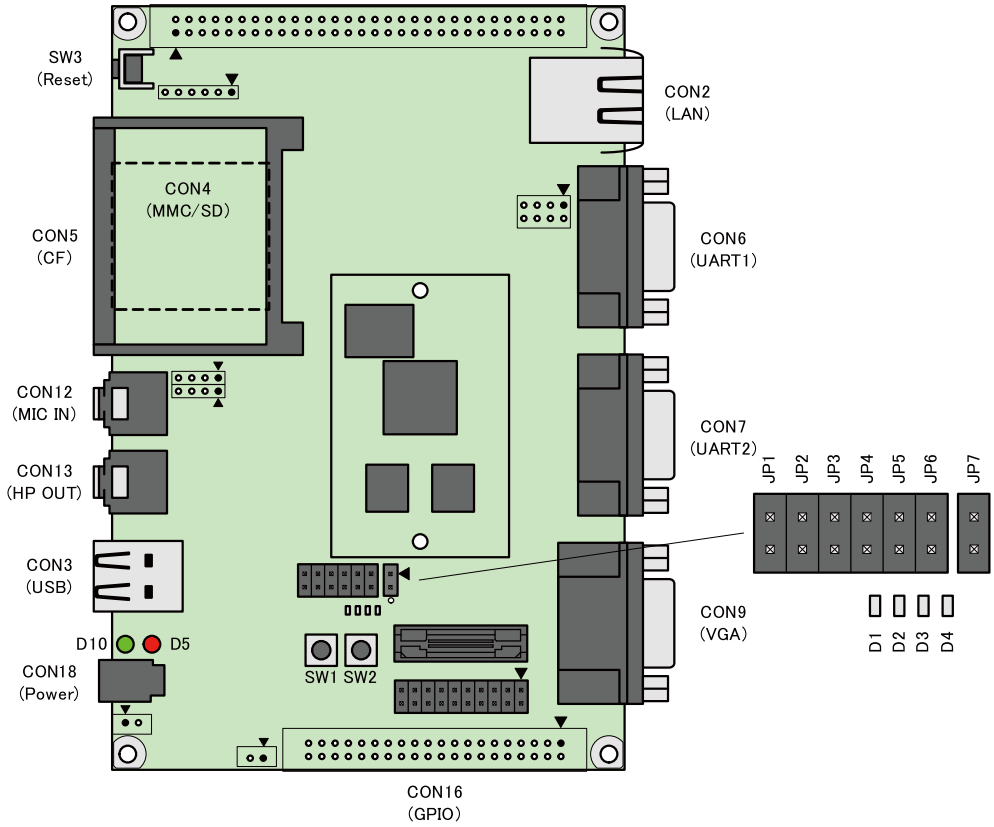

図 2.1. 見取り図

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インターフェースを持つ 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. ジャンパピンの設定について

開発ボードのジャンパピンは、「表 2.1. ジャンパピンの割り当て」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	-
JP7	CPU モジュール設定	CPU モジュールの型番識別に使用

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

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. フラッシュメモリブートモード」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。

有効にするには、ジャンパを「表 2.4. UART ブートモードジャンパー設定」のように設定してください。

表 2.4. 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

UART ブート機能は、フラッシュメモリのブートローダーが壊れた場合など、システム復旧のために使用します。詳しくは、「4.6. ブートローダーを出荷状態に戻す」を参照してください。

2.3.4. CPU モジュール設定

モジュールの型番によって、JP7 を下記のように設定してください。

あやまった設定をすると、正常に動作しないことがあります。

表 2.5. CPU モジュール設定

CPU モジュール型番	JP7 の状態
A50**-U** A50**-U**B(A50**Z-B)	ショート
A50**-U**C(A50**Z-C)	オープン

2.3.5. 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 2.6. シリアル通信設定」のように設定してください。

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

表 2.6. 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

2.3.6. 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 2.7. メモリマップ(フラッシュメモリ)」に示します。

表 2.7. 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

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

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

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

- ATDE Install Guide
- coLinux Guide

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

3.1. クロス開発環境パッケージのインストール

付属 CD の `cross-dev/deb` ディレクトリにクロス開発環境パッケージが用意されています。サポートしている開発環境は、「表 3.1. 開発環境一覧」のとおりです。通常は、arm クロス開発環境をインストールしてください。`cross-dev/deb/`クロスターゲットディレクトリ以下のパッケージをすべてインストールしてください。インストールは必ず root ユーザで行ってください。「図 3.1. インストールコマンド」のようにコマンドを実行します。

表 3.1. 開発環境一覧

クロスターゲット	説明
arm	通常の ARM クロス開発環境です。

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

図 3.1. インストールコマンド

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

[1]debian 系以外の Linux でも開発はできますが、本書記載事項すべてが全く同じように動作するわけではありません。各作業はお使いの Linux 環境に合わせた形で自己責任のもと行ってください。
[2]Armadillo の開発環境としては、ATDE v2.0 以降を推奨しています。

3.2. atmark-dist のビルドに必要なパッケージ

atmark-dist をビルドするためには、「表 3.2. atmark-dist のビルドに必要なパッケージ一覧」に示すパッケージを作業用 PC にインストールされている必要があります。作業用 PC の環境に合わせて適切にインストールしてください。

表 3.2. atmark-dist のビルドに必要なパッケージ一覧

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

現在インストールされているバージョンを表示するには、「図 3.2. インストール情報表示コマンド」のようにパッケージ名を指定して実行してください。

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

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

図 3.2. インストール情報表示コマンド

3.3. クロス開発用ライブラリパッケージの作成方法

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

まず、作成したいクロス開発用パッケージの元となるライブラリパッケージを取得します。元となるパッケージは、ARM 用のパッケージです。例えば、libjpeg6b の場合「`libjpeg6b_[version]_arm.deb`」というパッケージになります。

次のコマンドで、取得したライブラリパッケージをクロス開発用に変換します。

```
[PC ~]$ dpkg-cross --build --arch arm libjpeg6b_[version]_arm.deb
[PC ~]$ ls
libjpeg6b-arm-cross_[version]_all.deb libjpeg6b_[version]_arm.deb
```

図 3.3. クロス開発用ライブラリパッケージの作成

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

4.フラッシュメモリの書き換え方法

Armadillo のオンボード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。

- Armadillo の電源を切断しない
- Armadillo と作業用 PC を接続しているシリアルケーブルと LAN ケーブルを外さない

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。

ダウンローダの種類には、「表 4.1. ダウンローダ一覧」のようなものがあります。

表 4.1. ダウンローダ一覧

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

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

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 **root ユーザ**で行ってください。

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

図 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_[version].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。書き込み領域毎に指定するイメージファイルは、「表 4.2. リージョン名と対応するイメージファイル 」のようになります。

表 4.2. リージョン名と対応するイメージファイル [1]

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

[1]「x」にはバージョン番号の任意の数値が入ります。

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

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

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

4.3.1. 準備

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

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

4.3.2. 作業用 PC が Linux の場合

「図 4.2. ダウンロードコマンド」のようにコマンドを実行します。

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

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

図 4.2. ダウンロードコマンド

シリアルインターフェースが ttyS0 以外の場合は、「図 4.3. ダウンロードコマンド（ポート指定） 」のように`--port` オプションを使用してポートを指定してください。

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

図 4.3. ダウンロードコマンド（ポート指定） [1]

bootloader リージョンは、誤って書き換えることがないように簡易プロテクトされています。書き換える場合は、「図 4.4. ダウンロードコマンド（アンプロテクト）[1]」のように`--force-locked` オプションを使用して、プロテクトの解除をしてください。

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

図 4.4. ダウンロードコマンド（アンプロテクト） [1]

[1] コマンドは 1 行で入力します。

bootloader リージョンに誤ったイメージを書き込んでしまった場合、オンボードフラッシュメモリからの起動ができなくなります。この場合は「4.6. 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。

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. Hermit-At：Download ウィンドウ」が表示されます。

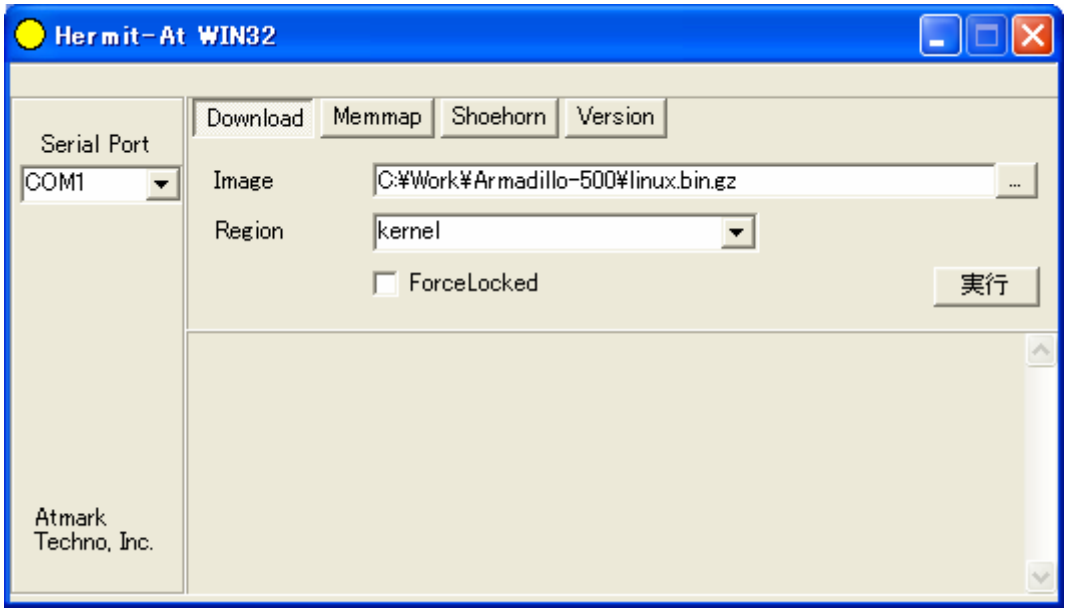

図 4.5. Hermit-At：Download ウィンドウ

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

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

すべて設定してから実行ボタンをクリックします。「図 4.6. Hermit-At：download ダイアログ」が表示されます。

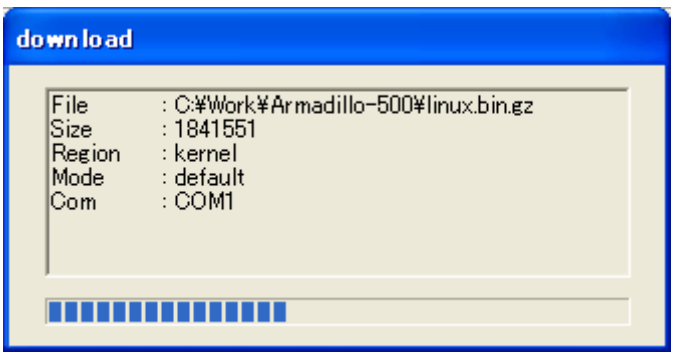

図 4.6. Hermit-At：download ダイアログ

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ダイアログはクローズされます。

4.4. tftpdl を使用してフラッシュメモリを書き換える

Armadillo のブートローダー機能の 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。

「2.3.2. オンボードフラッシュメモリブート」を参照して、Armadillo の起動モードを保守モードに変更し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. tftpdl コマンド例」は、Armadillo の IP アドレスを 192.168.10.10 に設定し、IP アドレスが 192.168.10.1 の tftp サーバー上にある、linux.bin.gz を kernel リージョンにを書き込む例です。

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

図 4.7. tftpdl コマンド例

実行すると、「図 4.8. tftpdl ログ」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。

Armadillo にログインし、「図 4.9. netflash コマンド例」のようにコマンドを実行します。

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

図 4.9. netflash コマンド例

オプションの"-r [デバイスファイル名]"で書き込み対象のリージョンを指定しています。「表 4.3. リージョンとデバイスファイルの対応」を参照してください。その他のオプションについては、netflash -h で詳細を確認する事ができます。

表 4.3. リージョンとデバイスファイルの対応

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

4.6. ブートローダーを出荷状態に戻す

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

4.6.1. 準備

Armadillo のジャンパを、「表 2.4. UART ブートモードジャンパー設定」のように設定してください。

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

4.6.2. 作業用 PC が Linux の場合

「図 4.10. shoehorn コマンド例」のようにコマンド[2]を実行してから、Armadillo 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

実行すると、「図 4.11. shoehorn ログ」のようにログが表示されます。

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

```
/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 ダウンローダを使用してフラッシュメモリを書き換える」を参照してブートローダー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. Hermit-At：Shoehorn ウィンドウ」が表示されます。

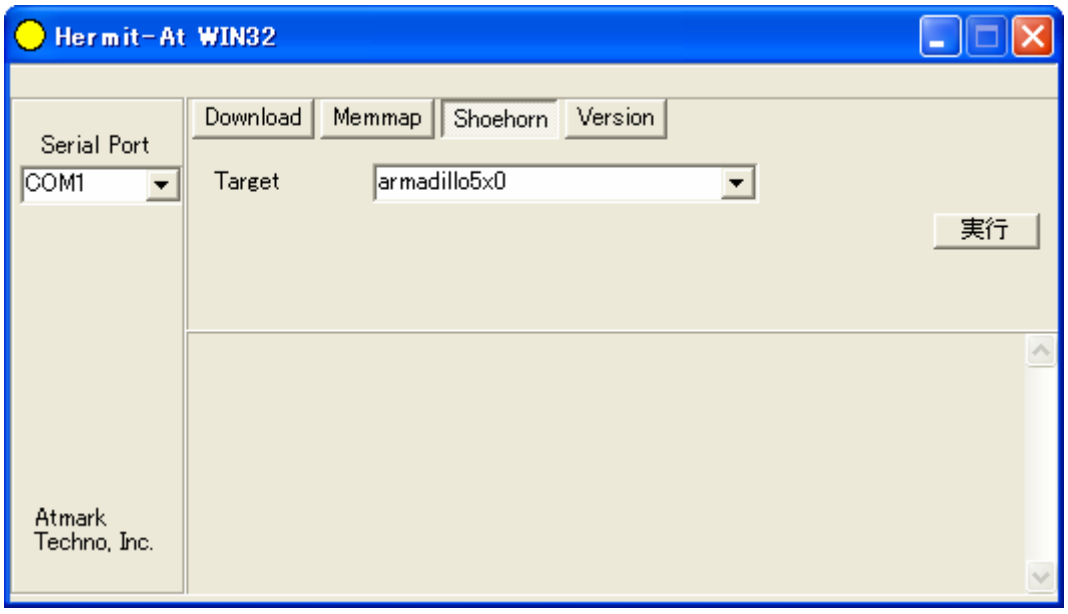

図 4.12. Hermit-At：Shoehorn ウィンドウ

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

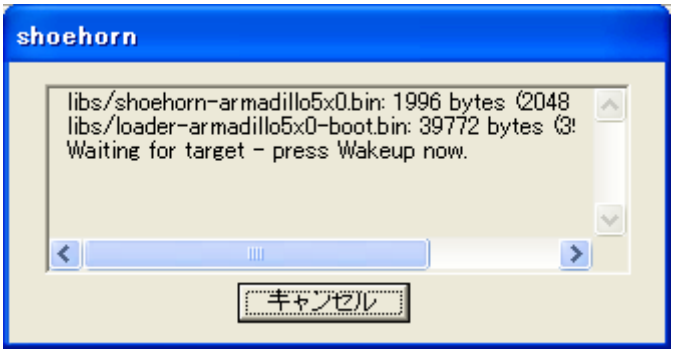

図 4.13. Hermit-At：shoehorn ダイアログ

ダイアログが表示されます。Armadillo 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 ダウンローダを使用してフラッシュメモリを書き換える」を参照してブートローダーの書き込みを行ってください。

4.7. ブートローダーの種類

Armadillo には複数のブートローダーが用意されています。ブートローダーの一覧は、「5.3. ブートローダーイメージのビルド」を参照してください。

5.ビルド

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

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

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. ソースコード準備」のように作業してください。

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

図 5.1. ソースコード準備

5.1.2. コンフィグレーション

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

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

続いて、使用するボードのベンダー名を聞かれます。「AtmarkTechno」と入力してください。

```
[PC ~/atmark-dist]$ make config
config/mkconfig > config.in
#
# No defaults found
```

```
#
*
* Vendor/Product Selection
*
*
* Select the Vendor you wish to target
*
Vendor (3com, ADI, Akizuki, Apple, Arcturus, Arnewsh, AtmarkTechno, Atmel, Avnet,
Cirrus, Cogent, Conexant, Cwlinux, CyberGuard, Cytek, Exys, Feith, Future, GDB,
Hitachi, Imt, Insight, Intel, KendinMicrel, LEOX, Mecel, Midas, Motorola, NEC,
NetSilicon, Netburner, Nintendo, OPENcores, Promise, SNEHA, SSV, SWARM, Samsung,
SecureEdge, Signal, SnapGear, Soekris, Sony, StrawberryLinux, TI, TeleIP,
Triscend, Via, Weiss, Xilinx, senTec) [SnapGear] (NEW) AtmarkTechno
```

次にプロダクト名を聞かれます。「表 5.1. プロダクト名一覧」から、使用する製品に対応するプロダクト名を入力してください。

表 5.1. プロダクト名一覧

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

以下は、Armadillo-210.Base の例です。

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

ビルドする開発環境を聞かれます。「default」と入力してください。

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

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

使用する C ライブラリを指定します。「None」を選択してください。

```
Libc Version (None, glibc, uC-libc, uClibc) [uClibc] (NEW) None
```

デフォルトの設定にするかどうか質問されます。「y」(Yes)を選択してください。

```
Default all settings (lose changes) (CONFIG_DEFAULTS_OVERRIDE) [N/y/?] (NEW) y
```

最後の 3 つの質問は「n」(No)と答えてください。

```
Customize Kernel Settings (CONFIG_DEFAULTS_KERNEL) [N/y/?] n
Customize Vendor/User Settings (CONFIG_DEFAULTS_VENDOR) [N/y/?] n
Update Default Vendor Settings (CONFIG_DEFAULTS_VENDOR_UPDATE) [N/y/?] n
```

質問事項が終わるとビルドシステムの設定を行います。すべての設定が終わるとプロンプトに戻ります。

5.1.3. ビルド

ビルドするには、atmark-dist ディレクトリで「図 5.2. ビルド」のようにコマンドを実行します。ビルドが完了すると、`atmark-dist/images` ディレクトリに `linux.bin.gz` と `romfs.img.gz` が作成されます。

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


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

図 5.2. 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 5.3. ユーザーランドイメージのカスタマイズ

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

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-*[version]*-source.tar.gz を作業ディレクトリに展開します。「図 5.4. ソースコード展開例」のように作業してください。

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

図 5.4. 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 と PROFILE を指定します。製品毎にパラメータが異なりますので、「表 5.2. ビルドオプション一覧」を参照してください。

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

表 5.2. ビルドオプション一覧

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

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

例えば、Armadillo-210(PROFILE=指定なし)の場合「図 5.5. ビルド例 1」のように実行します。

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


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

図 5.5. ビルド例 1

同様に、Armadillo-500 FX の場合「図 5.6. ビルド例 2」のように実行します。

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


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

図 5.6. ビルド例 2

6.コンパクトフラッシュシステム構築

6.1. コンパクトフラッシュシステム例

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

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

この章では、「表 6.1. コンパクトフラッシュシステム例」のようなコンパクトフラッシュシステムを例に、構築手順を説明します。

表 6.1. コンパクトフラッシュシステム例

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

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

6.2. コンパクトフラッシュの初期化

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

作業の前に、ジャンパピンを以下のように設定してください。

表 6.2. コンパクトフラッシュ初期化時のジャンパピン設定

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

6.2.1. ディスクフォーマット

「図 6.1. ディスク初期化方法」のように、ディスクをフォーマットします。

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

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

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

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

Command (m for help): p

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

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

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

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

図 6.1. ディスク初期化方法

6.2.2. ファイルシステムの作成

「図 6.2. ファイルシステムの構築」のように初期化したディスクのパーティションにファイルシステムを作成します。

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

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

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

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

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

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

図 6.2. ファイルシステムの構築

6.3. カーネルイメージを配置する

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

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

表 6.3. カーネルイメージのダウンロード先 URL

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

以下に Armadillo-500 での配置例を示します。

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

図 6.3. カーネルイメージの配置

6.4. ルートファイルシステムの構築

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

6.4.1. Debian GNU/Linux を構築する

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

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

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

表 6.4. debian アーカイブのダウンロード先 URL

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

以下に Armadillo-500 での構築例を示します。

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

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

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

[LOOP] に戻る

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

図 6.4. Debian アーカイブの構築例

6.4.2. atmark-dist イメージから構築する

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

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

表 6.5. atmark-dist イメージのダウンロード先 URL

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

以下に Armadillo-500 での構築例を示します。

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

図 6.5. romfs.img.gz からの作成例

6.5. 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. 起動デバイスの指定」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. ルートファイルシステム指定例」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

6.6. 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。

6.6.1. Debian システム

6.6.1.1. 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. 解決方法：modules.dep」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

6.6.2. atmark-dist システム

6.6.2.1. 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. 解決方法：fstab」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 開発ボード ハードウェアマニュアル」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. JTAG モード指定」のように jtag パラメータを on に設定します。

```
hermit> setenv jtag=on
```

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

付録 A. Hermit-At について

Hermit-At とは、Atmark Techno 製品のブートローダーに採用している高機能ダウンローダー/ブートローダーです。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]...
説明：カーネル起動オプションを設定します。

構文：clearenv
説明：設定されているオプションをクリアします。

図 A.1. setenv と clearenv

A.1.1. setenv/clearenv 使用例

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

hermit> clearenv
```

図 A.2. setenv と clearenv の使用例

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。
rootwait	ルートファイルシステムがアクセス可能になるまで待機します。
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. frob コマンド

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. 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. memmap の使用例

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. erase

A.4.1. 使用例

```
hermit> erase 0xa0fe0000
```

図 A.6. erase の使用例

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]...
説明：指定された内容に基づき TFTP ダウンロードを行い、フラッシュメモリに書き込みます。

図 A.7. 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. tftpdl の使用例

改訂履歴

バージョン	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 アーカイブの構築例」のアーカイブファイル名を変更
1.0.2	2007/10/19	・「3.1. クロス開発環境パッケージのインストール」を修正
1.0.3	2008/3/27	・「図 2.1. 見取り図」に JP7 の情報追加 ・「表 2.1. ジャンパピンの割り当て」に JP7 の情報を追加 ・「2.3.4. CPU モジュール設定」に JP7 の情報を追加
1.0.4	2008/10/02	・「図 6.3. カーネルイメージの配置」の URL 誤記を修正 ・タイトルを英語表記からカタカナ表記に ・「表 2.5. CPU モジュール設定」JP7 に関する注意事項を追記
1.0.5	2008/12/03	・「Development board」を「開発ボード」という表記に統一
1.0.6	2008/12/25	・「図 2.1. 見取り図」画像形式を SVG に変更 ・「図 6.3. カーネルイメージの配置」誤記修正
1.1.0	2009/03/18	・「1. はじめに」、「2. 作業の前に」、「3. 開発環境の準備」、「4. フラッシュメモリの書き換え方法」、「5. ビルド」 構成変更 ・誤記、表記ゆれを修正
1.1.1	2009/07/17	・「7. JTAG」の表記を変更 ・本文のレイアウト統一 ・表記ゆれを修正 ・「4. フラッシュメモリの書き換え方法」にコマンド例の説明を追記 ・「図 5.1. ソースコード準備」の誤記を修正

Armadillo-500 開発ボードソフトウェアマニュアル
Version 1.1.1-8d87fa8
2009/07/17

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