Armadillo-500 FX 液晶モデル
ソフトウェアマニュアル

A542701-D00Z

Version 1.0.0-f2105ac
2008/10/23

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]

Armadillo-500 FX 液晶モデル
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 1.0.0-f2105ac
2008/10/23

目次

図目次

表目次

1.はじめに

Armadillo-500 FX 液晶モデルは、中核機能を持った「FX ボード」とユーザインターフェースを実現する「インターフェースボード」から構成されています。FX ボードは Freescale 社製 ARM11 プロセッサ「i.MX31」、DDR SDRAM、フラッシュメモリを高集積に配置した高性能小型 CPU モジュール「Armadillo-500」を中心に、パネルコンピュータとしての機能が凝縮されています。インターフェースボードは、「ユーザインターフェース」を実現する部分（LCD の種類、ボタンの数、タッチパネルの種類など）と「外部インターフェース」を実現する部分（USB ポートやオーディオ入出力、SD スロットなど）で構成されています。

FX ボードをそのまま利用しインターフェースボードだけをカスタマイズ開発することで、パネルコンピュータ開発時のハードウェアに対する様々な要求に短期間で対応することが可能となります。

本書は Armadillo-500 FX 液晶モデル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 FX 液晶モデル スタートアップガイド」を参照してください。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。また、ハードウェアの仕様に関しては「Armadillo-500 ハードウェアマニュアル」と、「Armadillo-500 FX ボード ハードウェアマニュアル」、「Armadillo-500 FX インターフェースボード ハードウェアマニュアル」を参照してください。

1.1. 対象となる読者

- Armadillo-500 FX 液晶モデルのソフトウェアをカスタマイズされる方
- USB 接続 SSD 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-500 FX 液晶モデル(以下、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

本書は、以下のように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 /]$`	作業用 PC 上の一般ユーザで実行
`[a500-fx /]#`	Armadillo 上の特権ユーザで実行
`[a500-fx /]$`	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。

2.作業の前に

2.1. 見取り図

Armadillo-500 FX 液晶モデルの見取り図です。各インターフェースの配置場所等を確認してください。

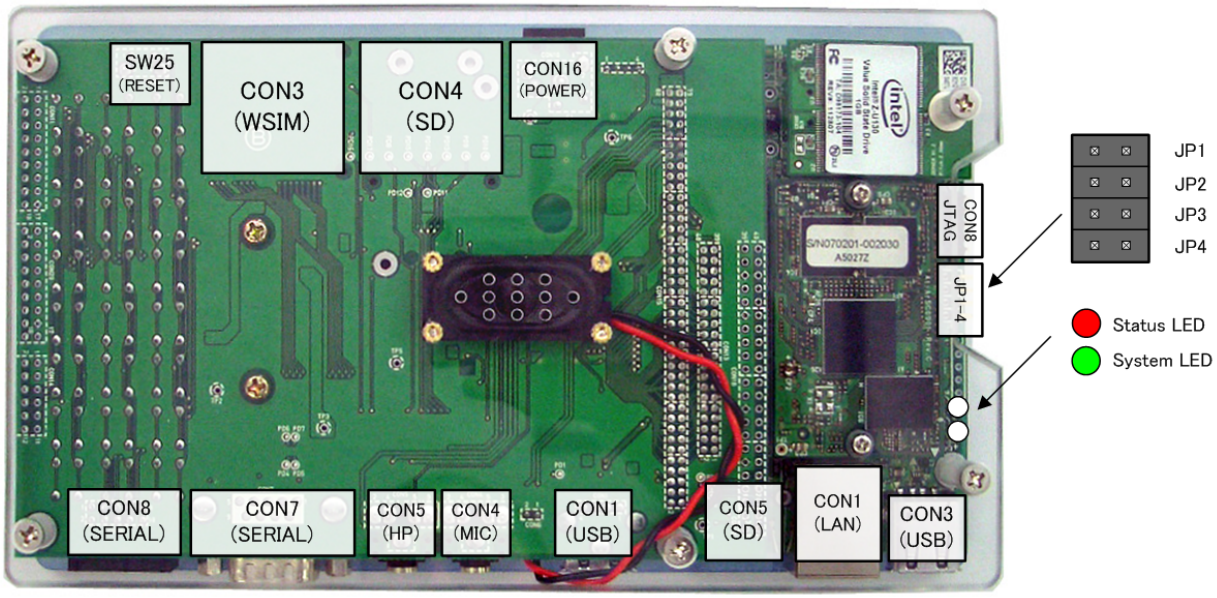

図 2.1. 見取り図

2.2. 準備するもの

Armadillo を使用する前に、次のものを準備してください。

作業用 PC とシリアルクロスケーブル	Linux または 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]	Armadillo 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

Armadillo のジャンパピンは、「表 2.1. ジャンパピンの機能割り当て」のように機能が割り当てられています。

表 2.1. ジャンパピンの機能割り当て

ジャンパ	機能	デフォルトソフトウェアでの使用状況
JP1	USB OTG ポートの ID 設定	予約済み
JP2	ユーザ設定	未使用
JP3	CPU 起動モード設定	—
JP4	ユーザ設定	ブートローダのモード設定に使用

2.3.1. CPU 起動モード設定

JP3 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、オンボード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	モード
オープン	オンボードフラッシュメモリブート
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 JP4 の設定によって、「表 2.3. オンボード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. オンボードフラッシュメモリブートモード

JP4	モード	説明
オープン	オートブート	電源投入後、Linux 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。

2.3.4. 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. シリアル通信設定」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

2.3.5. 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. メモリマップ(フラッシュメモリ)」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 | 0xa1ffffff	all	32MB	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、gzip 圧縮イメージに対応)
0xa0220000 | 0xa1fdffff	userland	29.75MB	ユーザランド領域 「`romfs.img(.gz)`」のイメージ (非圧縮イメージ、gzip 圧縮イメージに対応)
0xa1fe0000 | 0xa1ffffff	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-500 FX 液晶モデルのソフトウェア開発には、Debian/GNU Linux 系の OS 環境[1]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は「ATDE インストールガイド」を参照してください。

3.1. クロス開発環境パッケージのインストール

必要になるクロス開発環境パッケージは CD またはダウンロードサイトからインストールできます。

3.1.1. ダウンロードサイトからのインストール

ATDE の場合、以下のコマンドを実行して必要なパッケージをダウンロードサイトから簡単にインストールできます。インストールは必ず特権ユーザで行ってください。

```
[PC ~]# apt-get install a500fx-development-environment
```

図 3.1. ダウンロードサイトからのインストールコマンド

3.1.2. CD からのインストール

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 プロセッサ用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特権ユーザで行ってください。

CD からのインストールの場合、クロス開発環境パッケージをインストールする前に pkg-config パッケージをインストールしてください。

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。

[1]Debian/GNU Linux 4.0（Etch）を標準とします。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 FX 液晶モデルの開発環境としては、ATDE v2.0 以降を推奨しています。

```
[PC ~]# apt-get update
[PC ~]# apt-get install pkg-config
[PC ~]# cd [CD-ROM マウントディレクトリ]/cross-dev/deb/arm
[PC ~]# dpkg -i *.deb
```

図 3.2. CD からの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。

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」に示すパッケージが必要です。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。

ATDE(Atmark Techno Development Environment)を利用する場合、必要な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。

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 3.3. 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。

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

図 3.3. 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_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 libjpeg6b_[version]_arm.deb
[PC ~]$ ls
libjpeg6b-arm-cross_[version]_all.deb libjpeg6b_[version]_arm.deb
```

図 3.4. クロス開発用ライブラリパッケージの作成

Debian 4.0（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

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、「図 4.1. ダウンローダのインストール（Linux）」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**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
```

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. 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

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

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

Armadillo の起動モードを、保守モードに変更します。JP4 をショートして再起動してください。

Armadillo の CON7 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. ダウンロードコマンド」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. ダウンロードコマンド（ポート指定）」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）

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）」のようにプロテクトの解除を指定します。

```
[PC ~]$ hermit download -i loader-armadillo5x0-fx.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. Hermit-At：Download ウィンドウ」が表示されます。

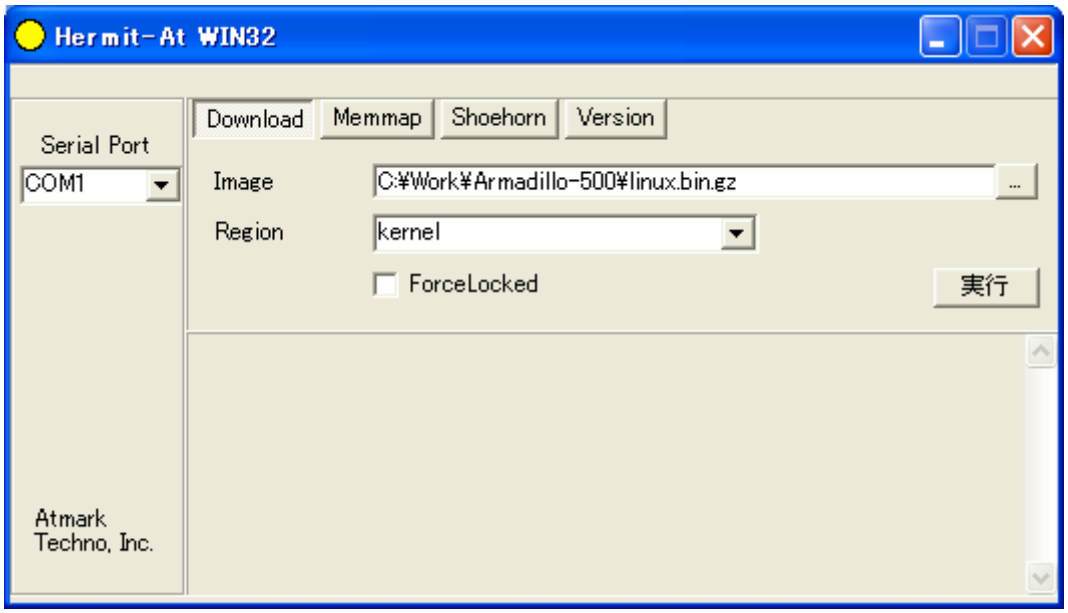

図 4.5. Hermit-At：Download ウィンドウ

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. Hermit-At：download ダイアログ」が表示されます。

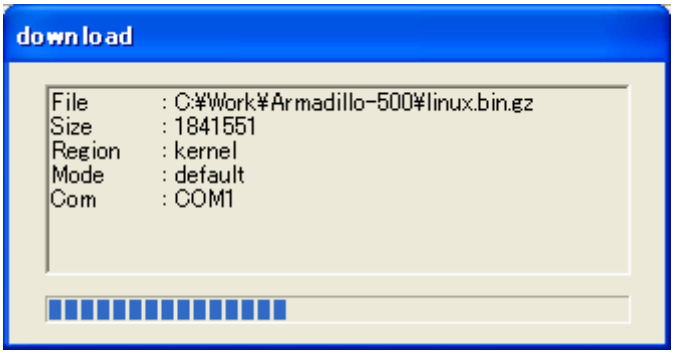

図 4.6. Hermit-At：download ダイアログ

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。

Armadillo の起動モードを保守モードに変更します。JP4 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. tftpdl コマンド例」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
①Armadillo-500 FX が使用する IP アドレス
②TFTP 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. 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。

```
[a500-fx ~]# netflash -k -n -u -r /dev/flash/nor.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

Armadillo のジャンパを、「表 4.3. 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 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

Armadillo の CON7 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. shoehorn コマンド例」のようにコマンド[1]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。

```
/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

[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。

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。

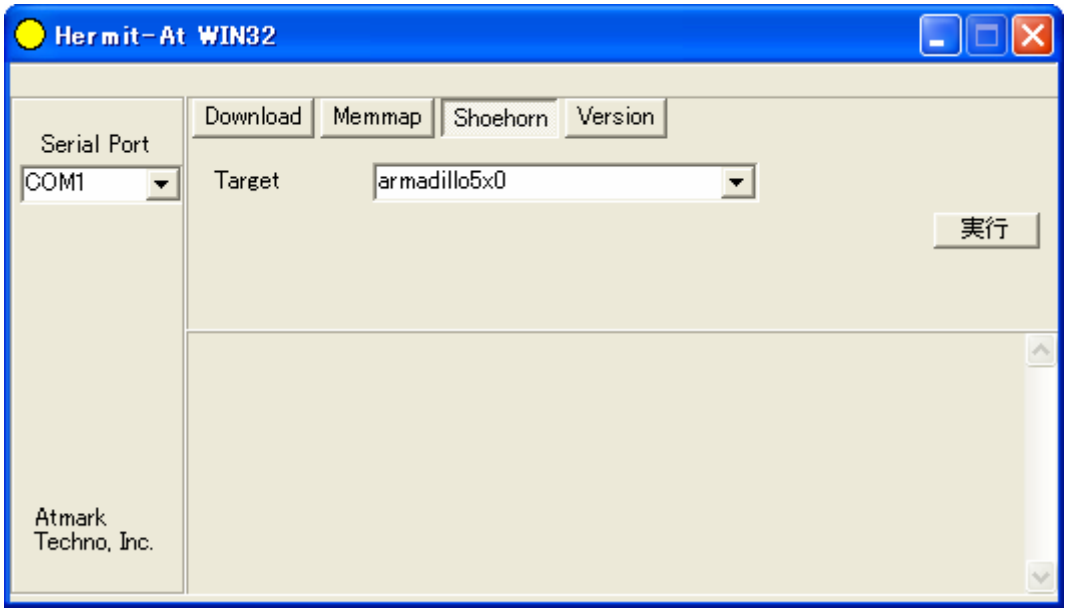

図 4.12. Hermit-At：Shoehorn ウィンドウ

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

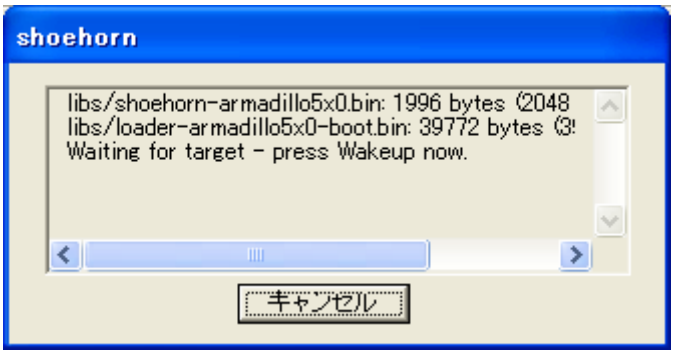

図 4.13. Hermit-At：shoehorn ダイアログ

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-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

Armadillo 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. コンフィグレーション」のようにコマンドを実行してください。

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

```
[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-500-FX.dev, Armadillo-9,
Armadillo-9.PCMCIA,
Armadillo-J.Base, Armadillo-J.Jffs2, Armadillo-J.Recover, SUZAKU,
SUZAKU-UQ-XUP) [Armadillo] Armadillo-500-FX.dev
*
* 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) [None] 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

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. 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 5.3. 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。「図 5.5. 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 5.5. 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 PROFILE=fx を指定してビルドします。「図 5.6. ビルド」のように実行してください。

```
[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

6.USB SSD システム構築

Armadillo-500 FX 液晶モデルでは、USB 接続 Solid State Disk（以降、SSD と表記）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 SSD システムの構築手順について説明します。

SSD に Linux カーネルをインストールしてブートさせることができません。ここで構築するファイルシステムを起動する時でも、Linux カーネルはフラッシュメモリに書き込まれている物を使用します。

この章では、「表 6.1. SSD システム例」のような SSD システムを例に、構築手順を説明します。

表 6.1. SSD システム例

パーティション	タイプ	容量	説明
/dev/sda1	ext3	1GB	ルートファイルシステムを配置する領域です。

6.1. SSD の初期化

ここでは、SSD をフォーマットし、パーティション 1 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. 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。

```
[a500-fx ~]# fdisk /dev/sda

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 (1-1011, default 1011):
Using default value 1011

Command (m for help): p

Disk /dev/sda: 1027 MB, 1027604480 bytes
32 heads, 62 sectors/track, 1011 cylinders
Units = cylinders of 1984 * 512 = 1015808 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        1011     1002881   83  Linux

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

Calling ioctl() to re-read partition table.
sd 1:0:0:0: [sda] 2007040 512-byte hardware sectors (1028 MB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 1:0:0:0: [sda] 2007040 512-byte hardware sectors (1028 MB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
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. 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。

```
[a500-fx ~]# mke2fs -j /dev/sda1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
125440 inodes, 250720 blocks
12536 blocks (5%) reserved for the super user
First data block=0
8 block groups
32768 blocks per group, 32768 fragments per group
15680 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 35 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. ルートファイルシステムの構築

ここでは、SSD 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。

6.2.1. 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 6.3. Debian アーカイブの展開例」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。

```
[a500-fx ~]# mount /dev/sda1 /mnt
[a500-fx ~]# cd /mnt

[a500-fx /mnt]# wget http://download.atmark-techno.com/armadillo-500/debian/
debian-etch-arm1.tgz
Connecting to download.atmark-techno.com [210.191.215.172]:80
debian-etch-1.tgz    100% |***********************|  **** KB    00:00 ETA
[a500-fx /mnt]# tar -xzf debian-etch-arm1.tgz
[a500-fx /mnt]# rm -f debian-etch-arm1.tgz

同様に、debian-etch-arm2.tgz から debian-etch-arm5.tgz について繰り返します。

[a500-fx /mnt]# cd ~
[a500-fx ~]# umount /mnt
```

図 6.3. Debian アーカイブの展開例

6.2.2. 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 SSD 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。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 6.4. romfs.img.gz からの作成例」で示します。

```
[a500-fx ~]# mount /dev/sda1 /mnt
[a500-fx ~]#  cd /mnt
[a500-fx /mnt]#  wget http://download.atmark-techno.com/armadillo-500/images/
romfs-a500-fx-[version].img.gz
Connecting to download.atmark-techno.com [210.191.215.172]:80
romfs-a500-fx-[version].img.gz 100% |**********************|  **** KB    00:00
ETA
[a500-fx /mnt]#  gzip -dc romfs-a500-[version].img.gz > romfs.img
[a500-fx /mnt]#  mkdir romfs
[a500-fx /mnt]#  mount -o loop romfs.img romfs
[a500-fx /mnt]#  (cd romfs/; tar cf - *) | (cd /mnt; tar xf -)
[a500-fx /mnt]# umount romfs
[a500-fx /mnt]# rm -rf romfs
[a500-fx /mnt]# rm -f romfs-a500-fx-[version].img.gz
[a500-fx /mnt]# cd ~
[a500-fx ~]# umount /mnt
```

図 6.4. romfs.img.gz からの作成例

6.3. SSD 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、SSD のルートファイルシステムで起動するためには、「図 6.5. ルートファイルシステム指定例」を実行してください。

```
hermit> setenv console=ttymxc0 root=/dev/sda1 rootwait usb-storage.delay_use=0
noinitrd video=mxcfb:KYOCERA-VGA,16bpp,enable
```

図 6.5. ルートファイルシステム指定例

6.4. 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。

6.4.1. Debian システム

6.4.1.1. 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 6.6. WARNING：modules.dep

解決方法

システムにログインし、「図 6.7. 解決方法：modules.dep」のようにコマンド[1]を実行します。

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

図 6.7. 解決方法：modules.dep

[1]「uname -r」の前後の記号はクォートではなくバッククォートです。

6.4.2. atmark-dist システム

6.4.2.1. 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 6.8. WARNING：fstab

解決方法

システムにログインし、/etc/fstab を「図 6.9. 解決方法：fstab」のように変更します。

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

/dev/sda1       /               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 6.9. 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. 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」のように設定します。

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

図 7.1. JTAG モード指定

JTAG モードに必ず on を指定します。

7.1.1. 設定例

```
hermit> console=ttymxc0 video=mxcfb:KYOCERA-VGA,16bpp,enable 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]...
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
hermit> setenv
hermit>
```

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。
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。
quit	frob モードを終了します。

A.3. memmap

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
説明：フラッシュメモリと DRAM のリージョン情報を一覧表示します。

図 A.3. memmap

A.3.1. 使用例

```
hermit> memmap
0xa0000000:0xa1ffffff FLA all bf:8K bl:4x32K/l,255x128K/l
0xa0000000:0xa001ffff FLA bootloader bf:8K bl:4x32K/l
0xa0020000:0xa021ffff FLA kernel bf:8K bl:16x128K
0xa0220000:0xa1fdffff FLA userland bf:8K bl:238x128K
0xa1fe0000:0xa1ffffff FLA config bf:8K bl:1x128K
0x80000000:0x87ffffff 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]
説明：指定されたアドレスから始まるブロックを消去します。

図 A.5. 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. erase の使用例」のようになります。

```
hermit> erase 0xa1fe0000
```

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	2008/10/20	・初版発行

Armadillo-500 FX 液晶モデルソフトウェアマニュアル
Version 1.0.0-f2105ac
2008/10/23

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