

Armadillo-400 シリーズ ソフトウェアマニュアル

Version 1.2.0
2010/06/08

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-400 シリーズ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

製作著作 © 2010 Atmark Techno, Inc.

Version 1.2.0
2010/06/08

目次

- 1. はじめに 9
 - 1.1. 対象となる読者 9
 - 1.2. 本書の構成 9
 - 1.3. 表記について 10
 - 1.3.1. フォント 10
 - 1.3.2. コマンド入力例 10
 - 1.3.3. アイコン 10
 - 1.4. 謝辞 11
 - 1.5. ソフトウェア使用に関する注意事項 11
 - 1.6. 商標について 11
- 2. システム概要 12
 - 2.1. Armadillo-400 シリーズ基本仕様 12
 - 2.2. Armadillo-420 ベーシックモデル基本仕様 13
 - 2.3. Armadillo-440 液晶モデル基本仕様 15
 - 2.4. メモリマップ 19
 - 2.5. ソフトウェア構成 20
 - 2.5.1. ブートローダー 20
 - 2.5.2. カーネル 20
 - 2.5.3. ユーザーランド 20
 - 2.5.4. ダウンローダー 21
 - 2.6. ブートモード 21
- 3. 作業の前に 22
 - 3.1. 準備するもの 22
 - 3.2. 接続方法 22
 - 3.3. シリアル通信ソフトウェアの設定 24
- 4. 開発環境の準備 25
 - 4.1. クロス開発環境パッケージのインストール 25
 - 4.2. Atmark-Dist のビルドに必要なパッケージのインストール 26
 - 4.3. クロス開発用ライブラリパッケージのインストール 26
- 5. フラッシュメモリの書き換え方法 28
 - 5.1. フラッシュメモリの書き込み領域について 28
 - 5.2. ダウンローダーのインストール 29
 - 5.2.1. 作業用 PC が Linux の場合 29
 - 5.2.2. 作業用 PC が Windows の場合 29
 - 5.3. ダウンローダーを使用してフラッシュメモリを書き換える 29
 - 5.3.1. 準備 30
 - 5.3.2. 作業用 PC が Linux の場合 30
 - 5.3.3. 作業用 PC が Windows の場合 31
 - 5.4. tftpd を使用してフラッシュメモリを書き換える 32
 - 5.5. netflash を使用してフラッシュメモリを書き換える 32
 - 5.6. ブートローダーを出荷状態に戻す 33
 - 5.6.1. 準備 33
 - 5.6.2. 作業用 PC が Linux の場合 33
 - 5.6.3. 作業用 PC が Windows の場合 35
- 6. ビルド 36
 - 6.1. カーネルイメージとユーザーランドイメージのビルド 36
 - 6.1.1. ソースコードの準備 36
 - 6.1.2. デフォルトコンフィギュレーションの適用 37
 - 6.1.3. ビルド 38
 - 6.1.4. イメージをカスタマイズする 39

- 6.1.5. ユーザーランドイメージにアプリケーションを追加する 43
- 6.2. ブートローダーイメージのビルド 44
 - 6.2.1. ソースコードの準備 44
 - 6.2.2. ビルド 44
- 7. カーネル/ユーザーランドの配置 46
 - 7.1. TFTP サーバーに配置する 46
 - 7.1.1. ファイルの配置 46
 - 7.1.2. ブートオプション 46
 - 7.2. ストレージに配置する 47
 - 7.2.1. パーティション分割 47
 - 7.2.2. ファイルシステムの作成 48
 - 7.2.3. カーネルイメージの配置 49
 - 7.2.4. ルートファイルシステムの構築 50
 - 7.2.5. カーネルパラメータの設定 52
 - 7.3. 設定を元に戻す 52
- 8. Linux カーネルデバイスドライバ仕様 54
 - 8.1. UART 54
 - 8.2. Ethernet 56
 - 8.3. SD/MMC/SDIO ホスト 56
 - 8.4. USB 2.0 ホスト 57
 - 8.5. フレームバッファ 57
 - 8.6. LED バックライト 58
 - 8.7. タッチスクリーン 58
 - 8.8. オーディオ 58
 - 8.9. GPIO 59
 - 8.9.1. GPIO sysfs 60
 - 8.9.2. Armadillo-200 シリーズ互換 GPIO ドライバー 61
 - 8.10. LED 62
 - 8.10.1. LED クラス 62
 - 8.10.2. Armadillo-200 シリーズ互換 LED ドライバー 63
 - 8.11. ボタン 64
 - 8.12. リアルタイムクロック 64
 - 8.13. ウォッチドッグタイマー 65
 - 8.14. I2C 65
 - 8.15. SPI 65
 - 8.16. one wire 67
 - 8.17. PWM 67
 - 8.18. CAN 68
 - 8.19. キーボード 70
 - 8.20. パワーマネジメント 71
 - 8.20.1. スリープ中の外部デバイスの扱いについて 72
- A. Hermit-At ブートローダー 74
 - A.1. version 74
 - A.1.1. version 使用例 74
 - A.2. info 74
 - A.2.1. info 使用例 74
 - A.3. memmap 75
 - A.3.1. memmap 使用例 75
 - A.4. mac 75
 - A.4.1. mac 使用例 75
 - A.5. md5sum 75
 - A.5.1. md5sum 使用例 76
 - A.6. erase 76

A.6.1. erase 使用例	76
A.7. setenv と clearenv	76
A.7.1. setenv/clearenv 使用例	77
A.7.2. Linux 起動オプション	77
A.8. setbootdevice	77
A.8.1. setbootdevice の使用例	78
A.9. frob	78
A.10. tftpd	79
A.10.1. tftpd の使用例	79
A.11. tftpboot	80
A.11.1. tftpboot の使用例	80
A.12. boot	81
A.12.1. boot 使用例	81
A.13. バージョンに関する注意	81

目次

- 2.1. Armadillo-420/440 ブロック図 13
- 2.2. Armadillo-420 ベーシックモデル見取り図 14
- 2.3. Armadillo-440 液晶モデル見取り図 17
- 3.1. Armadillo-440 液晶モデル接続例 23
- 3.2. Armadillo-420 ベーシックモデル接続例 24
- 4.1. インストールコマンド 25
- 4.2. インストール情報表示コマンド 26
- 4.3. クロス開発用ライブラリパッケージの作成 27
- 4.4. クロス開発用ライブラリパッケージのインストール 27
- 4.5. apt-cross コマンド 27
- 5.1. ダウンローダーのインストール (Linux) 29
- 5.2. ダウンロードコマンド 30
- 5.3. ダウンロードコマンド (ポート指定) 30
- 5.4. ダウンロードコマンド (アンプロテクト) 30
- 5.5. Hermit-At Win32 : Download ウィンドウ 31
- 5.6. Hermit-At Win32 : download ダイアログ 31
- 5.7. tftpd command 例 32
- 5.8. netflash コマンド例 33
- 5.9. shoehorn コマンド例 34
- 5.10. shoehorn コマンドログ 34
- 5.11. Hermit-At Win32 : Shoehorn ウィンドウ 35
- 5.12. Hermit-At Win32 : shoehorn ダイアログ 35
- 6.1. ソースコード準備 36
- 6.2. Atmark-Dist のビルド 38
- 6.3. Atmark-Dist のコンフィギュレーション 39
- 6.4. menuconfig: Main Menu 39
- 6.5. menuconfig: Kernel/Library/Defaults Selection 40
- 6.6. menuconfig: Do you wish to save your new kernel configuration? 41
- 6.7. menuconfig: Linux Kernel Configuration 42
- 6.8. menuconfig: Userland Configuration 43
- 6.9. ユーザーランドイメージのカスタマイズ 44
- 6.10. Hermit-At ソースアーカイブの展開 44
- 6.11. Hermit-At ビルド例 44
- 7.1. tftpboot コマンド 46
- 7.2. tftpboot コマンド例 47
- 7.3. パーティション分割手順 48
- 7.4. ファイルシステム作成手順 49
- 7.5. カーネルイメージの配置 50
- 7.6. Debian アーカイブによるルートファイルシステムの構築例 51
- 7.7. Atmark-Dist イメージによるルートファイルシステムの構築例 52
- 7.8. 起動デバイスの指定 52
- 7.9. ルートファイルシステム指定例 52
- 7.10. 起動デバイスにフラッシュメモリを指定する 52
- 7.11. clearenv で起動オプションを初期状態に戻す 53
- 8.1. CAN 通信速度計算 70
- A.1. version 構文 74
- A.2. version の使用例 74
- A.3. info 構文 74
- A.4. info の使用例 74
- A.5. memmap 構文 75

A.6. memmap の使用例	75
A.7. mac 構文	75
A.8. mac の使用例	75
A.9. md5sum 構文	75
A.10. md5sum の使用例	76
A.11. erase 構文	76
A.12. erase の使用例	76
A.13. setenv/clearenv 構文	76
A.14. setenv と clearenv の使用例	77
A.15. setbootdevice 構文	78
A.16. ブートデバイスにフラッシュメモリを指定する	78
A.17. ブートデバイスに TFTP サーバーを指定する	78
A.18. ブートデバイスに SD/MMC カードを指定する	78
A.19. tftpd 構文	79
A.20. tftpd の使用例	79
A.21. tftpboot 構文	80
A.22. tftpboot の使用例	80
A.23. boot 構文	81
A.24. boot の使用例	81

表目次

- 1.1. 使用しているフォント 10
- 1.2. 表示プロンプトと実行環境の関係 10
- 1.3. コマンド入力例での省略表記 10
- 2.1. Armadillo-400 シリーズ基本仕様 12
- 2.2. RTC オプションモジュール基本仕様 13
- 2.3. Armadillo-420 ベーシックモデル拡張インターフェースピン配置 14
- 2.4. 拡張ボード基本仕様 15
- 2.5. Armadillo-440 液晶モデル拡張インターフェースピン配置 18
- 2.6. Armadillo-420 フラッシュメモリ メモリマップ 19
- 2.7. Armadillo-440 フラッシュメモリ メモリマップ 19
- 2.8. ジャンパの設定 21
- 3.1. シリアル通信設定 24
- 4.1. Atmark-Dist のビルドに必要なパッケージ一覧 26
- 5.1. リージョン名と対応するイメージファイル 28
- 5.2. ダウンローダー一覧 29
- 5.3. リージョンとオプションの対応 32
- 5.4. リージョンとデバイスファイルの対応 33
- 6.1. プロダクト名一覧 37
- 7.1. microSD システムパーティション例 47
- 7.2. カーネルイメージのダウンロード先 URL 50
- 7.3. Debian アーカイブのダウンロード先 URL 51
- 7.4. Atmark-Dist イメージのダウンロード先 URL 51
- 8.1. シリアルインターフェースとデバイスファイルの対応 55
- 8.2. UART コンフィギュレーション 55
- 8.3. SD/MMC/SDIO ホストコントローラ コンフィギュレーション 57
- 8.4. フレームバッファとデバイスファイルの対応 57
- 8.5. タッチスクリーンイベント 58
- 8.6. オーディオ コンフィギュレーション 59
- 8.7. GPIO_NAME と GPIO ピンの対応 60
- 8.8. GPIO 入出力方向の設定 60
- 8.9. Armadillo-200 シリーズ互換 GPIO ドライバー GPIO 一覧 61
- 8.10. Armadillo-200 シリーズ互換 GPIO ドライバーデバイスファイル 61
- 8.11. Armadillo-200 シリーズ互換 GPIO ドライバー ioctl コマンド 62
- 8.12. LED 一覧 63
- 8.13. LED ノード 63
- 8.14. LED 操作コマンド 63
- 8.15. Armadillo-440 液晶モデルボタンイベント 64
- 8.16. I2C コンフィギュレーション 65
- 8.17. SPI コンフィギュレーション 66
- 8.18. one wire コンフィギュレーション 67
- 8.19. PWM sysfs 67
- 8.20. PWM コンフィギュレーション 68
- 8.21. CAN sysfs 68
- 8.22. CAN コンフィギュレーション 70
- 8.23. キーボードコンフィギュレーション 71
- 8.24. スリープモード 72
- 8.25. ウェイクアップ要因の指定 72
- A.1. よく使用される Linux 起動オプション 77
- A.2. frob コマンド 78
- A.3. tftpd オプション 79

1.はじめに

Armadillo シリーズは、ARM コアを搭載した高性能・低消費電力な小型汎用 CPU ボードです。標準 OS に Linux (Kernel 2.6 系) を採用しており、豊富なソフトウェア資産と実績のある安定性を提供します。また、全ての製品が標準でネットワークインターフェースを搭載し、Linux のネットワークプロトコルスタックと組み合わせて、容易にネットワーク対応機器の開発を実現します。

Armadillo-400 シリーズは、同クラスの従来製品より性能を向上しつつも、低消費電力を実現したモデルです。Armadillo-400 シリーズには、低価格の Armadillo-420 と拡張ボードによってマルチメディア機能を追加可能な Armadillo-440 の 2 種類の製品があります。

Armadillo-400 シリーズは、基本機能としてシリアル、Ethernet、USB、ストレージ(microSD)、GPIO など組み込み機器に必要とされる機能を備えています。Armadillo-440 はそれらに加え、LCD、タッチスクリーン、オーディオなどのマルチメディア機能を、拡張ボードによって追加可能です。さらに、Armadillo-400 シリーズでは、オプションモジュールによってリアルタイムクロックや無線 LAN などの機能を追加することができます。

Armadillo-440 に LCD 拡張ボードをセットにしたモデルを Armadillo-440 液晶モデルと呼びます。また、Armadillo-420 に RTC オプションモジュールをセットにしたモデルを Armadillo-420 ベーシックモデルと呼びます。

本書には、Armadillo-400 シリーズのソフトウェアをカスタマイズするために必要な情報が記載されています。

出荷状態のソフトウェアの操作方法については、「Armadillo-440 液晶モデル 開発セット スタートアップガイド」または「Armadillo-420 ベーシックモデル 開発セット スタートアップガイド」をご参照ください。また、ハードウェア仕様に関しては、「Armadillo-400 シリーズ ハードウェアマニュアル」をご参照ください。

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

本書は、Armadillo を使用して組み込みシステムを開発される方のうち、Armadillo のソフトウェアをカスタマイズされる方を対象としています。

1.2. 本書の構成

本書は、1 章から 8 章および Appendix から構成されています。

1 章から 3 章で、開発を始めるための準備について取り上げます。

4 章から 6 章で、開発環境を構築し、ブートローダー、カーネル、ユーザーランドのソースコードから一連のイメージファイルを作成する方法と、イメージファイルをターゲットとなる Armadillo に書き込む方法について説明します。

7 章では、カーネルとユーザーランドを Armadillo の内蔵 フラッシュメモリ以外の場所に配置する方法について説明します。

8 章では、Armadillo 独自の Linux カーネルデバイスドライバの仕様について記述します。

最後に、Appendix ではブートローダーの機能について説明します。

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っています。この場を借りて感謝の意を表します。

1.5. ソフトウェア使用に関する注意事項

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。

1.6. 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. システム概要

ソフトウェアの開発を開始する前に、本章ではシステム概要について解説します。

2.1. Armadillo-400 シリーズ基本仕様

Armadillo-400 シリーズの標準状態^[1]での基本仕様を「表 2.1. Armadillo-400 シリーズ基本仕様」に示します。また、ブロック図を「図 2.1. Armadillo-420/440 ブロック図」に示します。

表 2.1 Armadillo-400 シリーズ基本仕様

	Armadillo-420	Armadillo-440
プロセッサ	Freescale i.MX257 (ARM926EJ-S) 命令/データキャッシュ 16KByte/16KByte 内部 SRAM 128KByte	
システムクロック	CPU コアクロック：400MHz BUS クロック：133MHz	
RAM	LPDDR SDRAM：64MByte (16bit 幅)	LPDDR SDRAM：128MByte (16bit 幅)
ROM	NOR フラッシュメモリ：16MByte (16bit 幅)	NOR フラッシュメモリ：32MByte (16bit 幅)
シリアル	RS232C レベル×1 ポート フロー制御ピン有り (フルモデム) 最大 230.4 kbps	
	3.3V I/O レベル×2 ポート フロー制御ピン無し 最大 4Mbps	
USB 2.0 ホスト	High Speed×1 ポート	
	Full Speed×1 ポート	
LAN	10BASE-T/100BASE-TX×1 ポート	
ストレージ	microSD×1 4bit 幅、最大 208Mbps	
GPIO	3.3V I/O レベル×18 ピン	
プログラマブル LED	赤×1、緑×1、黄×1	
ボタン	タクトスイッチ×1	

^[1]Armadillo-400 シリーズは IO ピンのマルチプレクスにより機能を変更することができます。詳しくは、「Armadillo-400 シリーズ ハードウェアマニュアル」及び「8. Linux カーネルデバイスドライバ仕様」をご参照ください。

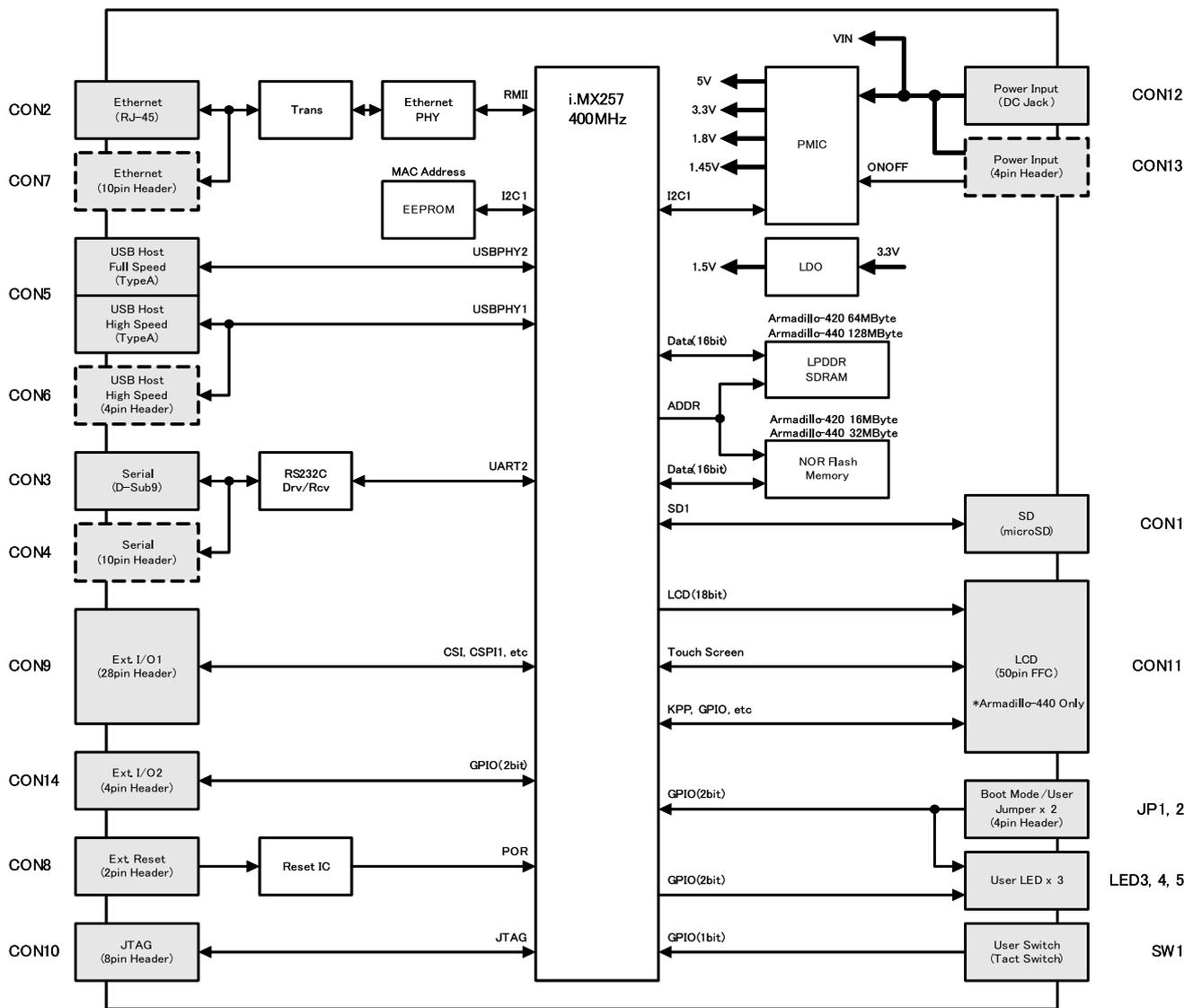


図 2.1 Armadillo-420/440 ブロック図

2.2. Armadillo-420 ベーシックモデル基本仕様

Armadillo-420 ベーシックモデルは、Armadillo-420 に Armadillo-400 RTC オプションモジュールを接続したモデルです。RTC オプションモジュールの基本仕様を、「表 2.2. RTC オプションモジュール基本仕様」に示します。

表 2.2 RTC オプションモジュール基本仕様

	Armadillo-400 RTC オプションモジュール
リアルタイムクロック	スーパーキャパシタによるバックアップ

Armadillo-420 ベーシックモデルの見取り図を「図 2.2. Armadillo-420 ベーシックモデル見取り図」に示します。また、CON9 および CON14 のピン配置を「表 2.3. Armadillo-420 ベーシックモデル拡張インターフェースピン配置」に示します。各インターフェースの配置場所等を確認してください。

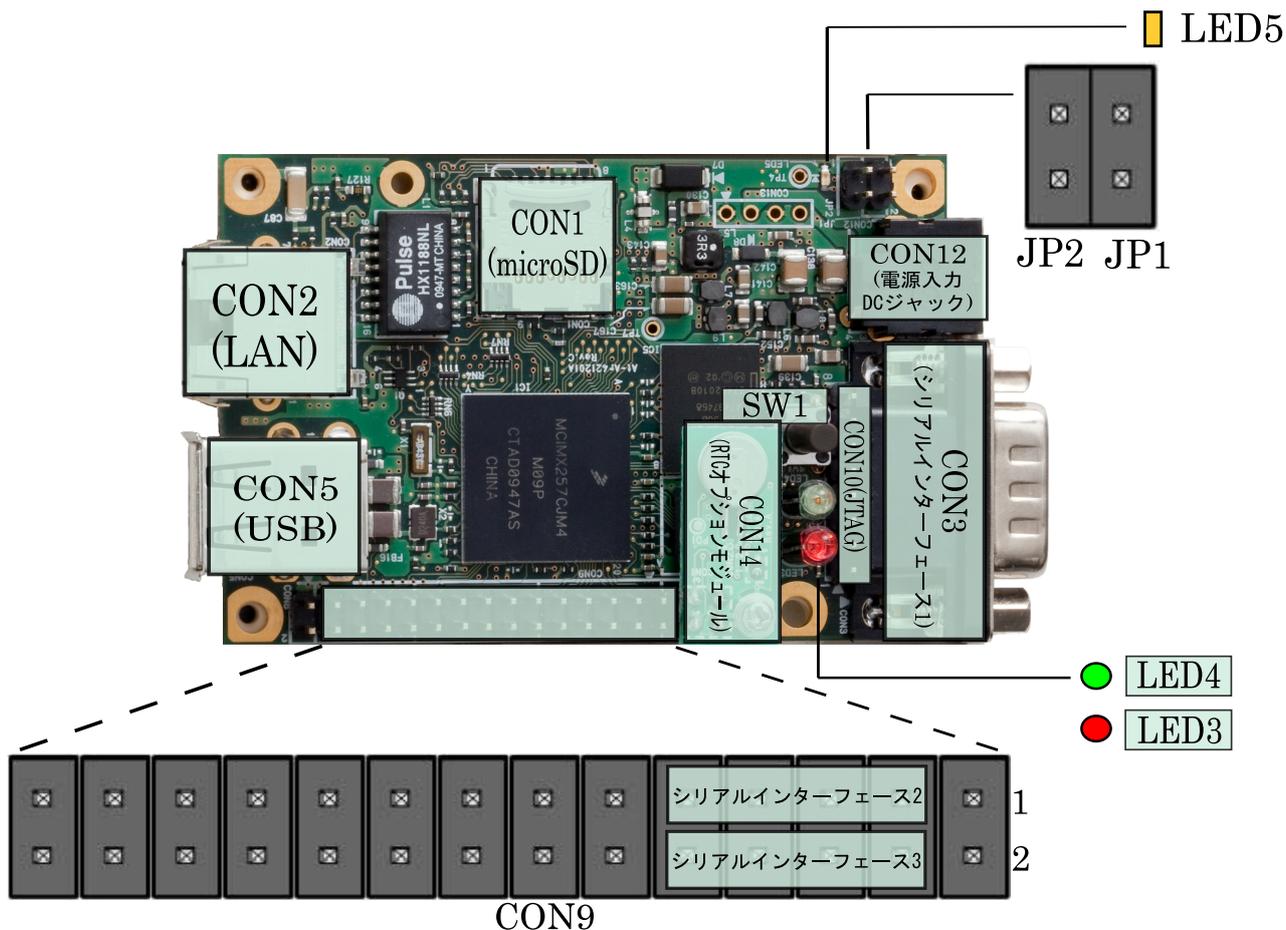


図 2.2 Armadillo-420 ベーシックモデル見取り図

表 2.3 Armadillo-420 ベーシックモデル拡張インターフェースピン配置

ピン番号	機能	備考
CON9 1	GPIO	100 kΩ プルアップ
CON9 2	GPIO	100 kΩ プルアップ
CON9 3	シリアルインターフェース 2 RXD	
CON9 4	シリアルインターフェース 3 RXD	
CON9 5	シリアルインターフェース 2 TXD	
CON9 6	シリアルインターフェース 3 TXD	
CON9 7	+3.3V	
CON9 8	+3.3V	
CON9 9	GND	
CON9 10	GND	
CON9 11	GPIO	100 kΩ プルアップ
CON9 12	GPIO	100 kΩ プルアップ
CON9 13	GPIO	100 kΩ プルアップ
CON9 14	GPIO	100 kΩ プルアップ
CON9 15	GPIO	100 kΩ プルアップ
CON9 16	GPIO	100 kΩ プルアップ

ピン番号	機能	備考
CON9 17	GPIO	100 kΩ プルアップ
CON9 18	GPIO	100 kΩ プルアップ
CON9 19	GND	
CON9 20	+3.3V	
CON9 21	GPIO	100 kΩ プルアップ
CON9 22	GPIO	100 kΩ プルアップ
CON9 23	GPIO	100 kΩ プルアップ
CON9 24	GPIO	100 kΩ プルアップ
CON9 25	GPIO	100 kΩ プルアップ
CON9 26	GPIO	100 kΩ プルアップ
CON9 27	GPIO	
CON9 28	GPIO	
CON14 1	+3.3V	
CON14 2	GND	
CON14 3	I2C2 SCL	22 kΩ プルアップ/オープンドレイン
CON14 4	I2C2 SDA	22 kΩ プルアップ/オープンドレイン



シリアルインターフェース 2 と 3 は +3.3V IO レベルとなっています。オプション^[2]の RS232C レベル変換アダプタを使用することで、RS232C レベルで使用することができます。

RS232C レベル変換アダプタは、シリアルインターフェース 2 に接続する場合は、RS232C レベル変換アダプタの 1 番ピン (黄色いケーブル) と CON9 1 ピンが合うように、シリアルインターフェース 3 に接続する場合は、RS232C レベル変換アダプタの 1 番ピンと CON9 2 ピンが合うように接続してください。

2.3. Armadillo-440 液晶モデル基本仕様

Armadillo-440 液晶モデルは、Armadillo-440 に Armadillo-440 LCD 拡張ボードを接続したモデルです。拡張ボードの基本仕様を、「表 2.4. 拡張ボード基本仕様」に示します。

表 2.4 拡張ボード基本仕様

	Armadillo-440 LCD 拡張ボード
オーディオ	Playback(ステレオ) / Capture(モノラル)
LCD	解像度 480 × 272 ピクセル RGB 565 カラー
タッチスクリーン	4 線抵抗膜式
リアルタイムクロック	スーパーキャパシタによるバックアップ
ボタン	タクトスイッチ×3

^[2]RS232C レベル変換アダプタはオプション品としてご購入いただけます。また、開発セットには付属しています。

Armadillo-440 液晶モデルの見取り図を「図 2.3. Armadillo-440 液晶モデル見取り図」に示します。また、CON9 および CON14 のピン配置を「表 2.5. Armadillo-440 液晶モデル拡張インターフェースピン配置」に示します。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 2.3 Armadillo-440 液晶モデル見取り図

表 2.5 Armadillo-440 液晶モデル拡張インターフェースピン配置

ピン番号	機能	備考
CON9 1	GPIO	100 kΩ プルアップ
CON9 2	GPIO	100 kΩ プルアップ
CON9 3	シリアルインターフェース 2 RXD	
CON9 4	シリアルインターフェース 3 RXD	
CON9 5	シリアルインターフェース 2 TXD	
CON9 6	シリアルインターフェース 3 TXD	
CON9 7	+3.3V	
CON9 8	+3.3V	
CON9 9	GND	
CON9 10	GND	
CON9 11	GPIO	100 kΩ プルアップ
CON9 12	GPIO	100 kΩ プルアップ
CON9 13	GPIO	100 kΩ プルアップ
CON9 14	GPIO	100 kΩ プルアップ
CON9 15	GPIO	100 kΩ プルアップ
CON9 16	GPIO	100 kΩ プルアップ
CON9 17	GPIO	100 kΩ プルアップ
CON9 18	GPIO	100 kΩ プルアップ
CON9 19	GND	
CON9 20	+3.3V	
CON9 21	GPIO	100 kΩ プルアップ
CON9 22	GPIO	100 kΩ プルアップ
CON9 23	GPIO	100 kΩ プルアップ
CON9 24	GPIO	100 kΩ プルアップ
CON9 25	GPIO	100 kΩ プルアップ
CON9 26	GPIO	100 kΩ プルアップ
CON9 27	GPIO	
CON9 28	GPIO	
CON14 1	+3.3V	
CON14 2	GND	
CON14 3	I2C2 SCL	22 kΩ プルアップ/オープンドレイン
CON14 4	I2C2 SDA	22 kΩ プルアップ/オープンドレイン



シリアルインターフェース 2 と 3 は +3.3V IO レベルとなっています。オプション^[3]の RS232C レベル変換アダプタを使用することで、RS232C レベルで使用することができます。

RS232C レベル変換アダプタは、シリアルインターフェース 2 に接続する場合は、RS232C レベル変換アダプタの 1 番ピン (黄色いケーブル) と CON9 1 ピンが合うように、シリアルインターフェース 3 に接続する場

^[3]RS232C レベル変換アダプタはオプション品としてご購入いただけます。また、開発セットには付属しています。

合は、RS232C レベル変換アダプタの 1 番ピンと CON9 2 ピンが合うように接続してください。



CON14 3 と CON14 4 ピンは linux-2.6.26-at7 (linux-a400-1.00.bin.gz) では デフォルトで GPIO として使用していました。linux-2.6.26-at8 (linux-a400-1.01.bin.gz) 以降では、デフォルトの設定で I2C2 として使用するよう変更されました。CON14 3 と CON14 4 ピンをご利用になる際は、デフォルトの設定が変更されているためご注意ください。

2.4. メモリマップ

Armadillo-400 シリーズは、標準で「表 2.6. Armadillo-420 フラッシュメモリ メモリマップ」、「表 2.7. Armadillo-440 フラッシュメモリ メモリマップ」に示すようにフラッシュメモリを分割して使用します。

表 2.6 Armadillo-420 フラッシュメモリ メモリマップ

物理アドレス	リージョン名	サイズ	説明
0xa0000000 0xa001ffff	bootloader	128KB	ブートローダーイメージを格納します
0xa0020000 0xa021ffff	kernel	2MB	カーネルイメージを格納します
0xa0220000 0xa0fdffff	userland	13.75MB	ユーザーランドイメージを格納します
0xa0fe0000 0xa0ffffff	config	128KB	設定情報を保存します

表 2.7 Armadillo-440 フラッシュメモリ メモリマップ

物理アドレス	リージョン名	サイズ	説明
0xa0000000 0xa001ffff	bootloader	128KB	ブートローダーイメージを格納します
0xa0020000 0xa021ffff	kernel	2MB	カーネルイメージを格納します
0xa0220000 0xa1fdffff	userland	29.75MB	ユーザーランドイメージを格納します

物理アドレス	リージョン名	サイズ	説明
0xa1fe0000 0xa1ffffff	config	128KB	設定情報を保存します

2.5. ソフトウェア構成

Armadillo-400 シリーズでは、以下のソフトウェアによって動作します。

2.5.1. ブートローダー

ブートローダーは、電源投入後に最初に動作するソフトウェアです。Armadillo-400 シリーズでは Hermit-At ブートローダー (以降、単に Hermit-At と記述します) を使用します。

Hermit-At にはオートブートモードと保守モードの2つの動作モードがあります。オートブートモードでは、あらかじめ指定された場所からカーネルイメージを RAM 上にロードし、カーネルをブートします。保守モードでは、フラッシュメモリの更新、ブートオプションの設定などを行います。詳しくは、付録 A Hermit-At ブートローダーを参照してください。

ブートローダーは、必ずフラッシュメモリのブートローダーリージョンに書き込まれている必要があります。

2.5.2. カーネル

Armadillo-400 シリーズでは、標準のカーネルとして Linux 2.6 系を使用します。

標準ではカーネルイメージはフラッシュメモリのカーネルリージョンに配置されます。カーネルイメージは、Hermit-At のブートオプションを変更することで、ストレージ(microSD)または TFTP サーバー上にも配置することができます。

2.5.3. ユーザーランド

Armadillo-400 シリーズでは、標準のユーザーランドのルートファイルシステムは Atmark-Dist と呼ばれるソースコードベースのディストリビューションから作成した initrd^[4] イメージを使用します。

また、標準ユーザーランドの他に、オプションとして Debian GNU/Linux ベースのユーザーランドも提供しています。

標準では initrd イメージはフラッシュメモリのユーザーランドリージョンに配置され、Hermit-At によって RAM disk に展開されます。initrd イメージは、Hermit-At のブートオプションを変更することで、TFTP サーバー上にも配置することができます。

ルートファイルシステムは、カーネルパラメータを設定することで、RAM disk 以外にストレージ(microSD/USB) または NFS サーバー^[5]上に配置することもできます。

カーネルとユーザーランドをフラッシュメモリ以外に配置する方法については、「7. カーネル/ユーザーランドの配置」で詳しく説明します。

^[4]initial RAM disk。一般的な Linux システムでは、initrd は HDD などにあるルートファイルシステムをマウントする前に一時的に使用する「ミニ」ルートファイルシステムとして使用されます。Armadillo-400 シリーズでは、initrd をそのままルートファイルシステムとして使用します。

^[5]カーネルで NFS サポートを有効にした場合

2.5.4. ダウンローダー

Armadillo の内蔵フラッシュメモリを書き換えるために、作業用 PC で動作するアプリケーションです。

Linux PC 上で動作するダウンローダーには Hermit-At ダウンローダーと Shoehorn-At があります。Hermit-At ダウンローダーは、ターゲットとなる Armadillo と協調動作を行い、Armadillo の内蔵フラッシュメモリを書き換えることができます。Shoehorn-At は、ブートローダーの復旧に使用します。

Windows PC 上で動作するダウンローダーは、Hermit-At Win32 と呼びます。Hermit-At Win32 は、ターゲットとなる Armadillo の内蔵フラッシュメモリを書き換える機能と、ブートローダーを復旧するための機能を両方有しています。

2.6. ブートモード

Armadillo-400 シリーズは、JP1 の設定によってオンボードフラッシュメモリブートモードと、UART ブートモードを選択することができます。

オンボードフラッシュメモリブートモードでは、フラッシュメモリのブートローダー領域に配置されたブートローダーが起動されます。

標準のブートローダーである Hermit-At では、JP2 の設定によって自動でカーネルをブートするオートブートモードか、各種設定を行うための保守モードを選択することができます。

なお、JP2 の設定によってオートブートモードが選択されている場合でも、起動時に SW1 が押下されている時は Hermit-At のオートブートキャンセル機能により保守モードで起動します。

UART ブートモードは、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、「5.6. ブートローダーを出荷状態に戻す」を参照してください。

Armadillo-400 シリーズの各ジャンパ設定でのブートモードを「表 2.8. ジャンパの設定」に示します。

表 2.8 ジャンパの設定

JP1	JP2	ブートモード
オープン	オープン	オンボードフラッシュメモリブート/オートブートモード
オープン	ショート	オンボードフラッシュメモリブート/保守モード
ショート	-	UART ブートモード

3. 作業の前に

3.1. 準備するもの

Armadillo-400 シリーズを使用した組み込みシステム開発には、以下の機材を準備する必要があります。

作業用 PC	Debian GNU/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 です。
シリアルクロスケーブル	Armadillo と作業用 PC を接続するための、D-Sub9 ピン（メス - メス）のクロス接続用ケーブルがです。
シリアル通信ソフトウェア	Linux では「minicom」、Windows では「Tera Term Pro」などです。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。

また、以下の機材があれば、より効率的に開発を進めることができます。

LAN ケーブル	Armadillo と LAN を経由した通信を行う場合に必要となります。作業用 PC と Armadillo は、スイッチングハブを介して接続してください ^[1] 。
----------	---

3.2. 接続方法

「[図 3.1. Armadillo-440 液晶モデル接続例](#)」もしくは「[図 3.2. Armadillo-420 ベーシックモデル接続例](#)」に示す接続例を参考に、Armadillo と作業用 PC および周辺機器を接続してください。

^[1] Armadillo-400 シリーズは Auto MDIX に対応しているため、作業用 PC と LAN ケーブルで直接接続することもできます。

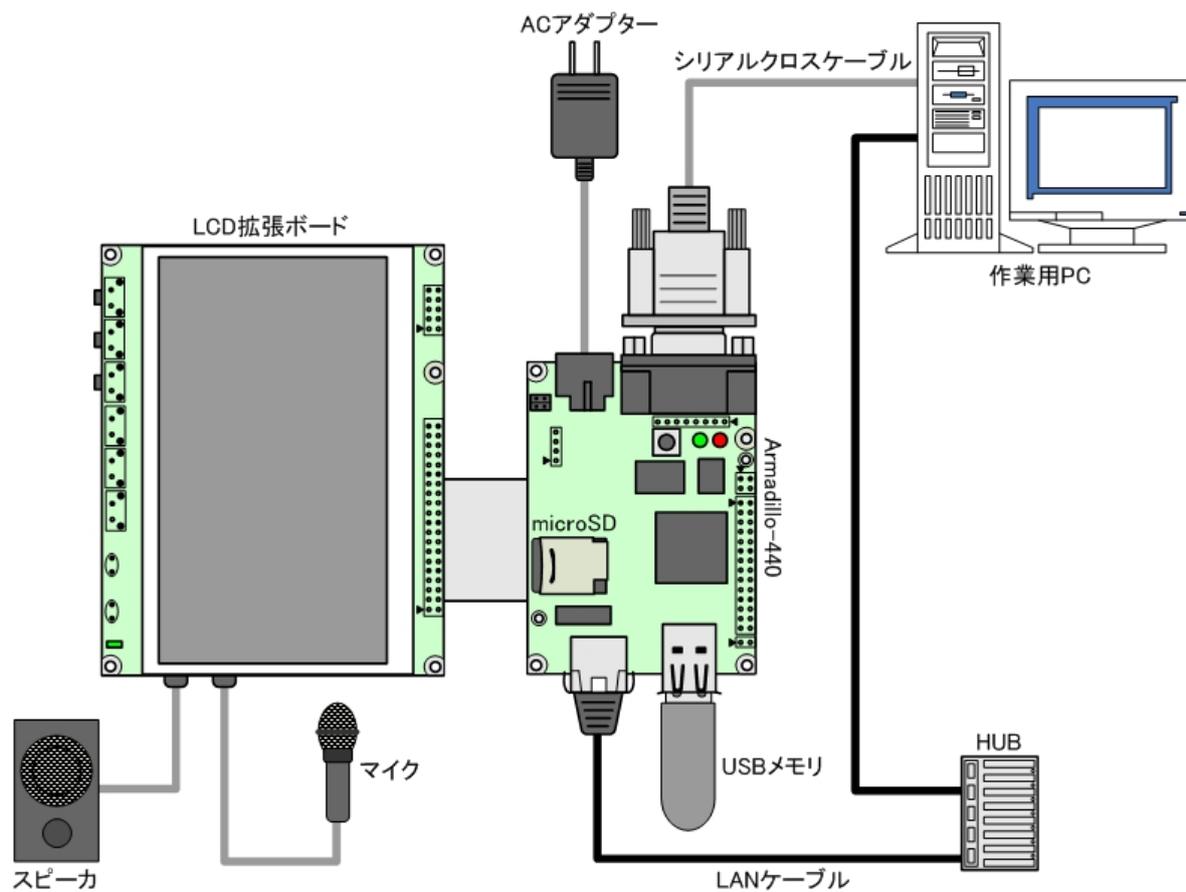


図 3.1 Armadillo-440 液晶モデル接続例

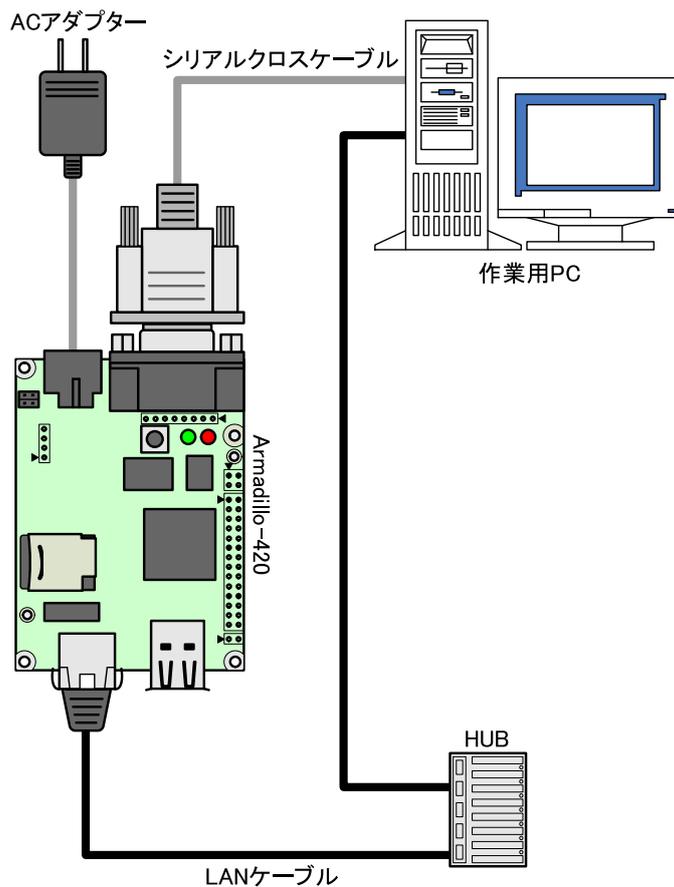


図 3.2 Armadillo-420 ベーシックモデル接続例

3.3. シリアル通信ソフトウェアの設定

作業用の PC から Armadillo のシリアルコンソールに接続する場合、作業用 PC のシリアル通信ソフトウェアの設定を、「表 3.1. シリアル通信設定」のように設定してください。

表 3.1 シリアル通信設定

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

4. 開発環境の準備

本章では、Armadillo のソフトウェア開発を行うための開発環境を、作業用 PC に構築する方法について説明します。

Armadillo-400 シリーズのソフトウェア開発には、Debian 系の Linux 環境^[1](Debian/GNU Linux 5.0 コードネーム lenny を標準とします) が必要です。

作業用 PC が Windows の場合、Windows 上に仮想的な 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は、「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 Install Guide」を参照してください。

ATDE には、標準で基本的な開発環境がインストールされているため、ATDE をお使いになる場合は、「4.1. クロス開発環境パッケージのインストール」と「4.2. Atmark-Dist のビルドに必要なパッケージのインストール」は、行う必要ありません。

4.1. クロス開発環境パッケージのインストール

Debian 系 Linux では、アプリケーションやライブラリの管理には Debian (deb) パッケージを使用します。

クロス開発を行うには、作業用 PC にクロス開発用のツールチェーンのパッケージと、ターゲットアーキテクチャ用のライブラリをクロス開発用に変換したパッケージをインストールする必要があります。

Debian 系 Linux では ARM 用のアーキテクチャとして、arm と armel の 2 つがあります。これは、ABI (Application Binary Interface) の違いによるものです。arm アーキテクチャは OABI を、armel アーキテクチャは EABI を意味します。

Armadillo-400 シリーズでは、EABI を標準の ABI としています。そのため、ターゲットアーキテクチャとして armel のパッケージをインストールする必要があります。

付属 DVD のクロス開発環境ディレクトリ (cross-dev/deb/) にクロス開発環境用のパッケージが用意されています。Armadillo-400 シリーズで開発を行う場合、通常は、armel ディレクトリにある、ARM EABI クロス開発環境をインストールしてください。

インストールは root ユーザーで行う必要があります。deb パッケージをインストールするには、「[図 4.1. インストールコマンド](#)」のようにコマンドを実行します。

```
[PC ~]$ sudo dpkg --install *.deb
```

図 4.1 インストールコマンド

^[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-400 シリーズの開発環境としては、ATDE v3.0 以降を推奨しています。



sudo は引数に与えられたコマンドを、別のユーザーとして実行するコマンドです。上記コマンドでは、root (スーパー)ユーザーとして、**dpkg** コマンドを実行します。

sudo を実行すると、パスワードを要求されることがあります。このとき入力するパスワードは、そのユーザーのパスワードであり、root パスワードではありません。



ご使用の開発環境に既に同一ターゲット向けのアーキテクチャ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既存環境をアンインストールするようにしてください。

4.2. Atmark-Dist のビルドに必要なパッケージのインストール

Armadillo 標準のディストリビューションである、Atmark-Dist をビルドするためには、「表 4.1. 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作業用 PC にインストールされている必要があります。

表 4.1 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
genext2fs	1.4.1-2.1 以降
file	4.26-1 以降
sed	4.1.5-6 以降
perl	5.10.0-19lenny2 以降
bison	1:2.3.dfsg-5 以降
flex	2.5.35-6 以降
libncurses5-dev	5.7+20081213-1 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 4.2. 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。package-name-pattern にはバージョンを表示したいパッケージ名のパターンを指定します。

```
[PC ~]$ dpkg --list [package-name-pattern]
```

図 4.2 インストール情報表示コマンド

4.3. クロス開発用ライブラリパッケージのインストール

Atmark-Dist に含まれないアプリケーションやライブラリをビルドする際に、付属 DVD やダウンロードサイトには用意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、クロス開発用ライブラリパッケージの作成方法およびそのインストール方法を紹介します。

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。取得するパッケージは、アーキテクチャをターゲットに、Debian ディストリビューションのバージョンを開発環境に合わせる必要があります。Armadillo-400 シリーズでは、アーキテクチャは `armel`、Debian ディストリビューションのバージョンは `lenny` (2010年3月現在の `stable`) になります。

例えば、`libjpeg62` の場合「`libjpeg62_[version]_armel.deb`」というパッケージになります。

Debian パッケージは、Debian Packages サイト^[3]から検索して取得することができます。

取得したライブラリパッケージをクロス開発用に変換するには、`dpkg-cross` コマンドを使用します。

```
[PC ~]$ dpkg-cross --build --arch armel libjpeg62_[version]_armel.deb
[PC ~]$ ls
libjpeg62-armel-cross_[version]_all.deb libjpeg62_[version]_armel.deb
```

図 4.3 クロス開発用ライブラリパッケージの作成

作成されたクロス開発用ライブラリパッケージをインストールします。

```
[PC ~]$ sudo dpkg -i libjpeg62-armel-cross_[version]_all.deb
```

図 4.4 クロス開発用ライブラリパッケージのインストール



Debian `lenny` 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。

`apt-cross` コマンドを使用すると、上記の一連の作業を 1 つのコマンドで行うことができます。

```
[PC ~]$ apt-cross --arch armel --suite lenny --install libjpeg62
```

図 4.5 apt-cross コマンド

`--arch` オプションにはアーキテクチャを、`--suite` オプションには Debian ディストリビューションのバージョンをそれぞれ指定し、`--install` オプションで取得/変換したパッケージをインストールすることを指定します。最後の引数には、パッケージ名を指定します。

^[3]<http://www.debian.org/distrib/packages>

5. フラッシュメモリの書き換え方法

本章では、Armadillo のオンボード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。

フラッシュメモリの書き換え方法には、大きくわけて 2 種類の方法があります。

1. 作業用 PC で動作するダウンローダーから、ターゲットとなる Armadillo にイメージを送信して、フラッシュを書き換える方法
2. ターゲットとなる Armadillo 自身で、リモートサーバーからイメージファイルを取得してフラッシュを書き換える方法

まず、「5.3. ダウンローダーを使用してフラッシュメモリを書き換える」で、1. の方法について説明します。次に、「5.4. tftpd を使用してフラッシュメモリを書き換える」および、「5.5. netflash を使用してフラッシュメモリを書き換える」で 2. の方法について説明します。



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。書き換えの際は次の点に注意してください。

- ・ 書き換え中に Armadillo の電源を切らない
- ・ 書き換え中に Armadillo と開発用 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

ブートローダーの書き換えに失敗するなどして起動できなくなった場合は、「5.6. ブートローダーを出荷状態に戻す」の手順に従ってブートローダーを復旧してください。

5.1. 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、リージョン（領域）名で指定することができます。各リージョンに指定するイメージファイルは、「表 5.1. 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 5.1 リージョン名と対応するイメージファイル

製品	リージョン名	ファイル名
Armadillo-440 液晶モデル	bootloader	loader-armadillo4x0-[version].bin
	kernel	linux-a400-[version].bin.gz
	userland	romfs-a440-[version].img.gz
Armadillo-420 ベーシックモデル	bootloader	loader-armadillo4x0-[version].bin
	kernel	linux-a400-[version].bin.gz
	userland	romfs-a420-[version].img.gz



Armadillo-420 と Armadillo-440 は、ブートローダー及びカーネルに共通のイメージファイルを使用します。

5.2. ダウンローダーのインストール

作業用 PC にダウンローダーをインストールします。

ダウンローダーには、「表 5.2. ダウンローダー一覧」に示すように複数の種類があります。

表 5.2 ダウンローダー一覧

ダウンローダー	OS タイプ	説明
Hermit-At ダウンローダー	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 Win32	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、ダウンローダー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。

5.2.1. 作業用 PC が Linux の場合

付属 DVD のダウンローダーディレクトリ (downloader/) 以下の deb パッケージディレクトリ (deb/) よりパッケージファイルを取得し、インストールします。

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

図 5.1 ダウンローダーのインストール (Linux)

5.2.2. 作業用 PC が Windows の場合

付属 DVD のダウンローダーディレクトリ (downloader/) 以下の win32 ディレクトリ (win32/) にある hermit-at-win_[version].zip を任意のフォルダに展開します。

5.3. ダウンローダー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 ダウンローダーおよび Hermit-AT Win32 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。

Hermit-At ダウンローダーおよび Hermit-AT Win32 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。

5.3.1. 準備

「表 2.8. ジャンパの設定」を参照しジャンパを適切に設定したあと Armadillo に電源を投入し、保守モードで起動してください。

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てください。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。

5.3.2. 作業用 PC が Linux の場合

作業用 PC が Linux の場合、**hermit-at** コマンドを使用し、「図 5.2. 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-at** コマンドのサブコマンドの 1 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」という指示になります。

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

図 5.2 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 5.3. 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^[1]。

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

図 5.3 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 5.4. ダウンロードコマンド (アンプロテクト)」のように `--force-locked` オプションを使用して、プロテクトを解除してください^[1]。

```
[PC ~]$ hermit download --input-file loader-armadillo4x0-[version].bin  
--region bootloader --force-locked
```

図 5.4 ダウンロードコマンド (アンプロテクト)



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「5.6. 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。

^[1]書面の都合上折り返して表記しています。実際にはコマンドは 1 行で入力します。

5.3.3. 作業用 PC が Windows の場合

作業用 PC が Windows の場合、hermit.exe を実行すると、「図 5.5. Hermit-At Win32 : Download ウィンドウ」が表示されます。

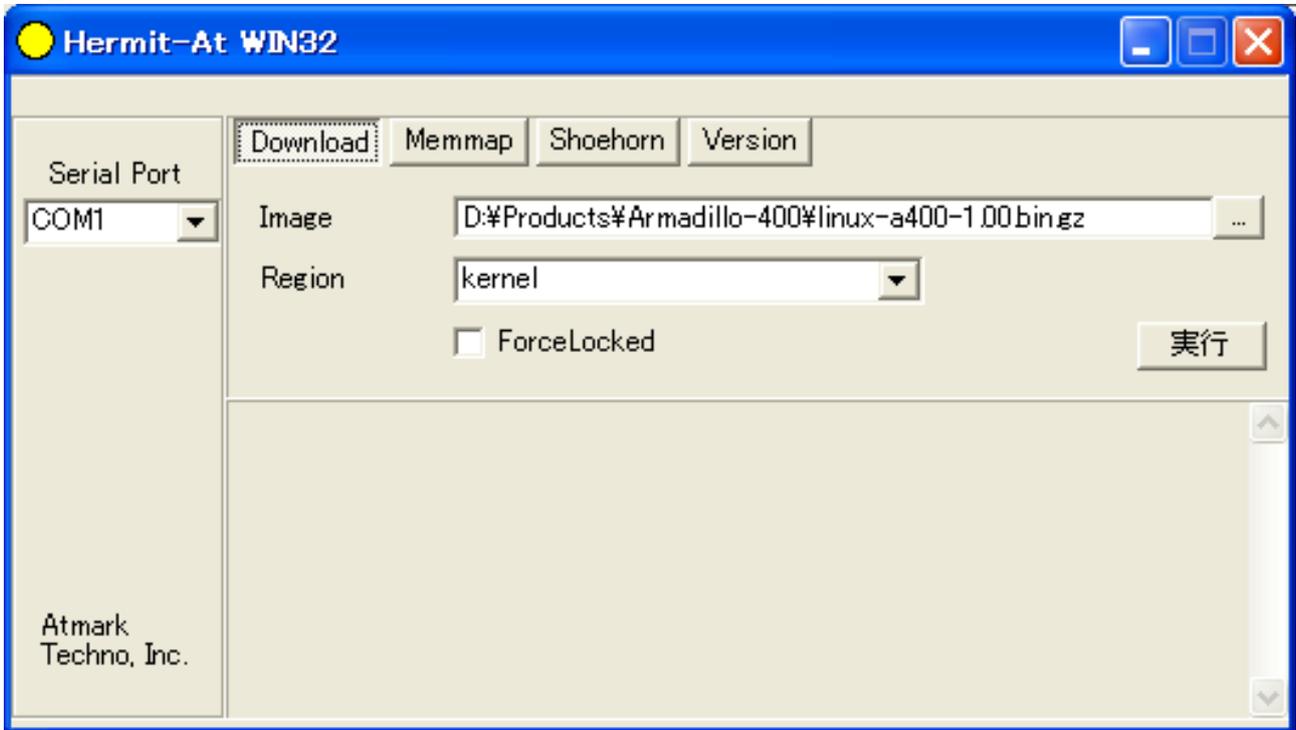


図 5.5 Hermit-At Win32 : Download ウィンドウ

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、直接ポート名を入力してください。

Image には書き込むファイルを、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 をチェックする必要があります。

すべて設定してから実行ボタンをクリックすると、書き込みが開始されます。書き込み中は、「図 5.6. Hermit-At Win32 : download ダイアログ」が表示され、ダウンロードの設定と進捗状況を確認することができます。

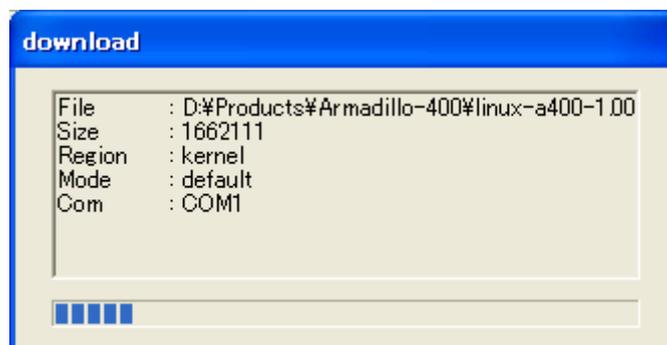


図 5.6 Hermit-At Win32 : download ダイアログ

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。

5.4. tftpd を使用してフラッシュメモリを書き換える

ここからは、Armadillo 自身でリモートサーバーからイメージファイルを取得してフラッシュメモリを書き換える方法について説明します。

Hermit-At ブートローダーの tftpd 機能を使用することで、ダウンロードを使用し書き込むよりも高速にフラッシュメモリを書き換えることができます。

tftpd 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、自分自身のフラッシュメモリを書き換えることができる機能です。



ATDE v3.0 以降では、標準で TFTP サーバー (atftpd) が動作していません。/var/lib/tftpboot/ ディレクトリにファイルを置くことで、TFTP によるアクセスが可能になります。

tftpd 機能を使用するには、ターゲットとなる Armadillo のジャンパを設定し、保守モードで起動してください。

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 5.7. tftpd 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 リージョンに書き込む例です。

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

図 5.7 tftpd コマンド例

書き込み対象には、ブートローダー、カーネル、ユーザーランドそれぞれのリージョンを指定することができます。書き込むリージョンとオプションの対応を、「表 5.3. リージョンとオプションの対応」に示します。

表 5.3 リージョンとオプションの対応

リージョン	オプション
ブートローダー	--bootloader
カーネル	--kernel
ユーザーランド	--userland

5.5. netflash を使用してフラッシュメモリを書き換える

Linux が動作している状態では、Linux アプリケーションの netflash を使用することでフラッシュメモリを書き換えることができます。

netflash は、接続されているネットワーク内にある 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、自分自身のフラッシュメモリを書き換えることができるコマンドです。



ATDE v3.0 以降では、標準で HTTP サーバー (lighttpd) が動作しています。/var/www/ ディレクトリにファイルを置くことで、HTTP によるアクセスが可能になります。

netflash を使用するには、Armadillo にログインし「図 5.8. 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 5.8 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 5.4. 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 で詳細を確認することができます。

表 5.4 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
ユーザーランド	/dev/flash/userland



bootloader リージョンは標準状態ではリードオンリー属性となっているため、**netflash** で書き換えることはできません。

5.6. ブートローダーを出荷状態に戻す

何らかの理由でブートローダー領域の内容が破壊されブートローダーが起動しなくなった場合、UART ブートモードを使用することでブートローダーを出荷状態に戻すことができます。

5.6.1. 準備

Armadillo のジャンパを、「表 2.8. ジャンパの設定」を参照し、UART ブートモードに設定してください。この時点では Armadillo は起動させないでください。

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。

5.6.2. 作業用 PC が Linux の場合

「図 5.9. shoehorn コマンド例」のようにコマンド^[2]を実行してから、Armadillo に電源を投入し、起動させてください。

^[2]書面の都合上折り返して表記しています。実際にはコマンドは 1 行で入力します。

```
[PC ~]$ shoehorn --boot --target armadillo4x0
--initrd /dev/null
--kernel /usr/lib/hermit/loader-armadillo4x0-boot-[version].bin
--loader /usr/lib/shoehorn/shoehorn-armadillo4x0.bin
--initfile /usr/lib/shoehorn/shoehorn-armadillo4x0.init
--postfile /usr/lib/shoehorn/shoehorn-armadillo4x0.post
```

図 5.9 shoehorn コマンド例

実行すると、「図 5.10. shoehorn コマンドログ」のようにログが表示されます。

```
/usr/lib/shoehorn/shoehorn-armadillo4x0.bin: 1272 bytes (2048 bytes buffer)
/usr/lib/hermit/loader-armadillo4x0-boot-v2.0.0.bin: 45896 bytes (45896 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
- Initializing for Mobile-DDR
Pinging loader
Detecting DRAM
- 32 bits wide
- start: 0x80000000 size: 0x04000000 last: 0x83ffffff
Total DRAM: 65536kB
Loading /usr/lib/hermit/loader-armadillo4x0-boot-v2.0.0.bin:
- start: 0x83000000 size: 0x0000b348 last: 0x8300b347
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 5.10 shoehorn コマンドログ

shoehorn コマンドが成功すると、ターゲットの Armadillo 上で Hermit At ブートローダーの UART ブートモード版 (loader-armadillo4x0-boot-[version].bin) が動作している状態になります。この状態で、「5.3. ダウンローダー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。

5.6.3. 作業用 PC が Windows の場合

hermit.exe を実行し Shoehorn ボタンをクリックすると、「図 5.11. Hermit-At Win32 : Shoehorn ウィンドウ」が表示されます。

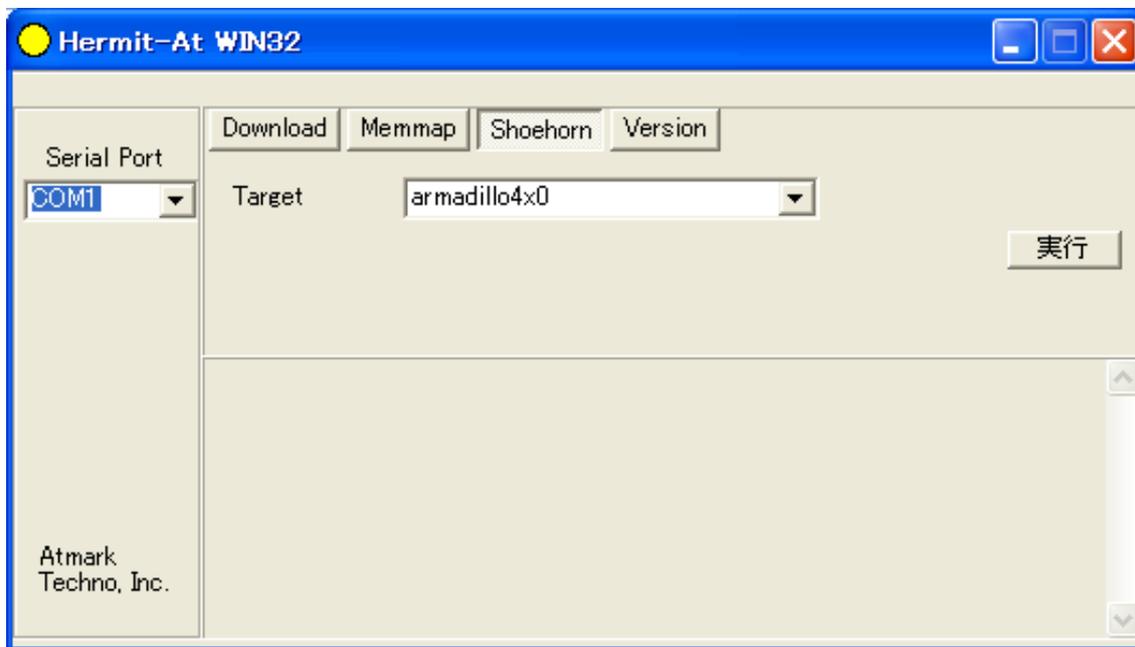


図 5.11 Hermit-At Win32 : Shoehorn ウィンドウ

Target に armadillo4x0 を選択して実行ボタンをクリックします。



図 5.12 Hermit-At Win32 : shoehorn ダイアログ

ダイアログが表示されます。Armadillo に電源を投入して起動してください。ダウンロードするための準備が完了すると自動的にダイアログはクローズされます。

この状態で、「5.3. ダウンローダー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。

6.ビルド

この章では、ソースコードから出荷イメージと同じイメージを作成する手順と、それをカスタマイズする方法について説明します。

開発環境を構築してない場合は、「4. 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。

より詳細な開発手順については「Atmark-Dist 開発者ガイド」を参照してください。

以下の例では、ホームディレクトリ (~/) 以下のディレクトリで作業を行います。



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 ユーザーではなく**一般ユーザー**で行ってください。

6.1. 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、ソースコードベースのディストリビューションである Atmark-Dist と、Linux カーネルのソースコードから、ユーザーランドとカーネルのイメージを作成する手順について説明します。

6.1.1. ソースコードの準備

まず最初に、ソースコードを取得します。付属 DVD の Atmark-Dist ソースアーカイブディレクトリ (source/dist) にある atmark-dist-[version].tar.gz と カーネルソースアーカイブディレクトリ (source/kernel) にある linux-[version].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 6.1. 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 atmark-dist-[version] atmark-dist
[PC ~]$ cd atmark-dist
[PC ~/atmark-dist]$ ln -s ../linux-[version] linux-2.6.x
```

図 6.1 ソースコード準備

6.1.2. デフォルトコンフィギュレーションの適用

ターゲットとなる Armadillo 用に Atmark-Dist をコンフィギュレーションします。Linux カーネルは Atmark-Dist の管理下にあるため、Atmark-Dist を適切に設定すれば、カーネルのコンフィギュレーションも適切に行われるようになっていきます。

以下の例のようにコマンドを入力し、コンフィギュレーションを開始します^[1]。

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

最初に、使用するボードのベンダー名を尋ねられます。「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尋ねられます。「表 6.1. 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 6.1 プロダクト名一覧

製品	プロダクト名	備考
Armadillo-440 液晶モデル	Armadillo-440	出荷時イメージ
Armadillo-420 ベーシックモデル	Armadillo-420	出荷時イメージ

以下は、Armadillo-440 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-440, Armadillo-500, Armadillo-500-FX.base, Armadillo-500-FX.dev,
Armadillo-9, Armadillo-9.PCMCIA, SUZAKU-V.SZ310, SUZAKU-V.SZ310-SID, SUZAKU-
V.SZ310-SIL, SUZAKU-V.SZ310-SIL-GPIO, SUZAKU-V.SZ410, SUZAKU-V.SZ410-SID, SUZAKU-
```

^[1]ここでは、CUI でのコンフィギュレーション方法を説明しています。Atmark Dist ではメニュー形式でのコンフィギュレーションも可能です。詳しくは「Atmark-Dist 開発者ガイド」をご参照ください。

```
V.SZ410-SIL, SUZAKU-V.SZ410-SIL-GPIO, SUZAKU-V.SZ410-SIV) [Armadillo-210.Base]
(NEW) Armadillo-440
```

ビルドする開発環境を尋ねられます。「default」と入力してください。Armadillo-400 シリーズでは、「default」と入力すると、armel (EABI)の開発環境が選択されます。

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

ビルドする libc を指定します。Armadillo-400 シリーズでは「None」を選択してください。「None」を選択することで、開発環境にインストールされているビルド済みの libc (glibc) を使用します。

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

既定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
```

設定の入力が完了するとビルドシステムに反映されてコンフィギュレーションが行われ、プロンプトに戻ります。

6.1.3. ビルド

ビルドするには、atmark-dist ディレクトリで「[図 6.2. Atmark-Dist のビルド](#)」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 ディレクトリにイメージファイルが作成されます。romfs.img がユーザーランドの、linux.bin がカーネルのイメージです。romfs.img.gz と linux.bin.gz はそれぞれのファイルを圧縮したものです。

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

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

図 6.2 Atmark-Dist のビルド

作成されたイメージは、「5. フラッシュメモリの書き換え方法」の手順に従って、ターゲットの Armadillo に書き込むことができます。通常は、フラッシュメモリの使用容量を少なくするため、圧縮イメージを書き込みます。

6.1.4. イメージをカスタマイズする

Atmark-Dist には、様々なアプリケーションやライブラリが含まれており、コンフィギュレーションによってそれらをイメージに含めたり、イメージから削除することができます。また、カーネルのコンフィギュレーションの変更を行うこともできます。

Atmark-Dist のコンフィギュレーションを変更するには、**make menuconfig** コマンドを使用します。

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

図 6.3 Atmark-Dist のコンフィギュレーション

make menuconfig を実行すると、「図 6.4. menuconfig: Main Menu」に示す Main Menu 画面が表示されます。

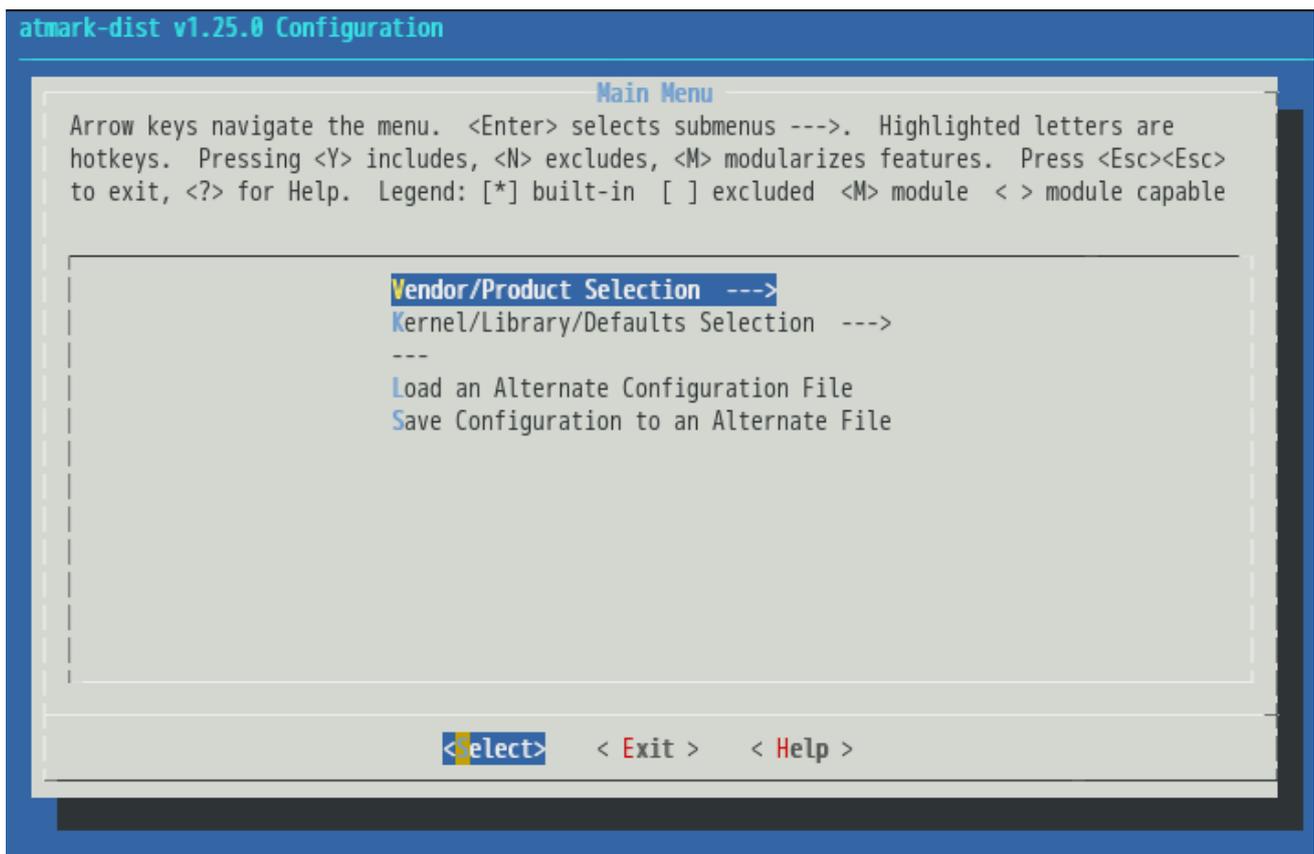


図 6.4 menuconfig: Main Menu

キーボードの上下キーでフォーカスを **Kernel/Library/Defaults Selection --->** に合わせ、Enter キーを押すと、Kernel/Library/Defaults Selection 画面が表示されます。

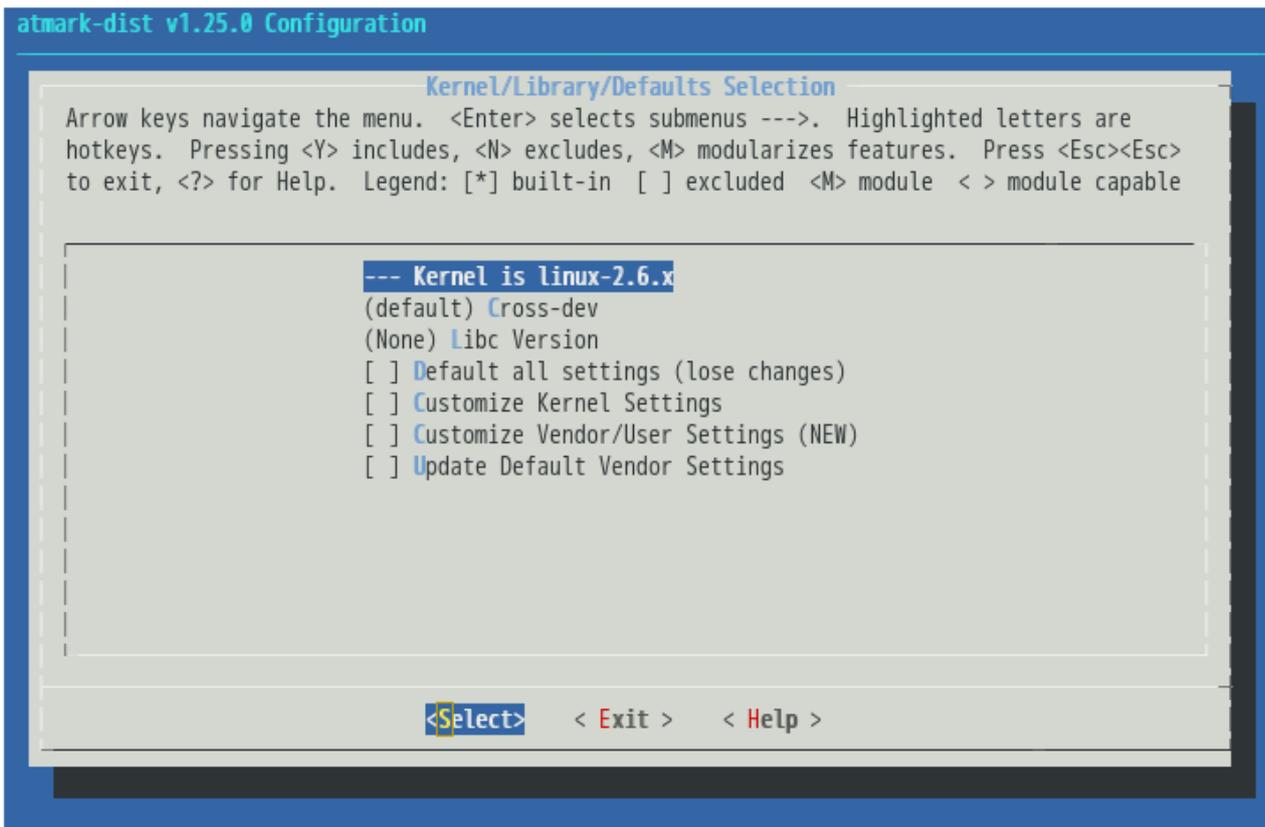


図 6.5 menuconfig: Kernel/Library/Defaults Selection

カーネルコンフィギュレーションを変更するには、**Customize Kernel Settings** を選択してください。また、ユーザーランドに含めるアプリケーションやライブラリを変更するには、**Customize Vendor/User Settings** を選択してください。ここでいう、「選択する」とは、上下キーで選択したい項目にフォーカスを合わせ、スペースキーを一度押し、*印を付けることを言います。

項目を選択したら、キーボードの左右キーで **Exit** にフォーカスを合わせ、Enter キーを押してください。そうすることで、Kernel/Library/Defaults Selection 画面を抜け、Main Menu 画面へ戻ります。

Main Menu 画面でも、**Exit** にフォーカスを合わせ、Enter キーを押してください。すると、**Do you wish to save your new kernel configuration?**と表示されますので、**Yes** にフォーカスを合わせたまま、Enter キーを押してください。

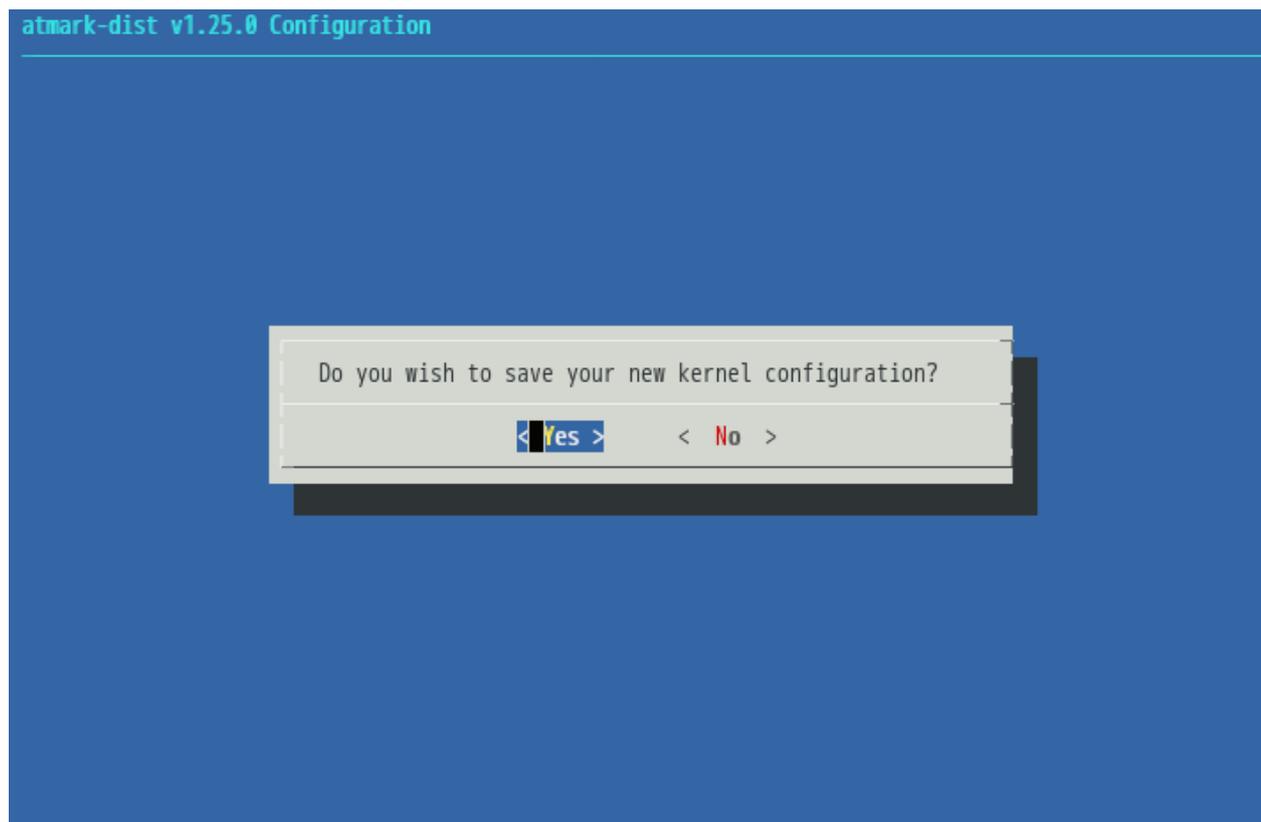


図 6.6 menuconfig: Do you wish to save your new kernel configuration?

Customize Kernel Settings を選択していた場合は、Linux Kernel Configuration 画面が表示されます。ここで、カーネルコンフィギュレーションを変更することができます。コンフィギュレーションが完了したら、Linux Kernel Configuration 画面で **Exit** にフォーカスを当てて Enter キーを押し、画面を抜けてください。

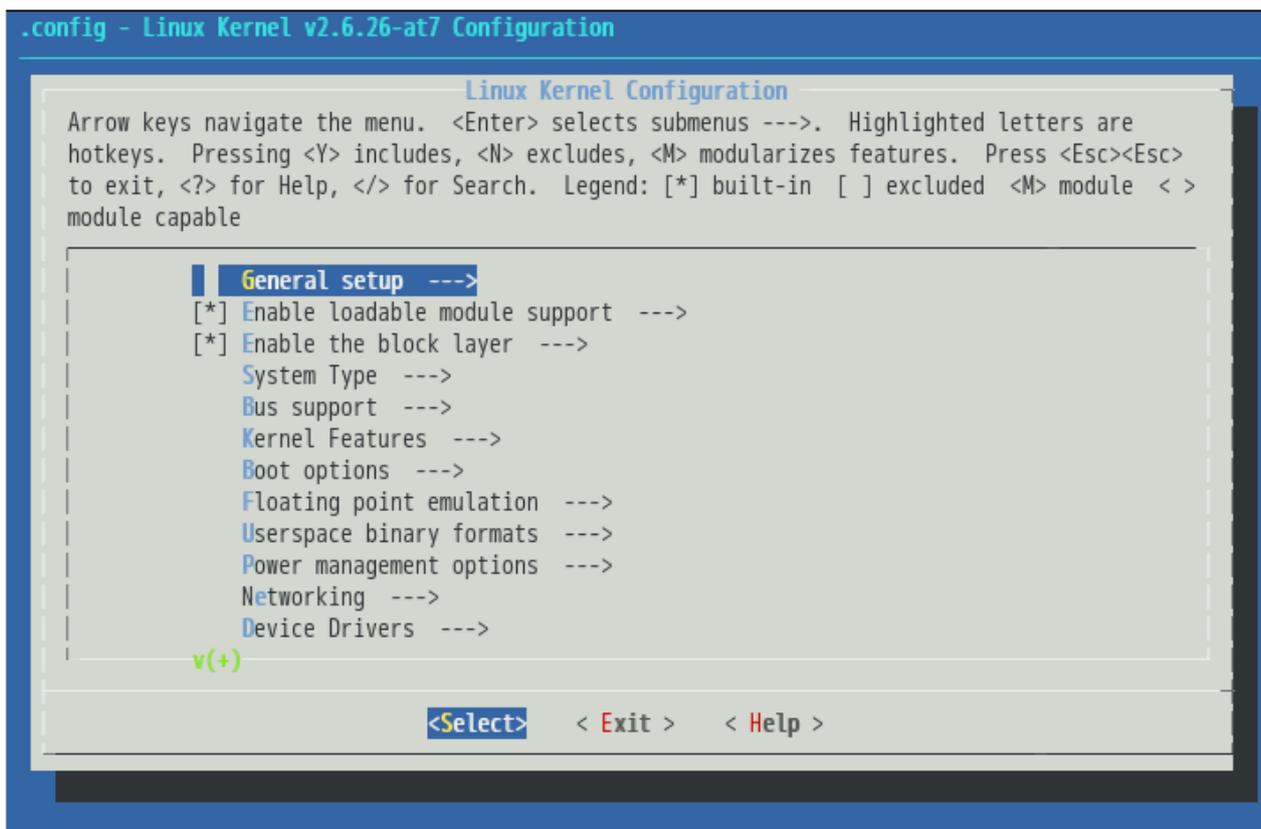


図 6.7 menuconfig: Linux Kernel Configuration

Customize Vendor/User Settings を選択していた場合は、Userland Configuration 画面が表示されます。ここで、ユーザーランドに含めるアプリケーションやライブラリを選択することができます。選択が完了したら、Userland Configuration 画面で **Exit** にフォーカスを当てて Enter キーを押し、画面を抜けてください。

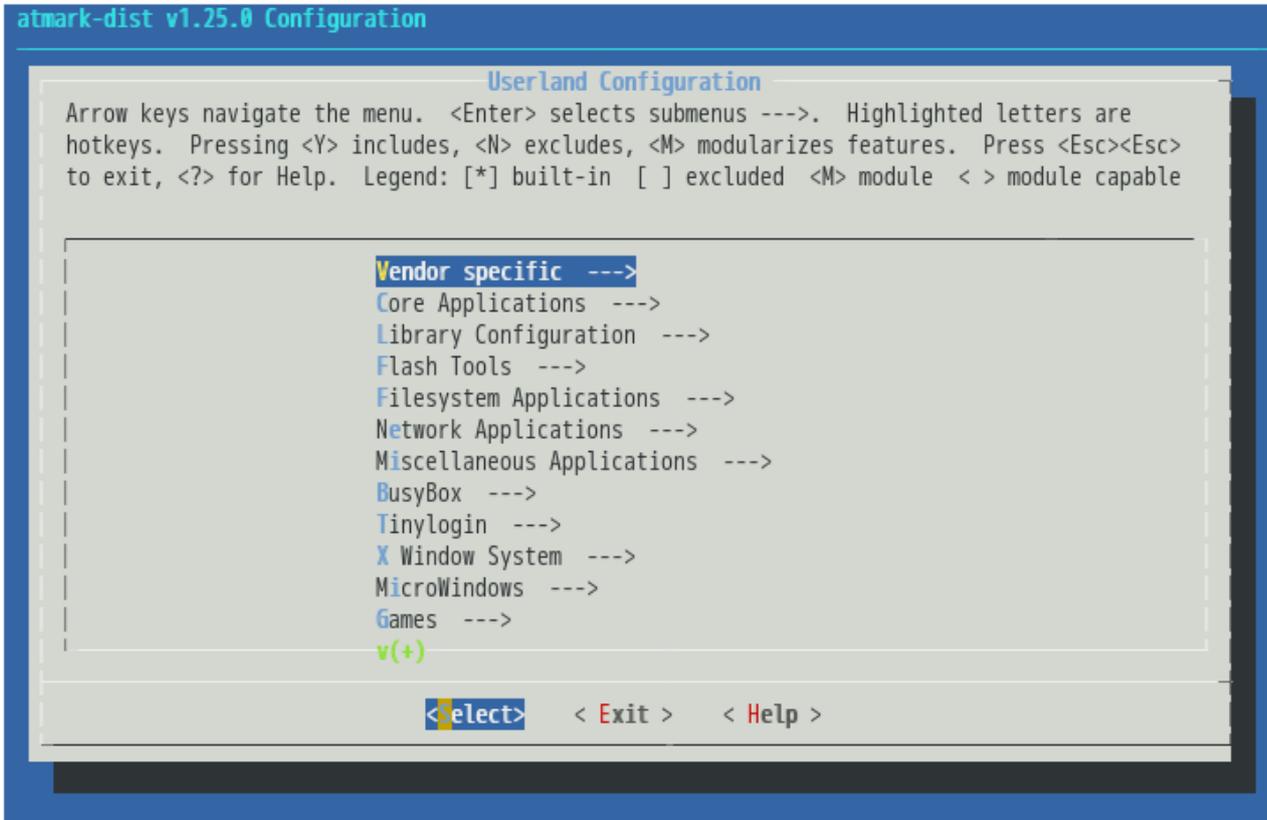


図 6.8 menuconfig: Userland Configuration

再び、**Do you wish to save your new kernel configuration?**と表示されますので、**Yes** にフォーカスを合わせたまま、Enter キーを押してください。

以上で、コンフィギュレーションの変更は完了です。

make menuconfig を使用したコンフィギュレーション方法の詳細については、「Atmark-Dist 開発者ガイド」を参照してください。

コンフィギュレーションを行ったあとは、「6.1.3. ビルド」の手順と同様に、**make** コマンドを実行すると、コンフィギュレーション結果を反映したイメージが作成されます。

6.1.5. ユーザーランドイメージにアプリケーションを追加する

ここでは、「6.1. カーネルイメージとユーザーランドイメージのビルド」で作成したユーザーランドに、自作のアプリケーションなど Atmark-Dist には含まれないファイルを追加する方法について説明します。

自作アプリケーションは、Out-Of-Tree コンパイル^[2]で作成し、~/sample/hello にあると仮定とします。

^[2]Out-Of-Tree コンパイルに関しては「Atmark-Dist 開発者ガイド」を参照してください

Atmark-Dist では、romfs ディレクトリにユーザーランドイメージに含めるファイルが置かれています。ここに自作アプリケーションを追加し、**make image** コマンドを実行することで、自作アプリケーションを含んだユーザーランドイメージを作成することができます。

```
[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 6.9 ユーザーランドイメージのカスタマイズ

作成されたユーザーランドイメージの /bin ディレクトリには、hello がインストールされています。

6.2. ブートローダーイメージのビルド

ここでは、Hermit-At ブートローダーのイメージをソースコードからビルドする手順について説明します。ソースコードのバージョンは、v2.0.0 以降を使用します。

6.2.1. ソースコードの準備

付属 DVD 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 6.10. Hermit-At ソースアーカイブの展開」のように作業してください。

```
[PC ~]$ tar zxvf hermit-at-[version]-source.tar.gz
[PC ~]$ ln -s hermit-at-[version] hermit-at
[PC ~]$ cd hermit-at
[PC ~/hermit-at]$
```

図 6.10 Hermit-At ソースアーカイブの展開

6.2.2. ビルド

Hermit-At v2.0.0 以降では、製品ごとに標準の設定が defconfig として用意されています。出荷イメージと同じイメージをビルドするには、「図 6.11. Hermit-At ビルド例」のようにコマンドを実行してください。

```
[PC ~/hermit-at]$ make armadillo4x0_defconfig
[PC ~/hermit-at]$ make

[PC ~/hermit-at]$ ls src/target/armadillo4x0/*.bin
loader-armadillo4x0-[version].bin
```

図 6.11 Hermit-At ビルド例

loader-armadillo4x0-[version].bin が作成されたブートローダーイメージです。

Hermit-At v2.0.0 以降では、Atmark-Dist と同様に **make menuconfig** でのコンフィギュレーションに対応しています。コンフィギュレーションを変更することにより、コマンドの追加/削除や、動作の変更を行うことができます。

7.カーネル/ユーザーランドの配置

Armadillo-400 シリーズでは、標準ではカーネルおよびユーザーランドイメージはフラッシュメモリに配置されており、ブートローダーによってカーネルのブート前に RAM 上に展開されます。

Armadillo-400 シリーズでは、フラッシュメモリ以外の場所にもカーネルおよびユーザーランドを配置することができます。

本章では、イメージの配置方法と、イメージの配置場所を変えたときに必要となるブートオプションの設定方法について説明します。

7.1. TFTP サーバーに配置する

Hermit-At ブートローダーは、TFTP サーバー上に配置されたカーネルまたはユーザーランドのイメージを取得し RAM 上に展開したあと起動する、tftpbboot 機能を有しています。

tftpbboot 機能を使用すると、フラッシュメモリにイメージを書くことなく起動できるため、開発の初期段階などイメージの更新が頻繁に行われる際に、効率よく作業することができます。

7.1.1. ファイルの配置

TFTP サーバーのルートディレクトリに、カーネルイメージとユーザーランドイメージを配置してください。



ATDE v3.0 以降では、標準状態で TFTP サーバー (atftpd) が動作しています。/var/lib/tftpbboot ディレクトリにファイルを置くことで、TFTP によるアクセスが可能になります。

7.1.2. ブートオプション

ターゲットとなる Armadillo のジャンパを適切に設定し、保守モードで起動してください。

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^[1]を入力します。

```
hermit> setbootdevice tftp [Armadillo IP adress] [tftp server IP adress]
        --kernel=kernel_image_file_name --userland=userland_image_file_name
```

図 7.1 tftpbboot コマンド

カーネルとユーザーランドのイメージは、どちらか一方だけ、もしくは両方指定できます。

^[1]書面の都合上折り返して表記しています。実際にはコマンドは 1 行で入力します。

TFTP サーバーの IP アドレスが 192.168.10.1、Armadillo の IP アドレスが 192.168.10.10 で、カーネルイメージのファイル名が `linux.bin.gz`、ユーザーランドのイメージのファイル名が `romfs.img.gz` の場合、以下ようになります^[2]。

```
hermit> setbootdevice tftp 192.168.10.10 192.168.10.1
        --kernel=linux.bin.gz --userland=romfs.img.gz
```

図 7.2 tftpboot コマンド例

`setbootdevice` コマンドでブートデバイスを TFTP サーバーに設定した場合、設定は保存され、起動時に毎回カーネルもしくはユーザーランドイメージを TFTP サーバーから取得するようになります。

7.2. ストレージに配置する

Armadillo-400 シリーズでは、カーネルイメージは microSD に、ユーザーランドのルートファイルシステムは microSD または USB メモリにも配置することができます。

ここでは、microSD にカーネルとユーザーランド両方のイメージを配置する手順を説明します。

パーティションは以下のようになります。

表 7.1 microSD システムパーティション例

パーティション	ファイルシステム	サイズ	説明
/dev/mmcbk0p1	ext2	32MB	起動パーティション、カーネルイメージを配置します
/dev/mmcbk0p2	ext3	残り	ルートファイルシステムを配置します

7.2.1. パーティション分割

最初に、microSD を 2 つのパーティションに分けます。

microSD をスロットに挿入し^[3]、「図 7.3. パーティション分割手順」のようにしてパーティションを構成してください。

^[2]書面の都合上折り返して表記しています。実際にはコマンドは 1 行で入力します。

^[3]Armadillo-400 シリーズの microSD スロットは、ロック式になっています。microSD カードの着脱方法に関しては「Armadillo-400 シリーズ ハードウェアマニュアル」をご参照ください。

```
[armadillo ~]# fdisk /dev/mmcblk0
The number of cylinders for this disk is set to 31104.
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
Selected partition 1

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4) p
Partition number (1-4): 1
First cylinder (1-31104, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-31104, default 31104): +32M

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4) p
Partition number (1-4): 2
First cylinder (979-31104, default 979):
Using default value 979
Last cylinder or +size or +sizeM or +sizeK (979-31104, default 31104):
Using default value 31104

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

Calling ioctl() to re-read partition table.
mmcblk0: p1 p2
mmcblk0: p1 p2
Syncing disks.

[armadillo ~]#
```

図 7.3 パーティション分割手順



使用する microSD カードによって仕様が異なるため、表示されるシリンダ数は手順通りとはならない場合があります。

7.2.2. ファイルシステムの作成

次に、「図 7.4. ファイルシステム作成手順」のようにして、各パーティションにファイルシステムを作成します。起動パーティションは EXT2 ファイルシステムに、ルートファイルシステムパーティションは EXT3 ファイルシステムにします。



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、`mke2fs` のオプションとして、必ず「`-O none`」を指定する必要があります。

```
[armadillo ~]# mke2fs -O none /dev/mmcblk0p1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
7840 inodes, 31288 blocks
1564 blocks (4%) reserved for the super user
First data block=1
4 block groups
8192 blocks per group, 8192 fragments per group
1960 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 39 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
[armadillo ~]# mke2fs -j /dev/mmcblk0p2
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
120576 inodes, 241008 blocks
12050 blocks (4%) reserved for the super user
First data block=0
8 block groups
32768 blocks per group, 32768 fragments per group
15072 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 27 mounts or
180.00 days, whichever comes first.  Use tune2fs -c or -i to override.
```

図 7.4 ファイルシステム作成手順

7.2.3. カーネルイメージの配置

microSD 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 7.2 カーネルイメージのダウンロード先 URL

製品	URL
Armadillo-420	http://download.atmark-techno.com/armadillo-420/image/linux-a400-[version].bin.gz
Armadillo-440	http://download.atmark-techno.com/armadillo-440/image/linux-a400-[version].bin.gz

以下に Armadillo-440 での配置例を示します。

```
[armadillo ~]# mount -t ext2 /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir /mnt/boot
[armadillo ~]# cd /mnt/boot
[armadillo /mnt/boot]# wget http://download.atmark-techno.com/armadillo-440/image/
linux-a400-[version].bin.gz
[armadillo /mnt/boot]# mv linux-a440-[version].bin.gz /mnt/boot/linux.bin.gz
[armadillo /mnt/boot]# cd
[armadillo ~]# umount /mnt
```

図 7.5 カーネルイメージの配置

7.2.4. ルートファイルシステムの構築

ここでは、microSD 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。

ルートファイルシステムは、Debian/GNU Linux もしくは Atmark-Dist で作成したルートファイルシステムを使用できます。

7.2.4.1. Debian GNU/Linux を構築する

Debian GNU/Linux を構築する場合、付属 DVD 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 GNU/Linux 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 GNU/Linux を構築する場合は、パーティションの空き容量が約 1GB 以上必要です。

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 7.3 Debian アーカイブのダウンロード先 URL

製品	URL
Armadillo-420/440 共通	http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-#.tgz ^[1]
	http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-a4x0.tgz

[1]注：#の部分は 1～5 まで

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt/
[armadillo ~]# mkdir tmp
[armadillo ~]# mount -t ramfs ramfs tmp
[armadillo ~]# cd tmp
[armadillo ~/tmp]# for N in 1 2 3 4 5 a4x0; do
> wget http://download.atmark-techno.com/armadillo-4x0/debian/debian-lenny-armel-$
{N}.tgz;
> gzip -cd debian-lenny-armel-${N}.tgz | (cd /mnt; tar xf -);
> sync;
> rm -f debian-lenny-armel-${N}.tgz;
> done
[armadillo ~/tmp]# cd
[armadillo ~]# umount tmp
[armadillo ~]# rmdir tmp
[armadillo ~]# umount /mnt
```

図 7.6 Debian アーカイブによるルートファイルシステムの構築例

7.2.4.2. Atmark-Dist イメージから構築する

Atmark-Dist で作成されるルートファイルシステムと同じルートファイルシステムを microSD 上に構築する方法を説明します。Debian を構築する場合に比べ、容量の少ない microSD ヘ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 で作成されるルートファイルシステムの initrd 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 7.4 Atmark-Dist イメージのダウンロード先 URL

製品	URL
Armadillo-420	http://download.atmark-techno.com/armadillo-420/image/romfs-a420- [version].img.gz
Armadillo-440	http://download.atmark-techno.com/armadillo-440/image/romfs-a440- [version].img.gz

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt/
[armadillo ~]# mkdir tmp
[armadillo ~]# mkdir romfs
[armadillo ~]# mount -t ramfs ramfs tmp
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-440/image/romfs-
a440-[version].img.gz -P tmp
[armadillo ~]# gzip -d tmp/romfs-a440-[version].img.gz
[armadillo ~]# mount -o loop tmp/romfs-a440-[version].img romfs/
[armadillo ~]# (cd romfs/; tar cf - *) | (cd /mnt; tar xf -)
[armadillo ~]# sync
[armadillo ~]# umount romfs
[armadillo ~]# rmdir romfs
[armadillo ~]# umount tmp
[armadillo ~]# rmdir tmp
[armadillo ~]# umount /mnt
```

図 7.7 Atmark-Dist イメージによるルートファイルシステムの構築例

7.2.5. カーネルパラメータの設定

microSD に配置したカーネルから起動するためには、Hermit-At を使用してカーネルパラメータを設定する必要があります。

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。

microSD のパーティション 1 に配置したカーネルイメージで起動するためには、「図 7.8. 起動デバイスの指定」を実行してください。

```
hermit> setbootdevice mmcblk0p1
```

図 7.8 起動デバイスの指定

ルートファイルシステムを microSD のパーティション 2 にする場合は、「図 7.9. 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=ttymxc1 root=/dev/mmcblk0p2 noinitrd rootwait
```

図 7.9 ルートファイルシステム指定例

7.3. 設定を元に戻す

ブートデバイスを初期状態のフラッシュメモリに戻す場合には、「図 7.10. 起動デバイスにフラッシュメモリを指定する」のコマンドを入力してください。

```
hermit> setbootdevice flash
```

図 7.10 起動デバイスにフラッシュメモリを指定する

また、setenv で設定した起動オプションを初期状態の設定に戻すには、clearenv コマンドを実行してください。

```
hermit> clearenv
```

図 7.11 `clearenv` で起動オプションを初期状態に戻す

8. Linux カーネルデバイスドライバー仕様

本章では、Armadillo-400 シリーズに固有な Linux カーネルのデバイスドライバーの仕様について説明します。

Armadillo-400 シリーズでは、カーネルコンフィギュレーションを変更することにより、標準で有効になっているもの以外の様々な機能を使用することができます。

カーネルコンフィギュレーションを変更する方法は、「6.1.4. イメージをカスタマイズする」を参照してください。

Armadillo-400 シリーズで標準以外の機能を追加する場合には、ボードオプションによりどのピンに機能を割り当てるか選択し、(標準で有効になっていない場合は)対応するデバイスドライバーを有効にする必要があります。ボードオプションは、Linux Kernel Configuration の **System Type -> Freescale MXC Implementations -> MX25 Options -> Armadillo-400 Board options** で変更することができます。

ボードオプションでは、一つのピンに複数の機能を割り当てることはできないようになっています。また、機能が割り当てられなかったピンは、GPIO として設定されます。

8.1. UART

i.MX25 プロセッサは UART1 から UART5 までの 5 つの UART モジュールを内蔵しています。Armadillo-400 シリーズでは標準状態で UART2 をシリアルインターフェース 1 (CON3)、UART3 をシリアルインターフェース 2 (CON9)、UART5 をシリアルインターフェース 3 (CON9)に使用しています。また、カーネルコンフィギュレーションを変更することにより、CON11 に UART3 及び UART4 の機能を割り当てることができます。UART9 もしくは UART11 に機能を割り当てた場合、CTS/RTS によるハードウェアフローコントロールを有効にするか無効にするかを選択することができます。

UART ドライバーは以下の機能を有します。

- ・ 7/8 bit 送受信
- ・ 1/2 ストップビット
- ・ None/Odd/Even パリティ
- ・ XON/XOFF ソフトウェアフローコントロール
- ・ CTS/RTS ハードウェアフローコントロール
- ・ モデム信号コントロール
- ・ スタンダード Linux シリアル API
- ・ 最大ボーレート 230.4Kbps(シリアルインターフェース 1) / 4Mbps^[1] (シリアルインターフェース 2, 3)

各シリアルインターフェースとデバイスファイルの対応を、「表 8.1. シリアルインターフェースとデバイスファイルの対応」に示します。

^[1]DMA 転送を有効にした場合。標準では DMA 転送は無効になっています。

表 8.1 シリアルインターフェースとデバイスファイルの対応

シリアルインターフェース	デバイスファイル	使用モジュール
シリアルインターフェース 1	/dev/ttymxcl	UART2
シリアルインターフェース 2	/dev/ttymxc2	UART3
シリアルインターフェース 3	/dev/ttymxc4	UART5

UART 機能に関連するカーネルコンフィギュレーションを「表 8.2. UART コンフィギュレーション」に示します。

表 8.2 UART コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
SERIAL_MXC	y	i.MX のシリアルドライバーを有効にします
SERIAL_MXC_CONSOLE	y	i.MX のシリアルドライバーを使ったシステムコンソールを有効にします
ARMADILLO400_UART3_CON9	y	CON9 の UART3 を有効にします CON9_3 を UART3_RXD に、 CON9_5 を UART3_TXD に使用します
ARMADILLO400_UART3_HW_FLOW_CON9	n	CON9 の UART3 のハードウェアフローコントロールを有効にします CON9_11 を UART3_RTS に、 CON9_13 を UART3_CTS に使用します ARMADILLO400_UART3_CON9 に依存します
ARMADILLO400_UART3_CON11	n	CON11 の UART3 を有効にします CON11_40 を UART3_RXD に、 CON11_41 を UART3_TXD に使用します ARMADILLO400_UART3_CON9 と排他です
ARMADILLO400_UART3_HW_FLOW_CON11	n	CON11 の UART3 のハードウェアフローコントロールを有効にします CON11_42 を UART3_RTS に、 CON11_43 を UART3_CTS に使用します ARMADILLO400_UART3_CON11 に依存します
ARMADILLO400_UART4_CON11	n	CON11 の UART4 を有効にします CON11_44 を UART4_RXD に、 CON11_45 を UART4_TXD に使用します

カーネルコンフィギュレーション	デフォルト	説明
ARMADILLO400_UART4_HW_FLOW_CON11	n	CON11 の UART4 のハードウェアフローコントロールを有効にします CON11_46 を UART4_RTS に、 CON11_47 を UART4_CTS に使用します ARMADILLO400_UART4_CON11 に依存します
ARMADILLO400_UART5_CON9	y	CON9 の UART5 を有効にします CON9_4 を UART5_RXD に、 CON9_6 を UART5_TXD に使用します
ARMADILLO400_UART5_HW_FLOW_CON9	n	CON9 の UART5 のハードウェアフローコントロールを有効にします CON9_12 を UART5_RTS に、 CON9_14 を UART5_CTS に使用します ARMADILLO400_UART5_CON9 に依存します

8.2. Ethernet

Armadillo-400 シリーズの Ethernet ドライバーは以下の機能を有します。

- ・ AutoNegotiation サポート
- ・ CarrierDetect サポート
- ・ Ethtool サポート
 - ・ link status
 - ・ 10/100Mbps Speed
 - ・ Full/Half Duplex
 - ・ AutoNegotiation enable/disable

8.3. SD/MMC/SDIO ホスト

i.MX25 プロセッサは、SD/MMC/SDIO ホストコントローラ(eSDHC)を二個内蔵しています。Armadillo-400 シリーズでは、標準状態で eSDHC1 を microSD スロット(CON1)に使用しています。また、カーネルコンフィギュレーションを変更することにより、CON9 に eSDHC2 の機能を割り当てることができます。

Armadillo-400 シリーズの SD/MMC/SDIO ホストドライバーは以下の機能を有します。

- ・ 4ビットモード
- ・ カードディテクト

microSD カードスロットにカードが挿入されると、`/dev/mmcblkN` (N は 0 または 1) として認識されます。

SD/MMC/SDIO ホスト機能に関連するカーネルコンフィギュレーションを「表 8.3. SD/MMC/SDIO ホストコントローラ コンフィギュレーション」に示します。

表 8.3 SD/MMC/SDIO ホストコントローラ コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
MMC	y	Linux カーネルの MMC/SD/SDIO カードサポートを有効にします
MMC_IMX_ESDHC1	y	i.MX25 の eSDHC ドライバーを有効にします
ARMADILLO400_SDHC2_CON9	n	CON9 の SDHC2 を有効にします CON9_15 から CON9_24 を使用します

8.4. USB 2.0 ホスト

Armadillo-400 シリーズの USB 2.0 ホストドライバーは以下の機能を有します。

- ・ EHCI 準拠
- ・ OTG 非サポート
- ・ USB High Speed ホスト × 1
- ・ USB Full Speed ホスト × 1

USB デバイスが検出されると、`/dev/sd*` にマップされます。

8.5. フレームバッファ

Armadillo-440 のビデオ出力機能は、フレームバッファデバイスとして実装されています。

フレームバッファデバイスドライバーは以下の機能を有します。

- ・ ダブルバッファサポート
- ・ 最大解像度 SVGA

Armadillo-440 液晶モデルの標準では、以下の設定になっています。

- ・ 解像度 480 × 272 ピクセル
- ・ RGB 565 カラー

フレームバッファとデバイスファイルの対応を、「表 8.4. フレームバッファとデバイスファイルの対応」に示します。

表 8.4 フレームバッファとデバイスファイルの対応

フレームバッファ	デバイスファイル
バックグラウンドプレーン	<code>/dev/fb0</code>
グラフィックウィンドウ	<code>/dev/fb1</code>

8.6. LED バックライト

Armadillo-440 の LED バックライト機能は、バックライトクラスとして実装されています。

`/sys/class/backlight/pwm-backlight` ディレクトリ以下のファイルによって、バックライトの制御を行うことができます。

輝度の調整は、`brightness` ファイルによって行うことができます。`brightness` ファイルに 0(消灯)~255(最高輝度)までの数値を書き込むことにより、輝度を変更することができます。

また、`brightness` ファイルを読むことにより現在の輝度を知ることができます。

バックライトの点灯/消灯は、`bl_power` ファイルによって行うことができます。`bl_power` に 0 を書き込むと消灯になり、1 を書き込むと点灯になります。

8.7. タッチスクリーン

Armadillo-440 のタッチスクリーン機能は、インプットデバイスとして実装されており、ユーザーランドとのインターフェースとしてイベントインターフェースを提供しています。

「表 8.5. タッチスクリーンイベント」に示すイベントが発生します。

表 8.5 タッチスクリーンイベント

Type	Code	Value
EV_KEY(1)	BTN_TOUCH(330)	0 or 1
EV_ABS(3)	ABS_X(0)	100 ~ 4000
EV_ABS(3)	ABS_Y(1)	100 ~ 4000
EV_ABS(3)	ABS_PRESSURE(24)	0 or 1

Armadillo-440 液晶モデルの標準状態では、タッチスクリーンのイベントデバイスは `/dev/input/event1` にマップされます。



イベントデバイスの番号は、検出された順番に割り振られます。そのため、USB キーボードなど他のインプットデバイスが起動時に検出されると、タッチスクリーンのイベントデバイス番号は変わる可能性があります。

8.8. オーディオ

Armadillo-400 シリーズのオーディオ機能は、ALSA デバイスとして実装されています。ALSA デバイスドライバーは以下の機能を有します。

- ・ Playback(2ch) / Capture(1ch)
- ・ サンプリング周波数 48k, 32k, 24k, 16k, 12k, 8k Hz
- ・ フォーマット Signed 16/20/24 bit, Little-endian

オーディオデバイスの制御は、ALSA ライブラリ (`libasound2`) を通じて行うことができます。



Armadillo-400 シリーズのオーディオドライバーでは、録音と再生を同時に行うことはできません。

i.MX25 では、オーディオマルチプレクスにより、オーディオ機能をどのピンで使用するかを選択することができます。Armadillo-400 シリーズではカーネルコンフィギュレーションでオーディオマルチプレクスの設定をおこなうことができます。標準では、オーディオマルチプレクスは AUD6 を使用するようになっており、オーディオ機能は CON11 に接続されます。コンフィギュレーションにより AUD5 を使用することで CON9 に接続することができます。

オーディオ 機能に関連するカーネルコンフィギュレーションを「表 8.6. オーディオ コンフィギュレーション」に示します。

表 8.6 オーディオ コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
SOUND	y	Linux カーネルのサウンドカードサポートを有効にします
SND	y	Linux カーネルの ALSA サポートを有効にします
SND_SOC	y	Linux カーネルの ASoC サポートを有効にします
SND_MXC_SOC	y	i.MX でオーディオ機能を実現するためのドライバを有効にします
SND_SOC_ARMADILLO440_WM8978	y	Armadillo-400 シリーズで WM8978 コーデックを使用したオーディオ機能を実現するためのドライバを有効にします
ARMADILLO400_AUD5_CON11	y	CON11 にオーディオ機能を出力します CON11_42 から CON11_47 を使用します
ARMADILLO400_AUD6_CON9	n	CON9 にオーディオ機能を出力します CON9_15, 17, 21, 22, 23, 24 を使用します ARMADILLO400_AUD5_CON11 と排他です

8.9. GPIO

Armadillo-400 シリーズの GPIO は、generic GPIO として実装されています。

ユーザーランドから GPIO を操作するためのインターフェースとしては、GPIO sysfs と Armadillo-200 シリーズ互換 GPIO ドライバーの 2 つがあります。標準状態では GPIO sysfs ドライバーが有効になっています。

カーネルコンフィギュレーションで機能が割り当てられなかったピンは、全て GPIO として設定されます。

8.9.1. GPIO sysfs

GPIO sysfs では、`/sys/class/gpio/(GPIO_NAME)` ディレクトリ以下のファイルで入出力方向の設定、出力レベルの設定、入出力レベルの取得を行うことができます。

GPIO_NAME ディレクトリと GPIO ピンの対応を「表 8.7. GPIO_NAME と GPIO ピンの対応」に示します。

表 8.7 GPIO_NAME と GPIO ピンの対応

GPIO_NAME	GPIO ピン	初期入出力方向	初期出力レベル
CON9_1	CON9 1 ピン	入力	-
CON9_2	CON9 2 ピン	入力	-
CON9_11	CON9 11 ピン	入力	-
CON9_12	CON9 12 ピン	入力	-
CON9_13	CON9 13 ピン	入力	-
CON9_14	CON9 14 ピン	入力	-
CON9_15	CON9 15 ピン	入力	-
CON9_16	CON9 16 ピン	入力	-
CON9_17	CON9 17 ピン	入力	-
CON9_18	CON9 18 ピン	入力	-
CON9_21	CON9 21 ピン	入力	-
CON9_22	CON9 22 ピン	入力	-
CON9_23	CON9 23 ピン	入力	-
CON9_24	CON9 24 ピン	入力	-
CON9_25	CON9 25 ピン	入力	-
CON9_26	CON9 26 ピン	入力	-
CON9_27	CON9 27 ピン	出力	LOW
CON9_28	CON9 28 ピン	出力	LOW

`/sys/class/gpio/(GPIO_NAME)/direction` ファイルで入出力方向の設定を行うことができます。「表 8.8. GPIO 入出力方向の設定」に示す設定を `direction` ファイルに書き込むことにより、入出力方向を設定します。また、`direction` ファイルを読み込むことで現在の設定を知ることができます。

表 8.8 GPIO 入出力方向の設定

設定	説明
high	入出力方向を出力に、出力レベルを HIGH レベルに設定します。出力レベルの取得/設定を行うことができます。
low	入出力方向を出力に、出力レベルを LOW レベルに設定します。出力レベルの取得/設定を行うことができます。
out	low を設定した場合と同じです。
in	入出力方向を入力に設定します。入力レベルの取得を行うことができます。

`/sys/class/gpio/(GPIO_NAME)/value` ファイルで出力レベルの設定、入出力レベルの取得を行うことができます。0 が LOW レベルを、1 が HIGH レベルを意味します。

8.9.2. Armadillo-200 シリーズ互換 GPIO ドライバー

Armadillo-200 シリーズ互換 GPIO ドライバーでは、対応するデバイスファイルに対して `ioctl` を発行することにより、GPIO の操作および状態の取得を行うことができます。



Armadillo-200 シリーズ互換 GPIO ドライバーは標準状態では無効になっています。有効にするには Linux カーネルコンフィギュレーションで、`CONFIG_GPIO_SYSFS` を無効に、`CONFIG_ARMADILLO2X0_GPIO` を有効にして、カーネルをビルドする必要があります。

Armadillo-200 シリーズ互換 GPIO ドライバーでの GPIO 名と GPIO ピンの対応を「表 8.9. Armadillo-200 シリーズ互換 GPIO ドライバー GPIO 一覧」に示します。

表 8.9 Armadillo-200 シリーズ互換 GPIO ドライバー GPIO 一覧

GPIO 名	GPIO ピン	初期入出力方向	初期出力レベル
GPIO0	CON9 21 ピン	入力	-
GPIO1	CON9 22 ピン	入力	-
GPIO2	CON9 23 ピン	入力	-
GPIO3	CON9 24 ピン	入力	-
GPIO4	CON9 25 ピン	入力	-
GPIO5	CON9 26 ピン	入力	-
GPIO6	CON9 27 ピン	出力	LOW
GPIO7	CON9 28 ピン	出力	LOW
GPIO8	CON9 11 ピン	入力	-
GPIO9	CON9 12 ピン	入力	-
GPIO10	CON9 13 ピン	入力	-
GPIO11	CON9 14 ピン	入力	-
GPIO12	CON9 15 ピン	入力	-
GPIO13	CON9 16 ピン	入力	-
GPIO14	CON9 17 ピン	入力	-
GPIO15	CON9 18 ピン	入力	-



Armadillo-200 シリーズ互換 GPIO ドライバーでの GPIO 名と対応する GPIO ピンの位置は、Armadillo-200 シリーズと同じになっています。そのため、Armadillo-200 シリーズ互換 GPIO ドライバーでは Armadillo-400 シリーズで使用可能な GPIO のうち、一部だけしか操作することができません。

デバイスファイルのパラメータは、以下の通りです。

表 8.10 Armadillo-200 シリーズ互換 GPIO ドライバーデバイスファイル

タイプ	メジャー番号	マイナー番号	デバイスファイル
キャラクタデバイス	10	185	/dev/gpio

ioctl の第 1 引数には、デバイスファイルのファイルディスクリプタを指定します。第 2 引数には、GPIO を操作するためのコマンドを指定します。

表 8.11 Armadillo-200 シリーズ互換 GPIO ドライバー ioctl コマンド

コマンド	説明	第 3 引数の Type
PARAM_SET	第 3 引数で指定する内容で GPIO の状態を設定します	struct gpio_param
PARAM_GET	第 3 引数で指定する内容で GPIO の状態を取得します	struct gpio_param
INTERRUPT_WAIT	第 3 引数で指定する内容で GPIO の割込みが発生するまで WAIT します	struct wait_param

第 3 引数には、()/include/linux/armadillo2x0_gpio.h に定義されている構造体「struct gpio_param」と「struct wait_param」を使用します。「struct gpio_param」は単方向リストになっているので、複数の GPIO を一度に制御する場合は next メンバを使用してください。また、リストの最後の next メンバには"0(NULL)"を指定してください。GPIO デバイスドライバーの詳細な使用方法については、GPIO 操作アプリケーション(atmark-dist/vendors/AtmarkTechno/Armadillo-440/gpioctrl)のソースコードを参考にしてください。

8.10. LED

Armadillo-400 シリーズの LED ドライバーは、LED クラスドライバーと Armadillo-200 シリーズ互換 LED ドライバーの 2 つがあります。標準状態では、LED クラスドライバーが有効になっています。

8.10.1. LED クラス

/sys/class/leds/(LED_NAME) ディレクトリ以下のファイルによって、LED の制御を行うことができます。

点灯/消灯の制御は、/sys/class/leds/(LED_NAME)/brightness ファイルによって行うことができます。brightness ファイルに 0 を書き込むと消灯、0 以外の数値を書き込むと点灯となります。

LED クラスでは、点滅などの制御はトリガーという仕組みを使用して行います。Armadillo-400 シリーズでは、mmc0、timer、heartbeat、default-on のトリガーを使用することができます。各文字列を、/sys/class/leds/(LED_NAME)/trigger ファイルに書き込むことでトリガーが有効になります。

mmc0 トリガーを有効にすると MMC/SD カードへの読み書きに連動して LED が点灯/消灯します。

timer トリガーにより、指定した周期で LED を点滅させることができます。timer トリガーを有効にすると、新しく /sys/class/leds/(LED_NAME)/delay_on と /sys/class/leds/(LED_NAME)/delay_off ファイルが作成されます。それぞれのファイルに点灯時間[msec]と消灯時間[msec]を書き込むことで LED が点滅します。

heartbeat トリガーを有効にすると、鼓動のように LED が点滅します。

default-on トリガーを有効にすると、点灯状態で起動します。

LED_NAME と対応する LED の一覧を「表 8.12. LED 一覧」に示します。

表 8.12 LED 一覧

LED_NAME	対応する LED	デフォルトトリガー
red	LED3	なし
green	LED4	default-on
yellow	LED5	なし

8.10.2. Armadillo-200 シリーズ互換 LED ドライバー

Armadillo-200 シリーズ互換 LED ドライバーでは、対応するデバイスファイルに対して ioctl を発行することにより、LED の操作を行うことができます。



Armadillo-200 シリーズ互換 LED ドライバーは標準状態では無効になっています。有効にするには Linux カーネルコンフィギュレーションで、CONFIG_LEDS_GPIO を無効に、CONFIG_ARMADILLO2X0_LED を有効にして、カーネルをビルドする必要があります。

LED に対応するデバイスファイルのパラメータは、以下の通りです。

表 8.13 LED ノード

タイプ	メジャー番号	マイナー番号	デバイスファイル
キャラクタデバイス	10	215	/dev/led

ioctl の第 1 引数には、デバイスファイルのファイルディスクリプタを指定します。第 2 引数には、LED を操作するためのコマンドを指定します。

表 8.14 LED 操作コマンド

コマンド	説明	第 3 引数の Type
LED_RED_ON	LED3(赤)を点灯します	なし
LED_RED_OFF	LED3(赤)を消灯します	なし
LED_RED_STATUS	LED3(赤)の点灯状態を取得します	状態を保存するバッファ(最小 1 バイト)
LED_RED_BLINKON	LED3(赤)の点滅を開始します	なし
LED_RED_BLINKOFF	LED3(赤)の点滅を停止します	なし
LED_RED_BLINKSTATUS	LED3(赤)の点滅状態を取得します	状態を保存するバッファ(最小 1 バイト)
LED_GREEN_ON	LED(緑)を点灯します	なし
LED_GREEN_OFF	LED4(緑)を消灯します	なし
LED_GREEN_STATUS	LED4(緑)の点灯状態を取得します	状態を保存するバッファ(最小 1 バイト)

コマンド	説明	第 3 引数の Type
LED_GREEN_BLINKON	LED4(緑)の点滅を開始します	なし
LED_GREEN_BLINKOFF	LED4(緑)の点滅を停止します	なし
LED_GREEN_BLINKSTATUS	LED4(緑)の点滅状態を取得します	状態を保存するバッファ(最小 1 バイト)

LED デバイスドライバーの詳細な使用方法については、サンプルの LED 制御アプリケーション(atmark-dist/vendors/AtmarkTechno/Armadillo-440/ledctrl)のソースコードを参考にしてください。

8.11. ボタン

Armadillo-400 シリーズでは、ボタン入力はインプットデバイスとして実装されており、ユーザーランドとのインターフェースとしてイベントインターフェースを提供しています。

Armadillo-440 液晶モデルでは、Armadillo-440 本体にオンボードタクトスイッチ× 1 と、LCD 拡張ボードにボタン× 3 が実装されています。

それぞれのボタンに対するイベントを、「表 8.15. Armadillo-440 液晶モデルボタンイベント」に示します。

表 8.15 Armadillo-440 液晶モデルボタンイベント

ボタン	Type	Code	Value
SW1	EV_KEY(1)	KEY_ENTER(28)	0 or 1
LCD_SW1	EV_KEY(1)	KEY_BACK(158)	0 or 1
LCD_SW2	EV_KEY(1)	KEY_MENU(139)	0 or 1
LCD_SW3	EV_KEY(1)	KEY_HOME(102)	0 or 1

Armadillo-440 液晶モデルの標準状態では、ボタンのイベントデバイスは /dev/input/event0 にマップされます。



イベントデバイスの番号は、検出された順番に割り振られます。そのため、USB キーボードなど他のインプットデバイスが起動時に検出されると、ボタンのイベントデバイス番号は変わる可能性があります。

8.12. リアルタイムクロック

Armadillo-440 LCD 拡張ボード及び Armadillo-400 RTC オプションモジュールには、リアルタイムクロックが搭載されています。

リアルタイムクロックは、/dev/rtc0 として認識されます。Armadillo-440 液晶モデルに RTC オプションモジュールを接続した場合、RTC オプションモジュールのリアルタイムクロックが /dev/rtc0 となり、LCD 拡張ボードのリアルタイムクロックは /dev/rtc1 として認識されます。

8.13. ウォッチドッグタイマー

Armadillo-400 シリーズで採用している i.MX25 プロセッサは、内蔵ウォッチドッグタイマーを有しています。

Armadillo-400 シリーズの標準ブートローダーでは、起動直後にこの内蔵ウォッチドッグタイマーを有効にします。標準状態でのタイムアウト時間は 10 秒に設定されます。

Linux カーネルでは、自動でウォッチドッグタイマーをキックします。

もし、何らかの要因で Linux カーネルがフリーズしてウォッチドッグタイマーをキックできなくなりタイムアウトが発生すると、システムリセットが発生します。

8.14. I2C

i.MX25 プロセッサは、I2C1 から I2C3 の 3 個の I2C コントローラーを内蔵しています。Armadillo-400 シリーズでは、標準で I2C1 はボード内蔵バスとして使用し、I2C2 は CON14 に、I2C3 は CON11 に割り当てています。

I2C バスドライバーは以下の機能を有します。

- ・ I2C マスターモード
- ・ 最大 400 kbps

I2C バスにスレーブデバイスを接続し、それを使用可能にするためには、デバイスに対応したチップドライバを有効にする必要があります。また、チップドライバが "new style"^[2] で記述されていた場合、struct i2c_board_info を適切に設定しなければいけません。Armadillo-400 シリーズでは、linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c の armadillo400_i2cN_board_info 配列(N はバスに対応した数値)に記述してください。

I2C 機能に関連するカーネルコンフィギュレーションを「表 8.16. I2C コンフィギュレーション」に示します。

表 8.16 I2C コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
ARMADILLO400_I2C2_CON14	y	CON14 の I2C2 を有効にします CON14_3 を I2C2_SCL に、CON14_4 を I2C2_SDA に使用します
ARMADILLO400_I2C3_CON11	y	CON11 の I2C3 を有効にします CON11_48 を I2C3_SCL に、CON11_49 を I2C3_SDA に使用します

8.15. SPI

i.MX25 プロセッサは、CSPI1 から CSPI3 の 3 個の SPI コントローラーを内蔵しています。Armadillo-400 シリーズでは、カーネルコンフィギュレーションにより、CSPI1 及び CSPI3 を CON9 に割り当てることが可能です。

SPI ドライバーは以下の機能を有します。

^[2]linux-2.6.26-at/Documentation/i2c/writing-clients 参照

- ・ SPI マスターモード
- ・ 複数スレーブセレクト

SPI バスにスレーブデバイスを接続し、それを使用可能にするためには、デバイスに対応したドライバを有効にする必要があります。また、struct spi_board_info を適切に設定しなければいけません。Armadillo-400 シリーズでは、linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c の armadillo400_spiN_board_info 配列(N はバスに対応した数値)に記述してください。

SPI 機能に関連するカーネルコンフィギュレーションを「表 8.17. SPI コンフィギュレーション」に示します。

表 8.17 SPI コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
SPI	n	Linux カーネルの SPI サポートを有効にします
SPI_MXC	n	i.MX の SPI ドライバを有効にします
ARMADILLO400_SPI1_CON9	n	CON9 の SPI1 を有効にします CON9_3 を CSPI1_MOSI、CON9_5 を CSPI1_MISO、CON9_13 を CSPI1_SCLK、CON9_26 を CSPI1_RDY として使用します
ARMADILLO400_SPI1_SS0_CON9_25	n	CON9_25 を SPI1_SS0 として使用します ARMADILLO400_SPI1_CON9 に依存します
ARMADILLO400_SPI1_SS1_CON9_11	n	CON9_11 を SPI1_SS1 として使用します ARMADILLO400_SPI1_CON9 に依存します
ARMADILLO400_SPI3_CON9	n	CON9 の SPI3 を有効にします CON9_4 を CSPI3_MOSI、CON9_6 を CSPI3_MISO、CON9_12 を CSPI3_SCLK、CON9_14 を CSPI3_RDY として使用します。
ARMADILLO400_SPI3_SS0_CON9_16	n	CON9_16 を SPI3_SS0 として使用します ARMADILLO400_SPI3_CON9 に依存します
ARMADILLO400_SPI3_SS1_CON9_18	n	CON9_18 を SPI3_SS1 として使用します ARMADILLO400_SPI3_CON9 に依存します
ARMADILLO400_SPI3_SS2_CON9_15	n	CON9_15 を SPI3_SS2 として使用します ARMADILLO400_SPI3_CON9 に依存します
ARMADILLO400_SPI3_SS2_CON9_17	n	CON9_17 を SPI3_SS3 として使用します ARMADILLO400_SPI3_CON9 に依存します

8.16. one wire

Armadillo-400 シリーズでは、CON9_2 と CON9_26 を one wire マスターとして使用することができます。CON9_2 では、i.MX25 プロセッサ内蔵の one wire コントローラを使用して機能を実現し、CON9_26 では GPIO one wire ドライバを用いて機能を実現しています。

one wire バスにスレーブデバイスを接続し、それを使用可能にするためには、デバイスに対応したスレーブデバイスドライバを有効にする必要があります。

one wire 機能に関連するカーネルコンフィギュレーションを「表 8.18. one wire コンフィギュレーション」に示します。

表 8.18 one wire コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
W1	n	Linux カーネルの one wire サポートを有効にします
W1_MASTER_MXC	n	i.MX25 内蔵コントローラを使用した one wire マスタードライバを有効にします CON9_2 を使用する場合には有効にしてください
W1_MASTER_GPIO	n	GPIO を使用した one wire マスタードライバを有効にします CON9_26 を使用する場合には有効にしてください
ARMADILLO400_W1_CON9_2	n	CON9_2 を one wire として使用します
ARMADILLO400_W1_CON9_26	n	CON9_26 を one wire として使用します

8.17. PWM

i.MX25 プロセッサは、PWM1 から PWM4 の 4 個の PWM モジュールを内蔵しています。Armadillo-400 シリーズでは、標準では PWM1 を LED バックライト (CON11_12)として使用しています。カーネルコンフィギュレーションを変更することにより、PWM2 を CON9_25 に、PWM4 を CON14_3 に割り当てることができます。

i.MX25 の PWM ドライバでは、/sys/class/mxc_pwm/(PWM_NAME) 以下のファイルに値を書き込むことで設定変更することができます。設定に使用するファイルを、「表 8.19. PWM sysfs」に示します。

表 8.19 PWM sysfs

ファイル名	説明
period_ns	PWM の周期を nsec 単位で設定します 設定可能な範囲は、17 から 2,147,483,647(約 20usec から 2sec)です
duty_ns	PWM の ON 時間(invert = 1 の場合は OFF 時間)を nsec 単位で設定します 設定可能な範囲は、0 < duty_ns < period_ns の範囲です
invert	1 に設定すると、PWM 出力が反転します
enable	1 に設定すると、PWM 出力が有効になります 0 で出力が停止します PWM 出力中でも、period_ns、duty_ns、invert は設定変更可能です

PWM 機能に関連するカーネルコンフィギュレーションを「表 8.20. PWM コンフィギュレーション」に示します。

表 8.20 PWM コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
MXC_PWM	y	i.MX25 の PWM ドライバーを有効にします
MXC_PWM_CLASS	y	sysfs 経由で PWM の設定を可能にします
ARMADILLO400_PWM2_CON9_25	n	CON9_25 を PWM2 として使用します PWM_NAME は CON9_25 です
ARMADILLO400_PWM4_CON14_3	n	CON14_3 を PWM4 として使用します PWM_NAME は CON14_3 です

8.18. CAN

i.MX25 プロセッサは、CAN1 及び CAN2 の 2 個の CAN コントローラ(FlexCAN)を内蔵しています。Armadillo-400 シリーズでは、カーネルコンフィギュレーションにより、CAN2 を CON14 に割り当てることが可能です。

CAN 機能は、SocketCAN フレームワークを使用して実現しています。SocketCAN については、linux-2.6.26-at/Documentation/networking/can.txt などを参照してください。

CAN ドライバーは以下の機能を有します。

- ・ 標準/拡張フォーマット対応
- ・ 最大 1Mbps

/sys/devices/platform/FlexCAN.1/ 以下のファイルに値を書き込むことで設定変更することができます。設定に使用するファイルを、「表 8.21. CAN sysfs」に示します^[3]。

表 8.21 CAN sysfs

ファイル名	説明	デフォルト値
br_clksrc	クロックソースを指定します bus を指定すると、クロックソースに 66.5MHz を使用します osc を指定すると、クロックソースに 24MHz を使用します	bus
br_presdiv	クロックソースのプリスケイラー分周値を設定します 1 から 8 を指定できます	7
br_propseg	プロパゲーションセグメントの値を設定します 1 から 8 を指定できます	5
br_pseg1	フェーズバッファセグメント 1 の値を設定します 1 から 8 を指定できます	5
br_pseg2	フェーズバッファセグメント 2 の値を設定します 1 から 8 を指定できます	8
br_rjw	リシンクロナイゼーションジャンプ幅を設定します 1 から 4 を指定できます	3

^[3]記述のないファイルは、後方互換性のために残されているものなので、使用しないでください。

ファイル名	説明	デフォルト値
bitrate	通信速度[bps]を示します 書き込みはできません	500000
std_msg	標準フォーマットに対応するかどうかを設定します 1 で対応、0 で非対応となります	1
ext_msg	拡張フォーマットに対応するかどうかを設定します 1 で対応、0 で非対応となります	1
maxmb	メッセージバッファの最大数を設定します 2 から 64 を指定できます	64
rx_maxmb	受信メッセージバッファのサイズを設定します 送信メッセージバッファのサイズは maxmb-rx_maxmb となります 1 から maxmb-1 を指定できます	32
state	現在のステータスを表示します "インターフェースのステータス::エラー状態"というフォーマットで表示されます インターフェースのステータスは、"Start"(up 状態)か"Stop"(down 状態)のいずれかです エラー状態は、"normal"(エラーなし)か"error passive"(エラーパッシブ状態)、"bus off"(バスオフ状態)のいずれかです	Stop::normal
boff_rec	自動的にバスオフから復帰するかどうかを設定します 0 で自動的に復帰、1 で復帰しません	1
listen	listen モード(受信のみ)にするかどうかを設定します 1 で listen モード有効、0 で無効になります	0
loopback	loopback モードにするかどうかを設定します 1 で loopback モード有効、0 で無効になります	0
smp	サンプリング時の動作を設定します 0 で 1 回のサンプルで受信したビットの値を決定します 1 で 3 回のサンプルをおこない多数決により受信したビットの値を決定します	1
srx_dis	自身が送信したフレームを受信するかどうかを設定します 0 で自身が送信したフレームを受信し、1 で受信しません	1
set_resframe	リモートフレームを受信した際、返信するデータフレームを設定します "ID#DATA"という形式で設定します ID には、16 進数 3 桁(標準フォーマット)もしくは 16 進数 8 桁を指定します DATA には、1 データあたり 16 進数 2 桁で、0 から 8 個までのデータを指定します データは、で区切ることもできます	なし
del_resframe	set_resframe で設定したデータフレームを削除します 削除するデータフレームの ID を 16 進数 3 桁(標準フォーマット)もしくは 16 進数 8 桁で指定してください	なし
show_resframe	set_resframe で設定したデータフレームを表示します 書き込みはできません	なし

ファイル名	説明	デフォルト値
wakeup	サスペンド時に CAN 受信によるウェイクアップを有効にするかどうかを設定します 1 でウェイクアップ有効、0 で無効となります	0
wak_src	サスペンド時にローパスフィルタを使用するかどうかを設定します 0 でフィルタリング無効、1 で有効になります	0

通信速度は以下の計算式により算出されます。

```
src_clk = 66,500,000 (br_clksrc = bus )
src_clk = 24,000,000 (br_clksrc = osc )
[bps] = src_clk / br_presdiv / (1 + br_propseg + br_pseg1 + br_pseg2)
```

図 8.1 CAN 通信速度計算

CAN 機能に関連するカーネルコンフィギュレーションを「表 8.22. CAN コンフィギュレーション」に示します。

表 8.22 CAN コンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
CAN	n	Linux カーネルの CAN サポートを有効にします
CAN_RAW	n	RAW_CAN プロトコルを有効にします
CAN_BCM	n	CAN_BCM プロトコルを有効にします
CAN_FLEXCAN	n	i.MX25 の FlexCAN ドライバを有効にします
ARMADILLO400_CAN2_CON14	n	CON14 を CAN2 として使用します CON14_3 を CAN2_TXCAN、CON14_4 を CAN2_RXCAN として使用します

8.19. キーパッド

i.MX25 プロセッサは、キーパッドコントローラを内蔵しています。Armadillo-400 では、カーネルコンフィギュレーションにより、CON11 をキーパッドに割り当てることが可能です。



キーパッドドライバーを有効にすると、イベントデバイスは /dev/input/event0 にマップされます。そのため、ボタン及びタッチスクリーンのイベントデバイス番号が変わります。

キーパッド機能に関連するカーネルコンフィギュレーションを「表 8.23. キーパッドコンフィギュレーション」に示します。

表 8.23 キーパッドコンフィギュレーション

カーネルコンフィギュレーション	デフォルト	説明
INPUT	y	Linux カーネルのインプットレイヤサポートを有効にします
INPUT_EVDEV	y	インプットレイヤのイベントデバイスサポートを有効にします
KEYBOARD_MXC	n	i.MX25 のキーパッドドライバを有効にします
ARMADILLO400_KEYPAD_CON11	n	CON11 のキーパッドを有効にします COL と ROW を少なくとも一つずつ有効にしてください
ARMADILLO400_KEYPAD_ROW0_CON11_40	n	CON11_40 を ROW0 として使用します
ARMADILLO400_KEYPAD_ROW1_CON11_41	n	CON11_41 を ROW1 として使用します
ARMADILLO400_KEYPAD_ROW2_CON11_42	n	CON11_42 を ROW2 として使用します
ARMADILLO400_KEYPAD_ROW3_CON11_43	n	CON11_43 を ROW3 として使用します
ARMADILLO400_KEYPAD_COLO_CON11_44	n	CON11_44 を COL0 として使用します
ARMADILLO400_KEYPAD_COL1_CON11_45	n	CON11_45 を COL1 として使用します
ARMADILLO400_KEYPAD_COL2_CON11_46	n	CON11_46 を COL2 として使用します
ARMADILLO400_KEYPAD_COL3_CON11_47	n	CON11_47 を COL3 として使用します
ARMADILLO400_KEYPAD_ROW4_CON11_48	n	CON11_48 を ROW4 として使用します
ARMADILLO400_KEYPAD_ROW5_CON11_49	n	CON11_49 を ROW5 として使用します

ROW 及び COL のどの範囲をキーパッドとして使用するかは、linux-2.6.26-at/arch/arm/maxh-mx25/armadillo400.c の armadillo440_keypad_data 変数に適切な値を指定してください。また、各ボタンがどのイベントに対応するかというキーマップは armadillo440_keymapping 変数で指定してください。

8.20. パワーマネジメント

Armadillo-400 シリーズは Linux パワーマネジメントのスリープ機能をサポートします。スリープ状態では、アプリケーションの実行は一時停止し、カーネルはサスペンド状態となります。スリープ状態では外部デバイスの動作を停止するため、消費電力を抑えることができます。スリープ状態から実行状態に復帰すると、カーネルのリジューム処理が行われた後、アプリケーションの実行を再開します。

/sys/power/state ファイルに、standby もしくは mem を書き込むことにより、スリープモードへ移行することができます。また、スリープモード中にウェイクアップ要因による割り込みが発生すると、スリープモードから実行状態へ復帰します。

各モードによる、状態の違いは「表 8.24. スリープモード」を参照してください。power-on suspend 状態に比べ、suspend-to-RAM の方がより少ない消費電力でスリープすることができます。

表 8.24 スリープモード

スリープモード	state ファイル に書き込む文字 列	i.MX25 パワー モード	ウェイクアップ要因
power-on suspend	standby	Doze モード	シリアル入力、タッチスクリーン入 力、ボタン入力
suspend-to- RAM	mem	Stop モード	ボタン入力

ウェイクアップ要因になることができるデバイスを、ウェイクアップ要因にするかどうかは、各デバイスに対応する sysfs エントリの power/wakeup ファイルで指定することができます。power/wakeup ファイルに enabled と書き込むとウェイクアップ要因になり、disabled と書き込むとウェイクアップ要因ではなくなります。

「表 8.25. ウェイクアップ要因の指定」を参照してください。

表 8.25 ウェイクアップ要因の指定

デバイス	sysfs ファイル	初期状態
シリアルインターフェース 1	/sys/devices/platform/mxcintuart.1/tty/ ttymxc1/power/wakeup	enabled
シリアルインターフェース 2	/sys/devices/platform/mxcintuart.2/tty/ ttymxc2/power/wakeup	disabled
シリアルインターフェース 3	/sys/devices/platform/mxcintuart.4/tty/ ttymxc4/power/wakeup	disabled
タッチスクリーン	/sys/devices/platform/imx_adc.0/power/ wakeup	enabled
ボタン	/sys/devices/platform/gpio-keys.0/power/ wakeup	enabled
キーパッド	/sys/devices/platform/mxc_keypad.0/power/ wakeup	enabled
FlexCAN	/sys/devices/platform/FlexCAN.1/wakeup	disabled

8.20.1. スリープ中の外部デバイスの扱いについて

Armadillo-400 シリーズでは、サスペンド処理で外部デバイスへの電源供給を全て停止します。

そのため、USB デバイスはスリープ状態に移行する前に、安全に取り外しができる状態にしておく必要があります。すなわち、USB メモリは umount しておく必要があります。リジューム時にデバイス検出が再度行われるため、USB デバイスはスリープ中に抜き差しすることができます。

一方で、microSD カードは、mount したままでスリープ状態に移行することができます。これを可能にするために、SD ホストドライバーではリジューム時にプローブ処理を行わず、同じカードが挿入されているものとして扱います。そのため、スリープ中に microSD の抜き差しを行うことはできません。

Ethernet デバイスは実行状態に復帰後、ケーブルを抜き差ししたときと同様の処理が行われるため、Auto-negotiation が有効になっている場合、リジューム後に Auto-negotiation が行われます。

初期状態の設定では、+3.3V_IO 出力はスリープ時に停止します。しかし、シリアルポート 2 および 4 をウェイクアップ要因に指定した場合、スリープ時も +3.3V_IO 出力が供給されます。

付録 A Hermit-At ブートローダー

Hermit-At は、アットマークテクノ製品に採用されている高機能ダウンローダー兼ブートローダーです。Armadillo を保守モードで起動すると、Hermit-At ブートローダーのプロンプトが表示されます。プロンプトからコマンドを入力することにより、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 Hermit-At ブートローダーの様々な機能を使用することができます。ここでは、代表的な機能について説明します。

A.1. version

バージョン情報を表示するコマンドです。

```
構文 : version
```

図 A.1 version 構文

A.1.1. version 使用例

```
hermit> version  
Hermit-At v2.0.0 (armadillo4x0) compiled at 23:03:08, Mar 08 2010
```

図 A.2 version の使用例

A.2. info

ボード情報を表示するコマンドです。

```
構文 : info
```

図 A.3 info 構文

A.2.1. info 使用例

```
hermit> info  
Board Type: 0x00000440  
Hardware ID: 0x00000300  
  DRAM ID: 0x00000002  
  Jumper: 0x00000001  
  Tact-SW: 0x00000000
```

図 A.4 info の使用例

A.3. memmap

フラッシュメモリと DRAM のメモリマップを表示するコマンドです。

```
構文 : memmap
```

図 A.5 memmap 構文

A.3.1. memmap 使用例

```
hermit> memmap
0xa0000000:0xa1fffffff FLA all bf:8K bl:4x32K/1,255x128K/1
0xa0000000:0xa001ffff FLA bootloader bf:8K bl:4x32K/1
0xa0020000:0xa021ffff FLA kernel bf:8K bl:16x128K
0xa0220000:0xa1fdffff FLA userland bf:8K bl:238x128K
0xa1fe0000:0xa1fffffff FLA config bf:8K bl:1x128K
0x80000000:0x87fffffff RAM dram-1
```

図 A.6 memmap の使用例

A.4. mac

MAC アドレスを表示するコマンドです。

```
構文 : mac
```

図 A.7 mac 構文

A.4.1. mac 使用例

```
hermit> mac
00:11:0c:00:00:00
```

図 A.8 mac の使用例

A.5. md5sum

メモリのある区間の md5sum 値を計算して表示するコマンドです。

```
構文 : md5sum <開始アドレス> <サイズ>
```

図 A.9 md5sum 構文

A.5.1. md5sum 使用例

bootloader リージョンの先頭から 1024 Bytes の区間の md5sum 値を計算して表示するには、「[図 A.10. md5sum の使用例](#)」のようにコマンドを実行します。

```
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
hermit> md5sum 0xa0000000 1024
fd44ce938f65726dc59669f537154429
```

図 A.10 md5sum の使用例

A.6. 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.11 erase 構文

A.6.1. erase 使用例

```
hermit> erase 0xa0fe0000
```

図 A.12 erase の使用例

A.7. 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 カーネルブート時にカーネルに渡されます。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。オプションを指定せずに実行すると、現在の設定を表示します。

```
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.13 setenv/clearenv 構文

A.7.1. setenv/clearenv 使用例

```
hermit> setenv console=ttymxc1
hermit> setenv
1: console=ttymxc1
hermit> clearenv
hermit> setenv
hermit>
```

図 A.14 setenv と clearenv の使用例

A.7.2. Linux 起動オプション

Linux 起動オプションの例を、「表 A.1. よく使用される Linux 起動オプション」に示します。この他のオプションについては、`linux-2.6/Documentation/kernel-parameters.txt` を参照してください。

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	カーネルコンソールとして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 <code>initrd</code> 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。



console オプションに `ttymxc1,2,4` を指定すると、次回起動時から Hermit-At が使用するシリアルインターフェースも変更されます。

A.8. setbootdevice

Linux カーネルを格納しているブートデバイスを指定するコマンドです。この設定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：setbootdevice flash
 説明：フラッシュメモリの kernel リージョンに格納されたカーネルイメージを RAM に展開してブートします

構文：setbootdevice tftp <クライアント IP アドレス> <サーバー IP アドレス> [--kernel=<path>] [--userland=<path>]
 説明：TFTP サーバーに置かれたカーネルまたは/およびユーザーランドイメージを取得し、RAM に展開してブートします

構文：setbootdevice mmcblk0pN
 説明：MMC/SD カードのパーティション N の /boot/ ディレクトリに置かれたカーネルイメージを RAM に展開してブートします

図 A.15 setbootdevice 構文

A.8.1. setbootdevice の使用例

フラッシュメモリに格納されたカーネルイメージでブートするには、「図 A.16. ブートデバイスにフラッシュメモリを指定する」のようにコマンドを実行します。

```
hermit> setbootdevice flash
```

図 A.16 ブートデバイスにフラッシュメモリを指定する

TFTP サーバー(IP アドレス: 192.168.10.10)に置かれた linux.bin.gz というファイル名のカーネルイメージを取得してブートするには、「図 A.17. ブートデバイスに TFTP サーバーを指定する」のようにコマンドを実行します。

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

図 A.17 ブートデバイスに TFTP サーバーを指定する

SD/MMC カードのパーティション 1 に格納されたカーネルイメージでブートするには、「図 A.18. ブートデバイスに SD/MMC カードを指定する」のようにコマンドを実行します。

```
hermit> setbootdevice mmcblk0p1
```

図 A.18 ブートデバイスに SD/MMC カードを指定する

A.9. 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。
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。
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。

frob コマンド	説明
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。
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。
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。

A.10. tftpd

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 : tftpd <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> [オプション]...
 説明 : 自 IP アドレスをクライアント IP アドレスに設定し、サーバー IP アドレスで指定された TFTP サーバーに置かれたイメージをダウンロードし、フラッシュメモリに書き込みます。

図 A.19 tftpd 構文

表 A.3 tftpd オプション

オプション	説明
--bootloader=filepath	bootloader リージョンに書き込むファイルを filepath で指定します。
--kernel=filepath	kernel リージョンに書き込むファイルを filepath で指定します。
--userland=filepath	userland リージョンに書き込むファイルを filepath で指定します。
--fake	ファイルのダウンロードだけを行い、フラッシュメモリには書き込まないように指定します。

A.10.1. tftpd の使用例

```

hermit> tftpd 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

programming: kernel
#####

completed!!
    
```

図 A.20 tftpd の使用例

A.11. tftpboot

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、RAM に展開してカーネルをブートするコマンドです。tftpd と異なり、フラッシュメモリの書き換えを行いません。また、setbootdevice で tftp を指定したときと異なり、設定は保存されません。

構文 : tftpboot <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> [オプション]...
 説明 : 自 IP アドレスをクライアント IP アドレスに設定し、サーバー IP アドレスで指定された TFTP サーバーに置かれたイメージをダウンロードし、RAM に展開したあとブートします。

図 A.21 tftpboot 構文

オプションには、「表 A.3. tftpd オプション」と同じものを指定することができます。--fake オプションを指定したときは、ファイルのダウンロードだけを行い、カーネルのブートを行いません。

A.11.1. tftpboot の使用例

```
hermit> tftpboot 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

Uncompressing kernel:net.....done.
Uncompressing ramdisk.....
.....
.....
.....done.
Linux version 2.6.26-at6 (2.6.26) (atmark@sv-build) (gcc version 4.3.2 (Debian
4.3.2-1.1) ) #6 PREEMPT Wed Mar 10 19:19:13 JST 2010
:
:
```

図 A.22 tftpboot の使用例

- ❶ カーネルおよびユーザーランドのイメージ RAM 上に展開しています。

- ② カーネルがブートされ、カーネルの起動ログが表示されます。

A.12. boot

setbootdevice で指定されたブートデバイスから Linux カーネルをブートするコマンドです。

構文 : boot

図 A.23 boot 構文

A.12.1. boot 使用例

```

hermit> boot
Uncompressing kernel.....done.
Uncompressing ramdisk.....done.
.....done.
Doing console=ttymxc1
Linux version 2.6.26-at6 (2.6.26) (atmark@sv-build) (gcc version 4.3.2 (Debian
4.3.2-1.1) ) #6 PREEMPT Wed Mar 10 19:19:13 JST 2010
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
Machine: Armadillo-440
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 32512
Kernel command line: console=ttymxc1
MXC IRQ initialized
:
:
    
```

- ①
- ②
- ③

図 A.24 boot の使用例

- ① カーネルおよびユーザーランドのイメージ RAM 上に展開しています。
- ② setenv でカーネル起動オプションを設定している場合、ここで表示されます。ここまでは Hermit-At が表示しています。
- ③ カーネルがブートされ、カーネルの起動ログが表示されます。

A.13. バージョンに関する注意

Armadillo-440 の基板リビジョン Rev.C1 以降 (S/N 100201-2195 以降) に hermit-at v2.0.0 をベースに生成したブートローダーイメージ(loader-armadillo4x0-v2.0.0.bin 等)及び linux-2.6.26-at7

をベースに生成したカーネルイメージ(linux-a400-1.00.bin.gz) を書き込むと、カーネルが起動しない不具合があります。この問題は、Rev.C1 以降の基板、hermit-at v2.0.0 及び linux-2.6.26-at7 という組み合わせのみで発生し、それ以外の組み合わせでは発生しません。

Armadillo-440 の基板リビジョン Rev.C1 以降 (S/N 100201-2195 以降) には、hermit-at v2.0.1 以降をベースに生成したブートローダーイメージ(loader-armadillo4x0-v2.0.1.bin 以降)を使用してください。^[1]

^[1]Armadillo-440 Rev.C1 以降は、出荷状態で loader-armadillo4x0-v2.0.1.bin 以降が書き込まれています。

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2010/03/12	<ul style="list-style-type: none"> ・ 初版発行
1.1.0	2010/04/28	<ul style="list-style-type: none"> ・ 「1. はじめに」、「2. システム概要」、「3. 作業の前に」、「5. フラッシュメモリの書き換え方法」、「6. ビルド」、「7. カーネル/ユーザーランドの配置」に Armadillo-420 に関する情報追記 ・ 「2.3. Armadillo-440 液晶モデル基本仕様」に CON14 3/4 ピンの機能変更について追記 ・ 「図 6.10. Hermit-At ソースアーカイブの展開」のディレクトリ名誤記修正 ・ 「表 7.2. カーネルイメージのダウンロード先 URL」、「図 7.5. カーネルイメージの配置」、「表 7.3. Debian アーカイブのダウンロード先 URL」、「図 7.6. Debian アーカイブによるルートファイルシステムの構築例」、「表 7.4. Atmark-Dist イメージのダウンロード先 URL」、「図 7.7. Atmark-Dist イメージによるルートファイルシステムの構築例」の URL 誤記修正 ・ 「A.13. バージョンに関する注意」に基板リビジョン Rev.C1 使用時の注意書き追加
1.2.0	2010/06/08	<ul style="list-style-type: none"> ・ 「表 2.3. Armadillo-420 ベーシックモデル拡張インターフェースピン配置」及び「表 2.5. Armadillo-440 液晶モデル拡張インターフェースピン配置」の CON9 5 の機能に関する誤記修正 ・ 「6.1.4. イメージをカスタマイズする」コンフィギュレーションの変更方法について追記 ・ 「8. Linux カーネルデバイスドライバー仕様」カーネルコンフィギュレーションで設定可能なドライバについて追記

Armadillo-400 シリーズソフトウェアマニュアル
Version 1.2.0
2010/06/08

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
