Armadillo-400 シリーズ
ソフトウェアマニュアル

Version 2.0.0
2015/10/26
linux-3.14-at 対応
Armadillo-420 対応

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

Armadillo 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

札幌本社

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

横浜営業所

〒 221-0835 横浜市神奈川区鶴屋町 3 丁目 30-4 明治安田生命横浜西口ビル 7F
TEL 045-548-5651 FAX 050-3737-4597

Version 2.0.0
2015/10/26

目次

図目次

表目次

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 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。基本機能としてシリアル、Ethernet、USB、ストレージ(microSD/SD)、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。さらに、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-400 シリーズは単体モデルの他に、WLAN オプションモジュールを搭載したモデルも用意しており、すぐに試作開発用や評価をおこなうことが可能です。各モデルの名称と構成を、「表 1.1. Armadillo-400 シリーズのモデル」に示します。

表 1.1 Armadillo-400 シリーズのモデル

名称	構成
Armadillo-420 ベーシックモデル	Armadillo-420 + Armadillo-400 シリーズ RTC オプションモジュール
Armadillo-420 WLAN モデル(AWL13 対応)	Armadillo-420 + Armadillo-400 シリーズ WLAN オプションモジュール (AWL13 対応)

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およびハードウェア仕様が記載されているマニュアルは、モデルにより異なります。各モデルの名称と対応するマニュアルを、「表 1.2. 各モデルとマニュアルの対応」に示します。

表 1.2 各モデルとマニュアルの対応

名称	ソフトウェアの操作方法	ハードウェア仕様
Armadillo-420 ベーシックモデル	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	Armadillo-400 シリーズ ハードウェアマニュアル
Armadillo-420 WLAN モデル(AWL13 対応)	Armadillo-420 WLAN モデル開発セット (AWL13 対応) スタートアップガイド	Armadillo-400 シリーズ ハードウェアマニュアル
	Armadillo-WLAN(AWL13) ソフトウェアマニュアル	Armadillo-WLAN(AWL13) ハードウェアマニュアル

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. 本書および関連ファイルのバージョンについて

本書を含めた関連マニュアル、ソースファイルやイメージファイルなどの関連ファイルは最新版を使用することをおすすめいたします。本書を読み進める前に、Armadillo サイト(http://armadillo.atmark-techno.com)から最新版の情報をご確認ください。

1.2. 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。

1.3. 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 4 章で、開発を始めるための準備について取り上げます。

5 章から 7 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。

8 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。

9 章では、Armadillo 独自の Linux カーネルデバイスドライバー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。

1.4. 表記について

1.4.1. 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 1.3 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

1.4.2. 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 1.4 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 1.5 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

1.4.3. 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。

1.5. 謝辞

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

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

- ご使用の前に必ず製品マニュアルおよび関連資料をお読みになり、使用上の注意を守って正しく安全にお使いください。
- マニュアルに記載されていない操作・拡張などを行う場合は、弊社 Web サイトに掲載されている資料やその他技術情報を十分に理解した上で、お客様自身の責任で安全にお使いください。
- 水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。
- 本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。
- 本製品を使用して、お客様の仕様による機器・システムを開発される場合は、製品マニュアルおよび関連資料、弊社 Web サイトで提供している技術情報のほか、関連するデバイスのデータシート等を熟読し、十分に理解した上で設計・開発を行ってください。また、信頼性および安全性を確保・維持するため、事前に十分な試験を実施してください。
- 本製品は、機能・精度において極めて高い信頼性・安全性が必要とされる用途(医療機器、交通関連機器、燃焼制御、安全装置等)での使用を意図しておりません。これらの設備や機器またはシステム等に使用された場合において、人身事故、火災、損害等が発生した場合、当社はいかなる責任も負いかねます。
- 本製品には、一般電子機器用(OA 機器・通信機器・計測機器・工作機械等)に製造された半導体部品を使用しています。外来ノイズやサージ等により誤作動や故障が発生する可能性があります。万一誤作動または故障などが発生した場合に備え、生命・身体・財産等が侵害されることのないよう、装置としての安全設計(リミットスイッチやヒューズ・ブレーカー等の保護回路の設置、装置の多重化等)に万全を期し、信頼性および安全性維持のための十分な措置を講じた上でお使いください。
- 無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構

内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

2.2. 取扱い上の注意事項

本製品に恒久的なダメージをあたえないよう、取扱い時には以下のような点にご注意ください。

破損しやすい箇所
: microSD コネクタおよびそのカバーは、破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。

本製品の改造
: 本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。

電源投入時のコネクタ着脱
: 本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN, USB, SD, マイク, ヘッドホン)以外へのコネクタ着脱は、絶対に行わないでください。

静電気
: 本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。

ラッチアップ
: 電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。

衝撃
: 落下や衝撃などの強い振動を与えないでください。

タッチパネルの操作
: LCD 拡張ボードのタッチパネル LCD モジュールは弾力性のある両面テープによって固定されています。液晶画面に強い力が加わった場合に両面テープがつぶれて液晶フレームと基板配線が接触する可能性があります。液晶画面を必要以上に強く押さないようご注意ください。

2.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて
: 本製品の標準出荷状態でプリインストールされている Linux 対応ソフトウェアは、個別に明示されている（書面、電子データでの通知、口頭での通知を含む）場合を除き、オープンソースとしてソースコードが提供されています。再配布等の権利については、各ソースコードに記載のライセンス形態にしたがって、お客様の責任において行使してください。また、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む）は、現状有姿（AS IS）にて提供します。お客様ご自身の責任において、使用用途・目的の適合について事前に十分な検討と試験を実施した上でお使いください。アットマークテクノは、当該ソフトウェアが特定の目的に適合すること、

[1]本書を含めた関連マニュアルで改造方法を記載している箇所および、コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2]改造やコネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。

ソフトウェアの信頼性および正確性、ソフトウェアを含む本製品の使用による結果について、お客様に対し何らの保証も行いません。

パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェアに付属する readme ファイル等をご参照ください。その他のバンドルソフトウェアについては、各提供元にお問い合わせください。

2.4. 書込み禁止領域について

EEPROM、CPLD および i.MX257 内蔵エレクトリカルヒューズ(e-Fuse)のデータは、本製品に含まれるソフトウェアで使用しています。正常に動作しなくなる可能性があるため、書込みを行わないでください。また、意図的に書込みを行った場合は保証対象外となります。

2.5. 電波障害について

Armadillo-400 シリーズは、クラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。VCCI-A

2.6. 保証について

本製品の本体基板は、製品に添付もしくは弊社 Web サイトに記載している「製品保証規定」に従い、ご購入から 1 年間の交換保証を行っています。添付品およびソフトウェアは保証対象外となりますのでご注意ください。

製品保証規定 http://www.atmark-techno.com/support/warranty-policy

2.7. 輸出について

- 当社製品は、原則として日本国内での使用を想定して開発・製造されています。
- 海外の法令および規則への適合については当社はなんらの保証を行うものではありません。
- 当社製品を輸出するときは、輸出者の責任において、日本国および関係する諸外国の輸出関連法令に従い、必要な手続を行っていただきますようお願いいたします。
- 日本国およびその他関係諸国による制裁または通商停止を受けている国家、組織、法人または個人に対し、当社製品を輸出、販売等することはできません。
- 当社製品および関連技術は、大量破壊兵器の開発等の軍事目的、その他国内外の法令により製造・使用・販売・調達が禁止されている機器には使用することができません。

2.8. 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。™、®マークは省略しています。
- SD、SDHC、SDXC、microSD、microSDHC、microSDXC、SDIO ロゴは SD-3C, LLC の商標です。

3. 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。

3.1. 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 3.1. Armadillo-400 シリーズ基本仕様」に示します。また、Armadillo-420 のブロック図を「図 3.1. Armadillo-420 ブロック図」に示します。

表 3.1 Armadillo-400 シリーズ基本仕様

	Armadillo-420
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)
ROM	NOR フラッシュメモリ：16MByte (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「9. Linux カーネル仕様」をご参照ください。

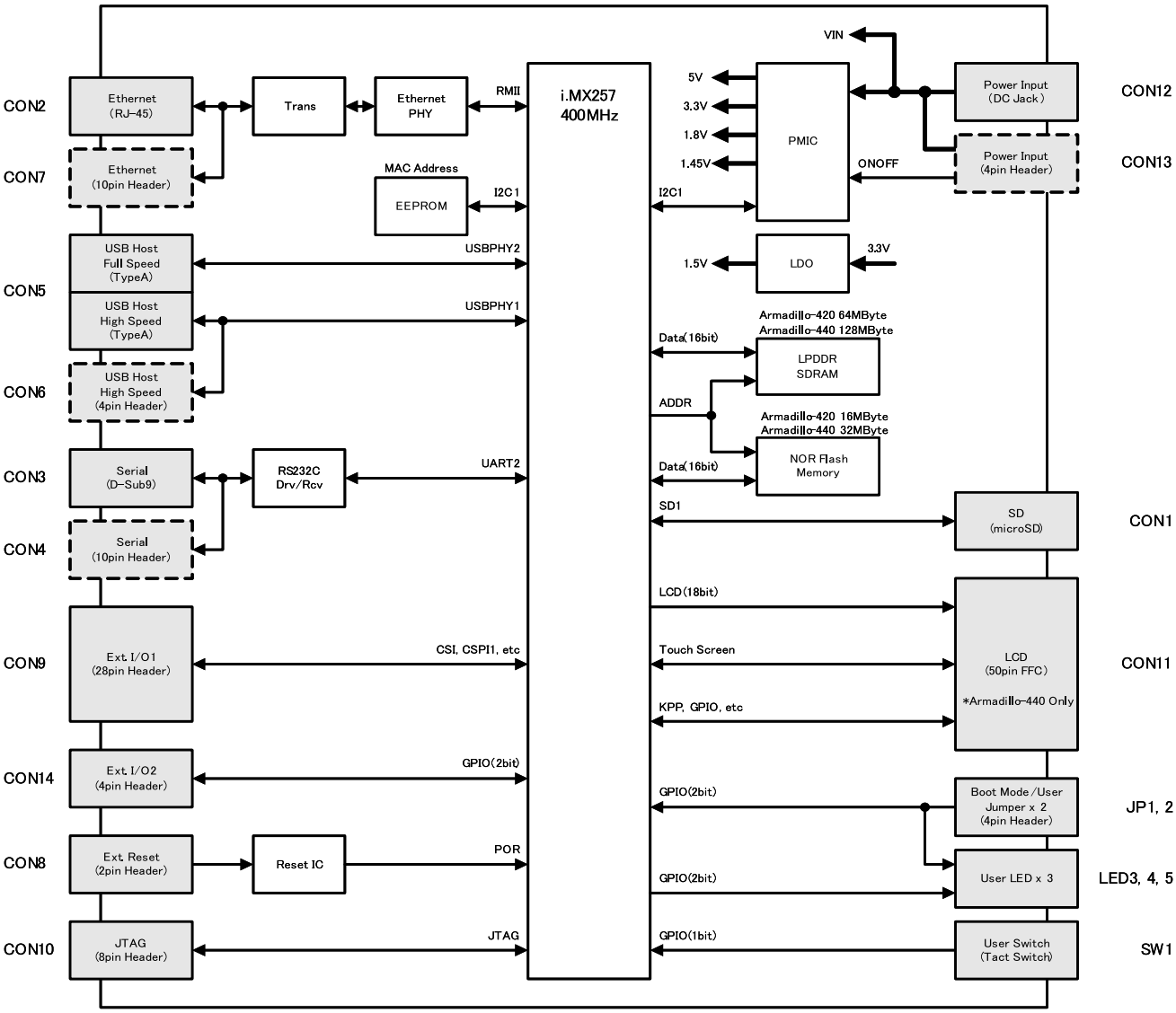


図 3.1 Armadillo-420 ブロック図

3.2. 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 シリーズ 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 3.2. RTC オプションモジュール基本仕様」に示します。

表 3.2 RTC オプションモジュール基本仕様

	Armadillo-400 シリーズ RTC オプションモジュール
リアルタイムクロック	電源切断後も一定時間動作可能

リアルタイムクロックのバックアップ時間は、RTC オプションモジュールの型番によって異なります。また、外部バッテリを接続することで長時間電源が切断されても時刻データを保持させることが可能です。詳細な仕

様については「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-420 ベーシックモデルの見取り図を「図 3.2. Armadillo-420 ベーシックモデル見取り図」に示します。また、標準イメージにおける、Linux カーネル起動後の拡張インターフェース(CON9 および CON14)の各ピンの状態を「表 3.3. Armadillo-420 ベーシックモデル拡張インターフェースデフォルト状態」に示します[2]。各インターフェースの配置場所等を確認してください。

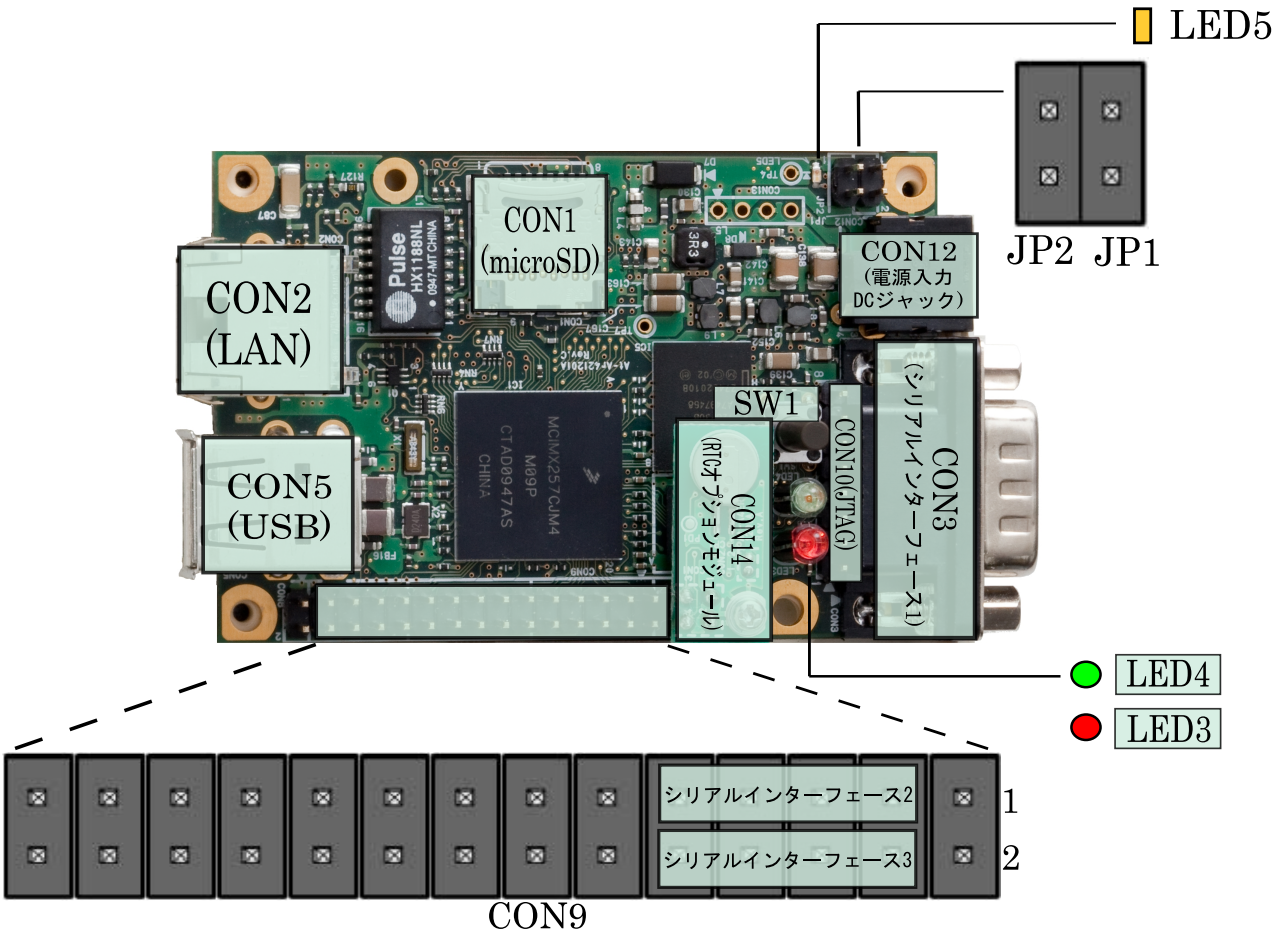


図 3.2 Armadillo-420 ベーシックモデル見取り図

表 3.3 Armadillo-420 ベーシックモデル拡張インターフェースデフォルト状態

ピン番号	機能	Input/ Output	Open Drain	Pull/Keeper[a]	Slew Rate	Drive Strength
CON14 1[b]	+3.3V_IO	Power	-	-	-	-
CON14 2	GND	Power	-	-	-	-
CON14 3	I2C2_SCL	Input/ Output	Enabled	22kΩ PU[c]	Slow	Std.
CON14 4	I2C2_SDA	Input/ Output	Enabled	22kΩ PU[c]	Slow	Std.
CON9 1	GPIO3_17	Input	Disabled	100kΩ PU	Slow	Std.
CON9 2	GPIO3_14	Input	Disabled	100kΩ PU	Slow	Std.

[2]Linux カーネル起動以前の状態に関しては、「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。

ピン番号	機能	Input/ Output	Open Drain	Pull/Keeper[a]	Slew Rate	Drive Strength
CON9 3	シリアルインターフェース 2 UART3_RXD	Input	Disabled	100kΩ PU	Slow	Std.
CON9 4	シリアルインターフェース 3 UART5_RXD	Input	Disabled	100kΩ PU	Slow	Std.
CON9 5	シリアルインターフェース 2 UART3_TXD	Output	Disabled	Disabled	Slow	Std.
CON9 6	シリアルインターフェース 3 UART5_TXD	Output	Disabled	Disabled	Slow	Std.
CON9 7	+3.3V_IO	Power	-	-	-	-
CON9 8	+3.3V_IO	Power	-	-	-	-
CON9 9	GND	Power	-	-	-	-
CON9 10	GND	Power	-	-	-	-
CON9 11	GPIO1_17	Input	Disabled	100kΩ PU	Slow	Std.
CON9 12	GPIO1_29	Input	Disabled	100kΩ PU	Slow	Std.
CON9 13	GPIO1_18	Input	Disabled	100kΩ PU	Slow	Std.
CON9 14	GPIO1_30	Input	Disabled	100kΩ PU	Slow	Std.
CON9 15	GPIO1_7	Input	Disabled	100kΩ PU	Slow	Std.
CON9 16	GPIO1_31	Input	Disabled	100kΩ PU	Slow	Std.
CON9 17	GPIO4_21	Input	Disabled	100kΩ PU	Slow	Std.
CON9 18	GPIO1_6	Input	Disabled	100kΩ PU	Slow	Std.
CON9 19	GND	Power	-	-	-	-
CON9 20	+3.3V_IO	Power	-	-	-	-
CON9 21	GPIO1_8	Input	Disabled	100kΩ PU	Slow	Std.
CON9 22	GPIO1_9	Input	Disabled	100kΩ PU	Slow	Std.
CON9 23	GPIO1_10	Input	Disabled	100kΩ PU	Slow	Std.
CON9 24	GPIO1_11	Input	Disabled	100kΩ PU	Slow	Std.
CON9 25	GPIO1_16	Input	Disabled	100kΩ PU	Slow	Std.
CON9 26	GPIO2_22	Input	Disabled	100kΩ PU	Slow	Std.
CON9 27	GPIO2_21	Output Low	Disabled	Disabled	Fast	Std.
CON9 28	GPIO3_15	Output Low	Disabled	Disabled	Fast	Std.

[a]PD=プルダウン、PU=プルアップ。
[b]RTC オプションモジュールの CON1 1 ピンに接続。以下、CON14 4 まで、RTC オプションモジュールと接続されます。
[c]RTC オプションモジュールで 1kΩ PU。

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 (黄色または緑に着色されたケーブル)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。

[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。

3.3. Armadillo-420 WLAN モデル(AWL13 対応)基本仕様

Armadillo-420 WLAN モデル(AWL13 対応)は、Armadillo-420 に Armadillo-400 シリーズ WLAN オプションモジュール(AWL13 対応)(以下、WLAN オプションモジュール(AWL13 対応))を接続したモデルです。WLAN オプションモジュール(AWL13 対応)の基本仕様を、「表 3.4. WLAN オプションモジュール(AWL13 対応)基本仕様」に示します。

表 3.4 WLAN オプションモジュール(AWL13 対応)基本仕様

	Armadillo-400 シリーズ WLAN オプションモジュール(AWL13 対応)
無線 LAN 規格	IEEE802.11b, IEEE802.11g, IEEE802.11n, IEEE802.11i
送受信周波数	2400MHz ～ 2483.5MHz(ch1 ～ 13)
アクセス方式	インフラストラクチャモード(STA[a]、AP[b][c])、アドホックモード
セキュリティ方式	64bit/128bit WEP, TKIP, AES
リアルタイムクロック	電源切断後も一定時間動作可能

[a]STA=ステーション。

[b]ファームウェア v4.3.2 以降で有効です。対応するデバイスドライバは v3.0.2 以降です。

[c]AP=アクセスポイント

リアルタイムクロックは、外部バッテリを接続することで長時間電源が切断されても時刻データを保持させることが可能です。詳細な仕様については「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-420 WLAN モデル(AWL13 対応)の見取り図を「図 3.3. Armadillo-420 WLAN モデル(AWL13 対応)見取り図」に示します。また、標準イメージにおける、Linux カーネル起動後の拡張インターフェース(CON9 および CON14)の各ピンの状態を「表 3.5. Armadillo-420 WLAN モデル(AWL13 対応)拡張インターフェースデフォルト状態」に示します[4]。各インターフェースの配置場所等を確認してください。

[4]Linux カーネル起動以前の状態に関しては、「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。

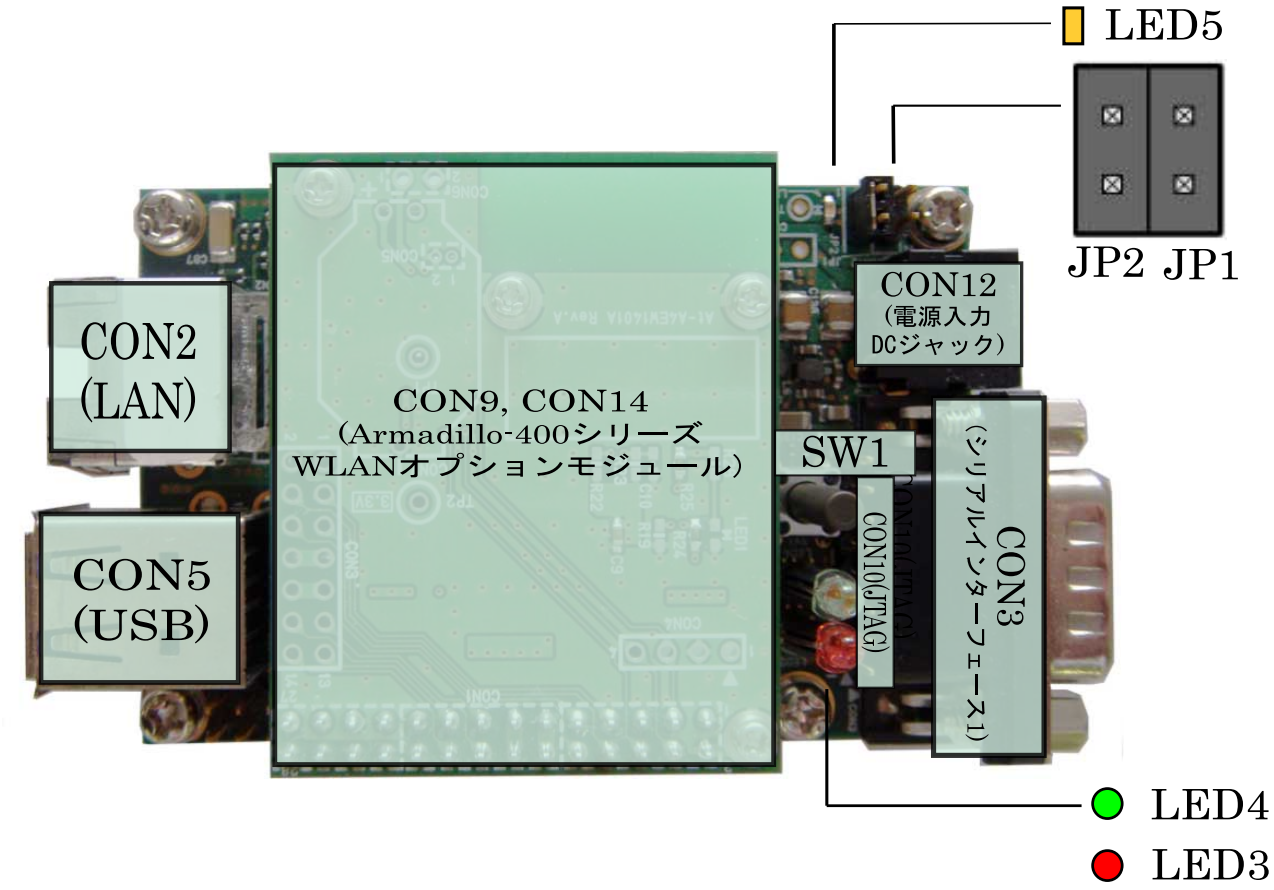


図 3.3 Armadillo-420 WLAN モデル(AWL13 対応)見取り図

表 3.5 Armadillo-420 WLAN モデル(AWL13 対応)拡張インターフェースデフォルト状態

ピン番号	機能	Input/ Output	Open Drain	Pull/ Keeper[a]	Slew Rate	Drive Strength
CON14 1[b]	+3.3V_IO	Power	-	-	-	-
CON14 2	GND	Power	-	-	-	-
CON14 3	I2C2_SCL	Input/ Output	Enabled	22kΩ PU[c]	Slow	Std.
CON14 4	I2C2_SDA	Input/ Output	Enabled	22kΩ PU[c]	Slow	Std.
CON9 1	SDHC2_PWREN(GPIO3_17)	Output	Disabled	100kΩ PU[d]	Slow	Std.
CON9 2	RTC_INT1(GPIO3_14)	Input	Disabled	22kΩ PU	Slow	Std.
CON9 3	GPIO1_14[e]	Input	Disabled	100kΩ PU	Slow	Std.
CON9 4	シリアルインターフェース 3 UART5_RXD	Input	Disabled	100kΩ PU	Slow	Std.
CON9 5	GPIO1_15[e]	Input	Disabled	100kΩ PU	Slow	Std.
CON9 6	シリアルインターフェース 3 UART5_TXD	Output	Disabled	Disabled	Slow	Std.
CON9 7	+3.3V_IO	Power	-	-	-	-
CON9 8	+3.3V_IO	Power	-	-	-	-
CON9 9	GND	Power	-	-	-	-
CON9 10	GND	Power	-	-	-	-
CON9 11	GPIO1_17[e]	Input	Disabled	100kΩ PU	Slow	Std.
CON9 12	GPIO1_29[e]	Input	Disabled	100kΩ PU	Slow	High

ピン番号	機能	Input/ Output	Open Drain	Pull/ Keeper[a]	Slew Rate	Drive Strength
CON9 13	GPIO1_18[e]	Input	Disabled	100kΩ PU	Slow	Std.
CON9 14	GPIO1_30[e]	Input	Disabled	100kΩ PU	Slow	High
CON9 15	SDHC2_WP(GPIO1_7)	Input	Disabled	100kΩ PU[f]	Slow	High
CON9 16	SDHC2_CMD	Input/ Output	Disabled	Disabled[g]	Fast	High
CON9 17	SDHC2_CD(GPIO4_21)	Input	Disabled	100kΩ PU[f]	Slow	High
CON9 18	SDHC2_CLK	Output	Disabled	Disabled	Fast	High
CON9 19	GND	Power	-	-	-	-
CON9 20	+3.3V_IO	Power	-	-	-	-
CON9 21	SDHC2_DATA0	Input/ Output	Disabled	Disabled[g]	Fast	High
CON9 22	SDHC2_DATA1	Input/ Output	Disabled	Disabled[g]	Fast	High
CON9 23	SDHC2_DATA2	Input/ Output	Disabled	Disabled[g]	Fast	High
CON9 24	SDHC2_DATA3	Input/ Output	Disabled	Disabled[g]	Fast	High
CON9 25	GPIO1_16	Input	Disabled	100kΩ PU	Slow	Std.
CON9 26	GPIO2_22	Input	Disabled	100kΩ PU	Slow	Std.
CON9 27	GPIO2_21	Output Low	Disabled	Disabled	Fast	Std.
CON9 28	GPIO3_15	Output Low	Disabled	Disabled	Fast	Std.

[a]PD=プルダウン、PU=プルアップ

[b]WLAN オプションモジュール(AWL13 対応)の CON1 1 ピンに接続。以下、CON9 24 まで、WLAN オプションモジュール(AWL13 対応)と接続されます。

[c]WLAN オプションモジュール(AWL13 対応)で 1kΩ PU。

[d]WLAN オプションモジュール(AWL13 対応)で 1kΩ PD。

[e]WLAN オプションモジュール(AWL13 対応)では未使用。

[f]WLAN オプションモジュール(AWL13 対応)で 10kΩ PD。

[g]WLAN オプションモジュール(AWL13 対応)で 47kΩ PU。

3.4. メモリマップ

Armadillo-400 シリーズは、標準で「表 3.6. Armadillo-420 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 3.6 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 | 0xa041ffff	kernel	4MB	カーネルイメージを格納します
0xa0420000 | 0xa0efffff	userland	10.875MB	ユーザーランドイメージを格納します
0xa0f00000 | 0xa0ffffff	config	1MB	設定情報を保存します

3.5. 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。

3.5.1. 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。

3.5.2. カーネル

Armadillo-400 シリーズは、Linux カーネルを使用しています。

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/SD)または TFTP サーバー上にも配置することができます。

3.5.3. 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[5] 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/SD/USB) または NFS サーバー[6]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、「8. カーネル/ユーザーランドの配置」で詳しく説明します。

3.5.4. 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。

[5]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。
[6]カーネルで NFS サポートを有効にした場合

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。

3.6. 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リージョン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、「6.6. ブートローダーを出荷状態に戻す」 を参照してください。

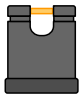
Armadillo-400 シリーズの各ジャンパ設定でのブートモードを「表 3.7. ジャンパの設定」に示します。

表 3.7 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

ジャンパのオープン、ショートとは

「オープン」とはジャンパピンにジャンパソケットを接続していない状態です。



「ショート」とはジャンパピンにジャンパソケットを接続している状態です。

4. 作業の前に

4.1. 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]。

4.2. 接続方法

「図 4.1. Armadillo-420 ベーシックモデル接続例」、または「図 4.2. Armadillo-420 WLAN モデル(AWL13 対応)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。

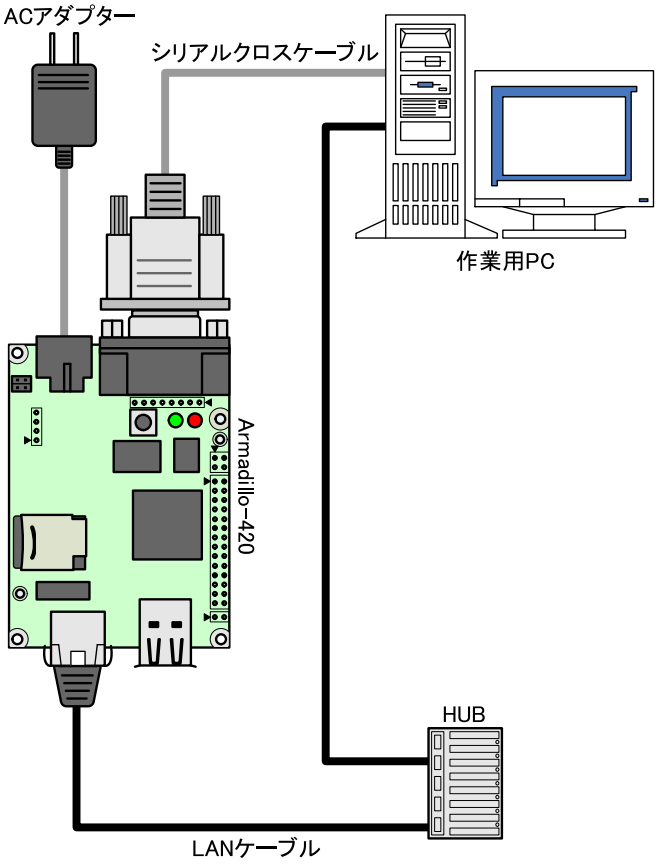


図 4.1 Armadillo-420 ベーシックモデル接続例

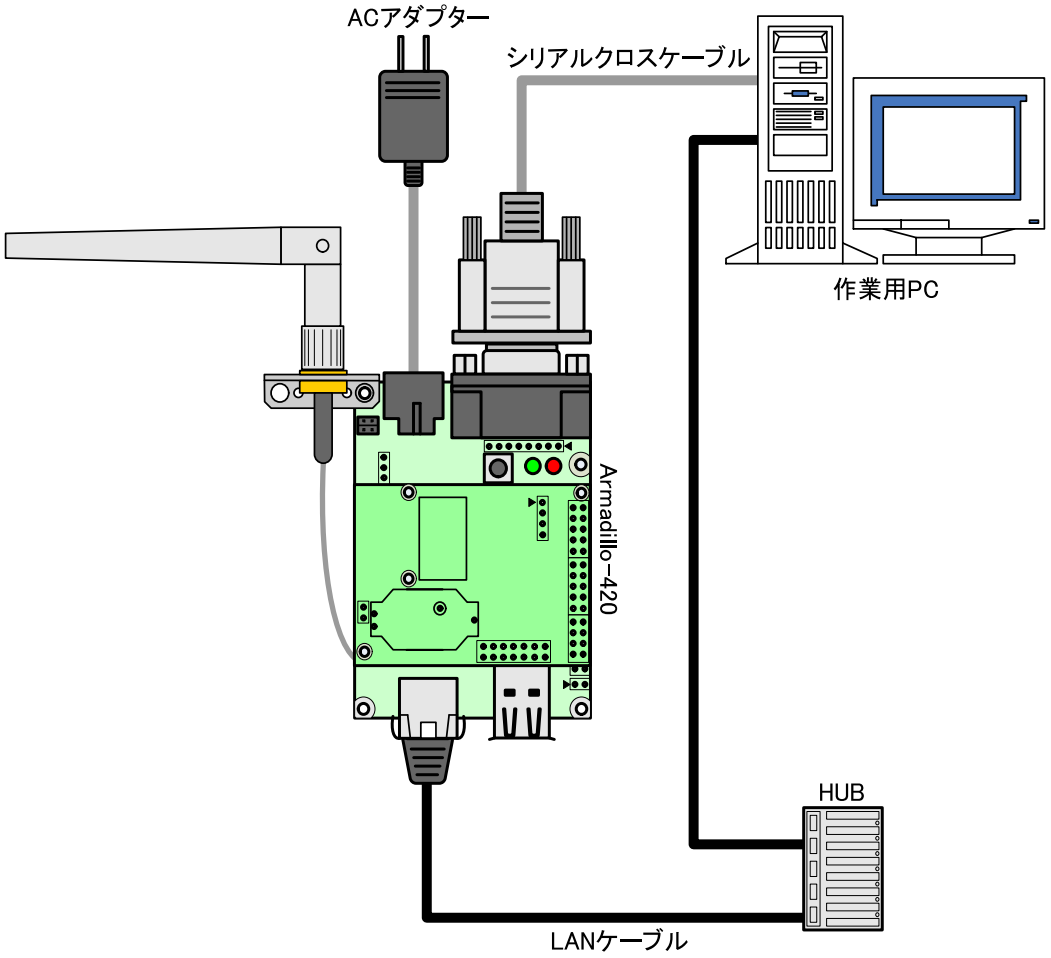


図 4.2 Armadillo-420 WLAN モデル(AWL13 対応)接続例

4.3. 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 4.1. シリアル通信設定」のように設定してください。また、シリアル通信ソフトウェアの横幅を 80 文字以上にしてください。横幅が 80 文字より小さい場合、コマンド入力中に表示が乱れることがあります。

表 4.1 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

5. 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]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。アットマークテクノでは、当社製品のソフトウェア開発や動作確認を簡単に行うために、VMware 仮想マシンのデータイメージを提供しています。この VMware 仮想マシンのデータイメージを ATDE(Atmark Techno Development Environment)と呼びます。ATDE の起動には仮想化ソフトウェアである VMware を使用します。ATDE のデータは、tar.xz 圧縮されています。環境に合わせたツールで展開してください。

5.1. VMware のインストール

ATDE5 を使用するためには、作業用 PC に VMware がインストールされている必要があります。VMware 社 Web ページ(http://www.vmware.com/)を参照し、利用目的に合う VMware 製品をインストールしてください。また、ATDE5 は tar.xz 圧縮されていますので、環境に合せたツールで展開してください。

VMware は、非商用利用限定で無償のものから、商用利用可能な有償のものまで複数の製品があります。製品ごとに異なるライセンス、エンドユーザー使用許諾契約書(EULA)が存在するため、十分に確認した上で利用目的に合う製品をご利用ください。

VMware や ATDE5 が動作しないことを未然に防ぐため、使用する VMware のドキュメントから以下の項目についてご確認ください。

- ホストシステムのハードウェア要件
- ホストシステムのソフトウェア要件
- ゲスト OS のプロセッサ要件

VMware のドキュメントは、VMware 社 Web ページ(http://www.vmware.com/)から取得することができます。

5.2. ATDE5 アーカイブの取得

「表 5.1. ATDE5 の種類」に示す ATDE5 のアーカイブのうちいずれか 1 つを作業用 PC にコピーします。ATDE5 のアーカイブは Armadillo サイト(http://armadillo.atmark-techno.com)または、開発セット付属の DVD から取得可能です。

[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。

表 5.1 ATDE5 の種類

ATDE5 アーカイブ	ベースの Debian GNU/Linux
atde5-amd64-*[version]*.tar.xz	64-bit PC(「amd64」)アーキテクチャ用 Debian GNU/Linux 7
atde5-i386-*[version]*.tar.xz	32-bit PC(「i386」)アーキテクチャ用 Debian GNU/Linux 7

本製品に対応している ATDE5 のバージョンは v20151026 以降です。

作業用 PC の動作環境(ハードウェア、VMware、ATDE5 の対応アーキテクチャなど)により、ATDE5 が正常に動作しない可能性があります。VMware 社 Web ページ(http://www.vmware.com/)から、使用している VMware のドキュメントなどを参照して動作環境を確認してください。

5.3. ATDE5 アーカイブの展開

ATDE5 のアーカイブを展開します。ATDE5 のアーカイブは、tar.xz 形式の圧縮ファイルです。

Windows での展開方法を「手順 5.1. Windows で ATDE5 のアーカイブ展開する」に、Linux での展開方法を「手順 5.2. Linux で tar.xz 形式のファイルを展開する」に示します。

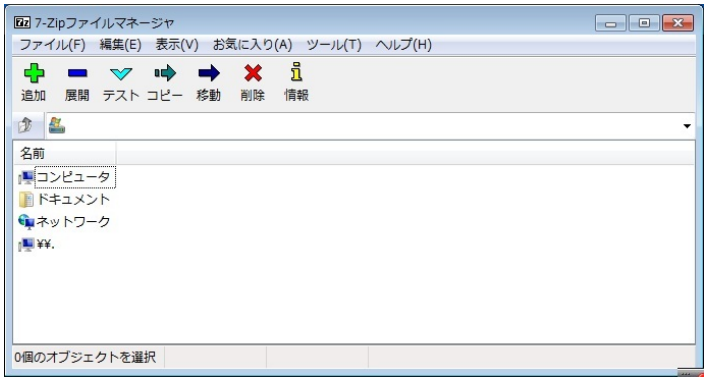
手順 5.1 Windows で ATDE5 のアーカイブ展開する

1. **7-Zip のインストール**

   7-Zip をインストールします。7-Zip は、圧縮解凍ソフト 7-Zip(http://sevenzip.sourceforge.jp)または、開発セット付属の DVD から取得可能です。

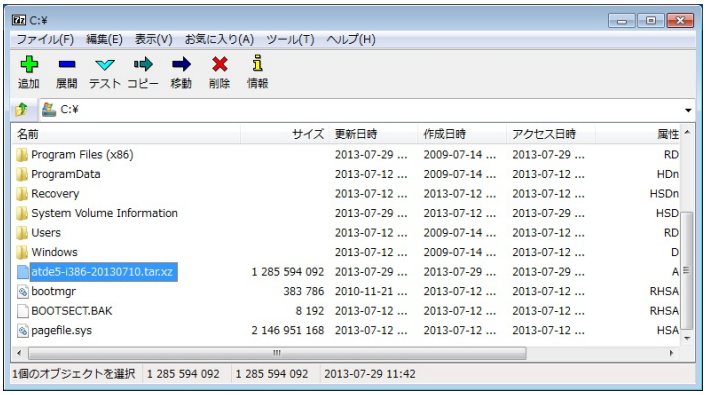
2. **7-Zip の起動**

   7-Zip を起動します。



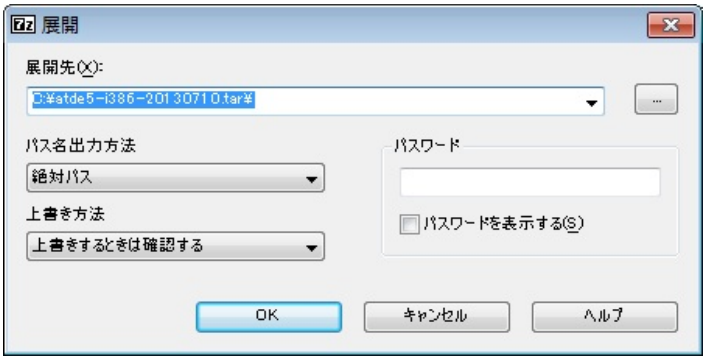
3. **xz 圧縮ファイルの選択**

   xz 圧縮ファイルを展開して、tar 形式のファイルを出力します。tar.xz 形式のファイルを選択して、「展開」をクリックします。



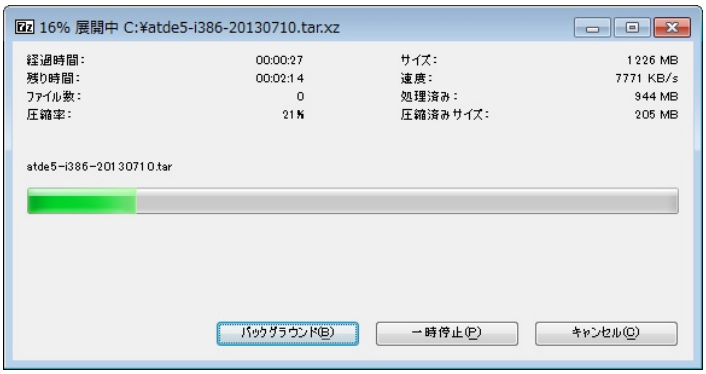
4. **xz 圧縮ファイルの展開先の指定**

   「展開先」を指定して、「OK」をクリックします。



5. **xz 圧縮ファイルの展開**

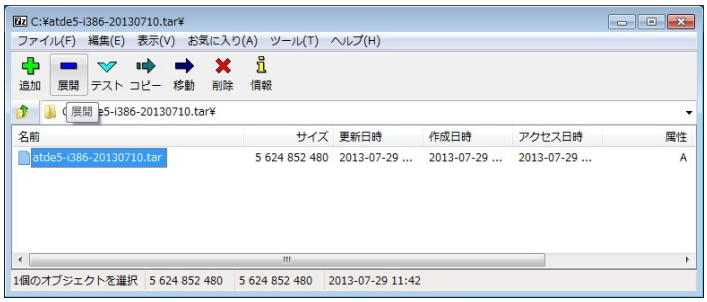
   展開が始まります。



6. **tar アーカイブファイルの選択**

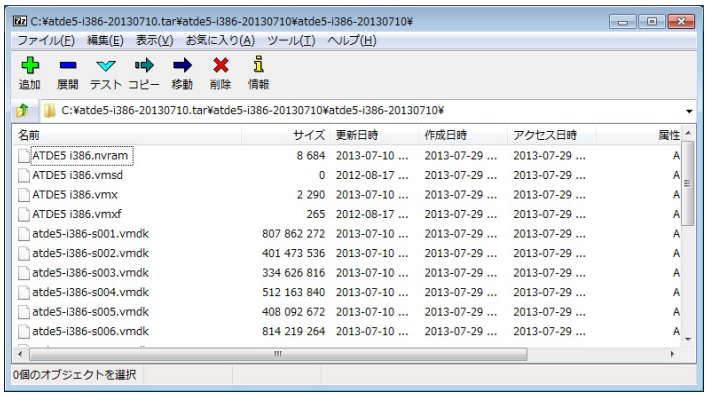
   xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。

   tar アーカイブファイルを出力したのと同様の手順で、tar アーカイブファイルから ATDE5 のデータイメージを出力します。tar 形式のファイルを選択して「展開」をクリックし、「展開先」を指定して、「OK」をクリックします。



7. 展開の完了確認

tar アーカイブファイルの展開が終了すると、ATDE5 アーカイブの展開は完了です。「展開先」に指定したフォルダに ATDE5 のデータイメージが出力されています。



手順 5.2 Linux で tar.xz 形式のファイルを展開する

1. tar.xz 圧縮ファイルの展開

tar の Jxf オプション使用して tar.xz 圧縮ファイルを展開します。

```
[ATDE ~]$ tar Jxf atde5-i386-[version].tar.xz
```

2. 展開の完了確認

tar.xz 圧縮ファイルの展開が終了すると、ATDE5 アーカイブの展開は完了です。atde5-i386-*[version]*ディレクトリに ATDE5 のデータイメージが出力されています。

```
[ATDE ~]$ ls atde5-i386-[version]/
ATDE5 i386.nvram      atde5-i386-s005.vmdk  atde5-i386-s013.vmdk
ATDE5 i386.vmsd       atde5-i386-s006.vmdk  atde5-i386-s014.vmdk
ATDE5 i386.vmx        atde5-i386-s007.vmdk  atde5-i386-s015.vmdk
ATDE5 i386.vmxf       atde5-i386-s008.vmdk  atde5-i386-s016.vmdk
atde5-i386-s001.vmdk  atde5-i386-s009.vmdk  atde5-i386-s017.vmdk
atde5-i386-s002.vmdk  atde5-i386-s010.vmdk  atde5-i386.vmdk
atde5-i386-s003.vmdk  atde5-i386-s011.vmdk
atde5-i386-s004.vmdk  atde5-i386-s012.vmdk
```

5.4. ATDE5 の起動

ATDE5 のアーカイブを展開したディレクトリに存在する仮想マシン構成(.vmx)ファイルを VMware 上で開くと、ATDE5 を起動することができます。ATDE5 にログイン可能なユーザーを、「表 5.2. ユーザー名とパスワード」に示します[2]。

表 5.2 ユーザー名とパスワード

ユーザー名	パスワード	権限
atmark	atmark	一般ユーザー
root	root	特権ユーザー

ATDE に割り当てるメモリおよびプロセッサ数を増やすことで、ATDE をより快適に使用することができます。仮想マシンのハードウェア設定の変更方法については、VMware 社 Web ページ (http://www.vmware.com/)から、使用している VMware のドキュメントなどを参照してください。

[2]特権ユーザーで GUI ログインを行うことはできません。

6. 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、「6.3. 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、「6.4. tftpdl を使用してフラッシュメモリを書き換える」および、「6.5. 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、「6.6. 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。

6.1. フラッシュメモリのリージョンについて

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 6.1. 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 6.1 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-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 WLAN モデル(AWL13 対応)	bootloader	loader-armadillo4x0-*[version]*.bin
	kernel	linux-a400-wlan-*[version]*.bin.gz
	userland	romfs-a420-wlan-awl13-*[version]*.img.gz

全てのモデルでブートローダは共通のイメージファイルを使用します。

各リージョンは、書き込みを制限することが可能です。書き込みを制限する理由は、誤動作や予期せぬトラブルにより、フラッシュメモリ上のデータが不意に破壊または消去されることを防ぐためです。

読み込みは、常時可能です。読み込みに制限を付けることはできません。

Linux が起動している場合、リージョンへの書き込み制限はコマンドで変更することが可能です。変更は、Sysfs の MTD クラスディレクトリ以下にある"ro"というファイルに"0"または"1"を書き込むことで行います。各リージョンの書き込み制限を変更することで、bootloader リージョンの書き換えを可能にしたり、kernel リージョンの書き換えを禁止したりすることができます。

各リージョンのデフォルト状態での書き込み制限の有無と、対応する MTD クラスディレクトリを以下に記載します。

表 6.2 リージョンのデフォルト状態での書き込み制限の有無と対応する MTD クラスディレクトリ

リージョン	書き込み制限	MTD クラスディレクトリ
bootloader	あり	/sys/class/mtd/mtd0
kernel	なし	/sys/class/mtd/mtd1
userland	なし	/sys/class/mtd/mtd2
config	なし	/sys/class/mtd/mtd3

以降の説明では、任意のリージョンを示す MTD クラスディレクトリを"/sys/class/mtd/[MTD]"のように表記します。

書き込み制限を解除するには、ro ファイルに 0 を書き込みます。

```
[armadillo ~]# echo 0 > /sys/class/mtd/[MTD]/ro
```

図 6.1 書き込み制限を外す

書き込みを制限するには、ro ファイルに 1 を書き込みます。

```
[armadillo ~]# echo 1 > /sys/class/mtd/[MTD]/ro
```

図 6.2 書き込みを制限する

6.2. 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 6.3. 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 6.3 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。

6.2.1. 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。

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

図 6.3 ダウンローダーのインストール（Linux）

6.2.2. 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。

6.3. 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。

6.3.1. 準備

「表 3.7. 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。

6.3.2. 作業用 PC が Linux の場合

作業用 PC が Linux の場合、**hermit** コマンドを使用し、「図 6.4. 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** 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 は書き込み対象のリージョンを指定するオプションです。下記の例では、「kernel リージョンに linux.bin.gz を書き込む」という指示になります。

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

図 6.4 ダウンロードコマンド

シリアルインターフェースが /dev/ttyS0 以外の場合は、「図 6.5. 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。

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

図 6.5 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 6.6. 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]。

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

図 6.6 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「6.6. 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。

bootloader リージョンにはブートローダーイメージの他にブートローダーのパラメータが保存されています。「図 6.6. ダウンロードコマンド（アンプロテクト）」の手順を行っただけでは、パラメータは書き換えられず、以前の設定が残ったままとなります。パラメータを初期状態に戻したい場合は、「6.7. ブートローダーのパラメータを出荷状態に戻す」に示す手順を実行してください。

6.3.3. 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 6.7. Hermit-At Win32：Download ウィンドウ」が表示されます。

[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。

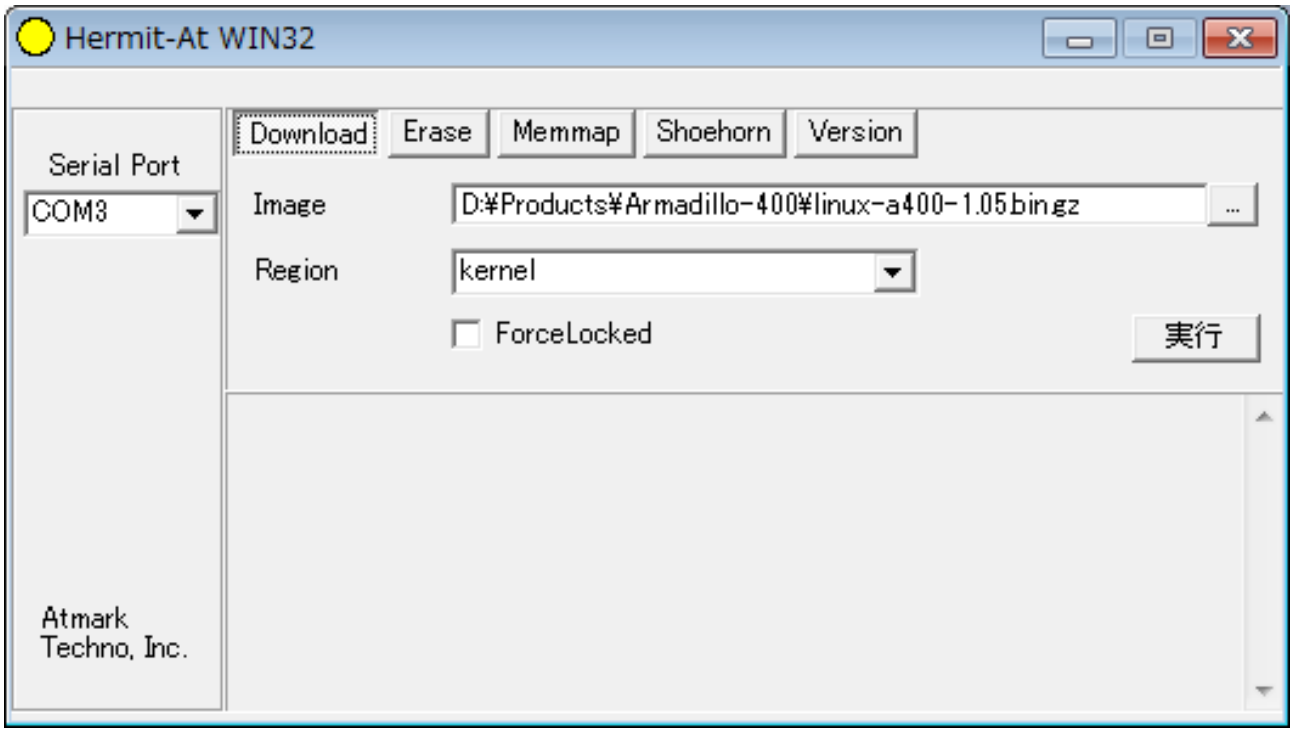


図 6.7 Hermit-At Win32：Download ウィンドウ

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 6.8. 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。

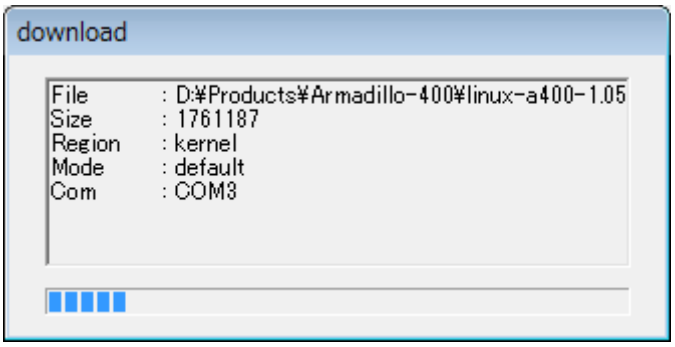


図 6.8 Hermit-At Win32：download ダイアログ

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。

bootloader リージョンにはブートローダーイメージの他にブートローダーのパラメータが保存されています。bootloader リージョンへのブートローターイメージの書き込み行っただけでは、パラメータは書き換えられず、

以前の設定が残ったままとなります。パラメータを初期状態に戻したい場合は、「6.7. ブートローダーのパラメータを出荷状態に戻す」に示す手順を実行してください。

6.4. tftpdl 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 tftpdl 機能を使用することで、ダウンローダーを使用して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。

tftpdl 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。

ATDE5 は、標準で TFTP サーバー (atftpd) が動作しています。/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。

tftpdl 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 6.9. tftpdl コマンド例」は、Armadillo の IP アドレスを 192.168.10.10 に設定し、IP アドレスが 192.168.10.1 の TFTP サーバー上にある、linux.bin.gz を kernel リージョンに書き込む例です。

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

図 6.9 tftpdl 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 6.4. リージョンとオプションの対応」に示します。

表 6.4 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

bootloader リージョンにはブートローダーイメージの他にブートローダーのパラメータが保存されています。「図 6.9. tftpdl コマンド例」の手順を行っただけでは、パラメータは書き換えられず、以前の設定が残ったまま

となります。パラメータを初期状態に戻したい場合は、「6.7. ブートローダーのパラメータを出荷状態に戻す」に示す手順を実行してください。

6.5. 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。

ATDE5 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 6.10. 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 6.10 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 6.5. リージョンとデバイスファイルの対応」を参照してください。その他のオプションについては、netflash -h で詳細を確認する事ができます。

表 6.5 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
bootloader[a]	/dev/flash/bootloader
kernel	/dev/flash/kernel
userland	/dev/flash/userland
config	/dev/flash/config

[a]デフォルト状態では書き込みが制限されています。詳細については「6.1. フラッシュメモリのリージョンについて」を参照してください。

6.6. 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リージョン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。

6.6.1. 準備

Armadillo のジャンパを、「表 3.7. 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。

6.6.2. 作業用 PC が Linux の場合

「図 6.11. shoehorn コマンド例」のようにコマンドを実行してください。「図 6.12. 電源投入タイミング」のログが表示されたら、Armadillo に電源を投入し、起動させてください。

```
[ATDE ~]$ 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 6.11 shoehorn コマンド例

```
Waiting for target - press Wakeup now.
```

図 6.12 電源投入タイミング

Armadillo に電源を投入すると、「図 6.13. 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 6.13 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。以降の手順は、ジャンパの設定変更や電源の切断をせずにおこなう必要があります。

「図 6.14. ブートローダの書き込みコマンド例」のようにブートローダの書き込みを行ってください[2]。

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

図 6.14 ブートローダの書き込みコマンド例

6.6.3. 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 6.15. Hermit-At Win32：Shoehorn ウィンドウ」が表示されます。

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

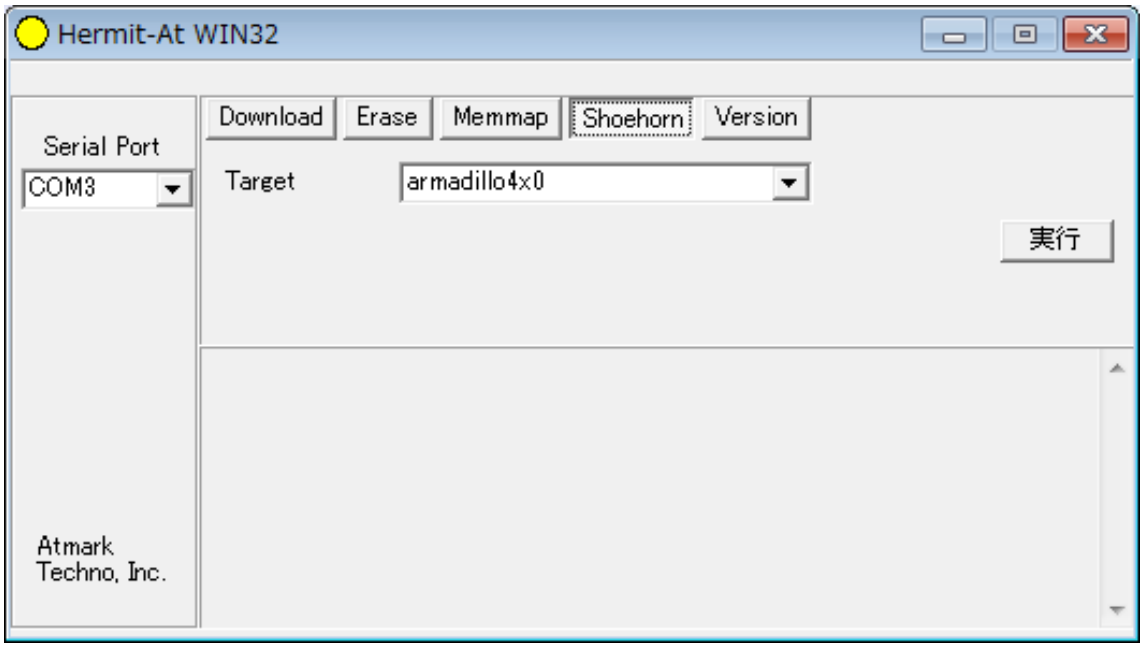


図 6.15 Hermit-At Win32：Shoehorn ウィンドウ

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

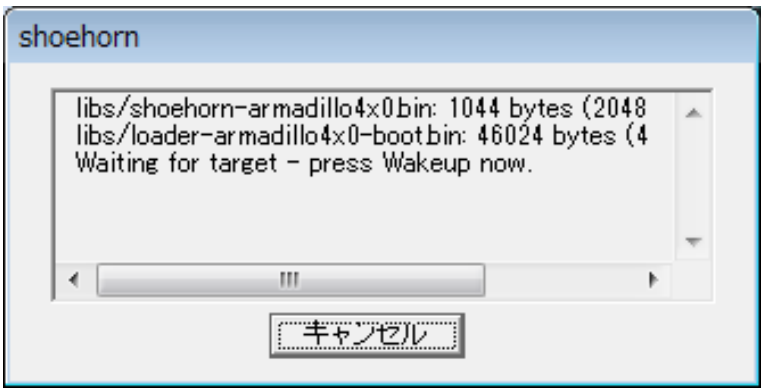


図 6.16 Hermit-At Win32：shoehorn ダイアログ

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。以降の手順は、ジャンパの設定変更や電源の切断をせずにおこなう必要があります。

ダウンロードをおこなう前に、一旦ブートローダリージョンを削除します。Erase ボタンをクリックすると、「図 6.17. Hermit-At Win32：Erase ウィンドウ」が表示されます。

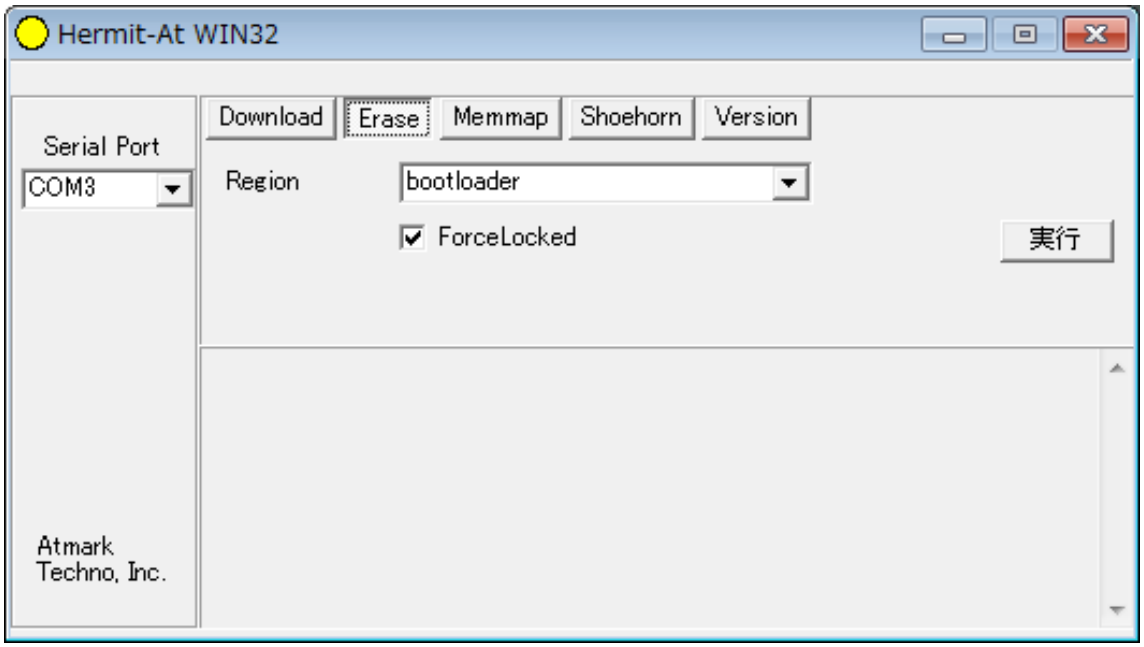


図 6.17 Hermit-At Win32：Erase ウィンドウ

Region に bootloader リージョンを選択し、Force Locked をチェックして実行ボタンをクリックします。ブートローダリージョンの削除中は、「図 6.18. Hermit-At Win32：Erase ダイアログ」が表示され、削除の設定と進捗状況を確認することができます。

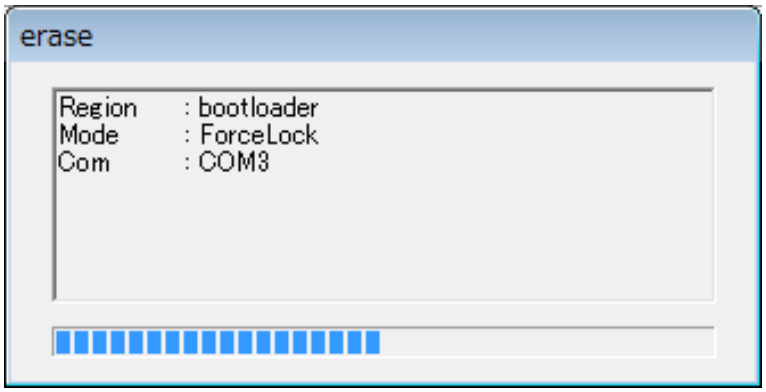


図 6.18 Hermit-At Win32：Erase ダイアログ

ブートローダリージョンの削除が完了すると、ダイアログはクローズされます。次にダウンロードをおこないます。Download ボタンをクリックすると、「図 6.19. Hermit-At Win32：Download ウィンドウ(Erase 後)」が表示されます。

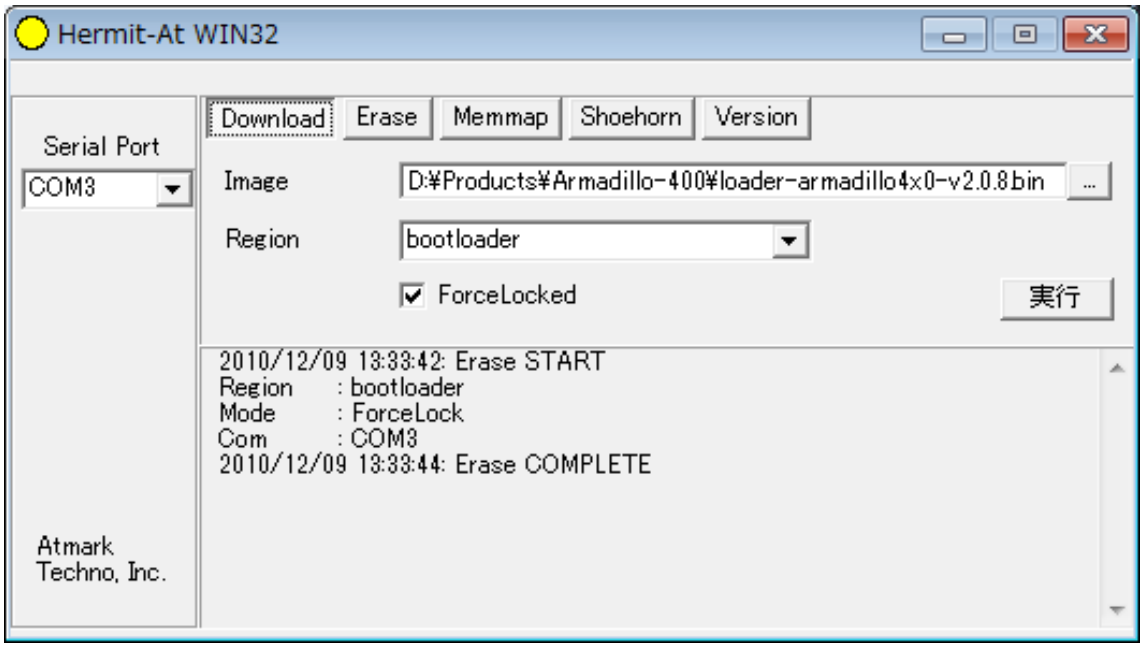


図 6.19 Hermit-At Win32：Download ウィンドウ(Erase 後)

Image にはブートローダイメージファイルを、Region には bootloader を指定し、Force Locked をチェックして実行ボタンをクリックします。ダウンロード中は、「図 6.20. Hermit-At Win32：Download ダイアログ(bootloader)」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。

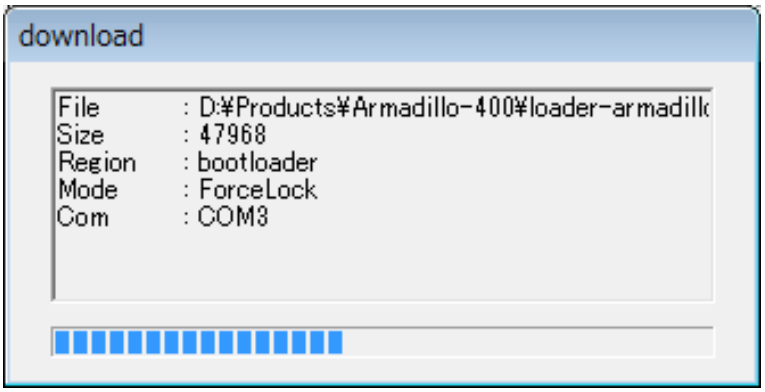


図 6.20 Hermit-At Win32：Download ダイアログ(bootloader)

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。

6.7. ブートローダーのパラメータを出荷状態に戻す

フラッシュメモリの bootloader リージョンには、ブートローダーイメージの他にブートローダーのパラメータが保存されています。これは、Armadillo の再起動後も設定を有効にするためです。パラメータと初期設定の対応を、「表 6.6. ブートローダーのパラメータ」に示します。

表 6.6 ブートローダーのパラメータ

パラメータ	初期設定	説明
Linux カーネルパラメータ	無し	Linux カーネル起動時にカーネルに渡すパラメータ
ブートデバイス	フラッシュメモリ	Linux カーネルを格納しているデバイスを指定する

ブートローダーのパラメータを出荷状態に戻すには、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。Linux カーネルパラメータを初期設定に戻すには、「図 6.21. Linux カーネルパラメータを初期設定に戻す」のようにコマンドを実行してください。[3]

```
hermit> clearenv
```

図 6.21 Linux カーネルパラメータを初期設定に戻す

ブートデバイスを初期設定のフラッシュメモリに戻すには、「図 6.22. ブートデバイスを初期設定に戻す」のようにコマンドを入力してください。[3]

```
hermit> setbootdevice flash
```

図 6.22 ブートデバイスを初期設定に戻す

[3]「6.6. ブートローダーを出荷状態に戻す」の手順を実行すると、パラメータが初期化されますので、この手順は必要ありません。但し、Hermit-AT Win32 v1.2.0 以前のバージョンを使用した場合、自動ではパラメータが初期化されないため、本手順を実行する必要があります。

7. ビルド手順

本章では、工場出荷イメージと同じイメージを作成する手順について説明します。

使用するソースコードは、開発セット付属の DVD に収録されています。最新版のソースコードは、Armadillo サイトからダウンロードすることができます。新機能の追加や不具合の修正などが行われているため、DVD に収録されているものよりも新しいバージョンがリリースされているかを確認して、最新バージョンのソースコードを利用することを推奨します。

Armadillo サイト - Armadillo-420 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-420/downloads

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。

7.1. Linux カーネル/ユーザーランドをビルドする

ここでは、「Atmark Dist」、「Linux カーネル」のソースコードからイメージファイルを作成する手順を説明します。

手順 7.1 Linux カーネル/ユーザーランドをビルド

1. **アーカイブの展開**

   各ソースコードアーカイブを展開します。

```
[ATDE ~]$ ls
atmark-dist-[version].tar.gz  linux-3.14-at[version].tar.gz
[ATDE ~]$ tar zxf atmark-dist-[version].tar.gz
[ATDE ~]$ tar zxf linux-3.14-at[version].tar.gz
[ATDE ~]$ ls
atmark-dist-[version]         linux-3.14-at[version]
atmark-dist-[version].tar.gz  linux-3.14-at[version].tar.gz
```

2. **シンボリックリンクの作成**

   Atmark Dist に、Linux カーネルのシンボリックリンクを作成します。

```
[ATDE ~]$ cd atmark-dist-[version]
[ATDE ~/atmark-dist-[version]]$ ln -s ../linux-3.14-at[version] linux-3.x
```

以降のコマンド入力例では、各ファイルからバージョンを省略した表記を用います。

3. AWL13 ドライバーの登録

この手順は、Armadillo-WLAN モジュール(AWL13)を使用する場合にのみ必要です。

AWL13 を使用するイメージを作成する場合は、カーネルソースの他に、AWL13 用のデバイスドライバー(AWL13 ドライバー)を Atmark-Dist に登録する必要があります。

付属 DVD の AWL13 ドライバーのソースアーカイブディレクトリ (source/awlan_driver) にある awl13-*[version]*.tar.gz を作業ディレクトリに展開します。

展開後、Atmark-Dist に AWL13 ドライバーのソースを登録するため、シンボリックリンクを作成します。「図 7.1. ソースコード準備(AWL13 ドライバー)」のように作業してください。

```
[ATDE ~]$ tar zxvf awl13-[version].tar.gz
[ATDE ~]$ ls
atmark-dist-[version].tar.gz  atmark-dist-[version]
linux-[version].tar.gz  linux-[version]
awl13-[version].tar.gz  awl13-[version]
[ATDE ~]$ cd atmark-dist
[ATDE ~/atmark-dist]$ ln -s ../awl13-[version] awl13
```

図 7.1 ソースコード準備(AWL13 ドライバー)

4. コンフィギュレーションの開始

コンフィギュレーションを開始します。ここでは、menuconfig を利用します。

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

```
 atmark-dist v1.45.0 Configuration
--------------------------------------------------------------------------------
                              Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ------------------------------------------------------------------
               Vendor/Product Selection  --->
               Kernel/Library/Defaults Selection  --->
               ---
               Load an Alternate Configuration File
               Save Configuration to an Alternate File


  ------------------------------------------------------------------

                   <Select>    < Exit >    < Help >
```

5. ベンダー/プロダクト名の選択

メニュー項目は、上下キーで移動することができます。下部の Select/Exit/Help は左右キーで移動することができます。選択するには Enter キーを押下します。 "Vendor/Product Selection --->"に移動して Enter キーを押下します。 Vendor には "AtmarkTechno" を選択し、AtmarkTechno Products には 「表 7.1. プロダクト名一覧」 から選択します。

表 7.1 プロダクト名一覧

製品	プロダクト名	備考
Armadillo-420 ベーシックモデル	Armadillo-420	出荷時イメージ
Armadillo-420 WLAN モデル(AWL13 対応)	Armadillo-420.WLAN-AWL13	出荷時イメージ

```
 atmark-dist v1.45.0 Configuration
------------------------------------------------------------------------------
                        Vendor/Product Selection
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ------------------------------------------------------------------
               --- Select the Vendor you wish to target
               (AtmarkTechno) Vendor
               --- Select the Product you wish to target
               (Armadillo-420) AtmarkTechno Products


  ------------------------------------------------------------------

                   <Select>    < Exit >    < Help >
```

6. デフォルトコンフィギュレーションの適用

前のメニューに戻るには、"Exit"に移動して Enter キーを押下します。 続いて、"Kernel/Library/Defaults Selection --->"に移動して Enter キーを押下します。"Default all settings (lose changes)"に移動して"Y"キーを押下します。押下すると"[*]"のように選択状態となります。

```
 atmark-dist v1.45.0 Configuration
--------------------------------------------------------------------------------
                    Kernel/Library/Defaults Selection
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ------------------------------------------------------------------
             --- Kernel is linux-3.x
             (default) Cross-dev
             (None) Libc Version
             [*] Default all settings (lose changes) (NEW)
             [ ] Customize Kernel Settings (NEW)
             [ ] Customize Vendor/User Settings (NEW)
             [ ] Update Default Vendor Settings (NEW)

  ------------------------------------------------------------------

                    <Select>    < Exit >    < Help >
```

7. コンフィギュレーションの終了

前のメニューに戻るため、"Exit"に移動して Enter キーを押下します。コンフィギュレーションを抜けるためにもう一度"Exit"に移動して Enter キーを押下します。

8. コンフィギュレーションの確定

コンフィギュレーションを確定させるために"Yes"に移動して Enter キーを押下します。

```
 atmark-dist v1.45.0 Configuration
--------------------------------------------------------------------------------


        ---------------------------------------------------------
            Do you wish to save your new kernel configuration?

                            < Yes >      <  No  >

        ---------------------------------------------------------
```

9. ビルド

コンフィギュレーションが完了するので、続いてビルドを行います。 ビルドは"make"コマンドを実行します。

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

ビルドログが表示されます。ビルドする PC のスペックにもよりますが、数分から十数分程度かかります。

10. **イメージファイルの生成確認**

ビルドが終了すると、atmark-dist/images/ディレクトリ以下にイメージファイルが作成されています。Armadillo-400 シリーズ では圧縮済みのイメージ(拡張子が".gz"のもの)を利用します。

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

7.2. 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。

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

図 7.2 Atmark-Dist のコンフィギュレーション

make menuconfig を実行すると、「図 7.3. menuconfig: Main Menu」に示す Main Menu 画面が表示されます。

```
 atmark-dist v1.45.0 Configuration
--------------------------------------------------------------------------------
                              Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ------------------------------------------------------------------------
               Vendor/Product Selection  --->
               Kernel/Library/Defaults Selection  --->
               ---
               Load an Alternate Configuration File
               Save Configuration to an Alternate File


  ------------------------------------------------------------------------

                   <Select>    < Exit >    < Help >
```

図 7.3 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。

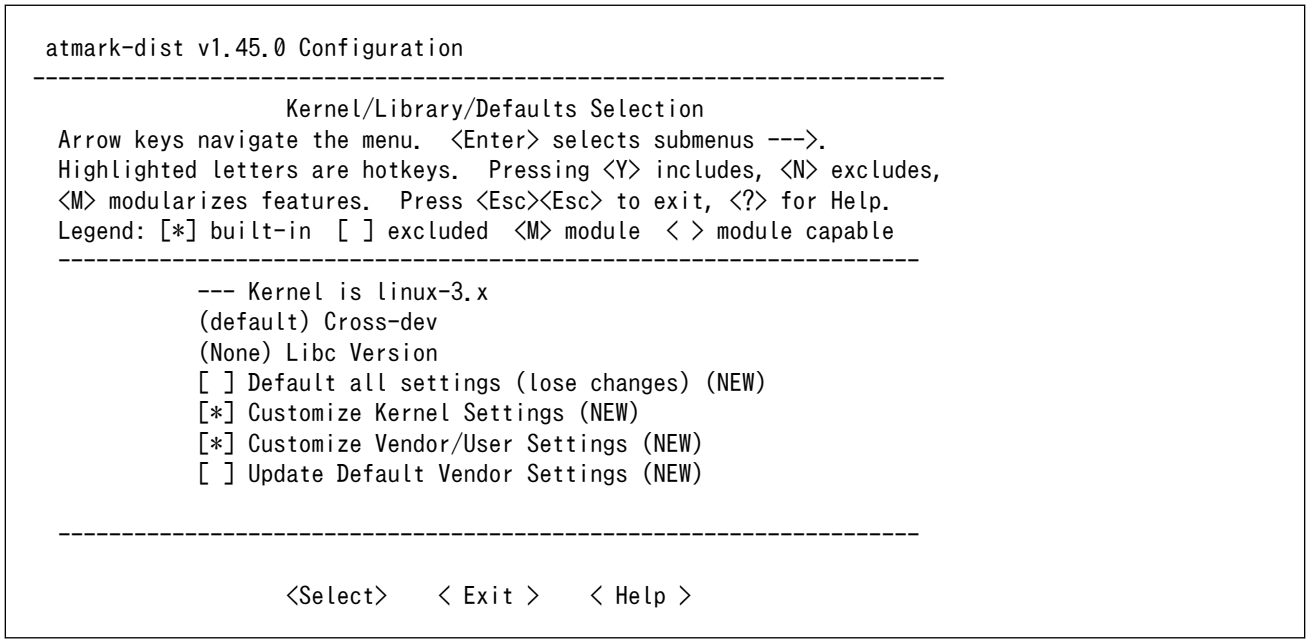


図 7.4 menuconfig: Kernel/Library/Defaults Selection

カーネルコンフィギュレーションを変更するには、**Cuntomize 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。

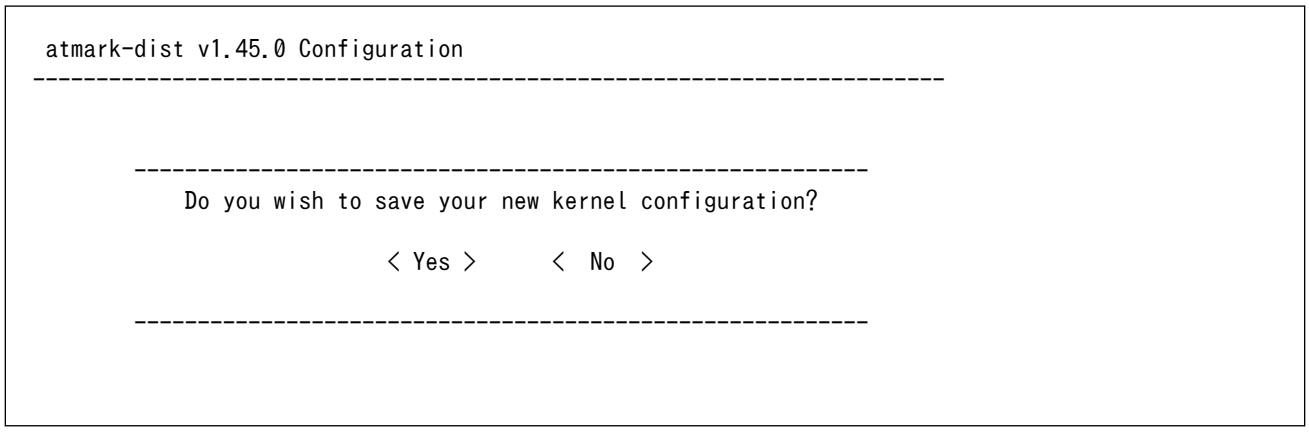


図 7.5 menuconfig: Do you wish to save your new kernel configuration?

Cuntomize Kernel Settings を選択していた場合は、Linux Kenrel 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。

```
.config - Linux/arm 3.14.36-at4 Kernel Configuration
------------------------------------------------------------------------------
                  Linux/arm 3.14.36-at4 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  --------------------------------------------------------------------
              -*- Patch physical to virtual translations at runtime
                  General setup  --->
              [*] Enable loadable module support  --->
              [*] Enable the block layer  --->
                  System Type  --->
                  Bus support  --->
                  Kernel Features  --->
                  Boot options  --->
                  CPU Power Management  --->
                  Floating point emulation  --->
                  Userspace binary formats  --->
                  Power management options  --->
              [*] Networking support  --->
                  Device Drivers  --->
                  File systems  --->
                  Kernel hacking  --->
                  Security options  --->
              -*- Cryptographic API  --->
                  Library routines  --->
              [ ] Virtualization  ----
  ------------------------------------------------------------------------
                <Select>    < Exit >    < Help >    < Save >    < Load >
```

図 7.6 menuconfig: Linux Kernel Configuration

AWL13 を使用するためのカーネルコンフィギュレーション変更について

Armadillo-420 のプロダクトを選択した場合に、Armadillo-WLAN(AWL13)を SDIO インターフェースで使用するには、以下のカーネルコンフィギュレーションを有効にします。

```
System Type  --->
    [*] Freescale i.MX family
          Freescale i.MX support  --->
            *** MX25 platforms: ***
        [*] Support Armadillo-420 platform
              Device options  --->
                  -*- Enable eSDHC2
              Armadillo-400 Board options  --->
                  [*] Enable SDHC2 at CON9
                  [*]   Enable PWREN for SDHC2 at CON9_1
```

図 7.7 Armadillo-WLAN 用 SDIO インターフェース有効化

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。

```
 atmark-dist v1.45.0 Configuration
--------------------------------------------------------------------------
                    Userland Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ----------------------------------------------------------------------
              Vendor specific  --->
              Fonts  --->
              Core Applications  --->
              Library Configuration  --->
              Flash Tools  --->
              Filesystem Applications  --->
              Network Applications  --->
              Miscellaneous Applications  --->
              BusyBox  --->
              Tinylogin  --->
              Qt  --->
              X Window System  --->
              MicroWindows  --->
              Games  --->
              Miscellaneous Configuration  --->
              Debug Builds  --->
              ---
              Load an Alternate Configuration File
              Save Configuration to an Alternate File
  ----------------------------------------------------------------------

                    <Select>    < Exit >    < Help >
```

図 7.8 menuconfig: Userland Configuration

AWL13 を使用する場合のユーザーランドコンフィギュレーション変更について

Armadillo-WLAN(AWL13)を使用するには、ユーザーランドコンフィギュレーションの **Vendor Specific** で **Armadillo-WLAN** を選択してください。

Armadillo-WLAN Products で AWL13 を選択した場合、使用するインターフェースとして SDIO または USB を選択できます。

また、ユーザーランドに組み込む AWL13 のファームウェアとして、STA (ステーション)と AP (アクセスポイント)を選択できます。 両方を選択した場合は、どちらのファームウェアを標準で使用するかを、**AWL13 Default Mode** で指定します。

```
Userland Configuration
    Vendor specific  --->
        [*] Armadillo-WLAN
        (AWL13) Armadillo-WLAN Products
        (SDIO) AWL13 Support interface
        --- AWL13 Firmware
        [*] AWL13 Station Mode
        [*] AWL13 Access Point Mode
        (STA) AWL13 Default Mode
```

図 7.9 AWL13 を使用する場合のコンフィギュレーション例(SDIO インターフェース、ステーションモード)

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、「7.1. Linux カーネル/ユーザーランドをビルドする」のステップ 9 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。

7.3. 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、「7.1. Linux カーネル/ユーザーランドをビルドする」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[1]で作成し、~/sample/hello にあると仮定とします。

[1]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。

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

図 7.10 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。

7.4. ブートローダーをビルドする

ここでは、ブートローダーである「Hermit-At」のソースコードからイメージファイルを作成する手順を説明します。

手順 7.2 ブートローダーをビルド

1. **ソースコードの準備**

   Hermit-At のソースコードアーカイブを準備し展開します。展開後、hermit-at ディレクトリに移動します。

   ```
   [ATDE ~]$ ls
   hermit-at-[version]-source.tar.gz
   [ATDE ~]$ tar zxf hermit-at-[version]-source.tar.gz
   [ATDE ~]$ ls
   hermit-at-[version]  hermit-at-[version]-source.tar.gz
   ```

   以降のコマンド入力例では、ブートローダーのソースファイルからバージョンを省略した表記を用います。

2. **デフォルトコンフィギュレーションの適用**

   Hermit-At ディレクトリに入り、Armadillo-400 シリーズ用のデフォルトコンフィギュレーションを適用します。ここでは例としてフラッシュメモリ起動用イメージを作成します。デフォルトコンフィグには armadillo4x0_defconfig を指定します。UART 起動用イメージを作成する場合は、armadillo4x0_boot_defconfig を指定してください。

   ```
   [ATDE ~]$ cd hermit-at
   [ATDE ~/hermit-at]$ make armadillo4x0_defconfig
   ```

3. **ビルド**

   ビルドには"make"コマンドを利用します。

   ```
   [ATDE ~/hermit-at]$ make
   ```

4. イメージファイルの生成確認

ビルドが終了すると、hermit-at/src/target/armadillo4x0/ディレクトリ以下にイメージファイルが作成されています。

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

8. 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。

8.1. 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、tftpboot 機能を有しています。

tftpboot 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。

8.1.1. 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。

ATDE5 は、標準状態で TFTP サーバー (atftpd) が動作しています。/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。

8.1.2. 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 8.1 tftpboot 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、以下のようになります[2]。

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

図 8.2 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。

8.2. 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/SD に、ユーザーランドのルートファイルシステムは microSD/SD または USB メモリにも配置することができます。

ここでは、例として microSD/SD にカーネルイメージとルートファイルシステム両方を配置する手順を説明します。

まず、microSD/SD に 1 つのパーティションを作成し、EXT3 ファイルシステムでフォーマットします。そこにルートファイルシステムを構築し、/boot/ ディレクトリにカーネルイメージを配置します。どのデバイスからカーネルイメージをロードするかは、Hermit-At のブートオプションで指定します。また、ルートファイルシステムがどこにあるかは、カーネルパラメーターで指定します。

8.2.1. パーティション作成

最初に、microSD/SD に 1 つのプライマリパーティションを作成します。

microSD/SD をスロットに挿入し[3]、「図 8.3. パーティション作成手順」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-420 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 124277.
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                                                    ❶
Selected partition 1

Command (m for help): n                                                    ❷
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-124277, default 1):                                      ❸
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-124277, default 124277):     ❹
Using default value 124277

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

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

[armadillo ~]#
```

図 8.3 パーティション作成手順

❶　まずは、既存のパーティションを削除します。複数のパーティションがある場合は、全て削除してください。

❷　新しくプライマリパーティションを作成します。

❸　開始シリンダにはデフォルト値(1)を使用するので、そのまま改行を入力してください。

❹　最終シリンダにもデフォルト値(124277)を使用するので、そのまま改行を入力してください。

❺　変更を microSD/SD に書き込みます。

使用する microSD/SD 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。

8.2.2. ファイルシステムの作成

次に、「図 8.4. ファイルシステム作成手順」のようにして、EXT3 ファイルシステムでフォーマットします。

```
[armadillo ~]# mke2fs -j /dev/mmcblk0p1
mke2fs 1.25 (20-Sep-2001)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
497984 inodes, 994220 blocks
49711 blocks (5%) reserved for the super user
First data block=0
31 block groups
32768 blocks per group, 32768 fragments per group
16064 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (8192 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 8.4 ファイルシステム作成手順

8.2.3. カーネルイメージの配置

microSD/SD 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 8.1 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

以下に Armadillo-420 での配置例を示します。

```
[armadillo ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir /mnt/boot
[armadillo ~]# cd /mnt/boot
[armadillo /mnt/boot]# wget http://download.atmark-techno.com/armadillo-420/image/linux-a400-
[version].bin.gz
[armadillo /mnt/boot]# mv linux-a400-[version].bin.gz /mnt/boot/linux.bin.gz
[armadillo /mnt/boot]# cd
[armadillo ~]# umount /mnt
```

図 8.5 カーネルイメージの配置

8.2.4. ルートファイルシステムの構築

ここでは、microSD/SD 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。

8.2.4.1. Debian GNU/Linux のルートファイルシステムを構築する

Debian GNU/Linux ルートファイルシステムアーカイブから、ルートファイルシステムを構築する手順を次に示します。

Debian GNU/Linux ルートファイルシステムアーカイブを準備しておきます。

表 8.2 Debian GNU/Linux ルートファイルシステムアーカイブのダウンロード先 URL

製品	URL
Armadillo-420	http://download.atmark-techno.com/armadillo-420/image/debian-wheezy-armel_a420_*[version]*.tar.gz

```
[ATDE ~]$ ls
debian-wheezy-armel_a420_[version].tar.gz
```

ルートファイルシステムを一旦 ATDE でマウントした microSD/SD カードに構築します。

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext3 /dev/sdb1 sd ❷
[ATDE ~]$ sudo tar zxf debian-wheezy-armel_a420_[version].tar.gz -C sd ❸
[ATDE ~]$ sudo umount sd ❹
[ATDE ~]$ rmdir sd ❺
```

❶ SD カードをマウントするための sd/ディレクトリを作成します。

❷ sd/ディレクトリに SD カードをマウントします。

❸ ルートファイルシステムアーカイブを sd/ディレクトリに展開します。

❹ sd/ディレクトリにマウントしたブートディスクの第 1 パーティションをアンマウントします。

❺ sd/ディレクトリを削除します。

アンマウントが完了する前に SD カードを作業用 PC から取り外すと、SD カードのデータが破損する場合があります。

8.2.4.2. 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/SD 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/SD へシステムを構築することができます。

ここで説明する例では、Atmark-Dist 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 8.3 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 ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# mkdir tmp
[armadillo ~]# mkdir romfs
[armadillo ~]# mount -t ramfs ramfs tmp
[armadillo ~]# wget http://download.atmark-techno.com/armadillo-420/image/romfs-a420-
[version].img.gz -P tmp
[armadillo ~]# gzip -d tmp/romfs-a420-[version].img.gz
[armadillo ~]# mount -o loop tmp/romfs-a420-[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 8.6 Atmark-Dist イメージによるルートファイルシステムの構築例

Atmark-Dist イメージは/etc/fstab の設定がフラッシュメモリ用になっているため、/dev/ram0 の行を書き換えて、microSD/SD 用に変更する必要があります。

```
[armadillo ~]# mount /dev/mmcblk0p1 /mnt/
[armadillo ~]# vi /mnt/etc/fstab
/dev/mmcblk0p1          /               ext3    defaults                0 1
proc                    /proc           proc    defaults                0 0
udev                    /dev            tmpfs   mode=0755               0 0
run                     /run            tmpfs   mode=0755               0 0
sysfs                   /sys            sysfs   defaults                0 0
[armadillo ~]# umount /mnt
```

図 8.7 fstab の変更例

8.2.5. ブートデバイスとカーネルパラメーターの設定

カーネルイメージをロードする場所は、Hermit-At のブートデバイス設定で指定します。また、ユーザーランドの場所は、カーネルパラメーターで指定します。

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/SD 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 8.8. ブートデバイスの指定」を実行してください。

```
hermit> setbootdevice mmcblk0p1
```

図 8.8 ブートデバイスの指定

ルートファイルシステムを microSD/SD のパーティション 1 にする場合は、「図 8.9. 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=ttymxc1,115200 root=/dev/mmcblk0p1 noinitrd rootwait
```

図 8.9 ルートファイルシステム指定例

ブートデバイスとカーネルパラメーターの設定を元に戻す方法については、「6.7. ブートローダーのパラメータを出荷状態に戻す」を参照してください。

9. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-400 シリーズ の Linux カーネル仕様について説明します。

9.1. デフォルトコンフィギュレーション

工場出荷状態のフラッシュメモリに書き込まれている Linux カーネルイメージには、デフォルトコンフィギュレーションが適用されています。 Armadillo-400 シリーズ用のデフォルトコンフィギュレーションが記載されているファイルは、Linux カーネルソースファイル(linux-3.14-at[version].tar.gz)に含まれる `arch/arm/configs/armadillo4x0_defconfig` です。

`armadillo4x0_defconfig` で有効になっている主要な設定を「表 9.1. Linux カーネル主要設定」に示します。

表 9.1 Linux カーネル主要設定

コンフィグ	説明
NO_HZ	Tickless System (Dynamic Ticks)
HIGH_RES_TIMERS	High Resolution Timer Support
PREEMPT	Preemptible Kernel
AEABI	Use the ARM EABI to compile the kernel
COMPACTION	Allow for memory compaction
BINFMT_ELF	Kernel support for ELF binaries

9.2. デフォルト起動オプション

工場出荷状態の Armadillo-400 シリーズ の Linux カーネルの起動オプションについて説明します。デフォルト状態では、次のように設定されています。

表 9.2 Linux カーネルのデフォルト起動オプション

起動オプション	説明
console=ttymxc1,115200	起動ログなどが出力されるイニシャルコンソールに ttymxc1 を、ボーレートに 115200bps を指定します。
root=/dev/ram0	ルートファイルシステムに RAM ディスクを指定します。

9.3. Linux ドライバ一覧

Armadillo-400 シリーズで利用することができるデバイスドライバについて説明します。 各ドライバで利用しているソースコードの内主要なファイルのパスや、コンフィギュレーションに必要な情報、及びデバイスファイルなどについて記載します。

9.3.1. Armadillo-400

Armadillo-400 シリーズの初期化手順やハードウェアの構成情報、ピンマルチプレクスの情報などが定義されています。

関連するソースコード

arch/arm/mach-imx/mach-armadillo4x0.c
arch/arm/mach-imx/armadillo4x0_extif.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
            *** MX25 platforms: ***
      [*] Support Armadillo-420 platform                          <MACH_ARMADILLO420>
            Device options  --->
            Armadillo-400 Board options  --->
                        :
```

9.3.2. フラッシュメモリ

Armadillo-400 シリーズでは、フラッシュメモリを制御するソフトウェアとして MTD(Memory Technology Device) を利用しています。MTD のキャラクタデバイスまたはブロックデバイスを経由して、ユーザーランドからアクセスすることができます。

関連するソースコード

drivers/mtd/cmdlinepart.c
drivers/mtd/maps/physmap.c
drivers/mtd/mtd_blkdevs.c
drivers/mtd/mtdblock.c
drivers/mtd/mtdchar.c
drivers/mtd/mtdconcat.c
drivers/mtd/mtdcore.c
drivers/mtd/mtdpart.c
drivers/mtd/mtdsuper.c
drivers/mtd/chips/cfi_cmdset_0001.c
drivers/mtd/chips/cfi_probe.c
drivers/mtd/chips/cfi_util.c
drivers/mtd/chips/chipreg.c
drivers/mtd/chips/gen_probe.c

デバイスファイル

デバイスファイル	デバイスタイプ	対応するパーティション名
/dev/mtd0	キャラクタ	bootloader
/dev/mtd0ro		
/dev/flash/bootloader		
/dev/flash/nor.bootloader		
/dev/mtdblock0	ブロック	
/dev/mtd1	キャラクタ	kernel
/dev/mtd1ro		
/dev/flash/kernel		
/dev/flash/nor.kernel		
/dev/mtdblock1	ブロック	
/dev/mtd2	キャラクタ	userland
/dev/mtd2ro		
/dev/flash/userland		
/dev/flash/nor.userland		
/dev/mtdblock2	ブロック	
/dev/mtd3	キャラクタ	config
/dev/mtd3ro		
/dev/flash/config		
/dev/flash/nor.config		
/dev/mtdblock3	ブロック	

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Memory Technology Device (MTD) support  --->                          <MTD>
        <*> Command line partition table parsing                      <MTD_CMDLINE_PARTS>
        [*] Read-only switching user interface support                        <MTD_RO_IF>
        <*> Caching block device access to MTD devices                        <MTD_BLOCK>
            RAM/ROM/Flash chip drivers  --->
            <*> Detect flash chips by Common Flash Interface (CFI) probe        <MTD_CFI>
            <*> Support for Intel/Sharp flash chips                    <MTD_CFI_INTELEXT>
            Mapping drivers for chip access  --->
            <*> Flash device in physical memory map                         <MTD_PHYSMAP>
```

9.3.3. UART

Armadillo-400 シリーズのシリアルは、i.MX257 の UART(Universal Asynchronous Receiver/Transmitter) を利用しています。

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、UART2 をコンソールとして利用しています。

フォーマット

データビット長: 7 or 8 ビット
ストップビット長: 1 or 2 ビット
パリティ: 偶数 or 奇数 or なし
フロー制御: CTS/RTS or XON/XOFF or なし
最大ボーレート: 4Mbps

関連するソースコード

drivers/tty/n_tty.c
drivers/tty/tty_buffer.c
drivers/tty/tty_io.c
drivers/tty/tty_ioctl.c
drivers/tty/tty_ldisc.c
drivers/tty/tty_ldsem.c
drivers/tty/tty_mutex.c
drivers/tty/tty_port.c
drivers/tty/serial/serial_core.c
drivers/tty/serial/imx.c

デバイスファイル

シリアルインターフェース	デバイスファイル
UART1	/dev/ttymxc0
UART2	/dev/ttymxc1
UART3	/dev/ttymxc2
UART5	/dev/ttymxc4

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
            *** MX25 platforms: ***
        [*] Support Armadillo-420 platform                      <MACH_ARMADILLO420>
              Device options  --->
                  -*- Enable UART2 module                       <SERIAL_MXC_SELECT2>
                  -*- Enable UART2 HW Flow Control              <SERIAL_MXC_HW_FLOW2>
                  [*] Enable UART3 module                       <SERIAL_MXC_SELECT3>  ❶
                  [*] Enable UART3 HW Flow Control              <SERIAL_MXC_HW_FLOW3>  ❷
                  [*] Enable UART5 module                       <SERIAL_MXC_SELECT5>  ❸
                  [*] Enable UART5 HW Flow Control              <SERIAL_MXC_HW_FLOW5>  ❹
              Armadillo-400 Board options  --->
                  [*] Enable UART3 at CON9                  <ARMADILLO4X0_UART3_CON9>  ❶
                  [*]   Enable UART3 HW flow control at CON9
                                                        <ARMADILLO4X0_UART3_HW_FLOW_CON9>  ❷
                  [*] Enable UART5 at CON9                  <ARMADILLO4X0_UART5_CON9>  ❸
                  [*]   Enable UART5 HW flow control at CON9
                                                        <ARMADILLO4X0_UART5_HW_FLOW_CON9>  ❹
Device Drivers  --->
    Character devices  --->
        [*] Enable TTY                                                        <TTY>
            Serial drivers  --->
                <*> IMX serial port support                            <SERIAL_IMX>
                [*]   Console on IMX serial port                  <SERIAL_IMX_CONSOL>
```

❶ CON9 に UART3 を割り当てる場合に設定します。デフォルトで有効化されています。

❷ 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。デフォルトで無効化されています。

❸ CON9 に UART5 を割り当てる場合に設定します。デフォルトで有効化されています。

❹ 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。デフォルトで無効化されています。

9.3.4. Ethernet

Armadillo-400 シリーズの Ethernet(LAN)は、i.MX257 の FEC(Fast Ethernet Controller)を利用しています。

機能

通信速度: 100Mbps(100BASE-TX), 10Mbps(10BASE-T)
通信モード: Full-Duplex(全二重), Half-Duplex(半二重)
Auto Negotiation サポート
キャリア検知サポート
リンク検出サポート

関連するソースコード

drivers/net/Space.c

drivers/net/loopback.c
drivers/net/mii.c
drivers/net/ethernet/freescale/fec_main.c
drivers/net/ethernet/freescale/fec_ptp.c
drivers/net/phy/mdio_bus.c
drivers/net/phy/phy.c
drivers/net/phy/phy_device.c

ネットワークデバイス

eth0

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Network device support  --->                                    <NETDEVICES>
        [*]   Ethernet driver support  --->                               <ETHERNET>
            [*]   Freescale devices                                <NET_VENDOR_FREESCALE>
            <*>     FEC ethernet controller (of ColdFire and some i.MX CPUs)       <FEC>
```

9.3.5. SD ホスト

Armadillo-400 シリーズの SD ホストは、i.MX257 の eSDHC(Enhanced Secured Digital Host Controller)を利用しています。

i.MX257 は SD ホストコントローラを 2 個搭載しており、Armadillo の CON1(SD インターフェース)に eSDHC1 が割り当てられています。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 を割り当てる事ができます。

機能

カードタイプ: SD/SDHC/SDXC
バス幅: 4bit
スピードモード: Default Speed(24MHz), High Speed(48MHz)
カードディテクトサポート(CON9 のみ)
ライトプロテクトサポート(CON9 のみ)

デバイスファイル

メモリカードの場合は、カードを認識した順番で/dev/mmcblkN (N は'0'からの連番)となります。
I/O カードの場合は、ファンクションに応じたデバイスファイルとなります。

関連するソースコード

drivers/mmc/card/block.c
drivers/mmc/card/queue.c
drivers/mmc/core/
drivers/mmc/host/sdhci-esdhc-imx.c
drivers/mmc/host/sdhci-pltfm.c
drivers/mmc/host/sdhci.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
              [*] Support Armadillo-420 platform                <MACH_ARMADILLO420>
                    Device options  --->
                      [*] Enable eSDHC1                          <MMC_MXC_SELECT1>
                      [*] Enable eSDHC2                          <MMC_MXC_SELECT2> ❶
                    Armadillo-400 Board options  --->
                      [*] Enable SDHC2 at CON9            <ARMADILLO4X0_SDHC2_CON9> ❶
                      [*]   Enable PWREN for SDHC2 at CON9_1
                                                  <ARMADILLO4X0_SDHC2_PWREN_CON9_1> ❶
Device Drivers  --->
    <*> MMC/SD/SDIO card support  --->                                          <MMC>
        [*]   Additional delay after SDIO reset             <MMC_DELAY_AFTER_SDIO_RESET>
              *** MMC/SD/SDIO Card Drivers ***
        <*>   MMC block device driver                                     <MMC_BLOCK>
        (8)     Number of minors per block device                   <MMC_BLOCK_MINORS>
        [*]     Use bounce buffer for simple hosts                  <MMC_BLOCK_BOUNCE>
              *** MMC/SD/SDIO Host Controller Drivers ***
        <*>   Secure Digital Host Controller Interface support              <MMC_SDHCI>
        <*>   SDHCI platform and OF driver helper                    <MMC_SDHCI_PLTFM>
        <*>     SDHCI support for the Freescale eSDHC/uSDHC i.MX controller
                                                                    <MMC_SDHCI_OF_ESDHC>
        [*]       Enforce to use multi-block transfer
                                        <MMC_SDHCI_ESDHC_IMX_FORCE_MULTIBLOCK_TRANSFER>
```

❶ CON9 に eSDHC2 を割り当てる場合に設定します。デフォルトでは無効化されています。

9.3.6. USB ホスト

Armadillo-400 シリーズの USB ホストは、i.MX257 の UTMI-USB-PHY および USBOH(Universal Serial Bus OTG and Host) を利用しています。

Armadillo-400 シリーズでは、USB ホストインターフェース下段の OTG ポートおよび USB ホストインターフェース上段の HOST ポートを利用することができます。

機能

Universal Serial Bus Specification Revision 2.0 準拠
Enhanced Host Controller Interface (EHCI)準拠
転送レート(OTG): USB2.0 High-Speed (480Mbps), Full-Speed (12Mbps), Low-Speed (1.5Mbps)
転送レート(Host): USB2.0 Full-Speed (12Mbps), Low-Speed (1.5Mbps)

デバイスファイル

メモリデバイスの場合は、デバイスを認識した順番で/dev/sdN (N は'a'からの連番)となります。

関連するソースコード

drivers/usb/chipidea/ci_hdrc_imx.c
drivers/usb/chipidea/ci_hdrc_msm.c

drivers/usb/chipidea/ci_hdrc_zevio.c
drivers/usb/chipidea/core.c
drivers/usb/chipidea/host.c
drivers/usb/chipidea/otg.c
drivers/usb/chipidea/usbmisc_imx.c
drivers/usb/host/ehci-hcd.c
drivers/usb/host/ehci-hub.c
drivers/usb/phy/phy-generic.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] USB support  --->                                          <USB_SUPPORT>
        <*>   Support for Host-side USB                                   <USB>
                *** USB Host Controller Drivers ***
        <*>     EHCI HCD (USB 2.0) support                      <USB_EHCI_HCD>
        <*>   ChipIdea Highspeed Dual Role Controller           <USB_CHIPIDEA>
        [*]     ChipIdea host controller                    <USB_CHIPIDEA_HOST>
              USB Physical Layer drivers  --->
            <*>     NOP USB Transceiver Driver                  <NOP_USB_XCEIV>
```

9.3.7. リアルタイムクロック

Armadillo-400 シリーズ RTC オプションモジュール、WLAN オプションモジュール(AWL13 対応)には、リアルタイムクロック(セイコーインスツル社製 S-35390A)が搭載されています。Armadillo-420 でリアルタイムクロック機能を使用する場合は、オプションモジュールと組み合わせて使用する必要があります。

リアルタイムクロックは I2C バスに接続された I2C スレーブデバイスとして動作します。リアルタイムクロックと I2C バスとの接続を「表 9.3. リアルタイムクロック I2C バス接続」に示します。

機能

アラーム割り込みサポート

デバイスファイル

/dev/rtc
/dev/rtc0

関連するソースコード

drivers/rtc/class.c
drivers/rtc/hctosys.c
drivers/rtc/interface.c
drivers/rtc/rtc-dev.c
drivers/rtc/rtc-lib.c
drivers/rtc/rtc-proc.c
drivers/rtc/rtc-s35390a.c
drivers/rtc/rtc-sysfs.c
drivers/rtc/systohc.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Real Time Clock  --->
        [*]   Set system time from RTC on startup and resume              <RTC_HCTOSYS>
        [*]   Set the RTC time based on NTP synchronization               <RTC_SYSTOHC>
        (rtc0)  RTC used to set the system time                     <RTC_HCTOSYS_DEVICE>
              *** RTC interfaces ***
        [*]   /sys/class/rtc/rtcN (sysfs)                             <RTC_INTF_SYSFS>
        [*]   /proc/driver/rtc (procfs for rtcN)                      <RTC_INTF_PROC>
        [*]   /dev/rtcN (character devices)                           <RTC_INTF_DEV>
        [*]     RTC UIE emulation on dev interface                <RTC_INTF_DEV_UIE_EMUL>
              *** I2C RTC drivers ***
        <*>   Seiko Instruments S-35390A                             <RTC_DRV_S35390A>
```

リアルタイムクロックは I2C バスに接続された I2C スレーブデバイスとして動作します。リアルタイムクロックと I2C バスとの接続を「表 9.3. リアルタイムクロック I2C バス接続」に示します。

表 9.3 リアルタイムクロック I2C バス接続

RTC 搭載ボード/オプションモジュール名	I2C バス	アドレス	優先順位
Armadillo-400 シリーズ RTC オプションモジュール	I2C	0x30	1
Armadillo-400 シリーズ WLAN オプションモジュール(AWL13 対応)	I2C	0x30	1

リアルタイムクロックは、デバイスファイルまたは sysfs ファイルを使用して操作することができます。デバイスファイルは/dev/rtc*N*(N は 0 から始まる数値)に、sysfs ファイルは/sys/class/rtc/rtc*N*/ディレクトリ以下に作成されます。リアルタイムクロックが一つだけ接続されている場合、/dev/rtc0 デバイスファイルまたは/sys/class/rtc/rtc0 ディレクトリ以下の sysfs ファイルでリアルタイムクロックを操作すことができます。

アラーム割り込みは、sysfs RTC クラスディレクトリ以下のファイルから利用できます。

wakealarm ファイルに UNIX エポックからの経過秒数、または先頭に+を付けて現在時刻からの経過秒数を書き込むと、アラーム割り込み発生時刻を指定できます。アラーム割り込み発生時刻を変更するには wakealarm ファイルに"+0"を書き込み、アラーム割り込みのキャンセル後に再設定する必要があります。アラーム割り込みの利用例を次に示します。

```
[armadillo ~]# cat /proc/interrupts | grep rtc-s35390a ❶
 95:          0  gpio-mxc  15  rtc-s35390a
[armadillo ~]# echo +60 > /sys/class/rtc/rtc0/wakealarm ❷
[armadillo ~]# cat /sys/class/rtc/rtc0/wakealarm ❸
1434522480
[armadillo ~]# cat /sys/class/rtc/rtc0/since_epoch ❹
1434522481
[armadillo ~]# cat /proc/interrupts | grep rtc-s35390a ❺
 95:          1  gpio-mxc  15  rtc-s35390a
```

❶ アラーム割り込みの発生回数を確認します。この例では 0 回です。

❷ アラーム割り込みの発生時刻を 60 秒後に設定します。秒単位は切り捨てられるため、アラーム発生時刻は厳密に 60 秒後とならない点に注意してください。

❸ アラーム割り込みの発生時刻(UNIX エポックからの経過秒数)を確認します。この例では 1434522480 秒です。

❹ 現在時刻(UNIX エポックからの経過秒数)を確認します。アラーム割り込みの発生時刻を超えるまで待ちます。

❺ 再度アラーム割り込みの発生回数を確認します。1 増えているのでアラーム割り込みが発生したことを確認できます。

デバイスファイル(/dev/rtc0)経由でもアラーム割り込みを利用することができます。サンプルプログラムなどのより詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/rtc.txt)を参照してください。

date コマンドを利用して、UNIX エポックからの経過秒数を日時に変換することができます。

```
[armadillo ~]# date --date=@`cat /sys/class/rtc/rtc0/since_epoch`
Wed Jun 17 15:29:30 JST 2015
```

9.3.8. LED

Armadillo-400 シリーズに搭載されているソフトウェア制御可能な LED には、GPIO が接続されています。Linux では、GPIO 接続用 LED ドライバ(leds-gpio)で制御することができます。

sysfs LED クラスディレクトリ

/sys/class/leds/red
/sys/class/leds/green
/sys/class/leds/yellow

関連するソースコード

drivers/leds/led-class.c
drivers/leds/led-core.c
drivers/leds/led-triggers.c
drivers/leds/leds-gpio-register.c
drivers/leds/leds-gpio.c
drivers/leds/trigger/ledtrig-default-on.c
drivers/leds/trigger/ledtrig-heartbeat.c
drivers/leds/trigger/ledtrig-timer.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] LED Support  --->                                          <NEW_LEDS>
        <*>   LED Class Support                                    <LEDS_CLASS>
              *** LED drivers ***
        <*>   LED Support for GPIO connected LEDs                  <LEDS_GPIO>
              *** LED Triggers ***
        [*]   LED Trigger support  --->                          <LEDS_TRIGGERS>
            <*>   LED Timer Trigger                          <LEDS_TRIGGER_TIMER>
            <*>   LED Heartbeat Trigger                  <LEDS_TRIGGER_HEARTBEAT>
            <*>   LED Default ON Trigger                <LEDS_TRIGGER_DEFAULT_ON>
```

Armadillo-400 シリーズの LED は、LED クラスとして実装されており、 LED クラス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。LED クラスディレクトリと各 LED の対応を次に示します。

表 9.4 LED クラスディレクトリと LED の対応

LED クラスディレクトリ	インターフェース	デフォルトトリガ
/sys/class/leds/red/	ユーザー LED3(赤色)	default-on
/sys/class/leds/green/	ユーザー LED4(緑色)	default-on
/sys/class/leds/yellow/	ユーザー LED5(黄色)	none

9.3.9. ユーザースイッチ

Armadillo-400 シリーズに搭載されているユーザースイッチには、GPIO が接続されています。GPIO が接続されユーザー空間でイベント(Press/Release)を検出することができます。Linux では、GPIO 接続用キーボードドライバ(gpio-keys)で制御することができます。

ユーザースイッチには、次に示すキーコードが割り当てられています。

表 9.5 キーコード

ユーザースイッチ	キーコード	イベントコード
SW1	KEY_1	28

ユーザースイッチを制御する GPIO 接続用キーボードドライバは次の通りです。

表 9.6 GPIO 接続用キーボードドライバ

ユーザースイッチ	GPIO 接続用キーボードドライバ
SW1	gpio-keys

デバイスファイル

ユーザースイッチ	デバイスファイル
SW1	/dev/input/event0[a]

[a]USB デバイスなどを接続してインプットデバイスを追加している場合は、番号が異なる可能性があります

関連するソースコード

drivers/input/evdev.c
drivers/input/ff-core.c
drivers/input/input-compat.c

drivers/input/input-mt.c
drivers/input/input-polldev.c
drivers/input/input.c
drivers/input/keyboard/gpio_keys.c
drivers/input/keyboard/gpio_keys_polled.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    Input device support  --->
        -*- Generic input layer (needed for keyboard, mouse, ...)              <INPUT>
        -*-   Matrix keymap support library                                <INPUT_MATRIXKMAP>
              *** Userland interfaces ***
        <*>   Event interface                                                 <INPUT_EVDEV>
              *** Input Device Drivers ***
        [*]   Keyboards  --->                                               <INPUT_KEYBOARD>
            <*>   GPIO Buttons                                               <KEYBOARD_GPIO>
            <*>   IMX keypad support                                         <KEYBOARD_IMX>
```

9.3.10. I2C

I2C インターフェースは、i.MX257 の I2C(Inter IC Module) を利用します。 i.MX25 プロセッサは I2C1 から I2C3 までの 3 つの I2C モジュールを内蔵しています。 Armadillo-400 シリーズでは、I2C1 はボード内蔵バスとして使用し、I2C2 は CON14 に割り当てています。

また、GPIO を利用した I2C バスドライバ(i2c-gpio)を利用することで、I2C バスを追加することができます。

標準状態で、CONFIG_I2C_CHARDEV が有効となっているためユーザードライバで I2C デバイスを制御することができます。

機能

最大転送レート: 400kbps (I2C1, I2C2)

デバイスファイル

/dev/i2c-0 (I2C1)
/dev/i2c-1 (I2C2)

関連するソースコード

drivers/i2c/i2c-boardinfo.c
drivers/i2c/i2c-core.c
drivers/i2c/i2c-dev.c
drivers/i2c/algos/i2c-algo-bit.c
drivers/i2c/busses/i2c-gpio.c
drivers/i2c/busses/i2c-imx.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
            *** MX25 platforms: ***
        [*] Support Armadillo-420 platform                        <MACH_ARMADILLO420>
              Device options  --->
                  -*- Enable I2C1 module                           <I2C_MXC_SELECT1>
                  -*- Enable I2C2 module                           <I2C_MXC_SELECT2>
              Armadillo-400 Board options  --->
                  [*] Enable I2C2 at CON14                  <ARMADILLO4X0_I2C2_CON14>
Device Drivers  --->
    <*> I2C support  --->                                                    <I2C>
        <*>   I2C device interface                                     <I2C_CHARDEV>
        [*]   Autoselect pertinent helper modules                   <I2C_HELPER_AUTO>
              I2C Hardware Bus support  --->
            <*> GPIO-based bitbanging I2C                                <I2C_GPIO>
            <*> IMX I2C interface                                        <I2C_MXC>
```

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、スレーブ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 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-3.14-at/arch/arm/mach-imx/armadillo4x0_extif.c の armadillo4x0_i2cN 配列(N はバスに対応した数値)に記述してください。

```
static struct i2c_board_info armadillo4x0_i2c1[] = {
    {
        I2C_BOARD_INFO("s35390a", 0x30),
    },
};
```

図 9.1 I2C i2c_board_info の設定

標準では、I2C1 バスの通信速度は 40kbps に設定されています。通信速度は、armadillo4x0_extif.c の以下の場所で設定されています。

```
static const struct imxi2c_platform_data mx25_i2c1_data __initconst = {
    .bitrate = 40000,
};
```

図 9.2 I2C 通信速度の設定

9.3.11. SPI

SPI インターフェースは、i.MX257 の CSPI(Configurable Serial Peripheral Interface)を利用します。Armadillo-400 シリーズはカーネルコンフィギュレーションにより、CSP1 及び CSPI3 を CON9 に割り当てる事が可能です。

標準状態では無効になっている CONFIG_SPI_SPIDEV を有効化すると、ユーザードライバで SPI デバイスを制御することができます。

機能

SPI マスターモード
複数スレーブセレクト
最大通信速度 約 16Mbps

関連するソースコード

drivers/spi/spi-bitbang.c
drivers/spi/spi-imx.c
drivers/spi/spi.c
drivers/spi/spidev.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
              [*] Support Armadillo-420 platform                  <MACH_ARMADILLO420>
                    Device options  --->
                      [*] Enable SPI1 module                        <SPI_MXC_SELECT1>  ❶
                      [*] Enable SPI3 module                        <SPI_MXC_SELECT3>  ❷
                    Armadillo-400 Board options  --->
                      [ ] Enable UART3 at CON9              <ARMADILLO4X0_UART3_CON9>  ❶
                      [ ] Enable UART5 at CON9              <ARMADILLO4X0_UART5_CON9>  ❷
                      [*] Enable SPI1 at CON9                <ARMADILLO4X0_SPI1_CON9>  ❶
                      [*]   Enable SPI1_SS0 at CON9_25 <ARMADILLO4X0_SPI1_SS0_CON9_25>  ❶
                      [*]   Enable SPI1_SS1 at CON9_11 <ARMADILLO4X0_SPI1_SS1_CON9_11>  ❶
                      [*] Enable SPI3 at CON9                <ARMADILLO4X0_SPI3_CON9>  ❷
                      [*]   Enable SPI3_SS0 at CON9_16 <ARMADILLO4X0_SPI3_SS0_CON9_16>  ❷
                      [*]   Enable SPI3_SS1 at CON9_18 <ARMADILLO4X0_SPI3_SS1_CON9_18>  ❷
                      [*]   Enable SPI3_SS2 at CON9_15 <ARMADILLO4X0_SPI3_SS2_CON9_15>  ❷
                      [*]   Enable SPI3_SS3 at CON9_17 <ARMADILLO4X0_SPI3_SS3_CON9_17>  ❷
Device Drivers  --->
    [*] SPI support  --->                                                      <SPI>  ❸
              *** SPI Master Controller Drivers ***
        -*-   Utilities for Bitbanging SPI masters                      <SPI_BITBANG>  ❸
        <*>   Freescale i.MX SPI controllers                               <SPI_MXC>  ❸
              *** SPI Protocol Masters ***
        < >   User mode SPI device driver support                       <SPI_SPIDEV>
```

❶ CON9 に SPI1 を割り当てる場合に設定します。デフォルトでは無効化されています。

❷ CON9 に SPI3 を割り当てる場合に設定します。デフォルトでは無効化されています。

❸ SPI を有効化する場合に設定します。デフォルトでは無効化されています。

SPI マスタードライバーとスレーブデバイスのドライバー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。struct spi_board_info は linux-3.*[version]*/arch/arm/mach-imx/armadillo4x0_extif.c の armadillo4x0_spiN_board_info (N はバスに対応した数値)に記述してください。SPI バスの通信速度は、それぞれのスレーブデバイスごとに設定します。

SPI のハードウェア、バスナンバー、struct spi_board_info の対応は次に示す通りです。

ハードウェア	バスナンバー	struct spi_bord_info 名
CSPI1	0	armadillo4x0_spi0_board_info
CSPI3	2	armadillo4x0_spi2_board_info

9.3.12. ウォッチドッグタイマー

Armadillo-400 シリーズのウォッチドッグタイマーは、i.MX257 の WDOG(Watchdog Timer) を利用します。

ウォッチドッグタイマーは、Hermit-At ブートローダーによって有効化されます。標準状態で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、ウォッチドッグタイマードライバの初期化時に、このタイムアウト時間を上書きします。標準状態のタイムアウト時間は 10 秒です。カーネルタイマーを利用して定期的にウォッチドッグタイマーをキックします。

何らかの要因でウォッチドッグタイマーのキックができなくなりタイムアウトすると、システムリセットが発生します。

関連するソースコード

drivers/watchdog/imx2_wdt.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Watchdog Timer Support  --->                                    <WATCHDOG>
        <*>   IMX2+ Watchdog                                            <IMX2_WDT>
```

i.MX257 の WDOG は、一度有効化すると無効化することができません。そのため、halt コマンドなどを実行して Linux カーネルを停止した場合は、ウォッチドッグタイマーのキックができなくなるためシステムリセットが発生します。

WDOG ドライバーの終了処理では、タイムアウト時間を WDOG の最大値である 128 秒に設定します。

9.3.13. 1-wire

Armadillo-400 シリーズは、CON9_2 と CON9_26 を 1-wire マスターとして使用する事ができます。CON9_2 は i.MX257 の 1-wire(1-wire Module) を利用し、CON9_26 は GPIO 1-wire ドライバーを用いて機能を実現しています。

デフォルト状態では、1-wire を利用することができません。1-wire を利用するためには、カーネルコンフィギュレーションを行い 1-wire マスタードライバーと 接続するスレーブデバイスのドライバーを有効にする必要があります。

関連するソースコード

drivers/w1/masters/mxc_w1.c
drivers/w1/w1.c
drivers/w1/w1_int.c

drivers/w1/w1_family.c
drivers/w1/w1_netlink.c
drivers/w1/w1_io.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                        <ARCH_MXC>
          Freescale i.MX support  --->
              [*] Support Armadillo-420 platform                  <MACH_ARMADILLO420>
                    Device options  --->
                      [*] Enable MX25 1-Wire module                 <W1_MXC_SELECT1>  ❶
                      [*] Enable GPIO 1-Wire module                 <W1_GPIO_SELECT1>  ❷
                    Armadillo-400 Board options  --->
                      [*] Enable one wire at CON9_2          <ARMADILLO4X0_W1_CON9_2>  ❶
                      [*] Enable one wire at CON9_26        <ARMADILLO4X0_W1_CON9_26>  ❷
Device Drivers  --->
    <*> Dallas's 1-wire support  --->                                          <W1>  ❶❷
          1-wire Bus Masters  --->
              <*> Freescale MXC 1-wire busmaster                    <W1_MASTER_MXC>  ❶
              <*> GPIO 1-wire busmaster                             <W1_MASTER_GPIO>  ❷
```

❶ CON9_2 を 1-wire として使用する場合に設定します。デフォルトでは無効化されています。

❷ CON9_26 を 1-wire として使用する場合に設定します。デフォルトでは無効化されています。

9.3.14. PWM

Armadillo-400 シリーズの PWM は、i.MX257 の PWM(Pulse-Width Modulator) を利用します。

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 に割り当てる事ができます。

関連するソースコード

drivers/pwm/core.c
drivers/pwm/pwm-imx.c
drivers/pwm/sysfs.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                    <ARCH_MXC>
          Freescale i.MX support  --->
              [*] Support Armadillo-420 platform              <MACH_ARMADILLO420>
                    Device options  --->
                      [*] Enable PWM2                          <MXC_PWM_SELECT2>  ❶
                      [*] Enable PWM4                          <MXC_PWM_SELECT4>  ❷
                    Armadillo-400 Board options  --->
                      [ ] Enable I2C2 at CON14           <ARMADILLO4X0_I2C2_CON14>  ❷
                      [*] Enable PWM2 at CON9_25         <ARMADILLO4X0_PWM2_CON9_25>  ❶
                      [*] Enable PWM4 at CON14_3         <ARMADILLO4X0_PWM4_CON14_3>  ❷
Device Drivers  --->
    [*] Pulse-Width Modulation (PWM) Support  --->                          <PWM>
        <*>   i.MX PWM support                                           <PWM_IMX>
```

❶ CON9_25 を PWM2 として使用する場合に設定します。デフォルトでは無効化されています。

❷ CON14_3 を PWM4 として使用する場合に有効化します。デフォルトでは無効化されています。

ARMADILLO4X0_PWM4_CON14_3 は、ARMADILLO4X0_I2C2_CON14 を無効化しなければ、有効化できません。

PWM 機能は使用する前に export する必要があります。export するためには、/sys/class/pwm/pwmchip*N*/export に 0 を書き込みます。export すると/sys/class/pwm/pwmchip*N*/pwm0 が生成されます。

```
[armadillo ~]# echo 0 > /sys/class/pwm/pwmchip0/export
```

図 9.3 pwmchip0 を export する

/sys/class/pwm/pwmchip*N*/export の *N* は、カーネルが PWM を認識した順に連番の値が設定されます。pwmchip*N* がどこにリンクされているかを調べる事で、i.MX25 のどのハードウェアに割当たっているか調べる事ができます。

```
[armadillo ~]# ls -l /sys/class/pwm/pwmchip0
lrwxrwxrwx    1 root     root             0 Jan  1  1970 /sys/class/pwm/
pwmchip0 -> ../../devices/platform/imx27-pwm.1/pwm/pwmchip0/
```

上記の例では、pwmchip0 は、imx27-pwm.1 にリンクされています。リンク先の sysfs ファイル名と、PWM のハードウェアとの対応は次の通りです。

ハードウェア	sysfs ファイル名
PWM2 CON9_25	imx27-pwm.1
PWM4 CON14_3	imx27-pwm.3

PWM 機能は、/sys/class/pwm/pwmchip*N*/pwm0 以下のファイルに値を書き込む事で設定変更する事ができます。設定に使用するファイルを、「表 9.7. PWM sysfs」に示します。

表 9.7 PWM sysfs

ファイル名	説明
enable	PWM の動作/停止を設定します 1: PWM は動作します。period、duty_cycle、polarity の設定に応じた波形を出力します 0: PWM は停止ます
period	PWM の周期を nsec 単位で設定します 設定可能な範囲は、17〜2,147,483,647 (約 20nsec から 2sec)です 設定変更は enable=0 の状態で行います
duty_cycle	PWM の High 期間(polarity=inversed の場合は Low 期間)を nsec 単位で設定します 設定可能な範囲は、0 < duty_cycle < period の範囲です 設定変更は enable=0 の状態で行います
polarity	PWM 波形の極性を設定します normal: PWM 波形は正転状態となります。duty_cycle で指定する時間は High 期間になります inversed: PWM 波形は反転状態となります。duty_cycle で指定する時間は Low 期間になります 設定変更は enable=0 の状態で行います

9.3.15. CAN

Armadillo-400 シリーズの CAN は、i.MX257 の FlexCAN(Controller Area Network) を利用します。

カーネルコンフィギュレーションを変更する事により、CAN2 を CON14 に割り当てる事ができます。

関連するソースコード

drivers/net/can/flexcan.c
drivers/net/can/dev.c
net/can/af_can.c
net/can/bcm.c
net/can/gw.c
net/can/proc.c
net/can/raw.c

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                          <ARCH_MXC>
          Freescale i.MX support  --->
              [*] Support Armadillo-420 platform                <MACH_ARMADILLO420>
                    Device options  --->
                      [*] Enable FlexCAN2 module                 <FLEXCAN_SELECT2>  ❶
                    Armadillo-400 Board options  --->
                      [ ] Enable I2C2 at CON14             <ARMADILLO4X0_I2C2_CON14>  ❶
                      [*] Enable CAN2 at CON14             <ARMADILLO4X0_CAN2_CON14>  ❶
[*] Networking support  --->                                                <NET>
    <*>   CAN bus subsystem support  --->                                    <CAN>  ❶
        <*>   Raw CAN Protocol (raw access with CAN-ID filtering)          <CAN_RAW>
        <*>   Broadcast Manager CAN Protocol (with content filtering)      <CAN_BCM>
        <*>   CAN Gateway/Router (with netlink configuration)               <CAN_GW>
        CAN Device Drivers  --->
            <*> Platform CAN drivers with Netlink support                  <CAN_DEV>
            [*]   CAN bit-timing calculation                  <CAN_CALC_BITTIMING>
            <*>   Support for Freescale FLEXCAN based chips            <CAN_FLEXCAN>  ❶
```

❶ CON9 に CAN2 を割り当てる場合に設定します。デフォルトでは無効化されています。ARMADILLO4X0_CAN2_CON14 は、ARMADILLO4X0_I2C2_CON14 を無効化しなければ、有効化できません。

CAN の転送速度の設定には **ip** コマンドを使用します。**iproute2** の **ip** コマンドで、通信速度を 125kpbs に設定するには、次のコマンドを実行します。

```
[armadillo ~]# ip link set can0 type can bitrate 125000 loopback off
```

転送速度が変更されたかどうかは、次のコマンドで表示される bitrate の数値から確認することができます。

```
[armadillo ~]# ip -s -d link show can0
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
    link/can  promiscuity 0
    can state STOPPED (berr-counter tx 0 rx 0) restart-ms 0
          bitrate 125000 sample-point 0.857
          tq 571 prop-seg 5 phase-seg1 6 phase-seg2 2 sjw 1
          flexcan: tseg1 4..16 tseg2 2..8 sjw 1..4 brp 1..256 brp-inc 1
          clock 66500000
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          0          0          0
    RX: bytes  packets  errors  dropped overrun mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
```

Armadillo-400 シリーズのユーザーランドに 標準で組み込まれている BusyBox の **ip** コマンドは、CAN の転送速度を設定できません。そのた

め CAN 機能を使用する場合には、ユーザーランドコンフィギュレーションを行い、**iproute2** を組み込んでおく必要があります。

```
Userland Configuration
  Network Applications --->
    [*] iproute2                チェックを入れる
```

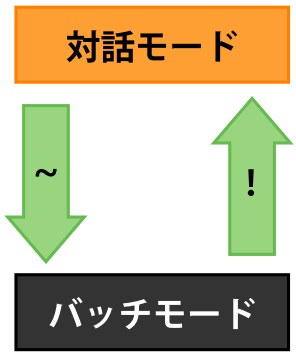
付録 A Hermit-At ブートローダー

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 カーネルパラメーターの設定等 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。

Hermit-AT のモード

Hermit-AT には、2 つのモードがあります。コマンドプロンプトを表示して対話的に動作する「対話モード」と、Hermit-AT ダウンローダと通信するための「バッチモード」です。バッチモードではコマンドプロンプトの表示や入力した文字の表示を行いませんが、コマンドの実行は可能です。

起動直後の Hermit-AT は必ず対話モードになっています。対話モードからバッチモードに移行するにはチルダ「~」を、バッチモードから対話モードに移行するにはエクスクラメーションマーク「!」を入力します。



Hermit-AT ダウンローダと通信を行った場合は、バッチモードに移行します。これは通信を確立するために Hermit-AT ダウンローダがチルダを送信するためです。

対話モードからバッチモードに移行したり、バッチモード中に入力したコマンドが成功した場合などは以下のように表示されます。

```
+OK
```

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
0xa0000000:0xa1ffffff 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.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 カーネルパラメーター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 [カーネルパラメーター]...
説明：カーネルパラメーターを設定します。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=ttymxc1,115200
hermit> setenv
1: console=ttymxc1,115200
hermit> clearenv
hermit> setenv
hermit>
```

図 A.14 setenv と clearenv の使用例

A.7.2. Linux カーネルパラメーター

Linux カーネルパラメーターの例を、「表 A.1. よく使用される Linux カーネルパラメーター」に示します。この他のオプションについては、linux-3.14/Documentation/kernel-parameters.txt を参照してください。

表 A.1 よく使用される Linux カーネルパラメーター

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 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。

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: 192.168.10.1)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 tftp 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

クライアント IP アドレスは、TFTP サーバと通信を行う際一時的に使用する IP アドレスです。同一ネットワーク内で重複しない IP アドレスを指定してください。

MMC/SD 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 MMC/SD 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 MMC/SD 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。
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. 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> [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 tftpdl 構文

表 A.3 tftpdl 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指定します。

A.10.1. tftpdl の使用例

```
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.20 tftpdl の使用例

クライアント IP アドレスは、TFTP サーバと通信を行う際一時的に使用する IP アドレスです。同一ネットワーク内で重複しない IP アドレスを指定してください。

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。tftpdl 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. tftpdl 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.                                    ❶
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                        ❷
    :
    :
```

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。

❷ 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。

クライアント IP アドレスは、TFTP サーバと通信を行う際一時的に使用する IP アドレスです。同一ネットワーク内で重複しない IP アドレスを指定してください。

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.                                ❶
Doing console=ttymxc1                                                      ❷
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                        ❸
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=ttymxc1,115200
MXC IRQ initialized
    :
    :
```

図 A.24 boot 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。

❷ setenv でカーネルパラメーター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。

❸ 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。

改訂履歴

バージョン	年月日	改訂内容
2.0.0	2015/10/26	・Linux 3.14 対応のため全面改版 **linux-2.6.26-at に対応した情報は、旧版(v1.x.x)のドキュメントを参照してください。下記 URL からダウンロードすることができます。** http://download.atmark-techno.com/armadillo-420/document/

Armadillo-400 シリーズソフトウェアマニュアル
Version 2.0.0
2015/10/26

株式会社アットマークテクノ

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

横浜営業所
〒 221-0835 横浜市神奈川区鶴屋町 3 丁目 30-4 明治安田生命横浜西口ビル 7F
TEL 045-548-5651 FAX 050-3737-4597