Armadillo-Box WS1
製品マニュアル

AB110-D00Z
AB110-C00Z

Version 1.0.0
2015/07/27

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-Box WS1 製品マニュアル

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 1.0.0
2015/07/27

目次

図目次

表目次

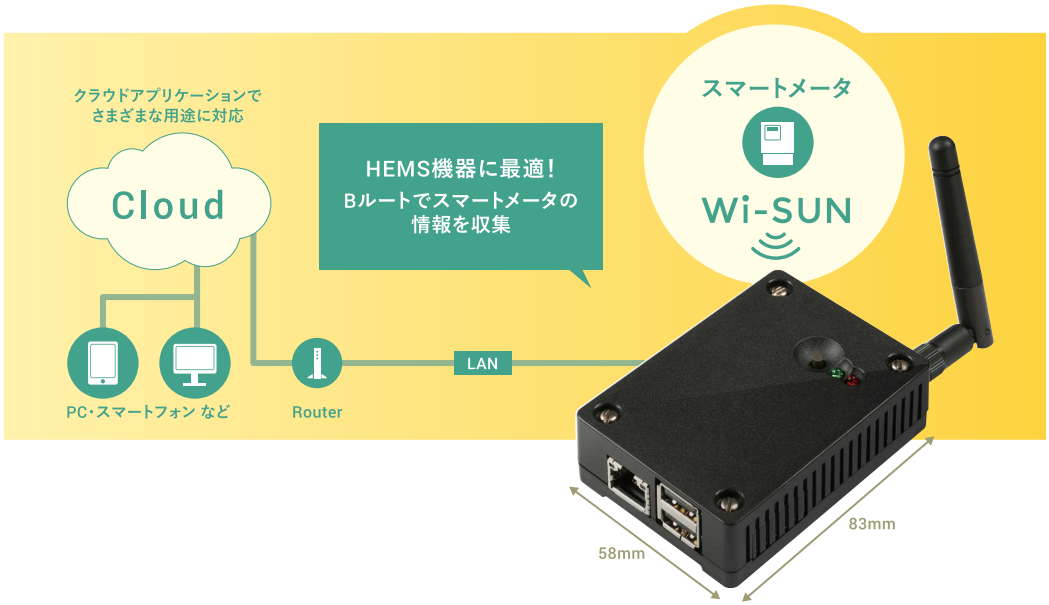
1. はじめに

このたびは Armadillo-Box WS1 をご利用いただき、ありがとうございます。

Armadillo-Box WS1 は、Wi-SUN に対応した HEMS 装置向けのプラットフォームです。スマートメーターの標準通信ネットワーク規格である Wi-SUN を標準搭載することで、すぐに HEMS 機器の開発を開始することができます。もちろんスマートメーター以外の様々な Wi-SUN 機器とも通信可能です。

Armadillo-Box WS1 では、ECHONET Lite のプロトコルスタックを用意しています。HEMS 標準プロトコルである ECHONET Lite を使うことで、簡単に HEMS 機器の開発が可能です。

Armadillo-Box WS1 には、標準 OS として Linux がプリインストールされています。そのため、オープンソースソフトウェアを含む多くのソフトウェア資産を活用し、自由にオリジナルのアプリケーションを開発することができます。開発言語としては、C/C++言語だけでなく、Oracle Java をサポートしています。使い慣れた言語で、Wi-SUN 経由で取得したデーターをクラウド側に転送したり、クラウドから Armadillo-Box WS1 や Wi-SUN 機器をコントロールするなど、Wi-SUN 機器とインターネットプロトコルとのゲートウェイを開発することが可能です。



以降、本書では他の Armadillo ブランド製品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. 本書で扱うこと扱わないこと

1.1.1. 扱うこと

本書では、Armadillo-Box WS1 の使い方、製品仕様(ソフトウェアおよびハードウェア)、製品を開発するために必要となる情報、その他注意事項について記載しています。Linux あるいは組み込み機器に不慣れな方でも読み進められるよう、コマンドの実行例なども記載しています。

1.1.2. 扱わないこと

本書では、一般的な Linux のプログラミング、デバッグ方法やツールの扱い方など、一般的な情報や、他に詳しい情報があるものは扱いません。また、(Armadillo-Box WS1 を使用した)最終製品あるいはサービスに、固有な情報や知識も含まれていません。

1.2. 本書で必要となる知識と想定する読者

本書は、読者として Armadillo-Box WS1 を使って、Wi-SUN と従来のネットワークを中継するゲートウェイ機器を開発するエンジニアを想定して書かれています。また、「Armadillo-Box WS1 を使うと、どのようなことが実現可能なのか 」を知りたいと考えている設計者・企画者も対象としています。Armadillo-Box WS1 は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

ソフトウェアエンジニア

端末からのコマンドの実行方法など、基本的な Linux の扱い方を知っているエンジニアを対象読者として想定しています。プログラミング言語として C/C++を扱えることは必ずしも必要ではありませんが、基礎的な知識がある方が理解しやすい部分もあります。

ハードウェアエンジニア

電子工学の基礎知識を有したエンジニアを対象読者として想定しています。回路図や部品表を読み、理解できる必要があります。

1.3. ユーザー限定コンテンツ

アットマークテクノ ユーザーズサイトで購入製品登録を行うと、製品をご購入いただいたユーザーに限定して公開している限定コンテンツにアクセスできるようになります。主な限定コンテンツには、下記のものがあります。

- リカバリ用ユーザーランドイメージ(工場出荷時と同等のもの)
- 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「18. ユーザー登録」を参照してください。

1.4. 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。本書を読み始める前に、Armadillo サイトで最新版の情報をご確認ください。

Armadillo サイト - Armadillo-Box WS1 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-box-ws1/downloads

1.5. 本書の構成

本書には、ご利用にあたっての注意事項や、ご購入時のソフトウェアの状態、ハードウェア・ソフトウェアをカスタマイズする場合に必要な情報などが記載されています。

◆ はじめにお読みください。

「1. はじめに」、「2. 注意事項」

◆ Armadillo-Box WS1 の仕様を紹介します。

「3. 製品概要」

◆ 工場出荷状態のソフトウェアの使い方や、動作を確認する方法を紹介します。

「4. Armadillo の電源を入れる前に」、「5. 起動と終了」、「6. 動作確認方法」、「7. コンフィグ領域 – 設定ファイルの保存領域」

◆ 工場出荷状態のソフトウェア仕様について紹介します。

「8. Linux カーネル仕様」、「9. ユーザーランド仕様」、「10. ブートローダー仕様」

◆ システム開発に必要な情報を紹介します。

「11. ビルド手順」、「12. フラッシュメモリの書き換え方法」、「13. 開発の基本的な流れ」

◆ ハードウェア仕様について紹介します。

「14. ハードウェア仕様」、「15. 電気的仕様」、「16. 組み立て」、「17. 形状図」

◆ ご購入ユーザーに限定して公開している情報の紹介やユーザー登録について紹介します。

「18. ユーザー登録」

1.6. 表記について

1.6.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

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

1.6.2. コマンド入力例

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

表 1.2 表示プロンプトと実行環境の関係

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

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

表 1.3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.6.3. 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。

1.7. 謝辞

Armadillo で使用しているソフトウェアの多くは 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. 電波障害について

Armadillo-Box WS1 は、2015 年 7 月 27 日現在 VCCI クラス B 情報技術装置の申請中です。

この装置は、VCCI クラス B 情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。取扱い説明書に従って正しい取扱いをして下さい。VCCI-B

2.3. 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

2.4. 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。

2.5. 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. 製品概要

3.1. 製品の特長

3.1.1. Armadillo とは

「Armadillo (アルマジロ)」は、ARM コアプロセッサ搭載・Linux 対応の組み込みプラットフォームのブランドです。Armadillo ブランド製品には以下の特長があります。

◆ ARM プロセッサ搭載・省電力設計

ARM コアプロセッサを搭載しています。１〜数ワット程度で動作する省電力設計で、発熱が少なくファンを必要としません。

◆ 小型・手のひらサイズ

CPU ボードは名刺サイズ程度の手のひらサイズが主流です。名刺１/３程度の小さな CPU モジュールや無線 LAN モジュール等、超小型のモジュールもラインアップしています。

◆ 標準 OS として Linux をプリインストール

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。ソースコードをオープンソースとして公開しています。

◆ 開発環境

Armadillo の開発環境として、「Atmark Techno Development Environment (ATDE)」を無償で提供しています。ATDE は、VMware など仮想マシン向けのデータイメージです。このイメージには、Linux デスクトップ環境をベースに GNU クロス開発ツールやその他の必要なツールが事前にインストールされています。ATDE を使うことで、開発用 PC の用意やツールのインストールなどといった開発環境を整える手間を軽減することができます。

3.1.2. Armadillo-Box WS1 とは

Armadillo-Box WS1 は、組み込みプラットフォームとして実績のある Armadillo をベースにした、HEMS 装置向けのプラットフォームです。"Wi-SUN"に準拠した無線モジュールを標準搭載し、Wi-SUN 準拠のスマートメーターと通信する事で、小型の HEMS 装置を開発する事が可能です。ネットワークに強い Linux を標準 OS としてプリインストールしているため、オープンソースソフトウェアを中心とした、各種ソフトウェア資産を活用できます。また Oracle Java 標準対応しているため、C/C++言語以外でのソフトウェア開発が可能です。

Wi-SUN モジュールを標準搭載

低消費電力かつ長距離通信が可能な無線規格"Wi-SUN"に準拠した無線モジュールを標準搭載しています。すぐにスマートメーターのと通信部分の開発を開始することができます。

Oracle Java 標準搭載

Oracle Java が標準搭載されていますので C や C++ 以外にも Java で開発することが可能です。

ECHONET Lite

HEMS 標準プロトコルである ECHONET Lite を使うことで、簡単に HEMS 機器の開発が可能です。ECHONET Lite を Java で実装している OpenECHO も利用可能です。OpenECHO はオープンソースで提供されています。利用可能な商用の ECHONET Lite プロトコルスタックもあります。

小型・低消費電力

組み込み機器プラットフォームとして実績のある Armadillo をベースに採用。小型で低消費電力を実現しています。

Linux 3.14 LTSI

基本 OS に、ネットワークに強い Linux を採用しています。また 長期的なメンテナンスに対応するために、LTSI バージョンの Linux 3.14 を採用しています。アプリケーションやネットワークスタックに、オープンソースソフトウェアを中心とした、各種ソフトウェア資産を活用できます。

3.2. 仕様

Armadillo-Box WS1 の主な仕様は次のとおりです。

表 3.1 仕様

プロセッサ	Freescale Semiconductor i.MX257(MCIMX257)
	ARM926EJ-S コア 命令/データキャッシュ 16KByte/16KByte 内部 SRAM 128KByte Thumb code(16bit 命令セット)サポート
システムクロック	CPU コアクロック: 400MHz BUS クロック: 133MHz 源発振クロック: 32.768kHz, 24MHz
RAM	LPDDR SDRAM: 128MByte バス幅 16bit
ROM	NOR 型フラッシュメモリ: 32MByte バス幅 16bit
LAN(Ethernet)	10BASE-T/100BASE-TX x 1
Wi-SUN	ROHM 製 BP35A1 搭載[a]
SD/MMC	microSD x 1[b]
USB	USB 2.0 Host x 2(High Speed x 1、Full Speed x 1)
シリアル(UART)	RS232C レベル x 1
カレンダー時計	非搭載[c]
LED	ユーザー LED x 2
スイッチ	ユーザースイッチ x 1
電源電圧	DC 5V±5%
消費電力[d]	約 1.3W
使用温度範囲	-10～60℃[e](ただし結露なきこと)
外形サイズ	58.0 x 83.0 x 24.3mm(突起部を除く)

[a]開発セット付属の Wi-SUN モジュール用外付けアンテナを接続可能。

[b]microSD スロットは活線挿抜に対応していません。また、ケース外から操作できません。

[c]リアルタイムクロックを搭載可能です。詳細につきましてはお問い合わせください。

[d]USB デバイス、SD デバイス等の外部機器の消費電力を除く。

[e]本体の使用温度範囲です。開発セット付属の AC アダプタの使用温度範囲は 0～40℃となります。

3.3. Armadillo-Box WS1 の外観

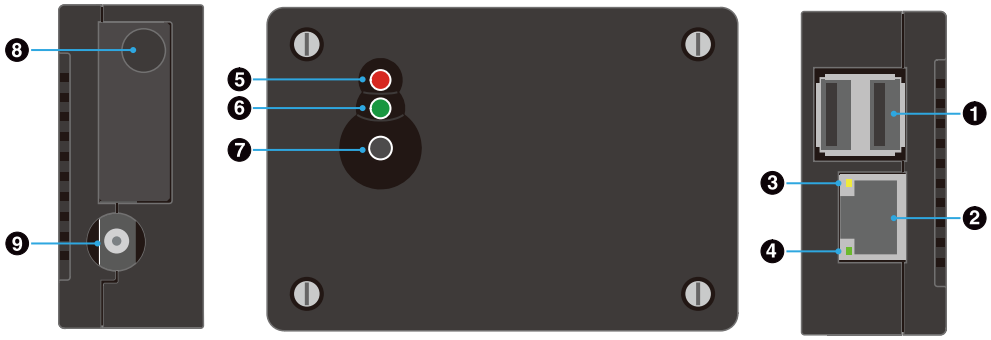


図 3.1 Armadillo-Box WS1 の外観

表 3.2 各部名称と機能

番号	名称	説明
1	USB コネクタ	USB メモリ等を接続します。
2	LAN コネクタ	LAN ケーブルを接続します。
3	LAN アクティビティ LED(黄色)	LAN ポートの使用状態を表し、送受信中は点灯します。
4	LAN リンク LED(緑色)	LAN ポートが使用可能な時に点灯します。
5	ユーザー LED(赤色)	ユーザーで自由に機能を設定できる LED です。
6	ユーザー LED(緑色)	ユーザーで自由に機能を設定できる LED です。
7	ユーザースイッチ	ユーザーで自由に機能を設定できるタクトスイッチです。
8	外付けアンテナ取り付け穴	Wi-SUN モジュール用外付けアンテナの取り付け穴です。Wi-SUN モジュールにはアンテナが内蔵されていますので、外付けアンテナを接続しなくても通信可能です。出荷状態では、キャップが取付けられています。
9	電源コネクタ	付属の AC アダプタを接続します。

3.4. ブロック図

Armadillo-Box WS1 のブロック図は次のとおりです。

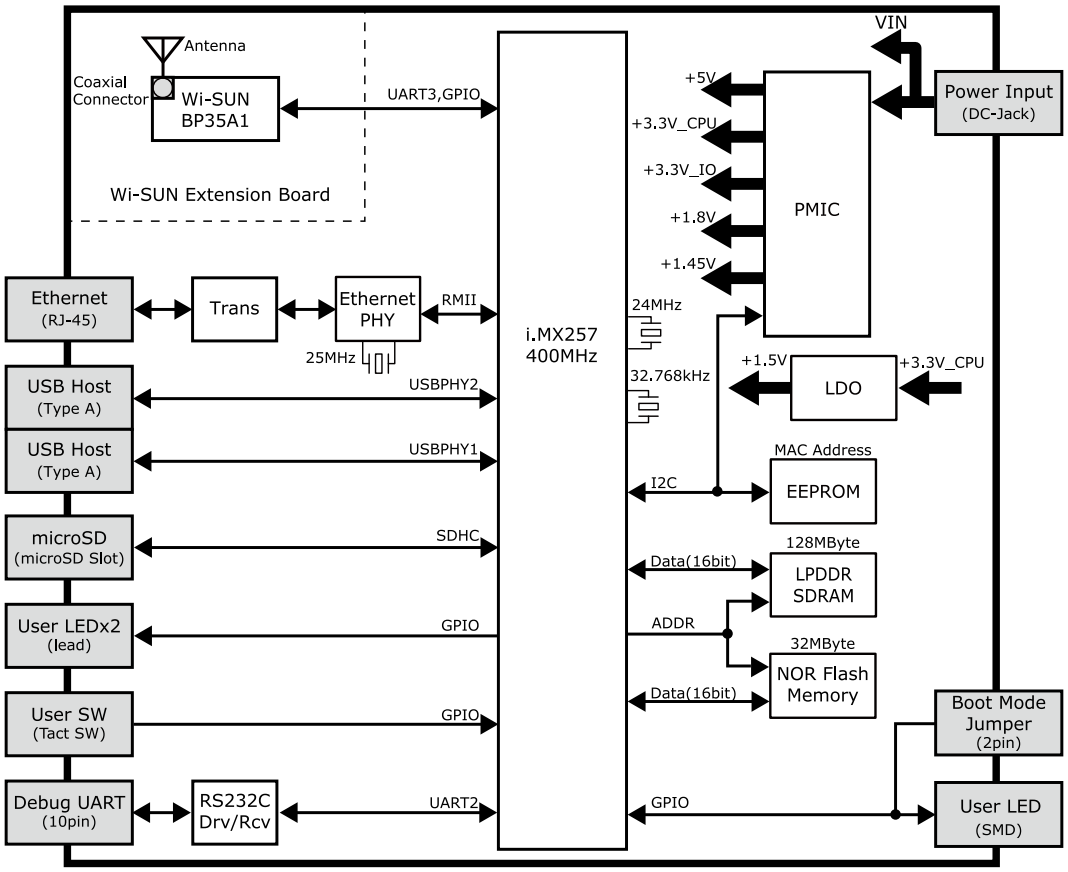


図 3.2 Armadillo-Box WS1 ブロック図

3.5. ソフトウェア構成

Armadillo-Box WS1 で動作するソフトウェアの構成について説明します。

Armadillo-Box WS1 で利用可能なソフトウェアを「表 3.3. Armadillo-Box WS1 で利用可能なソフトウェア」に示します。

表 3.3 Armadillo-Box WS1 で利用可能なソフトウェア

ソフトウェア	説明
Hermit-At	ブートローダーです。Linux カーネルを起動させる機能の他に、ダウンローダーと協調動作を行いフラッシュメモリを書き替える機能など様々な機能を持っています。工場出荷状態ではブートローダーイメージはフラッシュメモリに配置されています。
Linux カーネル	バージョン 3.14 の Linux カーネルです。工場出荷状態では Linux カーネルイメージはフラッシュメモリに配置されていますが、Hermit-At の機能により microSD カードに配置することもできます。
Atmark Dist	uClinux-dist をベースにしたアットマークテクノ製品向けの Linux ディストリビューションです。フラッシュメモリ向けのユーザーランドを提供します。工場出荷状態では Atmark Dist 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いますが、microSD カードなどのストレージに配置することもできます。

Armadillo-Box WS1 のフラッシュメモリのメモリマップを「表 3.4. フラッシュメモリ メモリマップ」に示します。

表 3.4 フラッシュメモリ メモリマップ

物理アドレス	パーティション名	サイズ	工場出荷状態で書き込まれているソフトウェア
0xA0000000 ｜ 0xA001FFFF	bootloader	128kByte	Hermit-At ブートローダーイメージ
0xA0020000 ｜ 0xA041FFFF	kernel	4MByte	Linux カーネルイメージ
0xA0420000 ｜ 0xA1EFFFFF	userland	26.875Mbyte	Atmark Dist ユーザーランドイメージ
0xA1F00000 ｜ 0xA1FFFFFF	config	1MByte	アプリケーションの設定情報など

4. Armadillo の電源を入れる前に

4.1. 準備するもの

Armadillo を使用する前に、次のものを必要に応じて準備してください。

作業用 PC	Linux または Windows が動作し、ネットワークインターフェースとシリアルインターフェースを持つ PC です。「4.2. 開発/動作確認環境の構築」を参照して、作業用 PC 上に開発/動作確認環境を構築してください。
ネットワーク環境	Armadillo と作業用 PC をネットワーク通信ができるようにしてください。
microSD カード	microSD スロットの動作を確認する場合などに利用します。
USB メモリ	USB の動作を確認する場合などに利用します。
tar.xz 形式のファイルを展開するソフトウェア	開発/動作確認環境を構築するために利用します。Linux では、tar[1]で展開できます。Windows では、7-Zip や Lhaz などが対応しています。7-Zip は、開発用 DVD に収録されています。

4.2. 開発/動作確認環境の構築

アットマークテクノ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。

仮想化ソフトウェアとして、VMware の他に Oracle VM VirtualBox が有名です。Oracle VM VirtualBox には以下の特徴があります。

- GPL v2(General Public License version 2)で提供されている[2]
- VMware 形式の仮想ディスク(.vmdk)ファイルに対応している

Oracle VM VirtualBox から ATDE を起動し、ソフトウェア開発環境として使用することができます。

ATDE は、バージョンにより対応するアットマークテクノ製品が異なります。本製品に対応している ATDE は、ATDE5 の v20150727 以降です。

ATDE5 は Debian GNU/Linux 7(コードネーム wheezy)をベースに、Armadillo-Box WS1 のソフトウェア開発を行うために必要なクロス開発ツールや、Armadillo-Box WS1 の動作確認を行うために必要なツールが事前にインストールされています。

[1]tar.xz 形式のファイルを展開するには Jxf オプションを指定します。
[2]バージョン 3.x までは PUEL(VirtulBox Personal Use and Evaluation License)が適用されている場合があります。

4.2.1. ATDE5 セットアップ

4.2.1.1. 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。

4.2.1.2. ATDE5 アーカイブの取得

「表 4.1. 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。

表 4.1 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 v20150727 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。

4.2.1.3. 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 4.1. Windows で ATDE5 のアーカイブ展開する」に、Linux での展開方法を「手順 4.2. Linux で tar.xz 形式のファイルを展開する」に示します。

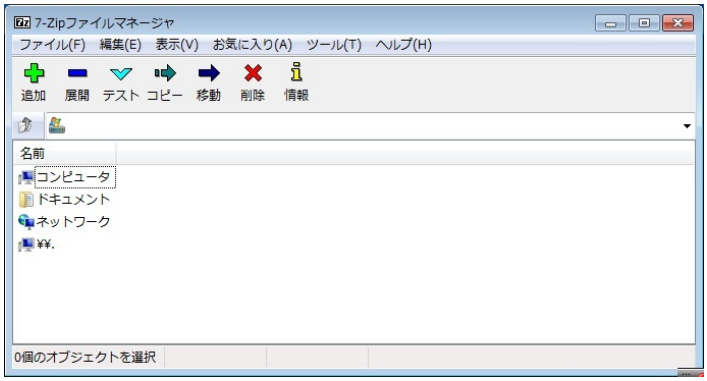
手順 4.1 Windows で ATDE5 のアーカイブ展開する

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。

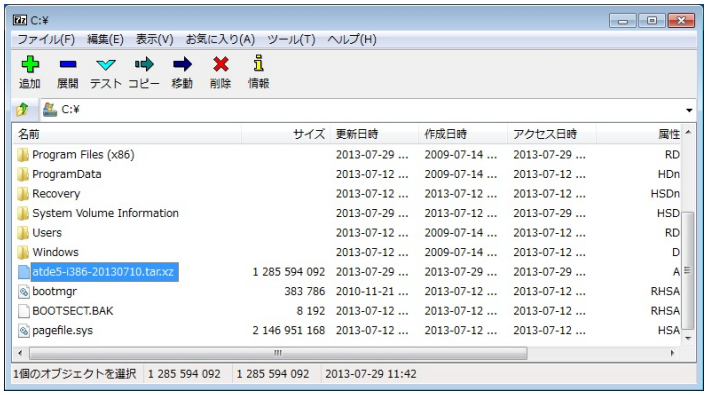
2. **7-Zip の起動**

   7-Zip を起動します。



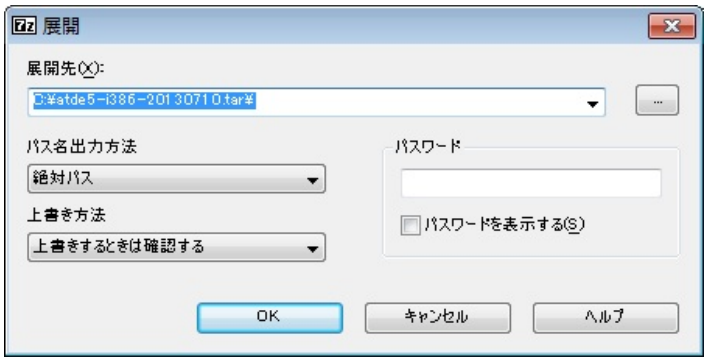
3. **xz 圧縮ファイルの選択**

   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。



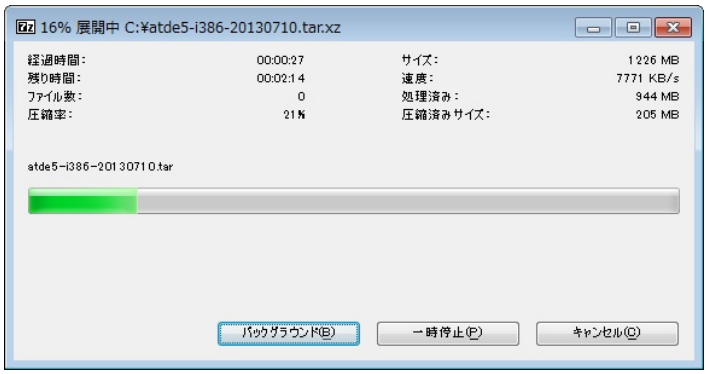
4. **xz 圧縮ファイルの展開先の指定**

   「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**

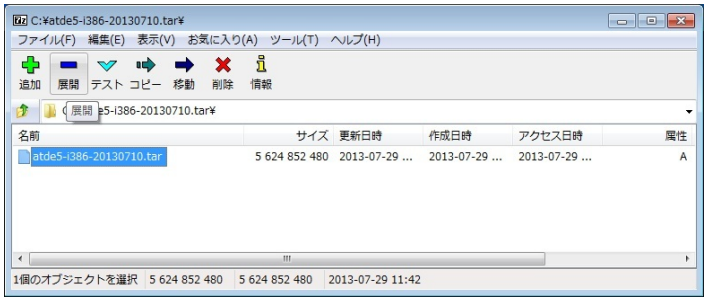
展開が始まります。



6. **tar アーカイブファイルの選択**

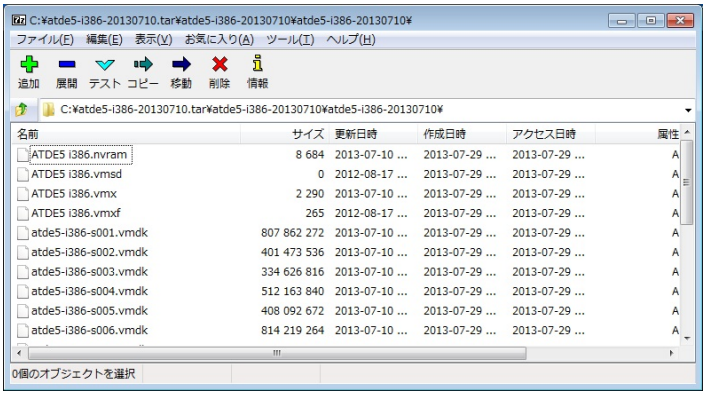
xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。

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 4.2 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。

```
[PC ~]$ 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。

```
[PC ~]$ 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
```

4.2.1.4. 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 4.2. ユーザー名とパスワード」に示します[3]。

表 4.2 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

[3]特権ユーザーで GUI ログインを行うことはできません。

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。

4.2.2. 取り外し可能デバイスの使用

VMware は、ゲスト OS (ATDE)による取り外し可能デバイス(USB デバイスや DVD など)の使用をサポートしています。デバイスによっては、ホスト OS (VMware を起動している OS)とゲスト OS で同時に使用することができません。そのようなデバイスをゲスト OS で使用するためには、ゲスト OS にデバイスを接続する操作が必要になります。

取り外し可能デバイスの使用方法については、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。

Armadillo-Box WS1 の動作確認を行うためには、「表 4.3. 動作確認に使用する取り外し可能デバイス」に示すデバイスをゲスト OS に接続する必要があります。

表 4.3 動作確認に使用する取り外し可能デバイス

デバイス	デバイス名
作業用 PC の物理シリアルポート	シリアルポート

4.2.3. コマンドライン端末(GNOME 端末)の起動

ATDE5 で、CUI (Character-based User Interface)環境を提供するコマンドライン端末を起動します。ATDE5 で実行する各種コマンドはコマンドライン端末に入力し、実行します。コマンドライン端末にはいくつかの種類がありますが、ここでは GNOME デスクトップ環境に標準インストールされている GNOME 端末を起動します。

GNOME 端末を起動するには、「図 4.1. GNOME 端末の起動」のようにデスクトップ左上のメニューから「端末」を選択してください。

図 4.1 GNOME 端末の起動

「図 4.2. GNOME 端末のウィンドウ」のようにウィンドウが開きます。

図 4.2 GNOME 端末のウィンドウ

4.2.4. シリアル通信ソフトウェア(minicom)の使用

シリアル通信ソフトウェア(minicom)のシリアル通信設定を、「表 4.4. シリアル通信設定」のように設定します。また、minicom を起動する端末の横幅を 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 4.4 シリアル通信設定

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

minicom の設定を開始するには、「図 4.3. minicom 設定方法」のようにしてください。設定完了後、デフォルト設定(dfl)に保存して終了します。

```
[ATDE ~]$ LANG=C minicom --setup
```

図 4.3 minicom 設定方法

minicom を起動させるには、「図 4.4. minicom 起動方法」のようにしてください。

```
[ATDE ~]$ LANG=C minicom --noinit --wrap --device /dev/ttyUSB0
```

図 4.4 minicom 起動方法

デバイスファイル名は、環境によって/dev/ttyS0 や/dev/ttyUSB1 など、本書の実行例とは異なる場合があります。

minicom を終了させるには、まず Ctrl+a に続いて q キーを入力します。その後、以下のように表示されたら「Yes」にカーソルを合わせて Enter キーを入力すると minicom が終了します。

```
+-----------------------+
| Leave without reset?  |
|     Yes       No      |
+-----------------------+
```

図 4.5 minicom 終了確認

Ctrl+a に続いて z キーを入力すると、minicom のコマンドヘルプが表示されます。

4.3. インターフェースレイアウト

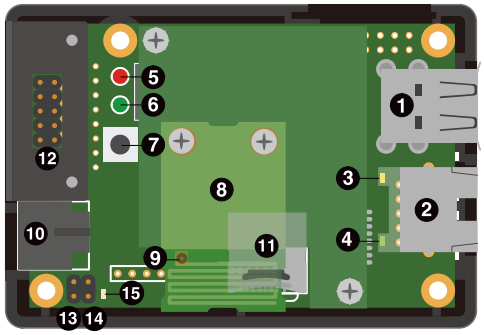


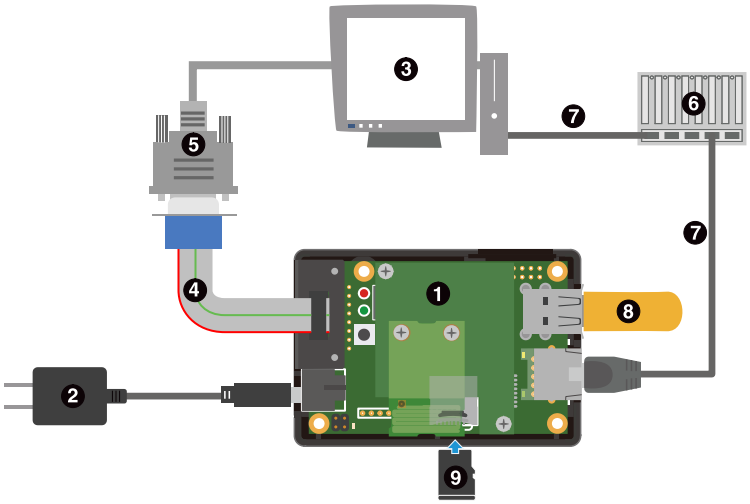
図 4.6 インターフェースレイアウト図

表 4.5 インターフェース内容

部品番号	インターフェース名	形状	備考
1	USB インターフェース	Type-A コネクタ(2 ポートスタック)	
2	LAN インターフェース	RJ-45 コネクタ	
3	LAN アクティビティ LED(黄色)	LED(面実装)	
4	LAN リンク LED(緑色)	LED(面実装)	
5	ユーザー LED(赤色)	LED(φ3mm)	
6	ユーザー LED(緑色)	LED(φ3mm)	
7	ユーザースイッチ	タクトスイッチ	
8	Wi-SUN モジュール	モジュール基板	
9	外付けアンテナインターフェース	小型同軸コネクタ	
10	電源入力インターフェース	DC ジャック	対応プラグ: EIAJ#2
11	microSD インターフェース	ヒンジタイプ	
12	デバッグシリアルインターフェース	ピンヘッダ 10 ピン(2.54 ピッチ)	
13	起動モード設定ジャンパ(JP1)	ピンヘッダ 2 ピン(2.54 ピッチ)	
14	起動モード設定ジャンパ(JP2)	ピンヘッダ 2 ピン(2.54 ピッチ)	
15	ユーザー LED(黄色)	LED(面実装)	

4.4. 接続方法

Armadillo-Box WS1 と周辺装置の接続例を次に示します。



1. Armadillo-Box WS1
2. AC アダプタ(5V)[4]
3. 作業用 PC
4. D-Sub 変換ケーブル[4]
5. シリアルクロスケーブル[4]
6. LAN HUB
7. LAN ケーブル
8. USB メモリ
9. microSD カード

図 4.7 Armadillo-Box WS1 の接続例

D-Sub 変換ケーブルを接続する場合は、外付けアンテナ取り付け穴のキャップを外してください。

microSD カードを挿抜する際には、Wi-SUN モジュールを取り外してください。 無理に挿抜した場合、microSD カードが正常に挿入されないなどの原因で、動作不良を起こす場合があります。 microSD カードの挿抜方法については「14.8.1. microSD カードの挿入方法」、「14.8.2. microSD カードの抜去方法」をご確認ください。

[4]Armadillo-Box WS1 開発セット付属品

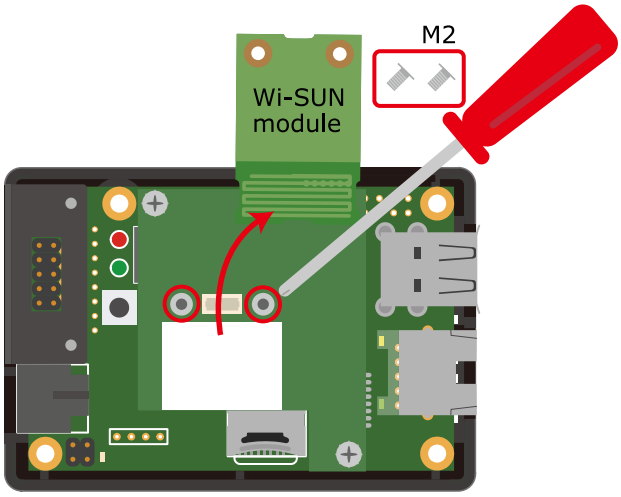


図 4.8 Wi-SUN モジュールの取り外し方

microSD カード挿抜後、取り外した Wi-SUN モジュールは元の状態に戻してください。

4.5. ジャンパピンの設定について

Armadillo-Box WS1 は、ジャンパピン(JP1 および JP2)の設定によりブートモードを選択する事ができます。

ジャンパピンの設定により選択されるブートモードを「表 4.6. ジャンパの設定」に示します。

表 4.6 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

JP1 は、オンボードフラッシュメモリブートモードと、UART ブートモードの選択を行います。

オンボードフラッシュメモリブートモードは、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を起動するモードです。

オンボードフラッシュメモリブートモードに設定されている場合、標準のブートローダーである Hermit-At は、JP2 の設定により 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かを判別します。

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、「12.5. ブートローダーが起動しなくなった場合の復旧作業」 を参照してください。

Armadillo-Box WS1 のジャンパピンの位置を「図 4.9. ジャンパピンの位置」に示します。

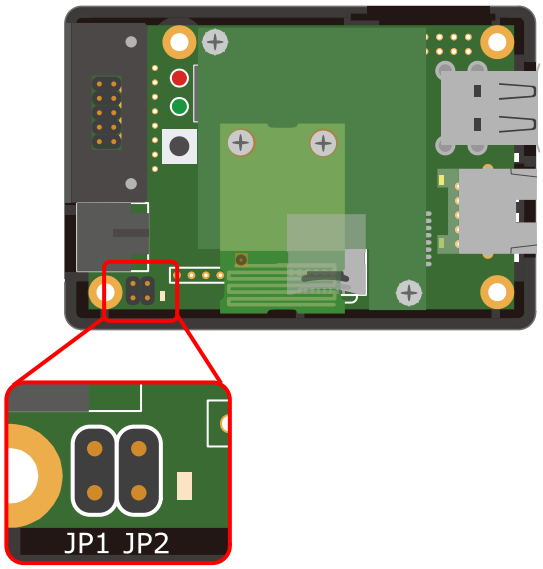
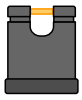


図 4.9 ジャンパピンの位置

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。

4.6. vi エディタの使用方法

vi エディタは、Armadillo に標準でインストールされているテキストエディタです。本書では、Armadillo の設定ファイルの編集などに vi エディタを使用します。

vi エディタは、ATDE にインストールされてる gedit や emacs などのテキストエディタとは異なり、モードを持っていることが大きな特徴です。vi のモードには、コマンドモードと入力モードがあります。コマンドモードの時に入力した文字はすべてコマンドとして扱われます。入力モードでは文字の入力ができます。

本章で示すコマンド例は ATDE で実行するよう記載していますが、Armadillo でも同じように実行することができます。

4.6.1. vi の起動

vi を起動するには、以下のコマンドを入力します。

```
[ATDE ~]# vi [file]
```

図 4.10 vi の起動

file にファイル名のパスを指定すると、ファイルの編集(*file* が存在しない場合は新規作成)を行います。vi はコマンドモードの状態で起動します。

4.6.2. 文字の入力

文字を入力するにはコマンドモードから入力モードへ移行する必要があります。コマンドモードから入力モードに移行するには、「表 4.7. 入力モードに移行するコマンド」に示すコマンドを入力します。入力モードへ移行後は、キーを入力すればそのまま文字が入力されます。

表 4.7 入力モードに移行するコマンド

コマンド	動作
i	カーソルのある場所から文字入力を開始
a	カーソルの後ろから文字入力を開始

入力モードからコマンドモードに戻りたい場合は、ESC キーを入力することで戻ることができます。現在のモードが分からなくなった場合は、ESC キーを入力し、一旦コマンドモードへ戻ることにより混乱を防げます。

日本語変換機能を OFF に

vi のコマンドを入力する時は ATDE の日本語入力システム(Mozc)を OFF にしてください。日本語入力システムの ON/OFF は、半角/全角キーまたは、Shift+Space キーで行うことができます。

「i」、「a」それぞれのコマンドを入力した場合の文字入力の開始位置を「図 4.11. 入力モードに移行するコマンドの説明」に示します。

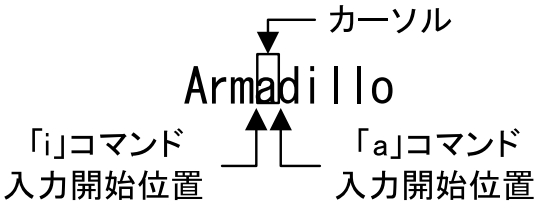


図 4.11 入力モードに移行するコマンドの説明

vi での文字削除

コンソールの環境によっては BS(Backspace)キーで文字が削除できず、「^H」文字が入力される場合があります。その場合は、「4.6.4. 文字の削除」で説明するコマンドを使用し、文字を削除してください。

4.6.3. カーソルの移動

方向キーでカーソルの移動ができますが、コマンドモードで「表 4.8. カーソルの移動コマンド」に示すコマンドを入力することでもカーソルを移動することができます。

表 4.8 カーソルの移動コマンド

コマンド	動作
h	左に 1 文字移動
j	下に 1 文字移動
k	上に 1 文字移動
l	右に 1 文字移動

4.6.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.9. 文字の削除コマンド」に示すコマンドを入力します。

表 4.9 文字の削除コマンド

コマンド	動作
x	カーソル上の文字を削除
dd	現在行を削除

「x」コマンド、「dd」コマンドを入力した場合に削除される文字を「図 4.12. 文字を削除するコマンドの説明」に示します。

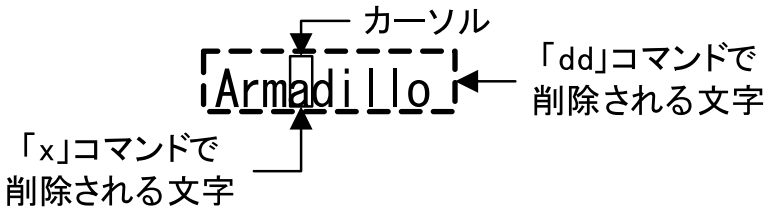


図 4.12 文字を削除するコマンドの説明

4.6.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.10. 保存・終了コマンド」に示します。

表 4.10 保存・終了コマンド

コマンド	動作
:q!	変更を保存せずに終了
:w *[file]*	ファイル名を *file* に指定して保存
:wq	ファイルを上書き保存して終了

保存と終了を行うコマンドは「:」(コロン)からはじまるコマンドを使用します。":"キーを入力すると画面下部にカーソルが移り入力したコマンドが表示されます。コマンドを入力した後 Enter キーを押すことで、コマンドが実行されます。

5. 起動と終了

5.1. 起動

Armadillo-Box WS1 のブートモードを保守モードに設定してください。ブートモードの設定方法は、「4.5. ジャンパピンの設定について」を参照してください。

Armadillo-Box WS1 に電源を接続すると、シリアル通信ソフトウェアには次のように表示されます。

```
Hermit-At v3.6.0 (armadillo-box-ws1) compiled at 20:51:51, Jul 24 2015
hermit>
```

図 5.1 電源投入直後のログ

Linux システムを起動するには、次のように "boot"コマンドを実行してください。コマンドを実行するとブートローダーが Linux システムを起動させます。シリアル通信ソフトウェアには Linux の起動ログが表示されます。

```
hermit> boot
Uncompressing  kernel.....................................................
.........................................................................
.....................................................................don
e.
Uncompressing ramdisk.....................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.........................................................................
.....................................................done.
Booting Linux on physical CPU 0x0
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.14.36-at2 (atmark@atde5) (gcc version 4.6.3 (Debian 4.6.3-14atma
rk1) ) #1 PREEMPT Fri Jul 24 20:37:52 JST 2015
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: Armadillo-440
Memory policy: Data cache writeback
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 32512
Kernel command line: console=ttymxc1,115200 root=/dev/ram0
PID hash table entries: 512 (order: -1, 2048 bytes)
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
allocated 262144 bytes of page_cgroup
please try 'cgroup_disable=memory' option if you don't want memory cgroups
Memory: 97984K/131072K available (4791K kernel code, 277K rwdata, 1596K rodata,
```

```
230K init, 294K bss, 33088K reserved)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    vmalloc : 0xc8800000 - 0xff000000   ( 872 MB)
    lowmem  : 0xc0000000 - 0xc8000000   ( 128 MB)
    modules : 0xbf000000 - 0xc0000000   (  16 MB)
      .text : 0xc0008000 - 0xc0645040   (6389 kB)
      .init : 0xc0646000 - 0xc067fa3c   ( 231 kB)
      .data : 0xc0680000 - 0xc06c56c4   ( 278 kB)
       .bss : 0xc06c56c4 - 0xc070f24c   ( 295 kB)
Preemptible hierarchical RCU implementation.
NR_IRQS:16 nr_irqs:16 16
MXC IRQ initialized
Switching to timer-based delay loop
sched_clock: 32 bits at 66MHz, resolution 15ns, wraps every 64585974768ns
Console: colour dummy device 80x30
Calibrating delay loop (skipped), value calculated using timer frequency.. 133.0
0 BogoMIPS (lpj=665000)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
Initializing cgroup subsys memory
Initializing cgroup subsys devices
Initializing cgroup subsys freezer
Initializing cgroup subsys blkio
CPU: Testing write buffer coherency: ok
Setting up static identity map for 0x80489d98 - 0x80489df0
devtmpfs: initialized
pinctrl core: initialized pinctrl subsystem
regulator-dummy: no parameters
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
imx25-pinctrl imx25-pinctrl.0: initialized IMX pinctrl driver
bio: create slab <bio-0> at 0
eSDHC1 Vcc: at 3300 mV
USB VBUS: at 5000 mV
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
i2c i2c-0: IMX I2C adapter registered
pps_core: LinuxPPS API ver. 1 registered
pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@l
inux.it>
PTP clock support registered
Advanced Linux Sound Architecture Driver Initialized.
Switched to clocksource mxc_timer1
NET: Registered protocol family 2
TCP established hash table entries: 1024 (order: 0, 4096 bytes)
TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
TCP: Hash tables configured (established 1024 bind 1024)
TCP: reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (junk in compressed archive); looks like an initrd
```

```
Freeing initrd memory: 24440K (c1000000 - c27de000)
futex hash table entries: 256 (order: -1, 3072 bytes)
audit: initializing netlink subsys (disabled)
audit: type=2000 audit(0.929:1): initialized
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
msgmni has been set to 239
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 249)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
imx-sdma imx25-sdma: loaded firmware 1.0
imx-sdma imx25-sdma: initialized
imx21-uart.1: ttymxc1 at MMIO 0x43f94000 (irq = 48, base_baud = 7500000) is a IM
X
console [ttymxc1] enabled
imx21-uart.2: ttymxc2 at MMIO 0x5000c000 (irq = 34, base_baud = 7500000) is a IM
X
brd: module loaded
loop: module loaded
physmap platform flash device: 02000000 at a0000000
physmap-flash: Found 1 x16 devices at 0x0 in 16-bit bank. Manufacturer ID 0x0000
89 Chip ID 0x00891c
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Using buffer write method
Using auto-unlock on power-up/resume
cfi_cmdset_0001: Erase suspend on write enabled
Creating 4 MTD partitions on "physmap-flash":
0x000000000000-0x000000020000 : "nor.bootloader"
0x000000020000-0x000000420000 : "nor.kernel"
0x000000420000-0x000001f00000 : "nor.userland"
0x000001f00000-0x000002000000 : "nor.config"
libphy: fec_enet_mii_bus: probed
PPP generic driver version 2.4.2
usbcore: registered new interface driver cdc_ether
usbcore: registered new interface driver net1080
usbcore: registered new interface driver cdc_subset
usbcore: registered new interface driver sierra_net
usbcore: registered new interface driver cdc_ncm
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
usbcore: registered new interface driver cdc_acm
cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
usbcore: registered new interface driver usb-storage
usbcore: registered new interface driver usbserial
usbcore: registered new interface driver sierra
usbserial: USB Serial support registered for Sierra USB modem
ci_hdrc ci_hdrc.0: EHCI Host Controller
ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus number 1
ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
ci_hdrc ci_hdrc.1: EHCI Host Controller
ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus number 2
ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
```

```
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
i2c /dev entries driver
imx2-wdt imx2-wdt.0: timeout 60 sec (nowayout=0)
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
mmc0: no vqmmc regulator found
mmc0: SDHCI controller on sdhci-esdhc-imx25.0 [sdhci-esdhc-imx25.0] using DMA
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
usbcore: registered new interface driver snd-usb-audio
usbcore: registered new interface driver snd-usb-caiaq
oprofile: no performance counters
oprofile: using timer interrupt.
Netfilter messages via NETLINK v0.30.
nf_conntrack version 0.5.0 (1912 buckets, 7648 max)
ipip: IPv4 over IPv4 tunneling driver
gre: GRE over IPv4 demultiplexor driver
ip_gre: GRE over IPv4 tunneling driver
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP: cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 10
ip6_tables: (C) 2000-2006 Netfilter Core Team
sit: IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
registered taskstats version 1
eSDHC1 Vcc: incomplete constraints, leaving on
regulator-dummy: incomplete constraints, leaving on
input: gpio-keys as /devices/platform/gpio-keys/input/input0
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
ALSA device list:
  No soundcards found.
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 24440KiB [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing unused kernel memory: 228K (c0646000 - c067f000)
Mounting proc: done
Starting fsck for root filesystem.
fsck 1.25 (20-Sep-2001)
/dev/ram0: clean, 1001/1224 files, 22001/24440 blocks
Checking root filesystem: done
Remounting root rw: done
Mounting sysfs: done
Mounting tmpfs on /dev: done
Mounting tmpfs on /run: done
Cleaning up system: done
Running local start scripts.
Starting the hotplug events dispatcher udevd:done
Synthesizing the initial hotplug events:done
Loading /etc/config: done
Changing file permissions: done
Mounting devpts: done
Starting syslogd: done
Starting klogd: done
```

```
Starting basic firewall: done
Setting hostname: done
Configuring network interfaces: fec imx25-fec.0 eth0: Freescale FEC PHY driver [
SMSC LAN8710/LAN8720] (mii_bus:phy_addr=imx25-fec-1:00, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpc (v1.20.2) started
Sending discover...
libphy: imx25-fec-1:00 - Link is Up - 100/Full
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
Sending select for 192.0.2.100...
Lease of 192.0.2.100 obtained, lease time 86400
done
Starting avahi.daemon: random: avahi-daemon urandom read with 35 bits of entropy
 available
done
Running local start script (/etc/config/rc.local).

atmark-dist v1.42.0 (AtmarkTechno/Armadillo-Box-WS1)
Linux 3.14.36-at2 [armv5tejl arch]

abws1-0 login:
```

図 5.2 起動ログ

Armadillo-Box WS1 の起動状態をユーザー LED から確認することができます。

起動状態	ユーザー LED(赤色)	ユーザー LED(緑色)
ブートローダーが保守モード	消灯	点灯
Linux の起動中	点灯	点灯
Linux が起動完了	消灯	点灯

Armadillo-Box WS1 の起動後に表示される次のメッセージは、エラーメッセージではありません。

```
random: nonblocking pool is initialized
```

このメッセージは、/dev/urandom が内部的に使用するプール領域の初期化完了を示します。

5.2. ログイン

起動が完了するとログインプロンプトが表示されます。「表 5.1. シリアルコンソールログイン時のユーザ名とパスワード」に示すユーザでログインすることができます。

表 5.1 シリアルコンソールログイン時のユーザ名とパスワード

ユーザ名	パスワード	権限
root	root	root ユーザ
guest	(なし)	一般ユーザ

5.3. 終了方法

安全に終了させる場合は、次のようにコマンドを実行し、「System halted.」と表示されたのを確認してから電源を切断します。

```
[armadillo ~]# halt
[armadillo ~]#
System is going down for system reboot now.

Starting local stop scripts.
Syncing all filesystems: done
Unmounting all filesystems: umount: udev busy - remounted read-only
umount: devtmpfs busy - remounted read-only
done
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processesimx2-wdt imx2-wdt.0: Device shutdown: Expect reboot!
reboot: System halted
```

図 5.3 終了方法

microSD カードなどのストレージをマウントしていない場合は、電源を切断し終了させることもできます。

「System halted.」と表示されてから約 128 秒後、Armadillo-Box WS1 は自動的に再起動します。詳しくは、「8.3.9. ウォッチドッグタイマー」を参照してください。

ストレージにデータを書き込んでいる途中に電源を切断した場合、ファイルシステム、及び、データが破損する恐れがあります。ストレージをアンマウントしてから電源を切断するようにご注意ください。

6. 動作確認方法

6.1. 動作確認を行う前に

工場出荷状態でフラッシュメモリに書き込まれているイメージファイルは、最新版ではない可能性があります。最新版のブートローダーおよび Linux カーネルイメージファイルは Armadillo サイトから、ユーザーランドイメージファイルはユーザーズサイトからダウンロード可能です。最新版のイメージファイルに書き換えてからのご使用を推奨します。

イメージファイルの書き換えについては、「12. フラッシュメモリの書き換え方法」を参照してください。

6.2. ネットワーク

ここでは、ネットワークの設定方法やネットワークを利用するアプリケーションについて説明します。

6.2.1. 接続可能なネットワーク

Armadillo-Box WS1 が接続可能なネットワークと Linux から使用するネットワークデバイスの対応を次に示します。

表 6.1 ネットワークとネットワークデバイス

ネットワーク	ネットワークデバイス
有線 LAN	eth0

6.2.2. デフォルト状態のネットワーク設定

ネットワーク設定は、/etc/config/interfaces に記述されています。デフォルト状態では、次のように設定されています。

表 6.2 デフォルト状態のネットワーク設定

インターフェース	種類	設定	起動時に有効化
lo	TCP/IP	ループバック	有効
eth0	TCP/IP	DHCP	有効

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
```

図 6.1 デフォルト状態の/etc/config/interfaces

6.2.3. 有線 LAN

ここでは有線 LAN の使用方法について説明します。

6.2.3.1. 有線 LAN インターフェースの有効化、無効化

無効化されている有線 LAN インターフェースを有効化するには、次のようにコマンドを実行します。

```
[armadillo ~]# ifup eth0
```

図 6.2 ネットワークインターフェース(eth0)の有効化

有効化されている有線 LAN インターフェースを無効化するには、次のようにコマンドを実行します。

```
[armadillo ~]# ifdown eth0
```

図 6.3 ネットワークインターフェース(eth0)の無効化

6.2.3.2. 有線 LAN のネットワーク設定を変更する

有線 LAN のネットワーク設定を変更する方法について説明します。

ネットワーク接続に関する不明な点については、ネットワークの管理者へ相談してください。

Armadillo-Box WS1 上の「/etc/config」以下にあるファイルを編集し、コンフィグ領域に保存することにより起動時のネットワーク設定を変更することができます。コンフィグ領域の保存については、「7. コンフィグ領域 – 設定ファイルの保存領域」を参照してください。

設定を変更する場合は、かならずネットワークインターフェースを無効化してから行ってください。変更してからネットワークインターフェースを無効化しても、「新しい設定」を無効化することになります。「古い設定」が無効化されるわけではありません。

6.2.3.2.1. 有線 LAN を固定 IP アドレスに設定する

「表 6.3. 有線 LAN 固定 IP アドレス設定例」の内容に設定する例を、「図 6.4. 有線 LAN の固定 IP アドレス設定」に示します。

表 6.3 有線 LAN 固定 IP アドレス設定例

項目	設定
IP アドレス	192.0.2.10
ネットマスク	255.255.255.0
ネットワークアドレス	192.0.2.0
ブロードキャストアドレス	192.0.2.255
デフォルトゲートウェイ	192.0.2.1

```
[armadillo ~]# vi /etc/config/interfaces
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet static
address 192.0.2.10
netmask 255.255.255.0
network 192.0.2.0
broadcast 192.0.2.255
gateway 192.0.2.1
```

図 6.4 有線 LAN の固定 IP アドレス設定

6.2.3.2.2. 有線 LAN を DHCP に設定する

DHCP に設定する例を、「図 6.5. DHCP 設定」に示します。

DHCP に設定するには、vi エディタで/etc/config/interfaces を、次のように編集します。

```
[armadillo ~]# vi /etc/config/interfaces
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
```

図 6.5 DHCP 設定

6.2.3.3. 有線 LAN の接続を確認する

有線 LAN で正常に通信が可能か確認します。設定を変更した場合、かならず変更したインターフェースを再度有効化してください。

同じネットワーク内にある通信機器と PING 通信を行います。以下の例では、通信機器が「192.0.2.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping 192.0.2.20
```

図 6.6 有線 LAN の PING 確認

6.2.4. DNS サーバー

DNS サーバーを指定する場合は、vi エディタで/etc/config/resolv.conf を編集します。

```
[armadillo ~]# vi /etc/config/resolv.conf
nameserver 192.0.2.1
```

図 6.7 DNS サーバーの設定

DHCP を利用している場合には、DHCP サーバーが DNS サーバーを通知する場合があります。この場合、/etc/config/resolv.conf は自動的に更新されます。

6.2.5. ファイアーウォール

Armadillo では、簡易ファイアーウォールが動作しています。設定されている内容を参照するには、「図 6.8. iptables」のようにコマンド実行してください。

```
[armadillo ~]# iptables --list
```

図 6.8 iptables

6.3. ストレージ

Armadillo-Box WS1 でストレージとして使用可能なデバイスを次に示します。

表 6.4 ストレージデバイス

デバイス種類	ディスクデバイス	先頭パーティション	インターフェース
microSD/microSDHC/microSDXC カード	/dev/mmcblk0	/dev/mmcblk0p1	microSD インターフェース
USB フラッシュメモリ	/dev/sd*[a]	/dev/sd*1	USB インターフェース

[a]USB ハブを利用して複数の USB メモリを接続した場合は、認識された順に sda sdb sdc ... となります。

6.3.1. ストレージの使用方法

ここでは、microSDHC カードを例にストレージの使用方法を説明します。以降の説明では、共通の操作が可能な場合に、microSD/microSDHC/microSDXC カードを microSD カードと表記します。

microSDXC カードを使用する場合は、事前に「6.3.2. ストレージのパーティション変更とフォーマット」を参照してフォーマットを行う必要があります。これは、Linux カーネルが exFAT ファイルシステムを扱うことができないためです。通常、購入したばかりの microSDXC カードは exFAT ファイルシステムでフォーマットされています。

Linux では、アクセス可能なファイルやディレクトリは、一つの木構造にまとめられています。あるストレージデバイスのファイルシステムを、この木構造に追加することを、マウントするといいます。マウントを行うコマンドは、mount です。

mount コマンドの典型的なフォーマットは、次の通りです。

```
mount -t [fstype] device dir
```

図 6.9 mount コマンド書式

-t オプションに続く fstype には、ファイルシステムタイプを指定します[1]。FAT32 ファイルシステムの場合は vfat[2]、EXT3 ファイルシステムの場合は ext3 を指定します。

device には、ストレージデバイスのデバイスファイル名を指定します。microSD カードのパーティション 1 の場合は/dev/mmcblk0p1、パーティション 2 の場合は/dev/mmcblk0p2 となります。

dir には、ストレージデバイスのファイルシステムをマウントするディレクトリを指定します。

microSD スロットに microSDHC カードを挿入した状態で「図 6.10. ストレージのマウント」に示すコマンドを実行すると、/mnt ディレクトリに microSDHC カードのファイルシステムをマウントします。microSD カード内のファイルは、/mnt ディレクトリ以下に見えるようになります。

```
[armadillo ~]# mount -t vfat /dev/mmcblk0p1 /mnt
```

図 6.10 ストレージのマウント

FAT32 ファイルシステムをマウントした場合、次の警告メッセージが表示される場合があります。

```
FAT-fs (mmcblk0p1): utf8 is not a recommended IO charset for
FAT filesystems, filesystem will be case sensitive!
```

これは無視して構いません。 UTF-8 ロケールでは結局はファイル名の表示を正しく処理できないためです。

ストレージを安全に取り外すには、アンマウントする必要があります。アンマウントを行うコマンドは、umount です。オプションとして、アンマウントしたいデバイスがマウントされているディレクトリを指定します。

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

図 6.11 ストレージのアンマウント

6.3.2. ストレージのパーティション変更とフォーマット

通常、購入したばかりの microSDHC カードや USB メモリは、一つのパーティションを持ち、FAT32 ファイルシステムでフォーマットされています。

パーティション構成を変更したい場合、fdisk コマンドを使用します。fdisk コマンドの使用例として、一つのパーティションで構成されている microSD カードのパーティションを、2 つに分割する例を「図 6.12. fdisk コマンドによるパーティション変更」に示します。一度、既存のパーティションを削除してから、新たにプライマリパーティションを二つ作成しています。先頭のパーティションには 100MByte、二つめのパーティションに残りの容量を割り当てています。先頭のパーティションは/dev/

[1]ファイルシステムタイプの指定は省略可能です。省略した場合、mount コマンドはファイルシステムタイプを推測します。この推測は必ずしも適切なものとは限りませんので、事前にファイルシステムタイプが分かっている場合は明示的に指定してください。
[2]通常、購入したばかりの microSDHC カードは FAT32 ファイルシステムでフォーマットされています。

mmcblk0p1、二つめは/dev/mmcblk0p2 となります。fdisk コマンドの詳細な使い方は、man ページ等を参照してください。

```
[armadillo ~]# fdisk /dev/mmcblk0

The number of cylinders for this disk is set to 62528.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-62528, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-62528, default 62528): +100M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (3054-62528, default 3054):
Using default value 3054
Last cylinder or +size or +sizeM or +sizeK (3054-62528, default 62528):
Using default value 62528

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

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

図 6.12 fdisk コマンドによるパーティション変更

FAT32 ファイルシステムでストレージデバイスをフォーマットするには、mkfs.vfat コマンドを使用します。また、EXT2 や EXT3 ファイルシステムでフォーマットするには、mke2fs コマンドを使用します。microSD カードのパーティション 1 を EXT3 ファイルシステムでフォーマットするコマンド例を、次に示します。

```
[armadillo ~]# mke2fs -j /dev/mmcblk0p1
```

図 6.13 EXT3 ファイルシステムの構築

6.4. LED

Armadillo-Box WS1 の LED は、GPIO が接続されているためソフトウェアで制御することができます。

利用しているデバイスドライバは LED クラスとして実装されているため、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 6.5 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	名称	デフォルトトリガ
/sys/class/leds/red/	ユーザー LED(赤色)	default-on
/sys/class/leds/green/	ユーザー LED(緑色)	default-on
/sys/class/leds/yellow/	ユーザー LED(黄色)	none

以降の説明では、任意の LED を示す LED クラスディレクトリを"/sys/class/leds/[LED]"のように表記します。

6.4.1. LED を点灯/消灯する

LED クラスディレクトリ以下の brightness ファイルへ値を書き込むことによって、LED の点灯/消灯を行うことができます。brightness に書き込む有効な値は 0〜255 です。

brightness に 0 以外の値を書き込むと LED が点灯します。

```
[armadillo ~]# echo 1 > /sys/class/leds/[LED]/brightness
```

図 6.14 LED を点灯させる

Armadillo-Box WS1 の LED には輝度制御の機能が無いため、0 (消灯)、1〜255 (点灯)の 2 つの状態のみ指定することができます。

brightness に 0 を書き込むと LED が消灯します。

```
[armadillo ~]# echo 0 > /sys/class/leds/[LED]/brightness
```

図 6.15 LED を消灯させる

brightness を読み出すと LED の状態が取得できます。

```
[armadillo ~]# cat /sys/class/leds/[LED]/brightness
0
```

図 6.16 LED の状態を表示する

6.4.2. トリガを使用する

LED クラスディレクトリ以下の trigger ファイルへ値を書き込むことによって LED の点灯/消灯にトリガを設定することができます。trigger に書き込む有効な値を次に示します。

表 6.6 trigger の種類

設定	説明
none	トリガを設定しません。
mmc0	microSD インターフェースのアクセスランプにします。
timer	任意のタイミングで点灯/消灯を行います。この設定にすることにより、LED クラスディレクトリ以下に delay_on, delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します。
heartbeat	心拍のように点灯/消灯を行います。
default-on	主に Linux カーネルから使用します。LED が点灯します。

以下のコマンドを実行すると、LED が 2 秒点灯、1 秒消灯を繰り返します。

```
[armadillo ~]# echo timer > /sys/class/leds/[LED]/trigger
[armadillo ~]# echo 2000 > /sys/class/leds/[LED]/delay_on
[armadillo ~]# echo 1000 > /sys/class/leds/[LED]/delay_off
```

図 6.17 LED のトリガに timer を指定する

trigger を読み出すと LED のトリガが取得できます。"[ ]"が付いているものが現在のトリガです。

```
[armadillo ~]# cat /sys/class/leds/[LED]/trigger
none mmc0 mmc1 [timer] heartbeat default-on
```

図 6.18 LED のトリガを表示する

6.5. ユーザースイッチ

Armadillo-Box WS1 のユーザースイッチのデバイスドライバは、インプットデバイスとして実装されています。インプットデバイスのデバイスファイルからボタンプッシュ/リリースイベントを取得することができます。

ユーザースイッチのインプットデバイスファイルと、各スイッチに対応したイベントコードを次に示します。

表 6.7 インプットデバイスファイルとイベントコード

ユーザースイッチ	インプットデバイスファイル	イベントコード
SW1	/dev/input/event0	2 (1)

インプットデバイスは検出された順番にインデックスが割り振られます。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、デバイスファイルのインデックスが異なる可能性があります。

6.5.1. イベントを確認する

ユーザースイッチのボタンプッシュ/リリースイベントを確認するために、ここでは evtest コマンドを利用します。evtest を停止するには、Ctrl+c を入力してください。

```
[armadillo ~]# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 2 (1)
Testing ... (interrupt to exit)
Event: time 2100.899311, type 1 (Key), code 2 (1), value 1
Event: time 2100.899311, -------------- Report Sync ------------ ❶
Event: time 2101.053143, type 1 (Key), code 2 (1), value 0
Event: time 2101.053143, -------------- Report Sync ------------ ❷
                :
[armadillo ~]#
```

❶ SW2 のボタンプッシュイベントを検出したときの表示。

❷ SW2 のボタンリリースイベントを検出したときの表示。

図 6.19 ユーザースイッチ: イベントの確認

6.6. Wi-SUN モジュール

Armadillo-Box WS1 は Wi-SUN モジュールとして ROHM 製 BP35A1 を搭載しています。

Wi-SUN モジュールは、TTY デバイスファイルから ASCII コマンドを使用した制御を行うことができます。Wi-SUN モジュールにアクセスする際の TTY デバイスファイルは/dev/ttymxc2 です。

6.6.1. 設定情報を取得する

Wi-SUN モジュールを制御する例として、BP35A1 の設定情報の取得を行います。

Wi-SUN モジュールに搭載されている BP35A1 の設定情報を取得する手順を次に示します。

手順 6.1 設定情報の取得

1. tip コマンドを実行して/dev/ttymxc2 に接続します。ボーレートは 115200bps です。

```
[armadillo ~]$ tip -l /dev/ttymxc2 -s 115200
Connected.
```

2. SKINFO コマンドを実行すると、BP35A1 の設定情報が表示されます。

```
SKINFO
EINFO FE80:0000:0000:0000:021D:1290:0004:0FBE 001D129000040FBE 21 FFFF FFFE
OK
```

3. tip を終了するには、"~."(チルダ「~」に続いてドット「.」)を入力します。

```
Disconnected.
[armadillo ~]$
```

その他の ASCII コマンドや、BP35A1 の詳細な情報については ROHM 製ドキュメントを参照してください。

「ROHM Sub-GHz シリーズ」サポートページ ドキュメントダウンロード | 半導体のローム ROHM

http://micro.rohm.com/jp/download_support/wi-sun

7. コンフィグ領域 – 設定ファイルの保存領域

コンフィグ領域は、設定ファイルなどを保存しハードウェアのリセット後にもデータを保持することができるフラッシュメモリ領域です。コンフィグ領域からのデータの読出し、またはコンフィグ領域への書込みは、flatfsd コマンドを使用します。

7.1. コンフィグ領域の読出し

コンフィグ領域を読み出すには以下のコマンドを実行します。読み出されたファイルは、「/etc/config」ディレクトリに作成されます。

```
[armadillo ~]# flatfsd -r
```

図 7.1 コンフィグ領域の読出し方法

デフォルトのソフトウェアでは、起動時に自動的にコンフィグ領域の読出しを行うように設定されています。コンフィグ領域の情報が壊れている場合、「/etc/default」ディレクトリの内容が反映されます。

7.2. コンフィグ領域の保存

コンフィグ領域を保存するには以下のコマンドを実行します。保存されるファイルは、「/etc/config」ディレクトリ以下のファイルです。

```
[armadillo ~]# flatfsd -s
```

図 7.2 コンフィグ領域の保存方法

コンフィグ領域の保存をおこなわない場合、「/etc/config」ディレクトリ以下のファイルへの変更は電源遮断時に失われます。

7.3. コンフィグ領域の初期化

コンフィグ領域を初期化するには以下のコマンドを実行します。初期化時には、「/etc/default」ディレクトリ以下のファイルがコンフィグ領域に保存され、且つ「/etc/config」ディレクトリにファイルが複製されます。

```
[armadillo ~]# flatfsd -w
```

図 7.3 コンフィグ領域の初期化方法

8. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-Box WS1 の Linux カーネル仕様について説明します。

8.1. 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-Box WS1 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/armadillo-box-ws1_defconfig` です。

`armadillo-box-ws1_defconfig` で有効になっている主要な設定を「表 8.1. Linux カーネル主要設定」に示します。

表 8.1 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

8.2. デフォルト起動オプション

工場出荷状態の Armadillo-Box WS1 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 8.2 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。

8.3. Linux ドライバ一覧

Armadillo-Box WS1 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。

8.3.1. Armadillo-Box WS1

Armadillo-Box WS1 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-armadillo-box-ws1.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-Box WS1 Base board                <MACH_ARMADILLO_BOX_WS1>
                          :
```

8.3.2. フラッシュメモリ

Armadillo-Box WS1 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  --->                    <CONFIG_MTD>
        <*> Command line partition table parsing              <CONFIG_MTD_CMDLINE_PARTS>
        <*> Caching block device access to MTD devices               <CONFIG_MTD_BLOCK>
            RAM/ROM/Flash chip drivers  --->
            <*> Detect flash chips by Common Flash Interface (CFI) probe <CONFIG_MTD_CFI>
            <*> Support for Intel/Sharp flash chips           <CONFIG_MTD_CFI_INTELEXT>
            Mapping drivers for chip access  --->
            <*> Flash device in physical memory map                <CONFIG_MTD_PHYSMAP>
```

8.3.3. UART

Armadillo-Box WS1 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-Box WS1 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

高速なボーレート(1Mbps 以上など)を設定して大量のデータを受信した場合、TTY バッファ(サイズ: 64kByte)が枯渇してユーザーランドからデー

> タが取得できない場合があります。この場合、ユーザーランドで ZMODEM のように確認応答と再送制御に対応したプロトコルを利用するなどの対策を行う必要があります。

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
UART4	/dev/ttymxc3
UART5	/dev/ttymxc4

カーネルコンフィギュレーション

```
Device Drivers  --->
    Character devices  --->
        [*] Enable TTY                                                  <TTY>
            Serial drivers  --->
                <*> IMX serial port support                      <SERIAL_IMX>
                [*]   Console on IMX serial port                <SERIAL_IMX_CONSOL>
```

8.3.4. Ethernet

Armadillo-Box WS1 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>
```

8.3.5. SD ホスト

Armadillo-Box WS1 の SD ホストは、i.MX257 の eSDHC(Enhanced Secured Digital Host Controller)を利用しています。

Armadillo-Box WS1 では、eSDHC1 のみを利用することができます。

機能

カードタイプ: microSD/microSDHC/microSDXC
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なし
ライトプロテクトサポートなし

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/mx_sdhci.c
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

```
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>
```

8.3.6. USB ホスト

Armadillo-Box WS1 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-Box WS1 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>
```

8.3.7. LED

Armadillo-Box WS1 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。

Linux カーネルでは、実装された LED を色の名前を命名して区別しています。

LED クラスディレクトリと LED の対応については、「表 6.5. LED クラスディレクトリと LED の対応」を参照してください。

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>
```

8.3.8. ユーザースイッチ

Armadillo-Box WS1 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 8.3 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	2

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 8.4 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-core.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>
        -*-   Polled input device skeleton                                     <INPUT_POLLDEV>
              *** Userland interfaces ***
        <*>   Event interface                                                    <INPUT_EVDEV>
              *** Input Device Drivers ***
        [*]   Keyboards  --->                                                 <INPUT_KEYBOARD>
            <*>   GPIO Buttons                                                 <KEYBOARD_GPIO>
            <*>   Polled GPIO buttons                                       <KEYBOARD_GPIO_POLLED>
```

8.3.9. ウォッチドッグタイマー

Armadillo-Box WS1 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 60 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. ユーザーランド仕様

本章では、工場出荷状態の Armadillo-Box WS1 のユーザーランドの基本的な仕様について説明します。

9.1. ルートファイルシステム

Armadillo-Box WS1 の標準ルートファイルシステムは、Atmark Dist で作成された initrd です。PC などで動作する 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-Box WS1 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。

initrd はメモリ上に配置されるため、ファイルに加えた変更は再起動すると全て元に戻ってしまいます。例外として/etc/config/ディレクトリ以下のファイルは、flatfsd コマンドを利用してフラッシュメモリに保存することができます。このフラッシュメモリ領域をコンフィグ領域と呼びます。

コンフィグ領域を利用することで、設定ファイルなどへの変更を再起動後も保持することができるようになっています。コンフィグ領域のより詳細な情報については「7. コンフィグ領域 – 設定ファイルの保存領域」を参照してください。

9.2. 起動処理

Armadillo-Box WS1 のユーザーランドの起動処理について説明します。ユーザーランドの起動処理は大きく分けて次の手順で初期化が行われています。

1. Linux カーネルが/sbin/init を実行し/etc/inittab の sysinit に登録されている/etc/init.d/rc スクリプトを実行
2. rc スクリプトの中で、「/etc/rc.d/」ディレクトリの起動スクリプトを順次実行
3. ローカル起動スクリプト(/etc/config/rc.local)を実行
4. /etc/inittab の respawn タブに登録されたものを実行

9.2.1. inittab

Linux カーネルは、ルートファイルシステムをマウントすると、/sbin/init を実行します。init プロセスは、コンソールの初期化を行い/etc/inittab に記載された設定に従ってコマンドを実行します。

デフォルト状態の Armadillo-Box WS1 の/etc/inittab は次のように設定されています。

```
::sysinit:/etc/init.d/rc

::respawn:/sbin/getty -L 115200 ttymxc1 vt102
#::respawn:/sbin/getty 38400 tty1 linux

::shutdown:/etc/init.d/reboot
::ctrlaltdel:/sbin/reboot
```

図 9.1 デフォルト状態の/etc/inittab

inittab の書式は、次のようになっています。

```
id:runlevel:action:process
```

図 9.2 inittab の書式

Armadillo-Box WS1 の init では、"id"フィールドに起動されるプロセスが使用するコンソールを指定することができます。省略した場合は、システムコンソールが使用されます。"runlevel"フィールドは未対応のため利用できません。

"action"フィールド及び"process"フィールドは、どのような状態(action)のときに何(process)を実行するかを設定することができます。action フィールドに指定可能な値を「表 9.1. inittab の action フィールドに設定可能な値」に示します。

表 9.1 inittab の action フィールドに設定可能な値

値	process を実行するタイミング
sysinit	init プロセス起動時
respawn	sysinit 終了後。このアクションで起動されたプロセスが終了すると、再度 process を実行する
shutdown	シャットダウンする時
ctrlaltdel	Ctrl-Alt-Delete キーの組み合わせが入力された時

9.2.2. /etc/init.d/rc

rc スクリプトでは、システムの基礎となるファイルシステムをマウントしたり、「/etc/rc.d/」ディレクトリ以下にある S から始まるスクリプト(初期化スクリプト)が実行できる環境を構築します。その後、初期化スクリプトを実行していきます。初期化スクリプトは、S の後に続く 2 桁の番号の順番で実行します。

9.2.3. /etc/rc.d/S スクリプト(初期化スクリプト)

初期化スクリプトでは、システムの環境を構築するもの、デーモン(サーバー)を起動するものの 2 つの種類があります。Armadillo-Box WS1 のデフォルト状態で登録されている初期化スクリプトを「表 9.2. /etc/rc.d ディレクトリに登録された初期化スクリプト」に示します。

表 9.2 /etc/rc.d ディレクトリに登録された初期化スクリプト

スクリプト	初期化内容
S03udev	udevd を起動し、Linux カーネルから発行された uevent をハンドリングします
S04flatfsd	flatfsd を使いコンフィグ領域(/etc/config/)を復元します
S05checkroot	システム関連のファイルのパーミッション設定や、オーナーを設定します
S06mountdevsubfs	udevd 起動後にマウントする必要のあるファイルシステムをマウントします
S10syslogd, S20klogd	ログデーモンを起動します
S25module-init-tools	/etc/modules に記載されたカーネルモジュールをロードします
S30firewall	ファイヤーウォールの設定を行います
S30hostname	hostname を設定します
S40networking	ネットワーク関連の初期化を行い、インターネットスーパーサーバー(inetd)を起動します
S71avahi	ネットワークデーモンを起動します
S99rc.local	コンフィグ領域(/etc/config/)に保存された rc.local を実行します

9.2.4. /etc/config/rc.local

コンフィグ領域に保存された rc.local は、ユーザーランドイメージを変更することなく、起動時に特定の処理を行うことができるようになっています。

Armadillo-Box WS1 では、システム起動時に呼び出されはしますが、特に何もしていません。

デフォルト状態の/etc/config/rc.local は次のように記載されています。

```
#!/bin/sh

. /etc/init.d/functions

PATH=/bin:/sbin:/usr/bin:/usr/sbin

#
# Add your temporary commands to run at boot time
#
```

図 9.3 デフォルト状態の/etc/config/rc.local

9.3. プリインストールアプリケーション

デフォルトのユーザーランドにインストールされているアプリケーションを一覧します。

・ /bin

```
addgroup       e2fsck         ipaddr         mount          setarch
adduser        echo           ipcalc         mountpoint     setserial
ash            ed             iplink         mpstat         sh
base64         egrep          iproute        mt             sleep
busybox        ethtool        iprule         mv             stat
cat            evtest         iptunnel       netflash       stty
catv           false          java           nice           su
chattr         fdflush        keytool        ntpclient      sync
chgrp          fgrep          kill           pidof          tar
chmod          flatfsd        linux32        ping           tip
chown          fsck           linux64        pipe_progress  touch
conspy         fsck.ext2      ln             powertop       true
cp             fsync          login          printenv       tune2fs
cpio           getopt         ls             ps             umount
cttyhack       grep           lsattr         pwd            uname
date           gunzip         lzop           reformime      usleep
dd             gzip           mail           rev            vi
delgroup       hostname       makemime       rm             watch
deluser        hush           mkdir          rmdir          zcat
df             hwclock        mke2fs         rpm
dmesg          ionice         mknod          run-parts
dnsdomainname  iostat         mktemp         scriptreplay
dumpkmap       ip             more           sed
```

・ /usr/bin

```
[              dirname        ipcs           openvt         sha1sum        unexpand
[[             dos2unix       iptables-xml   passwd         sha256sum      uniq
add-shell      du             joe            patch          sha512sum      unix2dos
ar             dumpleases     kbd_mode       pgrep          showkey        unlzma
arping         eject          killall        pkill          smemcap        unlzop
awk            env            killall5       pmap           softlimit      unxz
```

```
basename      envdir        last          printf        sort          unzip
beep          envuidgid     less          pscan         split         uptime
bunzip2       ether-wake    logger        pstree        strings       users
bzcat         expand        logname       pwdx          sudo          uudecode
bzip2         expr          lpq           readahead     sudoedit      uuencode
cal           fdformat      lpr           readlink      sum           vi
chat          fgconsole     lsof          realpath      sv            vlock
chpst         find          lspci         remove-shell  tac           volname
chrt          flock         lsusb         renice        tail          wall
chvt          fold          lzcat         reset         tcpsvd        wc
cksum         free          lzma          resize        tee           which
clear         fuser         lzopcat       rpm2cpio      test          who
cmp           groups        md5sum        rtcwake       time          whoami
comm          hd            mesg          runsv         timeout       whois
crontab       head          microcom      runsvdir      top           xargs
cryptpw       hexdump       mkfifo        rx            tr            xz
curl          hostid        mkpasswd      script        traceroute    xzcat
cut           id            nmeter        seq           traceroute6   yes
dc            ifplugd       nohup         setkeycodes   tty
deallocvt     install       nslookup      setsid        ttysize
diff          ipcrm         od            setuidgid     udpsvd
```

- /sbin

```
acpid             flash_unlock      losetup           route
adjtimex          freeramdisk       lsmod             runlevel
arp               fsck.minix        makedevs          setconsole
avahi-daemon      fsck.msdos        man               slattach
blkid             fsck.vfat         mdev              start-stop-daemon
blockdev          getty             mkdosfs           sulogin
bootchartd        halt              mkfs.minix        swapoff
depmod            hdparm            mkfs.msdos        swapon
devmem            ifconfig          mkfs.vfat         switch_root
dosfsck           ifdown            mkswap            sysctl
fbsplash          ifenslave         modinfo           syslogd
fdisk             ifup              modprobe          tunctl
findfs            init              pivot_root        udevadm
flash_erase       insmod            poweroff          udevd
flash_eraseall    klogd             raidautorun       udhcpc
flash_info        loadkmap          reboot            vconfig
flash_lock        logread           rmmod             watchdog
```

- /usr/sbin

```
brctl                iptables             setlogcons
chpasswd             iptables-restore     svlogd
chroot               iptables-save        ubiattach
crond                loadfont             ubidetach
dhcprelay            lpd                  ubimkvol
fakeidentd           popmaildir           ubirmvol
get-board-info       rdate                ubirsvol
get-board-info-abws1 rdev                 ubiupdatevol
ip6tables            readprofile          udhcpd
ip6tables-restore    sendmail             visudo
ip6tables-save       setfont              xtables-multi
```

9.4. 有用なアプリケーションについて

デフォルトのユーザーランドにインストールされているアプリケーションの中から、いくつかをピックアップし概要を説明します。

表 9.3 アプリケーション概要説明

アプリケーション	概要
Java	オブジェクト指向プログラミング言語です。Armadillo-Box WS1 では Oracle Java が使用可能です。
cURL	ファイルを送信または受信するコマンドラインツールです。幅広いインターネットプロトコルをサポートします。Armadillo-Box WS1 では、curl コマンドにて実行が可能です。

10. ブートローダー仕様

本章では、ブートローダーの起動モードや利用することができる機能について説明します。

10.1. ブートローダー起動モード

ブートローダーが起動すると、ジャンパピン JP2 の状態により、2 つのモードのどちらかに遷移します。ジャンパピンの設定の詳細については、「4.5. ジャンパピンの設定について」を参照してください。

表 10.1 ブートローダー起動モード

起動モードの種別	JP2 状態	説明
保守モード	ショート	各種設定が可能な Hermit-At コマンドプロンプトが起動します。
オートブートモード	オープン	電源投入後、自動的に Linux カーネルを起動させます。

10.1.1. Linux でコンソールを使用する

Armadillo-Box WS1 を保守モードで起動し、boot コマンドを実行してください。

```
hermit> boot
```

図 10.1 boot コマンドで Linux を起動する

10.2. ブートローダーの機能

Hermit-At の保守モードでは、Linux カーネルの起動オプションの設定やフラッシュメモリの書き換えなどを行うことできます。

保守モードで利用できるコマンドを、「表 10.2. 保守モードコマンド一覧」に示します。

表 10.2 保守モードコマンド一覧

コマンド	説明
tftpdl erase program download	フラッシュメモリを書き換える場合に使用します
memmap	フラッシュメモリのメモリマップを表示します
setbootdevice setenv clearenv	OS の起動設定をする場合に使用します
boot tftpboot	OS を起動する場合に使用します
mac	MAC アドレスを表示します
frob	簡易的にメモリアクセスする場合に使用します
md5sum	メモリ空間の MD5 サム値を表示する場合に使用します
info	ハードウェアの情報を表示します
version	ブートローダーのバージョンを表示します

各コマンドのヘルプを表示するには「図 10.2. hermit コマンドのヘルプを表示」のようにします。

```
hermit> help [コマンド]
```

図 10.2 hermit コマンドのヘルプを表示

10.2.1. コンソールの指定方法

ブートローダーおよび Linux カーネルのコンソールを指定するには、後述する Linux カーネル起動オプションを設定する場合の setenv コマンドで行います。Linux カーネル起動オプションの console パラメータは、ブートローダーのコンソールにも影響する仕組みとなっています。

コンソール指定子とそれに対応するログ表示先/保守モードプロンプト出力先を「表 10.3. コンソール指定子とログ出力先」に示します。

表 10.3 コンソール指定子とログ出力先

コンソール指定子	ログ出力先[a]
ttymxc1	デバッグシリアルインターフェース
none	なし

[a]ブートローダーの再起動後に反映されます

10.2.2. Linux カーネルイメージの指定方法

ブートローダーが OS を起動させる場合、フラッシュメモリに書き込まれた Linux カーネルイメージか、microSD カードに保存されているイメージファイルを指定することができます。

Linux カーネルイメージを指定するには、"setbootdevice"コマンドを使用します。「表 10.4. Linux カーネルイメージ指定子」に示す指定子を設定することができます。

表 10.4 Linux カーネルイメージ指定子

指定子	Linux カーネルイメージの配置場所
flash	フラッシュメモリの kernel パーティションに書き込まれたイメージ
mmcblk0p1	microSD カードのパーティション 1 に保存されている/boot/linux.bin.gz ファイル "p1"はパーティションを示しており、"p2"とするとパーティション 2 のファイルを指定可能

10.2.3. Linux カーネル起動オプションの指定方法

Linux カーネルには様々な起動オプションがあります。詳しくは、Linux の解説書や、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/kernel-parameters.txt)を参照してください。

ここでは Armadillo-Box WS1 で使用することができる、代表的な起動オプションを「表 10.5. Linux カーネルの起動オプションの一例」に紹介します。

表 10.5 Linux カーネルの起動オプションの一例

オプション指定子	説明
console=	起動ログなどが出力されるイニシャルコンソールを指定します。 次の例では、コンソールに ttymxc1 を、ボーレートに 115200 を指定しています。 `console=ttymxc1,115200`

オプション指定子	説明

root=	ルートファイルシステムが構築されているデバイスを指定します。
	デバイスには Linux カーネルが認識した場合のデバイスを指定します。
	initrd をルートファイルシステムとする場合には、以下の例のように設定します。
	`root=/dev/ram0`
	SD カードにルートファイルシステムを配置する場合には、SD カードのデバイスファイルを指定します。次の例では、デバイスに microSD カードの第 2 パーティションを指定しています。
	`root=/dev/mmcblk0p2`

rootwait	"root="で指定したデバイスが利用可能になるまでルートファイルシステムのマウントを遅らせます。
noinitrd	initrd を利用しないことを明示します。
mem	Linux カーネルが利用可能なメモリの量を指定します。RAM の一部を専用メモリとして利用したい場合などに設定します。

11. 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-Box WS1 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-box-ws1/downloads

工場出荷イメージの作成に必要な Oracle Java SE Embedded は、Oracle 社 Web ページ(http://www.oracle.com/)から取得してください。Armadillo-Box WS1 では、「Oracle Java SE Embedded version 8」の「ARMv5 Linux - SoftFP ABI, Little Endian」を使用します。

Java SE Embedded - Downloads

http://www.oracle.com/technetwork/java/embedded/embedded-se/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。

11.1. 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」のソースコードと、「Oracle Java SE Embedded 8」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 11.1 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と、Java SE Embedded のアーカイブを展開します。

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

```
atmark-dist-[version]          ejdk[version]          linux-3.14-at[version]
atmark-dist-[version].tar.gz   ejdk-[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 カーネルおよび Java SE Embedded 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
[ATDE ~/atmark-dist-[version]]$ ln -s ../ejdk[version] ejdk
```

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. **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 v1.42.0 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 >
```

4. **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 "Armadillo-Box-WS1" を選択します。

```
 atmark-dist v1.42.0 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-Box-WS1) AtmarkTechno Products


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

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

5. **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 v1.42.0 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 >
```

6. **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。

7. **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 v1.42.0 Configuration
------------------------------------------------------------------------------


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

                                < Yes >      <  No  >

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

8. **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。

9. **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-Box WS1 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
```

11.2. 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 11.2 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.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-Box WS1 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 armadillo_box_ws1_defconfig を指定します。UART 起動用イメージを作成する場合は、armadillo-box-ws1_boot_defconfig を指定してください。

```
[ATDE ~]$ cd hermit-at
[ATDE ~/hermit-at]$ make armadillo_box_ws1_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/armadillo-box-ws1/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/armadillo-box-ws1/loader-armadillo-box-ws1-*.bin
src/target/armadillo-box-ws1/loader-armadillo-box-ws1-[version].bin
```

12. フラッシュメモリの書き換え方法

本章では、Armadillo-Box WS1 のフラッシュメモリに書き込まれているイメージファイルを更新す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、大きく分けて以下の 3 種類の方法があります。

表 12.1 フラッシュメモリの書き換え方法

方法	特徴
netflash を使用する	・ イメージファイルをネットワークまたはストレージで転送するため書き換えが高速 ・ Armadillo で Linux にログインできる必要がある
ダウンローダーを使用する	・ イメージファイルをシリアルで転送するため書き換えが低速 ・ Armadillo でブートローダーが起動できればよい
TFTP を使用する	・ イメージファイルをネットワークで転送するため書き換えが高速 ・ Armadillo でブートローダーが起動できればよい

フラッシュメモリを書き換えるためには、Linux またはブートローダーが起動している必要があります。フラッシュメモリに書き込まれているブートローダーが起動しない状態になってしまった場合は、「12.5. ブートローダーが起動しなくなった場合の復旧作業」を参照してブートローダーを復旧してください。

ダウンローダーを使用してユーザーランドイメージなどサイズの大きなイメージファイルを書き換えると非常に時間がかかります。これは、イメージファイルを Armadillo に転送する際にシリアルの転送速度がボトルネックとなるためです。サイズの大きなイメージファイルを書き換える場合は netflash または TFTP を使用する方法を推奨します。

12.1. フラッシュメモリのパーティションについて

フラッシュメモリの書き換えは、パーティション毎に行います。パーティションは"リージョン"とも呼ばれます。

各パーティションのサイズはフラッシュメモリ内には保存されていません。ブートローダーと Linux カーネルそれぞれが同じパーティションテーブルを保持することにより、一意的に扱うことができるようになっています。

各パーティションは、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。

各パーティションのデフォルト状態での書き込み制限の有無と、対応するイメージファイル名を「表 12.2. パーティシ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イメージファイル名」に示します。

表 12.2 パーティシ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イメージファイル名

パーティション	書き込み制限	イメージファイル名	備考
bootloader	あり	loader-armadillo-box-ws1-*[version]*.bin	ブートローダーイメージを配置するパーティションです。
kernel	なし	linux-abws1-*[version]*.bin.gz	Linux カーネルイメージを配置するパーティションです。
userland	なし	romfs-abws1-*[version]*.img.gz	ユーザーランドイメージを配置するパーティションです。
config	なし	なし	ユーザーランドアプリケーション"flatfsd"が Flat file-system(フラッシュメモリ向けファイルシステム)を構築するパーティションです。使用方法については「7. コンフィグ領域 – 設定ファイルの保存領域」を参照してください。

工場出荷状態でフラッシュメモリに書き込まれているイメージファイルは、最新版ではない可能性があります。最新版のブートローダー、Linux カーネルイメージファイルは Armadillo サイトから、ユーザーランドイメージファイルはユーザーズサイトからダウンロード可能です。最新版のイメージファイルに書き換えてからのご使用を推奨します。

ダウンローダーでは、書き込みが制限されているパーティションを"ロック(locked)されている"と呼びます。このパーティションを強制的に書き換える場合は、"--force-locked"というオプションを付けます。他のオプションについては、「12.3. ダウンローダーを使用してフラッシュメモリを書き換える」を参照してください。

Linux が動いている場合は、書き込みが制限されているパーティションを書き換えることはできません。そのため、bootloader パーティションを netflash で書き換えることはできません。

12.2. 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 を利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。ここでは、netflash を利用して次に示す場所に存在するイメージファイルをフラッシュメモリに書き込む手順を紹介します。

- Web サーバー上のイメージファイル
- ストレージ上のイメージファイル

netflash コマンドのヘルプは次の通りです。

```
[armadillo ~]# netflash -h
usage: netflash [-bCfFhijklntuv?] [-c console-device] [-d delay] [-o offset] [-r flash-device]
                [net-server] file-name

    -b      don't reboot hardware when done
    -C      check that image was written correctly
    -f      use FTP as load protocol
    -F      force overwrite (do not preserve special regions)
    -h      print help
    -i      ignore any version information
    -H      ignore hardware type information
    -j      image is a JFFS2 filesystem
    -k      don't kill other processes (or delays kill until
        after downloading when root filesystem is inside flash)
    -K      only kill unnecessary processes (or delays kill until
        after downloading when root filesystem is inside flash)
    -l      lock flash segments when done
    -n      file with no checksum at end (implies no version information)
    -p      preserve portions of flash segments not actually written.
    -s      stop erasing/programming at end of input data
    -t      check the image and then throw it away
    -u      unlock flash segments before programming
    -v      display version number
```

図 12.1 netflash コマンドのヘルプ

"-r"オプションに指定するフラッシュメモリのデバイスファイルとパーティションの対応を次に示します。

表 12.3 フラッシュメモリのパーティションとデバイスファイル

パーティション	デバイスファイル
kernel	/dev/flash/kernel
userland	/dev/flash/userland
config	/dev/flash/config

bootloader パーティションは書き込みが制限されているため、netflash で書き換えることはできません。

12.2.1. Web サーバー上のイメージファイルを書き込む

ATDE では、標準で Web サーバー(lighttpd)が動作しており、/var/www/ディレクトリ以下に置かれたファイルはネットワーク経由でダウンロードすることができます。netflash は、HTTP によるファイルのダウンロードをサポートしています。

ここでは、ATDE とネットワーク通信ができることを前提に、ATDE からイメージファイルをダウンロードして kernel パーティションに書き込む手順を説明します。

手順 12.1 Web サーバー上のイメージファイルを書き込む

1. ATDE の/var/www/ディレクトリに Linux カーネルイメージファイルを置きます。

```
[ATDE ~]$ ls
linux-abws1-[version].bin.gz
[ATDE ~]$ cp linux-abws1-[version].bin.gz /var/www/
```

2. Web サーバー上のイメージファイルの URL(http://[ATDE の IP アドレス]/linux-abws1-*[version]*.bin.gz)を指定して netflash コマンドを実行します。次の例では、ATDE の IP アドレスが「192.0.2.1」であることを想定しています。

```
[armadillo ~]# netflash -b -k -n -u -s -r /dev/flash/kernel http://192.0.2.1/linux-abws1-[version].bin.gz
.....................................................................
.....................................................................
(省略)
.....................................
netflash: got "http://192.0.2.1/linux-abws1-[version].bin.gz", length=3388178
netflash: programming FLASH device /dev/flash/kernel
.................
```

3. Armadillo のプロンプトが表示されるとフラッシュメモリの書き換えは完了です。次回起動時から書き換えた Linux カーネルイメージで起動します。

```
[armadillo ~]#
```

12.2.2. ストレージ上のイメージファイルを書き込む

ストレージ(microSD カードや USB メモリ)をマウントすることで、ストレージに保存されたイメージファイルをフラッシュメモリに書き込むことができます。

ここでは microSD カードに保存されているイメージファイルを userland パーティションに書き込む手順を説明します。

手順 12.2 microSD カード上のイメージファイルを書き込む

1. microSD カードを/mnt/ディレクトリにリードオンリーでマウントします。

```
[armadillo ~]# mount -o ro /dev/mmcblk0p1 /mnt
kjournald starting.  Commit interval 5 seconds
EXT3-fs (mmcblk0p1): using internal journal
EXT3-fs (mmcblk0p1): mounted filesystem with ordered data mode
[armadillo ~]# ls /mnt
romfs-abws1-[version].img.gz
```

2. microSD カード上のイメージファイルのパス(/mnt/romfs-abws1-*[version]*.img.gz)を指定して netflash コマンドを実行します。

```
[armadillo ~]# netflash -b -k -n -u -s -r /dev/flash/userland /mnt/romfs-abws1-[version].img.gz
.....................................................................
.....................................................................
```

```
（省略）
.....................................................
netflash: got "/mnt/romfs-abws1-[version].img.gz", length=14176995
netflash: programming FLASH device /dev/flash/userland
.....................................................................
```

3. Armadillo のプロンプトが表示されるとフラッシュメモリの書き換えは完了です。次回起動時から書き換えたユーザーランドイメージで起動します。

```
[armadillo ~]#
```

4. microSD カードをアンマウントします。

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

12.3. 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

Linux を起動できない場合やブートローダーを更新する場合は、ダウンローダー(hermit)を使用してフラッシュメモリを書き換える必要があります。hermit は ATDE に標準でインストールされています。

hermit は Armadillo のブートローダーと協調動作を行いフラッシュメモリを書き換えることができます。hermit とブートローダー間の通信には、シリアル[1]が使用されます。

hermit のヘルプは次の通りです。

[1]通信速度(ボーレート)は、115200bps です

```
[ATDE ~]# hermit
Usage: hermit [options] command [command options]
Available commands: download, erase, help, go, map, terminal, upload, md5sum
Armadillo-J command: firmupdate
Multiple commands may be given.
General options (defaults) [environment]:
    -e, --ethernet
    -i, --input-file <path>
    --netif <ifname> (eth0) [HERMIT_NETIF]
    --memory-map <path>
    --port <dev> (/dev/ttyS0) [HERMIT_PORT]
    -o, --output-file <path>
    --remote-mac <MAC address>
    -v, --verbose
    -V, --version
Download/Erase options:
    -a, --address <addr>
    -b, --baudrate <baudrate>
    --force-locked
    -r, --region <region name>
Memory map options:
    --anonymous-regions
Md5sum  options:
    -a, --address <addr>
    -r, --region <region name>
    -s, --size <size>
```

図 12.2 hermit コマンドのヘルプ

ここでは、bootloader パーティションを書き換える手順について説明します。

手順 12.3 ダウンローダーを使用して書き換える

1. ブートローダーが保守モードで起動するように設定します。設定方法については、「10.1. ブートローダー起動モード」を参照してください。

2. Armadillo が保守モードで起動したことを確認するために、ATDE で minicom を起動しておきます。デバイスファイル名(/dev/ttyUSB0)は、ご使用の環境により ttyUSB1 や ttyS0、ttyS1 などになる場合があります。Armadillo に接続されているシリアルポートのデバイスファイルを指定してください。

   ```
   [ATDE ~]$ LANG=C minicom --noinit --wrap --device /dev/ttyUSB0
   ```

3. Armadillo に電源を投入します。ブートローダーが保守モードで起動すると、次のように保守モードのプロンプトが表示されます。

   ```
   hermit>
   ```

4. minicom を終了させシリアルポート(/dev/ttyUSB0)を開放します。

5. bootloader パーティションと書き込むイメージファイル(loader-armadillo-box-ws1-[version].bin)を指定して hermit コマンドを実行します。bootloader パーティションを更新する場合は、必ず"--force-locked"オプションを指定する必要があります。

```
[ATDE ~]$ hermit download --input-file loader-armadillo-box-ws1-[version].bin --region
bootloader --force-locked --port /dev/ttyUSB0
serial: completed 0x0000a92c (43308) bytes.
```

書き込みが制限されているパーティションを書き換える場合、"--force-locked"オプションを指定する必要があります。

6. ATDE のプロンプトが表示されるとフラッシュメモリの書き換えは完了です。次回起動時から書き換えたブートローダーイメージで起動します。

```
[ATDE ~]$
```

12.4. TFTP を使用してフラッシュメモリを書き換える

Hermit-At ブートローダーの tftpdl 機能を使用することで、Linux が動いていない時でも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 では、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 12.3. tftpdl コマンド例」は、Armadillo の IP アドレスを 192.0.2.10 に設定し、IP アドレスが 192.0.2.1 の TFTP サーバー上にある、romfs.img.gz を userland パーティションにを書き込む例です。

```
hermit> tftpdl 192.0.2.10 192.0.2.1 --blksize=1024 --userland=romfs.img.gz
```

図 12.3 tftpdl コマンド例

書き込み対象となるパーティションを指定するオプションと、パーティションの対応を次に示します。

表 12.4 パーティションとオプションの対応

パーティション	オプション
bootloader	--bootloader
kernel	--kernel

パーティション	オプション
userland	--userland
config	--config

tftpdl は、TFTP プロトコルを使用して TFTP サーバーからイメージファイルをダウンロードします。デフォルトのデータブロックサイズが 512Byte であるため、イメージファイルの最大サイズがブロック番号の桁溢れが発生しない 33554431Byte(32MByte - 1Byte)に制限されます。これよりもサイズの大きいイメージファイルをダウンロードする場合は、"--blksize"オプションを利用してデータブロックサイズを増やす必要があります。

"--blksize"オプションには、IP フラグメンテーションが起きないデータブロックサイズを指定する必要があります。

12.5. ブートローダーが起動しなくなった場合の復旧作業

フラッシュメモリの bootloader パーティションを誤ったイメージファイルで書き換えたり、書き換え中に Armadillo の電源を切断してしまった場合、ブートローダーが起動しなくなる場合があります。フラッシュメモリのブートローダーが起動しなくなった場合は、プロセッサ(i.MX257)の UART ブート機能を利用して復旧する必要があります。

ブートローダーの復旧手順を次に示します。

手順 12.4 ブートローダーの復旧

1. Armadillo の電源が切断されていることを確認します。
2. Armadillo のジャンパを、「表 4.6. 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。
3. ATDE で shoehorn コマンドを実行します。デバイスファイル名(/dev/ttyUSB0)は、ご使用の環境により ttyUSB1 や ttyS0、ttyS1 などになる場合があります。Armadillo に接続されているシリアルポートのデバイスファイルを指定してください。

```
[ATDE ~]$ shoehorn --boot --target armadillo4x0 \
--initrd /dev/null \
--kernel /usr/lib/hermit-3/loader-armadillo-box-ws1-boot-[version].bin \
--loader /usr/lib/shoehorn/shoehorn-armadillo4x0.bin --initfile \
/usr/lib/shoehorn/shoehorn-armadillo4x0.init --postfile \
/usr/lib/shoehorn/shoehorn-armadillo4x0.post --port /dev/ttyUSB0
/usr/lib/shoehorn/shoehorn-armadillo4x0.bin: 1300 bytes (2048 bytes buffer)
/usr/lib/hermit-3.3/loader-armadillo-box-ws1-boot-[version].bin: 50224
bytes (50224 bytes buffer)
/dev/null: 0 bytes (0 bytes buffer)
Waiting for target - press Wakeup now.
```

4. Armadillo に電源を投入します。

```
Initializing target...
Writing SRAM loader...
```

```
Pinging loader
Initialising hardware:
- flushing cache/TLB
- Switching to 115200 baud
- Get board IDs
- Initializing for Mobile-DDR
Pinging loader
Detecting DRAM
- 16 bits wide
- start: 0x80000000 size: 0x08000000 last: 0x87ffffff
Total DRAM: 131072kB
Loading /usr/lib/hermit-3/loader-armadillo-box-ws1-boot-[version].bin:
- start: 0x80800000 size: 0x0000c430 last: 0x8080c42f
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 0x80800000
[ATDE ~]$
```

5. shoehorn コマンドが成功すると、Armadillo の RAM 上で Hermit-At ブートローダーが動作している状態になります。Armadillo の電源を切断せずに、hermit コマンドでフラッシュメモリの bootloader パーティションにブートローダーイメージを書き込みます。

```
[ATDE ~]$ hermit erase --region bootloader download --input-file loader-armadillo-box-
ws1-[version].bin --region bootloader --force-locked --port /dev/ttyUSB0
erasing region bootloader
serial: completed 0x0000c584 (50564) bytes.
```

6. ATDE のプロンプトが表示されるとフラッシュメモリの書き換えは完了です。次回起動時から書き換えたブートローダーイメージで起動します。

```
[ATDE ~]$
```

13. 開発の基本的な流れ

本章では、Armadillo-Box WS1 を用いたシステム開発の一連の流れについて説明します。

1. ユーザーオリジナルアプリケーションを作成する
2. Atmark Dist にユーザーオリジナルアプリケーションを組み込む
3. システムの最適化を行う
4. オリジナルプロダクトのコンフィギュレーションを更新する

以降では、上記ステップについて順を追って説明します。

13.1. ユーザーオリジナルアプリケーションを作成する

ここでは、システムのメイン機能となるアプリケーションプログラムを作成する方法を説明します。ほとんどのシステムでは、ユーザーオリジナルなアプリケーションを実装するものと思います。本章では定番である「Hello world!」を例に、C 言語でアプリケーションプログラムのソースコードを作成し、コンパイル、動作確認する方法について説明します。

まずは、ATDE 上で動作する「Hello World!」を作成してみましょう。テキストエディタ[1]には gedit を利用します。

```
[ATDE ~]$ mkdir hello
[ATDE ~]$ cd hello
[ATDE ~/hello]$ gedit main.c &
```

図 13.1 ディレクトリを作成後、テキストエディタ(gedit)を起動

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("Hello World!\n");

    return EXIT_SUCCESS;
}
```

図 13.2 「Hello World!」のソース例(main.c)

作成したソースコードが意図した通りに動作するか、ATDE 上で動作するようにコンパイルして実行し、動作の確認をしましょう。

[1] ATDE には、gedit、emacs や vi などのテキストエディタがあらかじめインストールされています。

```
[ATDE ~/hello]$ gcc main.c -o hello ❶
[ATDE ~/hello]$ ls
hello  main.c
[ATDE ~/hello]$ ./hello ❷
Hello World!
```

❶ ATDE 上で動作するようにコンパイルするには「gcc」コマンドを使用します。

❷ コンパイルされた実行ファイル(hello)を実行

図 13.3 ATDE 上で動作するように main.c をコンパイルし実行

意図した通りに実行できましたね。では次に Armadillo が実行できるようにコンパイルを行います。Armadillo のアプリケーションを作成するには、クロスコンパイルが基本的な手法となります。先に示している、ブートローダー、Linux カーネル、ユーザランドイメージもクロスコンパイルされています。

クロスコンパイルとは、別のアーキテクチャで動作する実行ファイルを作成することです。ATDE など、通常の PC は、i386 または amd64 と言われるアーキテクチャとなっています。Armadillo-Box WS1 では armel というアーキテクチャが使われています。Armadillo-Box WS1 で実行することができる実行ファイルを ATDE 上で作成する方法を説明します。

Armadillo-Box WS1 上で動作するようにコンパイルする場合は、コンパイラ(gcc)に armel アーキテクチャ用のもの(arm-linux-gnueabi-gcc)を利用します。

```
[ATDE ~/hello]$ arm-linux-gnueabi-gcc main.c -o hello
[ATDE ~/hello]$ ls
hello  main.c
```

図 13.4 Armadillo-Box WS1 上で動作するように main.c をクロスコンパイル

Armadillo-Box WS1 に実行ファイルをコピーして動作の確認を行います。ここでは ATDE で動作している HTTP サーバーにファイルを一旦アップロードし、Armadillo-Box WS1 にそのファイルをダウンロードさせています。また、ATDE の IP アドレスが「192.0.2.11」であることを想定しています。

```
[ATDE ~/hello]$ cp hello /var/www/
```

図 13.5 HTTP サーバーに hello をアップロード

```
[armadillo ~]# curl -O http://192.0.2.11/hello
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5156  100  5156    0     0  22079      0 --:--:-- --:--:-- --:--:-- 22614
```

図 13.6 ATDE から hello をダウンロード

ダウンロードしたばかりのファイルには実行権限がついていないため、chmod コマンドで実行権限を付与して実行してみましょう。

```
[armadillo ~]# ls
hello
[armadillo ~]# chmod +x hello
[armadillo ~]# ./hello
Hello World!
```

図 13.7 Armadillo-Box WS1 上で hello を実行

13.2. Atmark Dist にユーザーオリジナルアプリケーションを組み込む

「13.1. ユーザーオリジナルアプリケーションを作成する」では、Armadillo-Box WS1 上で動作することができる実行ファイルを作成することができました。続いて、Atmark Dist にそのアプリケーションを組み込み、ユーザーランドのイメージファイル(romfs.img.gz)に自動的にインストールされるように作業を行います。

はじめに hello アプリケーションをビルドするための Makefile を作成します。この Makefile は、Atmark Dist のビルドシステムに hello を組み込むために必要となります。テキストエディタで作成します。

```
TARGET = hello

CROSS_COMPILE ?= arm-linux-gnueabi-
CC = $(CROSS_COMPILE)gcc
CFLAGS = -Wall -Wextra -O3

all: $(TARGET)

hello: main.o
    $(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@

%.o: %.c
    $(CC) $(CFLAGS) -c -o $@ $<

clean:
    $(RM) *~ *.o hello
```

図 13.8 hello 用の Makefile

Makefile が正しく作成できたかを確認するために、一度ビルドしてみましょう。ビルドには make コマンドを利用します。

```
[ATDE ~/hello]$ make
arm-linux-gnueabi-gcc -Wall -Wextra -O3 -c -o main.o main.c
arm-linux-gnueabi-gcc  main.o  -o hello
[ATDE ~/hello]$ ls
Makefile  hello  main.c  main.o
```

図 13.9 hello を make

makefile の記述ルールは次のようになります。

```
ターゲット: 依存ファイル 1 依存ファイル 2
        コマンド 1
        コマンド 2
```

make コマンドに続けて入力することによりターゲットを指定することができます。ターゲットを指定しない場合は、makefile のルールで最初に記述されているターゲットが実行されます。

「図 13.8. hello 用の Makefile」では、ターゲット指定をしない場合は、"all"ターゲットが実行されます。clean ターゲットを指定し make すると、一時ファイルなどが消去されます。

```
[ATDE ~/hello]$ make clean
rm -f *~ *.o hello
```

図 13.10 clean ターゲット指定した例

Atmark Dist では、製品(システム)固有の設定やファイルなどを製品毎にディレクトリに分けて管理されています。このディレクトリをプロダクトディレクトリといいます。アットマークテクノ製品の場合、開発セット用の標準イメージに対応するプロダクトディレクトリが製品毎に用意されています。

ここでは、Armadillo-Box WS1 のプロダクトディレクトリをコピーしてオリジナルプロダクトを作成し、そのオリジナルプロダクトに hello を組み込みます。オリジナルプロダクトの名前は、"my-product"とします。なお、「~/atmark-dist」を配置していない場合は、「11.1. Linux カーネル/ユーザーランドをビルドする」を参照して配置してください。

```
[ATDE ~/hello]$ cd ~/atmark-dist/
[ATDE ~/atmark-dist]$ cp -r vendors/AtmarkTechno/Armadillo-Box-WS1/ vendors/AtmarkTechno/my-product
[ATDE ~/atmark-dist]$ cp -r ../hello/ vendors/AtmarkTechno/my-product/
```

図 13.11 オリジナルプロダクトを作成し hello ディレクトリをコピー

続いて、hello を Atmark Dist のビルドシステムに組み込みます。プロダクトディレクトリ(atmark-dist/vendors/AtmarkTechno/my-product/)にある Makefile をテキストエディタで開き、次のように 34 行目を追加します。

```
29 comma := ,
30 empty :=
31 space := $(empty) $(empty)
32
33 SUBDIR_y =
34 SUBDIR_y += hello/
35
```

図 13.12 オリジナルプロダクト(my-product)に hello を登録

「図 13.8. hello 用の Makefile」では、romfs ディレクトリ(atmark-dist/romfs/)にファイルをインストールするための romfs ターゲットに対応していないため、ビルドされた実行ファイルは作成されますが、ユーザーランドイメージに実行ファイルがインストールされることはありません。ユーザーランドイメージに自動的にインストールされるように、romfs ターゲットを追加しましょう。ここでは、Armadillo 上の/usr/bin/ディレクトリ以下に hello がインストールされるように記述してみます。(18-19 行目を追加)

```
12 %.o: %.c
13      $(CC) $(CFLAGS) -c -o $@ $<
14
15 clean:
16      $(RM) *~ *.o hello
17
18 romfs: hello
19      $(ROMFSINST) /usr/bin/hello
```

図 13.13 romfs ターゲットの追加

これで、my-product に hello が追加されました。my-product をビルドして、イメージファイルを書き換えてみましょう。「11.1. Linux カーネル/ユーザーランドをビルドする」の手順の中で、AtmarkTechno Products に"Armadillo-Box-WS1"を選択している箇所では"my-product"を選択します。ビルドして出来上がったユーザーランド(romfs.img.gz)をフラッシュメモリに書き込むには、「12. フラッシュメモリの書き換え方法」を参照してください。

フラッシュメモリを書き換えた後 Armadillo を再起動すると、/usr/bin/hello が組み込まれたユーザーランドとなっています。

```
[armadillo ~]# ls /usr/bin/hello
/usr/bin/hello
[armadillo ~]# hello
Hello World!
```

図 13.14 hello が組み込まれたユーザーランドイメージ

13.3. システムの最適化を行う

ここでは、システム開発の最終段階の最適化について説明します。

ベースとした Armadillo-Box WS1 では、システムに不要なアプリケーションなどが含まれていると思います。不要なアプリケーションを省くことでイメージファイルがスリムになり起動速度が向上したり、空きメモリ容量が増えるなどのシステムの負荷が軽減します。

また、セキュリティーについても考慮すべきでしょう。Armadillo のデフォルトの root パスワードは、「root」となっています。デフォルトのままにしてしまうと簡単にハッキングされてしまう恐れがあります。

必要のないアプリケーションを削除したり、パスワードの変更を行うには、make menuconfig などを行いシステムを変更します。

手順 13.1 必要のないアプリケーションを削除する

1. make menuconfig を行い「Kernel/Library/Defaults Selection --->」を選択します。

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

```
 atmark-dist v1.42.0 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 >
```

2. 「Customize Vendor/User Settings」を選択して"Exit"を 2 回して「Do you wish to save your new kernel configuration?」で"Yes"とします。

```
 atmark-dist v1.42.0 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 >
```

3. Userland Configuration メニューが表示されます。

```
 atmark-dist v1.42.0 Configuration
--------------------------------------------------------------------------
                       Userland Confgiguration
  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  --->
  ------------------------------------------------------------------

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

4. ここでは、例として「java」を削除してみます。「Miscellaneous Applications --->」を選択しメニューをスクロールすると java の項目があります。

```
 atmark-dist v1.42.0 Configuration
--------------------------------------------------------------------------
                     Miscellaneous Applications
  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
  ------------------------------------------------------------------
              [*] java
              [*]   Oracle Java SE Embedded 8
              (compact1)     profile
              (minimal)     vm
              ---     extension
              [ ]     sunec
              [ ]     sunpkcs11
              [ ]     locales
              [ ]     charsets
              [ ]     nashorn
  ------------------------------------------------------------------

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

5. 「java」にカーソルを合わせて"N"を押下し選択を解除してください。そして、"Exit"を 2 回選択して「Do you wish to save your new kernel configuration?」で"Yes"とすることで設定を保存することができます。

```
  ------------------------------------------------------------------
              [ ] java
```

手順 13.2 root パスワードを変更する

1. 「手順 13.1. 必要のないアプリケーションを削除する」と同様に、make menuconfig を使い「Userland Configuration」メニューを開きます。

2. 「Vendor specific --->」を選択します。

```
 atmark-dist v1.42.0 Configuration
------------------------------------------------------------------------------
                             Vendor specific
  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
  ---------------------------------------------------------------------
               [ ] change root password
               (Auto) generate file-system option
               --- Applications
               --- Kernel modules


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

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

3. 「change root passwd」を選択すると、root パスワードを変更することができます。

```
  ---------------------------------------------------------------------
               [*] change root password
                    root password: "root"
               (Auto) generate file-system option
               --- Applications
               --- Kernel modules

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

13.4. オリジナルプロダクトのコンフィギュレーションを更新する

make menuconfig で修正を加えたコンフィギュレーションは、一時ファイルとして保存されています。一時ファイルは make clean や make distclean などで Atmark Dist をクリーンアップした場合に削除されてしまいます。再度コンフィギュレーションを復元するためには、一からコンフィギュレーション手順を再現しなくてはなりません。

Atmark Dist をクリーンアップした場合でも、設定したコンフィギュレーションを恒久的に復元させることができるように、プロダクトのデフォルトコンフィギュレーションを上書き更新する手順を説明します。

手順 13.3 プロダクトのデフォルトコンフィギュレーションを上書き更新する

1. 「手順 13.1. 必要のないアプリケーションを削除する」と同様に、make menuconfig を使い「Kernel/Library/Defaults Selection」メニューを開きます。

2. 「Update Default Vendor Settings」を選択しておきます。「Customize Vendor/User Settings」でコンフィギュレーションを変更した場合などに、自動的にプロダクトのデフォルトコンフィギュレーションが上書き更新されるようになります。

```
 atmark-dist v1.42.0 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 >
```

「Update Default Vendor Settings」を選択した場合に更新されるデフォルトコンフィグファイルを「表 13.1. デフォルトコンフィグファイル」に示します。

表 13.1 デフォルトコンフィグファイル

対象	デフォルトコンフィギュレーションファイル
Linux カーネル	[プロダクトディレクトリ]/config.linux-3.x[a]
Userland	[プロダクトディレクトリ]/config.vendor
Busybox-1.20.2	[プロダクトディレクトリ]/config.busybox-1.20.2

[a] ファイルが存在しない場合は、Linux カーネルのデフォルトコンフィグが使用されます

Linux カーネルのデフォルトコンフィグレーションが make distclean で削除されないようにするには

デフォルトコンフィグファイルのうち、Linux カーネルのデフォルトコンフィグレーションは、make distclean を実行すると削除されるようになっています。この挙動が望ましくない場合は、[プロダクトディレクトリ]/Makefile の distclean ターゲットで config.$(LINUXDIR) を削除しないよう、次のように書き換えてください。

```
distclean: clean
	rm -f etc/DISTNAME
```

図 13.15 distclean ターゲットの変更例

14. ハードウェア仕様

Armadillo-Box WS1 のハードウェア仕様について説明します。

14.1. インターフェースレイアウト

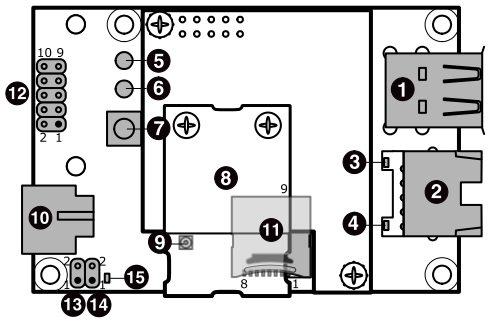


図 14.1 Armadillo-Box WS1 インターフェースレイアウト

表 14.1 Armadillo-Box WS1 搭載コネクタ、スイッチ型番一覧

部品番号	インターフェース名	型番	メーカー
1	USB インターフェース	UBA-4RSD14T-4D(LF)(SN)	J.S.T. Mfg.
2	LAN インターフェース	TM11R-5M2-88-LP	HIROSE ELECTRIC
3	LAN アクティビティ LED(黄色)	SML-310YTT86	ROHM
4	LAN リンク LED(緑色)	SML-310MTT86	ROHM
5	ユーザー LED(赤色)	SLR-342VC3F	ROHM
6	ユーザー LED(緑色)	SLR-342MC3F	ROHM
7	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC
8	Wi-SUN モジュール	BP35A1	ROHM
9	外付けアンテナインターフェース	MS-156C	HIROSE ELECTRIC
10	電源入力インターフェース	HEC3690-015210	HOSIDEN
11	microSD インターフェース	DM3C-SF	HIROSE ELECTRIC
12	デバッグシリアルインターフェース	A1-10PA-2.54DSA(71)	HIROSE ELECTRIC
13	起動モード設定ジャンパ(JP1)	A1-04PA-2.54DSA(71)	HIROSE ELECTRIC
14	起動モード設定ジャンパ(JP2)		
15	ユーザー LED(黄色)	SML-310YTT86	ROHM

14.2. USB インターフェース

USB ホストインターフェースを 2 つ搭載しています。信号線は i.MX257 の USB コントローラに接続されています。

USB インターフェースから USB デバイスに供給する電源は、電源入力 VIN と電源 IC で生成される +5V のどちらかを選択することが可能です。 電源の選択は i.MX257 の NFWE_B ピンで行い、i.MX257 の NFWE_B ピンから Low レベル出力で電源入力 VIN、High レベル出力で電源 IC で生成される+5V 電源が供給されます。

電源入力 VIN を使用する場合、供給可能な電流は各ポート最大 500mA となります。 電源 IC で生成される+5V 電源を使用する場合、供給可能な電流は 2 ポートの合計で最大 300mA となります。

表 14.2 USB 仕様

コネクタ位置	データ転送モード	コントローラ	PHY
USB コネクタ 下段	USB 2.0 High Speed(480Mbps) Full Speed(12Mbps) Low Speed(1.5Mbps)	OTG[a]	USBPHY1[b]
USB コネクタ 上段	USB 2.0 Full Speed(12Mbps) Low Speed(1.5Mbps)	HOST[a]	USBPHY2[b]

[a]i.MX257 内蔵 USB コントローラ
[b]i.MX257 内蔵 USB PHY

データ転送モードにある括弧内の転送速度は規格上の最大値を示しております。実際の転送速度がシステム要件を十分に満たすことをご確認の上、ご使用ください。

表 14.3 USB インターフェース 信号配列

ピン番号	ピン名	I/O	説明
1	+5V_USB	Power	USB 電源(VIN または+5V)
2	USB1-	In/Out	USB1 のマイナス側信号、i.MX257 の USBPHY1_DM ピンに接続
3	USB1+	In/Out	USB1 のプラス側信号、i.MX257 の USBPHY1_DP ピンに接続
4	GND	Power	電源(GND)
5	+5V_USB	Power	USB 電源(VIN または+5V)
6	USB2-	In/Out	USB2 のマイナス側信号、i.MX257 の USBPHY2_DM ピンに接続
7	USB2+	In/Out	USB2 のプラス側信号、i.MX257 の USBPHY2_DP ピンに接続
8	GND	Power	電源(GND)

Armadillo サイト [http://armadillo.atmark-techno.com/]にて、動作確認済み USB デバイス情報を随時更新していますのでご確認ください。

14.3. LAN インターフェース

10BASE-T/100BASE-TX に対応した LAN インターフェースを搭載しています。 信号線は Microchip Technology 製 PHY(LAN8720AI-CP)を経由して、i.MX257 の Ethernet コントローラ(FEC)に接続されています。 AUTO-MDIX 機能を搭載しており、ストレートケーブルまたはクロスケーブルを自動認識して送受信端子を切り替えます。

表 14.4 LAN インターフェース 信号配列

ピン番号	ピン名	I/O	説明
1	TX+	In/Out	差動のツイストペア送信/受信 1(+)
2	TX-	In/Out	差動のツイストペア送信/受信 1(−)
3	RX+	In/Out	差動のツイストペア送信/受信 2(+)
4	-	-	5 ピンと接続後に 75Ω 終端
5	-	-	4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	差動のツイストペア送信/受信 2(−)

ピン番号	ピン名	I/O	説明
7	-	-	8 ピンと接続後に 75Ω 終端
8	-	-	7 ピンと接続後に 75Ω 終端

表 14.5 LAN LED

名称(色)	状態	説明
LAN アクティビティ LED(黄色)	消灯	データを送受信していない
	点灯	データを送受信している
LAN リンク LED(緑色)	消灯	リンクが確立されていない
	点灯	リンクが確立されている

14.4. ユーザー LED

ユーザー側で自由に利用できる LED を 3 つ(赤色、緑色、黄色)搭載しています。

表 14.6 ユーザー LED の接続

名称(色)	説明
ユーザー LED(赤色)	i.MX257 の NFALE ピンに接続(Low: 消灯、High: 点灯)
ユーザー LED(緑色)	i.MX257 の NFCLE ピンに接続(Low: 消灯、High: 点灯)
ユーザー LED(黄色)	i.MX257 の BOOT_MODE0 ピンに接続(Low: 消灯、High: 点灯)

14.5. ユーザースイッチ

ユーザー側で自由に利用できるスイッチを 1 つ搭載しています。

表 14.7 ユーザースイッチの接続

名称	説明
ユーザースイッチ	i.MX257 の NFWP_B ピンに接続(ON: Low、OFF: High)

14.6. Wi-SUN モジュール

ROHM 製 Wi-SUN 対応無線モジュール(BP35A1)を搭載しています。

表 14.8 Wi-SUN モジュール仕様

無線規格	ARIB STD-T108 準拠
無線周波数	920MHz 帯
対応規格	ARIB STD-T108 規格対応 IEEE802.15.4g パケット対応
伝送電力	20mW 出力
受信感度	-103dBm(TYP.) (100kbps、BER < 0.1%)

上記は Wi-SUN モジュール単体での受信感度となり、Armadillo-Box WS1 ではコネクタ損失、基板、ケースとの干渉等により、受信感度が低下します。 実際の仕様がシステム要件を十分に満たすことをご確認の上、ご使用ください。

Wi-SUN モジュールには、外付けアンテナコネクタが搭載されています。受信感度が弱い時は、外付けアンテナの使用が効果的です。 外付けアンテ

ナの組み立てについては、「16.2. Wi-SUN モジュール用外付けアンテナの組み立て」をご参照ください。

14.7. 電源入力インターフェース

電源供給用の DC ジャックを搭載しています。AC アダプタのジャック形状は EIAJ RC-5320A 準拠(電圧区分 2)です。

図 14.2 AC アダプタの極性マーク

14.8. microSD インターフェース

microSD スロットを搭載しています。信号線は、i.MX257 の SD/MMC コントローラ(SDHC1)に接続されています。

microSD インターフェースに供給する電源は、i.MX257 の NFRE_B ピンで ON/OFF 制御が可能です。 i.MX257 の NFRE_B ピンから Low レベル出力で電源が供給され、High レベル出力で電源が切断されます。

microSD インターフェースから供給可能な電流は、LAN インターフェース、デバッグシリアルインターフェース、内部回路の合計で最大 200mA となります。

表 14.9 microSD 信号配列

ピン番号	ピン名	I/O	説明
1	SD1_DAT2	In/Out	データバス(bit2)、i.MX257 の SD1_DATA2 ピンに接続
2	SD1_DAT3	In/Out	データバス(bit3)、i.MX257 の SD1_DATA3 ピンに接続
3	SD1_CMD	In/Out	コマンド/レスポンス、i.MX257 の SD1_CMD ピンに接続
4	VDD	Power	電源(+3.3V_CPU)
5	SD1_CLK	Out	クロック、i.MX257 の SD1_CLK ピンに接続
6	VSS	Power	電源(GND)
7	SD1_DAT0	In/Out	データバス(bit0)、i.MX257 の SD1_DATA0 ピンに接続
8	SD1_DAT1	In/Out	データバス(bit1)、i.MX257 の SD1_DATA1 ピンに接続
9	SD1_CD*	In	カード検出(Low：カード挿入、High：カード未挿入)、i.MX257 の NFRB(GPIO3_31) ピンに接続

Armadillo サイト [http://armadillo.atmark-techno.com/]にて、動作確認済み microSD カード情報を随時更新していますのでご確認ください。

microSD インターフェースは活線挿抜に対応しておりません。microSD カードの挿抜は、電源を切断してから行ってください。

microSD カードを挿抜する際には、Wi-SUN モジュールを取り外してください。 無理に挿抜した場合、microSD カードが正常に挿入されないなどの原因で、動作不良を起こす場合があります。

14.8.1. microSD カードの挿入方法

1. カバーの穴に指の爪を引っ掛けてスライドさせ、ロックを解除してください。

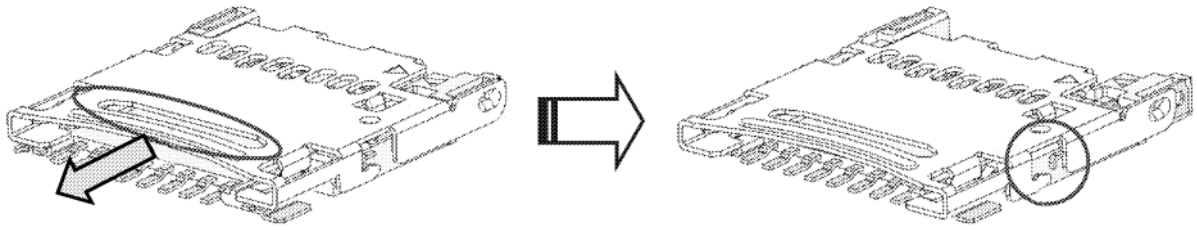


図 14.3 カードの挿入 1

2. カバーを開け、カードを挿入してください。カードは端子面が内側になるように挿入してください。挿入後は、完全にカードが奥まで入っていることを確認してください。

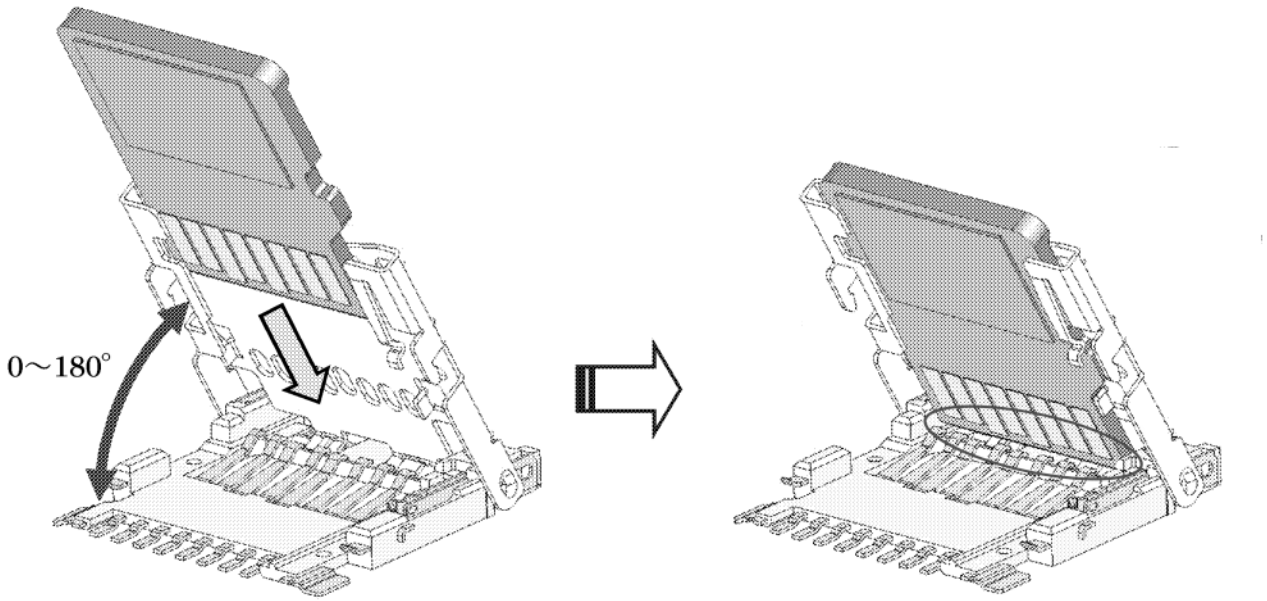


図 14.4 カードの挿入 2

3. カバーを閉じてください。カバーを閉じると信号端子のバネの力でカバーが浮き上がった状態になります。

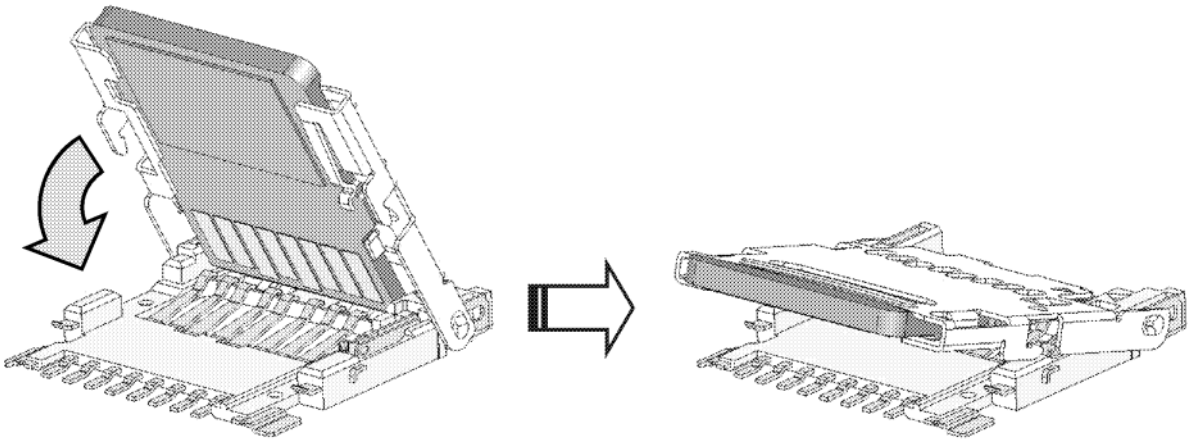
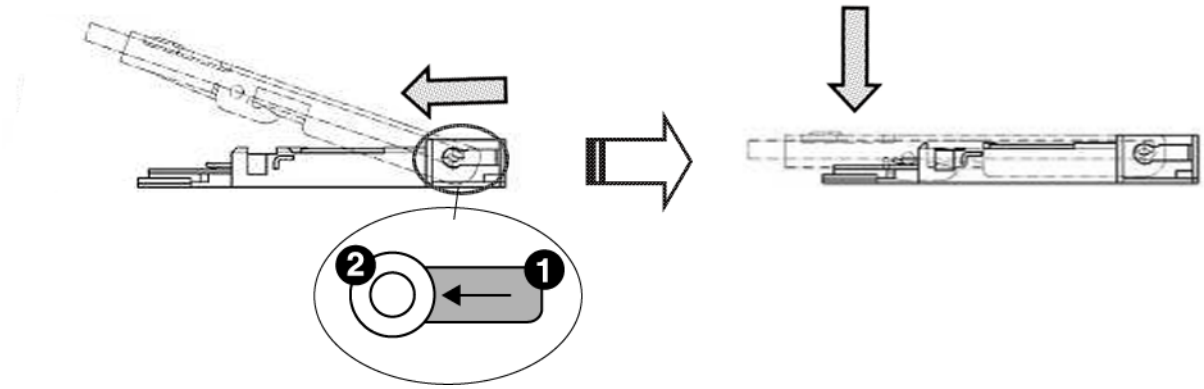


図 14.5 カードの挿入 3

4. 「図 14.6. カードの挿入 4」のように、カバーの軸が長穴の最も左側にある状態で、上から指で軽く押さえてください。



❶ 長穴

❷ 軸

図 14.6 カードの挿入 4

カバーを押さえる際は、必ずカバーの軸が長穴の最も左側にある状態で行ってください。 カバーを押さえる位置が合っていない状態でロックすると、「図 14.9. 異常なカード挿入状態(カードと基板が平行でない)」の様になり、カード検出が機能せず、動作不良となります。

5. カバーの穴に指の爪を引っ掛けて、カチっと音がするまでカバーをスライドさせ、モールドの△マークとカバーの△マークをそろえてください。

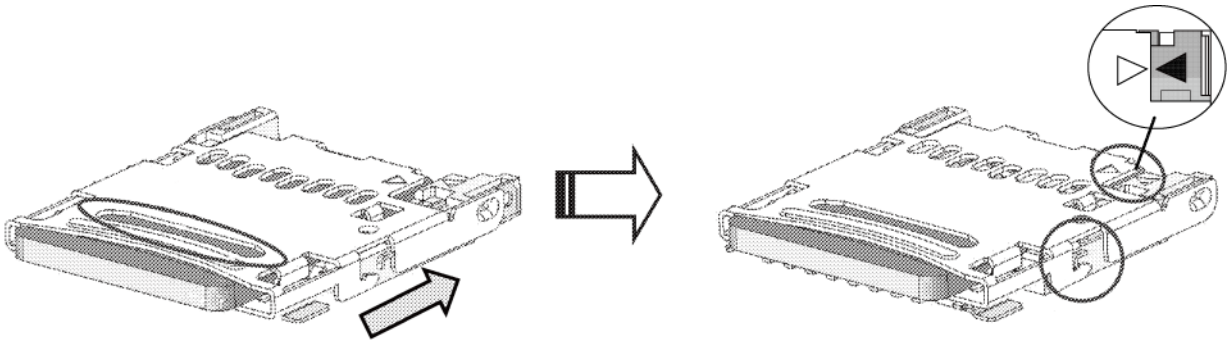


図 14.7 カードの挿入 5

カードが正常に挿入されているか確認してください。

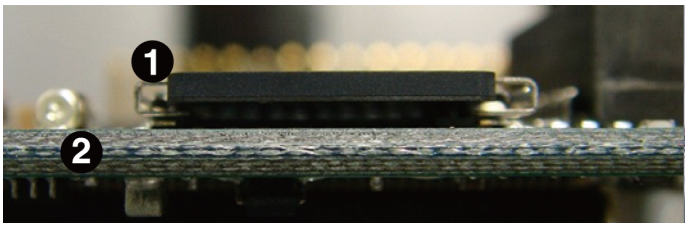
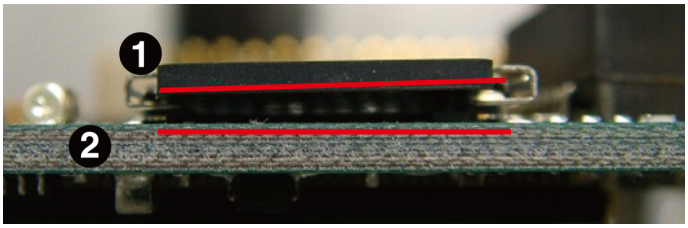


図 14.8 正常なカード挿入状態(カードと基板が平行)



❶ カード
❷ 基板

図 14.9 異常なカード挿入状態(カードと基板が平行でない)

14.8.2. microSD カードの抜去方法

1. カバーの穴に指の爪を引っ掛けてスライドさせ、ロックを解除してください。ロックを解除すると信号端子のバネの力でカバーが浮き上がります。

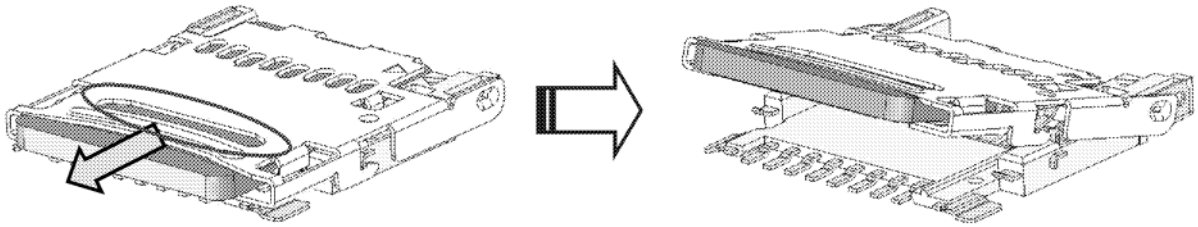


図 14.10 カードの抜去 1

2. カバーを開け、カードを抜去してください。

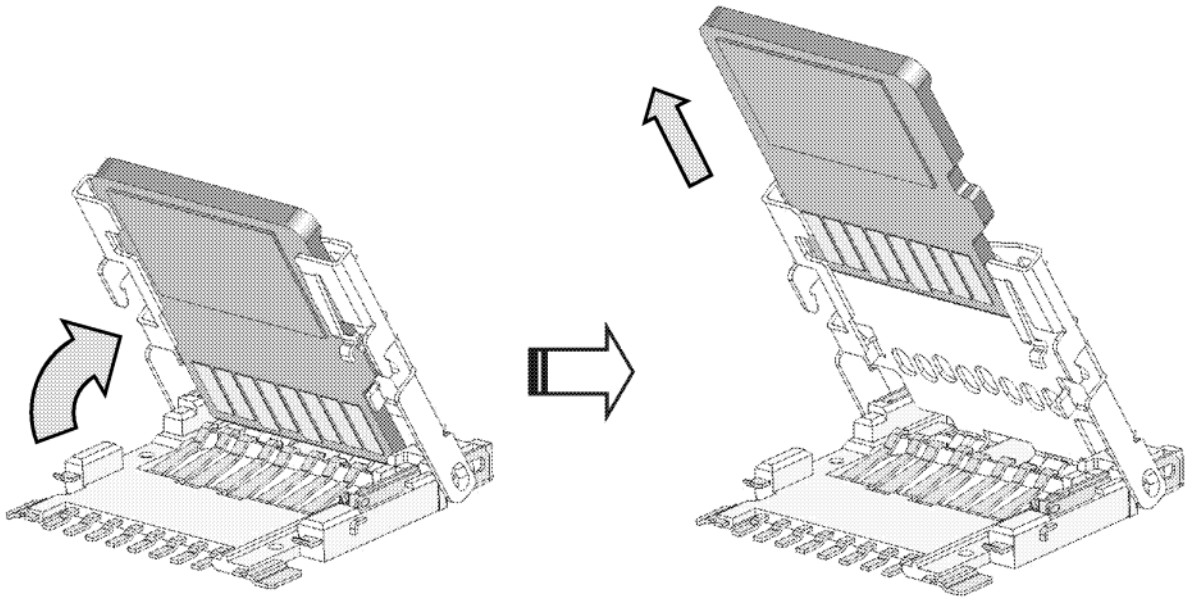


図 14.11 カードの抜去 2

3. カバーを閉じ、カバーの穴に指の爪を引っ掛けて、カチっと音がするまでカバーをスライドさせ、モールドの△マークとカバーの△マークをそろえてください。

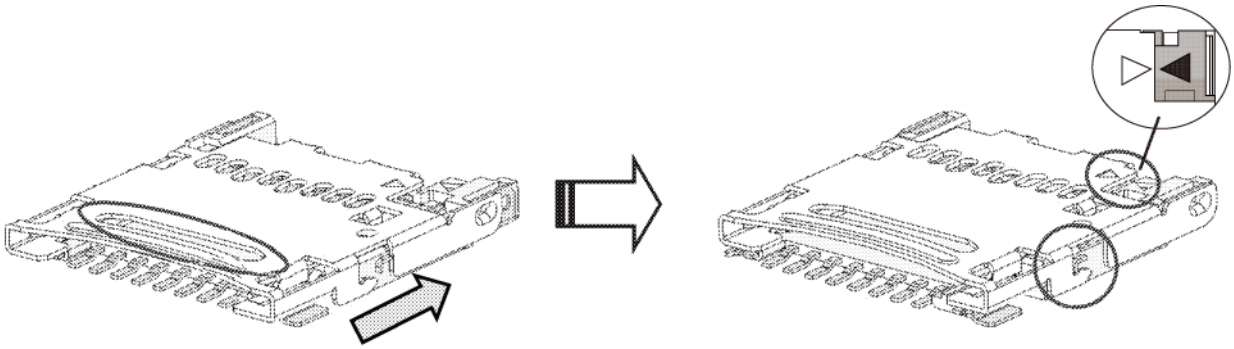


図 14.12 カードの抜去 3

microSD コネクタは microSD カードの挿入・未挿入に関わらず、必ずカバーをロックした状態でご使用ください。 microSD カードの未挿入時に

カバーのロックが解除されていると、コネクタ内部の接点がカバーに接触して動作不良を起こす場合があります。

microSD カード未挿入時に microSD コネクタを上から押さないでください。 コネクタ内部の接点がカバーに接触して動作不良を起こす場合があります。

14.9. デバッグシリアルインターフェース

非同期(調歩同期)のデバッグ用シリアルインターフェースを搭載しています。 信号線は RS232C レベル変換 IC を経由して、i.MX257 の UART コントローラ(UART2)に接続されています。

RS232C レベル変換 IC は、i.MX257 の BOOT_MODE1 ピンでシャットダウンすることが可能です。 BOOT_MODE1 ピンから Low レベル出力でシャットダウンモード、High レベル出力で通常モードとなります。

信号レベル	RS232C
最大データ転送レート	230.4kbps
フロー制御	CTS、RTS、DTR、DSR、DCD、RI

開発セット付属の D-sub コネクタ変換ケーブルを接続して、シリアルクロスケーブルで PC と通信可能です。

表 14.10 デバッグシリアルインターフェース 信号配列

ピン番号	信号名	I/O	機能
1	DCD2	In	キャリア検出、i.MX257 の UART1_RTS ピンに接続
2	DSR2	In	データセットレディ、i.MX257 の UART1_TXD ピンに接続
3	RXD2	In	受信データ、i.MX257 の UART2_RXD ピンに接続
4	RTS2	Out	送信要求、i.MX257 の UART2_CTS ピンに接続
5	TXD2	Out	送信データ、i.MX257 の UART2_TXD ピンに接続
6	CTS2	In	送信可能、i.MX257 の UART2_RTS ピンに接続
7	DTR2	Out	データ端末レディ、i.MX257 の UART1_RXD ピンに接続
8	RI2	In	被呼表示、i.MX257 の UART1_CTS ピンに接続
9	GND	Power	電源(GND)
10	+3.3V_CPU	Power	電源(+3.3V_CPU)

14.10. 起動モード設定ジャンパ

Armadillo-Box WS1 の起動モードを設定するジャンパと起動モードの状態を表す LED を搭載しています。

表 14.11 起動モード設定ジャンパ(JP1) 信号配列

ピン番号	ピン名	I/O	説明
1	JP1	In	i.MX257 の BOOT_MODE0 ピンに接続(10kΩ プルダウン)
2	JP1PU	Out	3.3V_CPU で 390Ω プルアップ

表 14.12 起動モード設定ジャンパ(JP2) 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	JP2	In	i.MX257 の NFC_CE0 ピンに接続(3.3V_CPU で 10kΩ プルアップ)

表 14.13 ジャンパの設定

JP1	JP2	起動モード	
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	

15. 電気的仕様

15.1. 絶対最大定格

表 15.1 絶対最大定格

Parameter	Symbol	Min	Max	Units	Conditions
Power Supply Voltage Range	VIN	-0.3	5.25	V	
Input Voltage Range	VI	-0.5	OVDD+0.3	V	OVDD=+3.3V_CPU, +3.3V_IO
Operating Temperature Range[a]	Topr	-10	60	℃	結露なきこと

[a]本体の使用温度範囲です。開発セット付属の AC アダプタの使用温度範囲は 0～40℃となります。

絶対最大定格は、あらゆる使用条件や試験状況において、瞬時でも超えてはならない値です。上記の値に対して余裕をもってご使用ください。

15.2. 推奨動作条件

表 15.2 推奨動作条件

Parameter	Symbol	Min	Typ	Max	Units	Conditions
Power Supply Voltage Range	VIN	4.75	-	5.25	V	
Operating Ambient Temperature Range	Ta	-10	25	60	℃	結露なきこと

15.3. 入出力インターフェースの電気的仕様

表 15.3 入出力インターフェース電源の電気的仕様

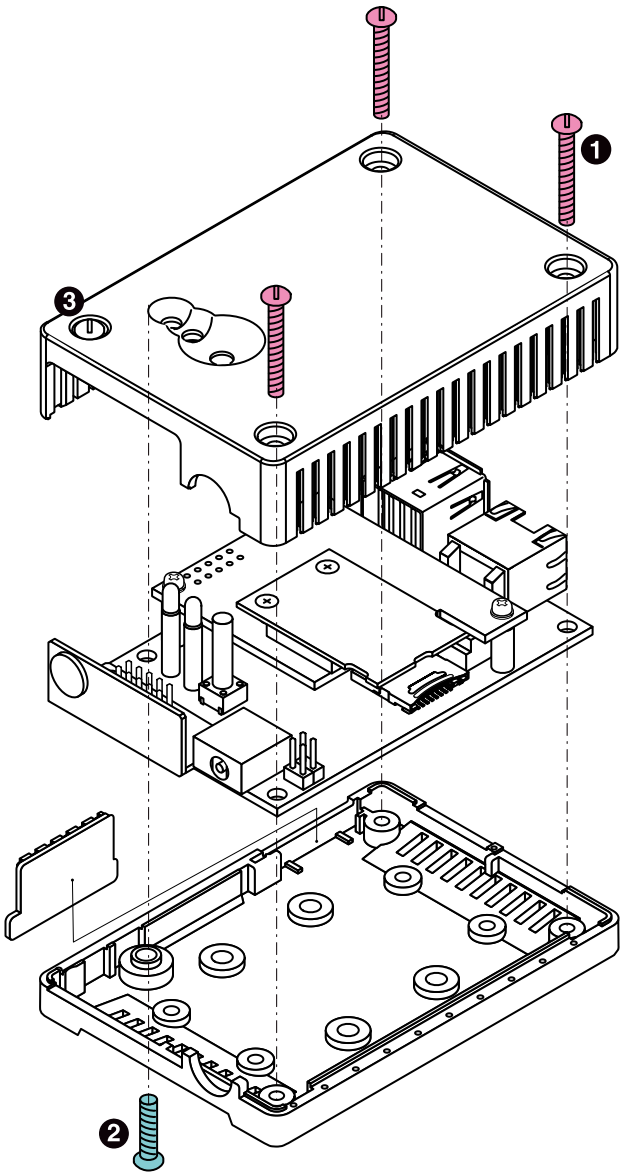
Parameter	Symbol	Min	Typ	Max	Units	Conditions
Power Supply Voltage	+3.3V_CPU	3.135	3.3	3.465	V	
	+3.3V_IO	3.135	3.3	3.465	V	
	+5V	4.75	5.0	5.25	V	

16. 組み立て

Armadillo- Box WS1 の組み立てについて説明します。

16.1. ケースの組み立て

基板をケースに収め、付属のネジで固定してください。



❶ タッピングネジ(M2.6、L=21mm)×3

❷ タッピングネジ(M3、L=12mm)×1

❸ 飾りネジです。ボンド止めされているので、無理に取り外さないでください。

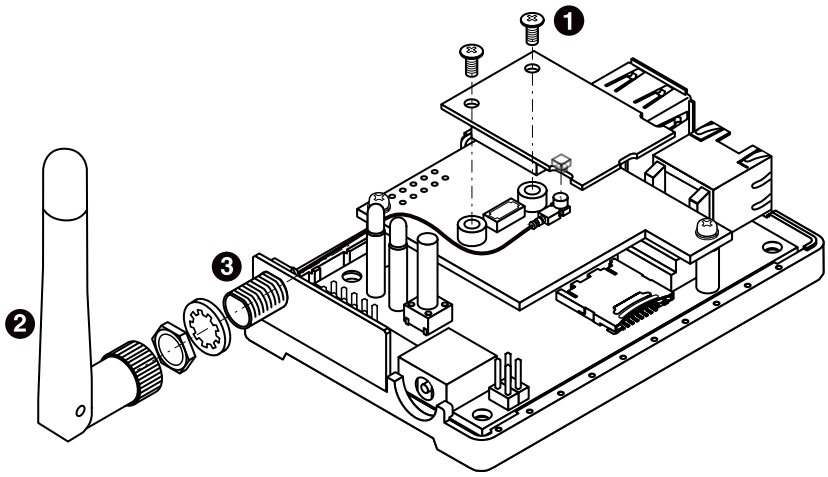
図 16.1 ケースの組み立て

ネジをきつく締め過ぎると、ケースが破損する恐れがありますので、十分にご注意ください。

16.2. Wi-SUN モジュール用外付けアンテナの組み立て

M2 のネジを 2 箇所外して、Wi-SUN モジュールを取り外します。アンテナケーブルを Wi-SUN モジュールの外付けアンテナコネクタに接続してください。反対の SMA コネクター側は、外付けアンテナ取り付け穴に固定してください。出荷状態では穴にキャップが取付けられているので、外してください。

アンテナを、ケーブルの SMA コネクターに接続します。



❶ 小ネジ(M2、L=4mm)×2

❷ Wi-SUN モジュール用アンテナ

❸ Wi-SUN モジュール用アンテナケーブル

図 16.2 外付けアンテナの組み立て

アンテナケーブルを接続する際、無理な力を加えると破損の原因となりますので十分に注意してください。

アンテナケーブルを引き抜く際は、専用の引き抜き治具(U.FL-LP-N-2/ヒロセ電機)を用いて行うことを推奨します。引き抜き治具を用いずに引き抜いた場合、コネクタの変形やケーブルの断線等の原因となります。

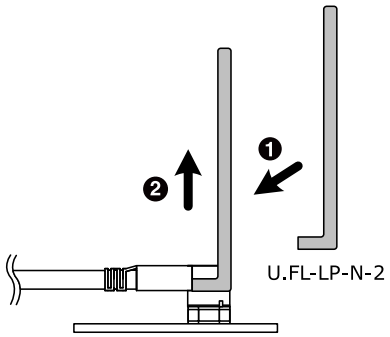


図 16.3 アンテナケーブルの引き抜き方法

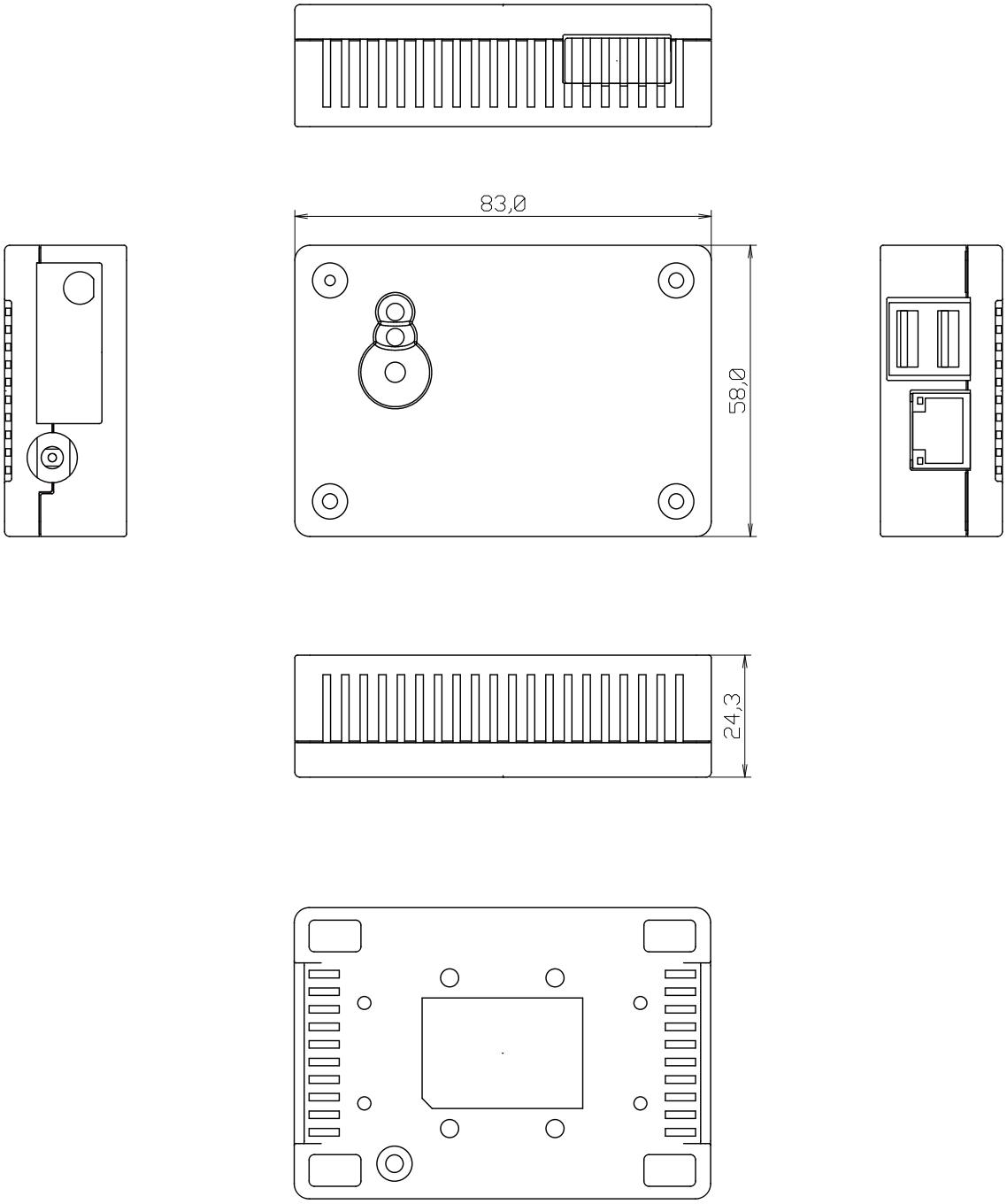
外付けアンテナコネクタにアンテナケーブルを長期間接続した場合、コネクタ内部のバネ弾性力がなくなり、内蔵アンテナが使用できなくなることがあります。 量産機器に組込んでご使用いただく場合、外付けアンテナから内蔵アンテナへの接続変更は推奨できません。

Wi-SUN モジュールの接続コネクタは複数回の挿抜を想定した仕様になっておりません。挿抜回数は 10 回以内としてください。

Wi-SUN モジュールを取り付ける時の M2 のネジの締め付けトルクは 1kgf・cm とし、締め付け過ぎにご注意ください。

17. 形状図

Armadillo-Box WS1 の外形寸法は次のとおりです。



[Unit : mm]

図 17.1 Armadillo-Box WS1 の外形寸法

18. ユーザー登録

アットマークテクノ製品をご利用のユーザーに対して、購入者向けの限定公開データの提供や大切なお知らせをお届けするサービスなど、ユーザー登録すると様々なサービスを受けることができます。サービスを受けるためには、「アットマークテクノ ユーザーズサイト」にユーザー登録をする必要があります。

ユーザー登録すると次のようなサービスを受けることができます。

- 製品仕様や部品などの変更通知の閲覧・配信
- 購入者向けの限定公開データのダウンロード
- 該当製品のバージョンアップに伴う優待販売のお知らせ配信
- 該当製品に関する開発セミナーやイベント等のお知らせ配信

詳しくは、「アットマークテクノ ユーザーズサイト」をご覧ください。

アットマークテクノ ユーザーズサイト

https://users.atmark-techno.com/

18.1. 購入製品登録

ユーザー登録完了後に、購入製品登録することで、「購入者向けの限定公開データ」をダウンロードすることができるようになります。

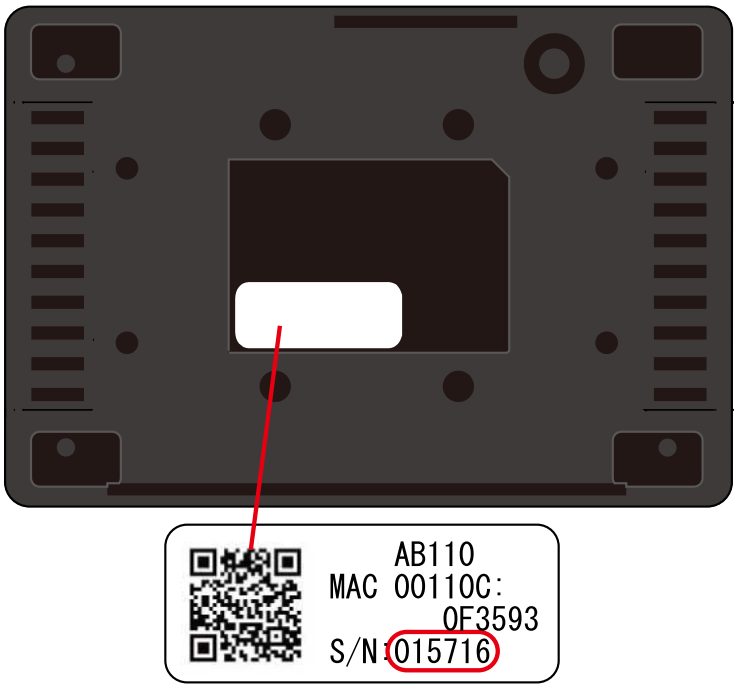
Armadillo-Box WS1 購入製品登録

https://users.atmark-techno.com/armadillo-box-ws1/register

Armadillo-Box WS1 の購入製品登録を行うには、ユーザーズサイトで「シリアル番号」の入力および「正規認証ファイル」のアップロードを行う必要があります

18.1.1. シリアル番号を確認する方法

シリアル番号は、ケース貼付シールに記載された 6 桁の数値です。次の例では、シリアル番号が「015716」であることが確認できます。



シリアル番号を「Armadillo-Box WS1 購入製品登録」ページの「シリアル番号」欄に入力してください。

18.1.2. 正規認証ファイルを取り出す手順

Armadillo にログインし、コマンドを実行すると正規認証ファイルが生成されます。そのファイルを USB メモリを使用して、PC に取り込んでください。

1. ATDE で minicom を立ち上げて、Armadillo-Box WS1 に root ユーザーでログインします。デバイスファイル名(/dev/ttyUSB0)は、ご使用の環境により ttyUSB1 や ttyS0、ttyS1 などになる場合があります。Armadillo に接続されているシリアルポートのデバイスファイルを指定してください。

```
atmark@atde5:~$ LANG=C minicom --noinit --wrap --device /dev/ttyUSB0

abws1-0 login: root
Password:
[root@abws1-0 (ttymxc1) ~]#
```

2. "get-board-info"コマンドを実行して正規認証ファイル(board-info.txt)を作成します。

```
[root@abws1-0 (ttymxc1) ~]# get-board-info
[root@abws1-0 (ttymxc1) ~]# ls
board-info.txt
[root@abws1-0 (ttymxc1) ~]#
```

3. USB メモリを接続して、正規認証ファイルを USB メモリにコピーします。次の例では、USB メモリが 1 つしか Armadillo に接続されていないことを想定しています。

```
[root@abws1-0 (ttymxc1) ~]# mount /dev/sda1 /mnt
[root@abws1-0 (ttymxc1) ~]# cp board-info.txt /mnt/
[root@abws1-0 (ttymxc1) ~]# umount /mnt
[root@abws1-0 (ttymxc1) ~]#
```

4. 正規認証ファイルをコピーした USB メモリを PC に接続し、正規認証ファイルを PC に取り込んでください。

その後、取り出した正規認証ファイルを「Armadillo-Box WS1 購入製品登録」ページの「正規認証ファイル」欄に指定し、アップロードしてください。

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2015/07/27	・初版発行

Armadillo-Box WS1 製品マニュアル
Version 1.0.0
2015/07/27

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