Armadillo-IoT ゲートウェイ スタンダードモデル 製品マニュアル

AG401-D00Z
AG401-D01Z
AG401-D02Z

Version 1.0.0
2014/12/16

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-IoT ゲートウェイスタンダードモデル製品マニュアル

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
2014/12/16

目次

図目次

表目次

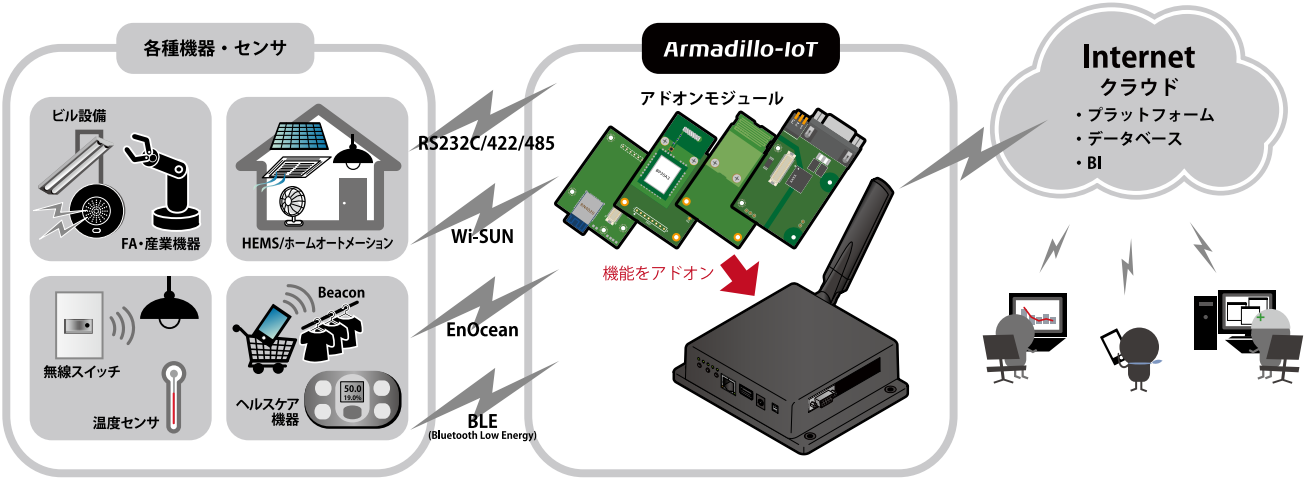
1. はじめに

このたびは Armadillo-IoT ゲートウェイ スタンダードモデルをご利用いただき、ありがとうございます。

Armadillo-IoT ゲートウェイ スタンダードモデル(以下、Armadillo-IoT)は、各種センサとネットワークとの接続を中継する IoT 向けゲートウェイの開発プラットフォームです。ハードウェアやソフトウェアをカスタマイズして、オリジナルのゲートウェイを素早く、簡単に開発することができます。

Armadillo-IoT は、センサ接続用インターフェースとして、RS232C/422/485、接点入出力など一般的なセンサ接続に広く使われるインターフェースの他、EnOcean や Wi-SUN、Bluetooth Low Energy など新しい省電力無線通信規格にも対応しています。これらの機能は専用の「アドオンモジュール」を付け替えることで、用途に応じて柔軟に構成できます。アドオンモジュールのインターフェース仕様は公開されているので、必要に応じてオリジナルのアドオンモジュールを開発することもできます。また、WAN(Wide Area Network)用インターフェースとして、LAN、無線 LAN(IEEE 802.11b/g/n)の他、モバイル通信(3G)も利用可能です。

Armadillo-IoT は標準 OS として Linux がプリインストールされているため、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、Java や Ruby などをサポートしています。さらに MQTT クライアントなど、クラウドサービスと親和性の高いソフトウェアスタックが用意され、ソフトウェア面でも開発の自由度と開発しやすさの両立を図っています。



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-IoT の使い方、製品仕様(ソフトウェアおよびハードウェア)、オリジナルの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。

また、Armadillo-IoT の機能をサポートする専用アプリケーションについても、その使い方を中心に説明しています。

Armadillo-IoT は一つの機器だけで完結するものではなく、接続するセンサや、クラウドシステムなどとの連携が不可欠です。そのため、参照すべきドキュメントも多岐に渡ります。本書では、アットマークテクノが運営する Armadillo サイトやユーザーズサイトを始め、開発に有用な情報を得る方法についても、随時説明しています。

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、各種モジュールの詳細仕様など、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-IoT 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-IoT を使ってオリジナルの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-IoT 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-IoT 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)
- アドオンモジュール回路図
- 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、「19. 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-IoT ゲートウェイ スタンダードモデル ドキュメント・ダウンロード

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

1.5. 本書の構成

本書には、Armadillo-IoT をベースに、オリジナルの製品を開発するために必要となる情報を記載しています。また、取扱いに注意が必要な事柄についても説明しています。

◆ **はじめにお読みください。**

「1. はじめに」、「2. 注意事項」

◆ **Armadillo-IoT ゲートウェイ スタンダードモデル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」

◆ **アドオンモジュールの開発や、ハードウェアをカスタマイズする場合に必要な情報を紹介します。**

「14. ハードウェア仕様」、「15. アドオンモジュール」、「16. オプション品」、「17. 設計情報」

◆ **ソフトウェアのカスタマイズ方法を紹介します。**

「18. Howto」

◆ **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。**

「19. 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

フォント例	説明
[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. 取扱い上の注意事項

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

破損しやすい箇所	BtoB コネクタは破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。
本製品の改造	本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。
電源投入時のコネクタ着脱	本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN、SD/SDIO、USB)以外へのコネクタやカードの着脱は、絶対に行わないでください。
静電気	本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。
ラッチアップ	電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。
衝撃	落下や衝撃などの強い振動を与えないでください。

2.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて	本製品の標準出荷状態でプリインストールされている Linux 対応ソフトウェアは、個別に明示されている（書面、電子データでの通知、口頭での通知を含む）場合を除き、オープンソースとしてソースコードが提供されています。再配布等の権利については、各ソースコードに記載のライセンス形態にしたがって、お客様の責任において行使してください。また、本製品に含まれるソフトウェア（付属のドキュメント等も含む）は、現状有姿（AS IS）にて提供します。お客様ご自身の責任において、使用用途・目的の適合について事前に十分な検討と試験を実施した上でお使いください。アットマークテクノは、当該ソフトウェアが特定の目的に適合すること、ソフトウェアの信頼性および正確性、ソフトウェアを含む本製品の使用による結果について、お客様に対し何らの保証も行いません。 パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェ

[1] コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2] コネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。

アに付属する readme ファイル等をご参照ください。その他のバンドルソフトウェアについては、各提供元にお問い合わせください。

本製品の標準出荷状態でプリインストールされている以下のソフトウェアは、オープンソースソフトウェアではありません。

- Oracle Java SE Embedded 8
- ボード情報取得ツール(get_board_info)

2.4. 書込み禁止領域について

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

2.5. 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

2.6. 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。

2.7. 商標について

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

Armadillo-IoT ゲートウェイは、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、IoT/M2M 向けのゲートウェイを簡単に、素早く開発するためのプラットフォームです。高い自由度と、開発のしやすさ、組み込み機器としての堅牢性をバランスよく兼ね備えており、オリジナルの商用 IoT ゲートウェイを市場のニーズに合わせてタイムリーに開発したい方に好適です。

アドオンモジュールで機能拡張

拡張用のインターフェースを 2 個搭載しており、任意のアドオンモジュールを接続可能です。RS232C/RS422/RS485 やデジタル入出力、アナログ入力等の有線接続用のアドオンモジュールや、Bluetooth Low Energy、EnOcean、Wi-SUN 等の省電力無線通信規格に対応したアドオンモジュールが標準ラインナップされています。

また、アドオンモジュール用のインターフェース規格は公開されているため、オリジナルのモジュールを開発できます。アドオンモジュールのみを開発するだけで様々な要求に対応することができるため、CPU ボードから全て開発する場合に比べて、開発期間とコストを低減できます。

モバイル通信(3G)対応

モバイル通信用に、3G 対応モジュールを搭載可能です。Armadillo-IoT 専用回線プランも各社から提供されており、3G 対応機能をすぐに導入できます。

Linux をベースとしたソフトウェアスタック

標準 OS として Linux を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、Ruby や Oracle Java にも対応しているため、C/C++言語以外でのソフトウェア開発が可能です。

クラウド対応

MQTT クライアントなど、クラウドシステムと相性の良いソフトウェアスタックをプリインストール。また、各社のクラウドサービス対応エージェントが、Armadillo-IoT 向けにポーティング済みなので、クラウドと連携したシステムが開発しやすくなっています。

3.2. 仕様

Armadillo-IoT ゲートウェイ スタンダードモデルの主な仕様は次のとおりです。

表 3.1 仕様

プロセッサ	Freescale 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)	RJ-45 x 1 100BASE-TX/10BASE-T, AUTO-MDIX 対応	
無線 LAN	Armadillo-WLAN(AWL13)搭載(オプション)[a][b] IEEE 802.11b/g/n	
モバイル通信	3G モジュール搭載(オプション)[a]	
シリアル(UART)	3.3V CMOS x 1	
SD/MMC	microSD スロット x 1、SD スロット x 1[b]	
温度センサ	測定精度: ±2℃	
AD コンバーター	分解能: 8bit	
USB	USB 2.0 Host x 1	
カレンダー時計	リアルタイムクロック 外部バックアップ用電源入力コネクタ搭載	
アドオンモジュール[c]	無線	Wi-SUN[a]、EnOcean[a]、Bluetooth 4.1/LE
	有線	シリアル(RS232C/RS422/RS485)
	接点入出力	デジタル入力 x 2/デジタル出力 x 2/アナログ入力 x 2
スイッチ	スイッチ x 3	
LED	LED x 5	
電源電圧	DC 8V～17V	
使用周囲温度	-10～60℃(ただし結露なきこと)[d][e]	

外形サイズ	ケース	125.6(W) x 45(H) x 125(D)mm(突起部を除く)
	ベースボード	105.5 x 108 mm(突起部を除く)

[a]外付けアンテナ接続可能
[b]無線 LAN と SD スロットは排他利用
[c]任意のアドオンモジュールを 2 個搭載可能
[d]高温時 3G モジュールの通信を停止するなど消費電力をセーブした場合
[e]基板単体での動作温度範囲は-20℃〜70℃

3.3. Armadillo-IoT ゲートウェイ スタンダードモデルの外観

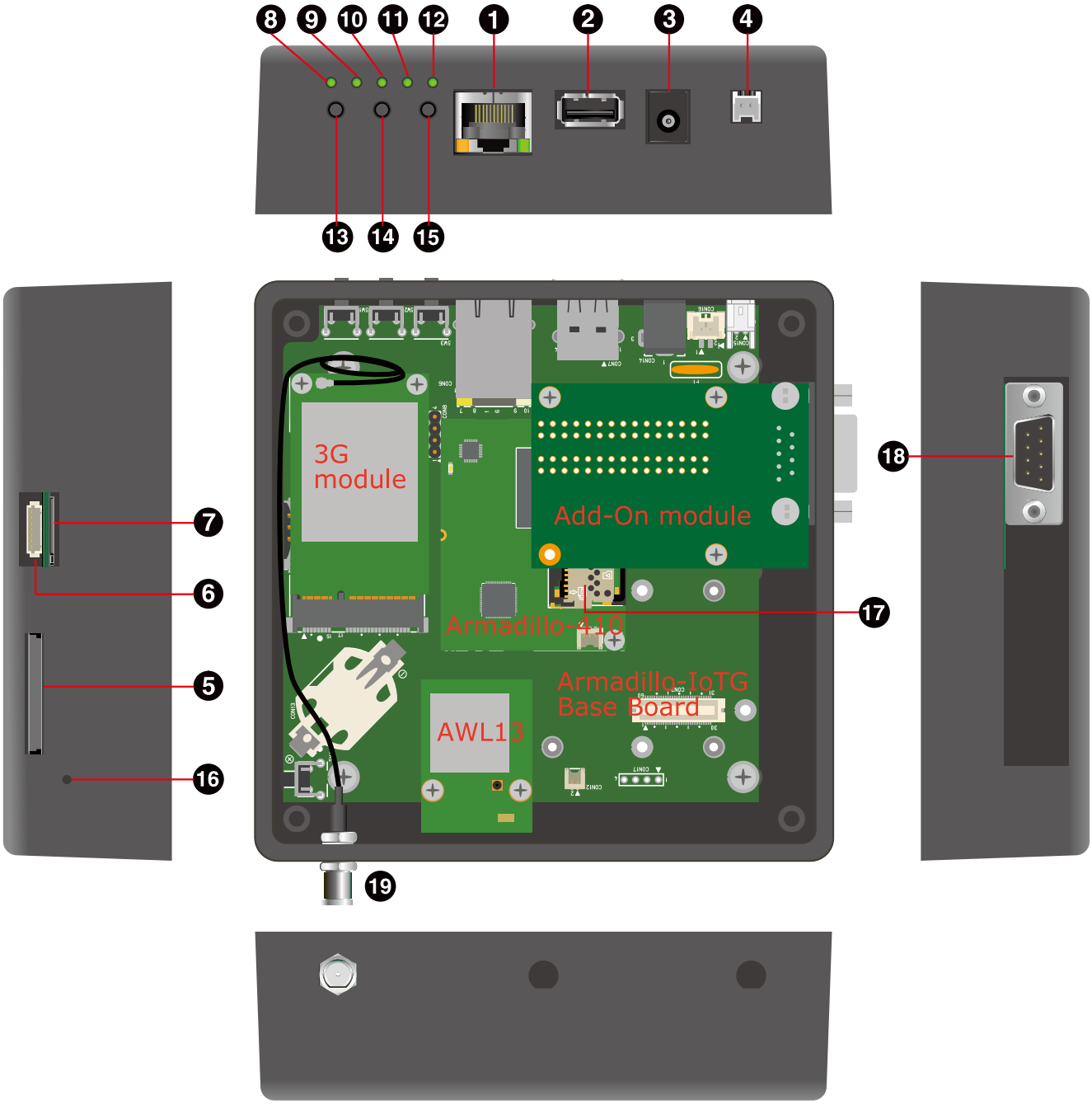


図 3.1 Armadillo-IoT ゲートウェイ スタンダードモデルの外観

表 3.2 各部名称と機能

番号	名称	説明
1	LAN コネクタ	LAN ケーブルを接続します。
2	USB コネクタ	USB メモリ等を接続します。

番号	名称	説明
3	電源コネクタ 1	付属の AC アダプタを接続します。
4	電源コネクタ 2	付属の AC アダプタ以外の電源ケーブルを接続します。
5	SD スロット	SD カードを挿入します。
6	デバッグシリアルコネクタ	付属の USB シリアル変換ケーブルを接続します。
7	microSIM スロット	microSIM カードを挿入します。
8	ユーザー LED1	ユーザーで自由に機能を設定できる緑色 LED です。
9	ユーザー LED2	
10	ユーザー LED3	
11	ユーザー LED4	
12	3G LED	3G モジュールの状態を表す緑色 LED です。
13	ユーザースイッチ 1	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。
14	ユーザースイッチ 2	
15	ユーザースイッチ 3	
16	リセットスイッチ	リセット用のタクトスイッチです。
17	microSD スロット	microSD カードを接続します。
18	シリアルコネクタ	シリアルクロスケーブルを接続します。
19	アンテナコネクタ	付属のアンテナを接続します。

3.4. ブロック図

Armadillo-IoT ゲートウェイ スタンダードモデルのブロック図は次の通りです。

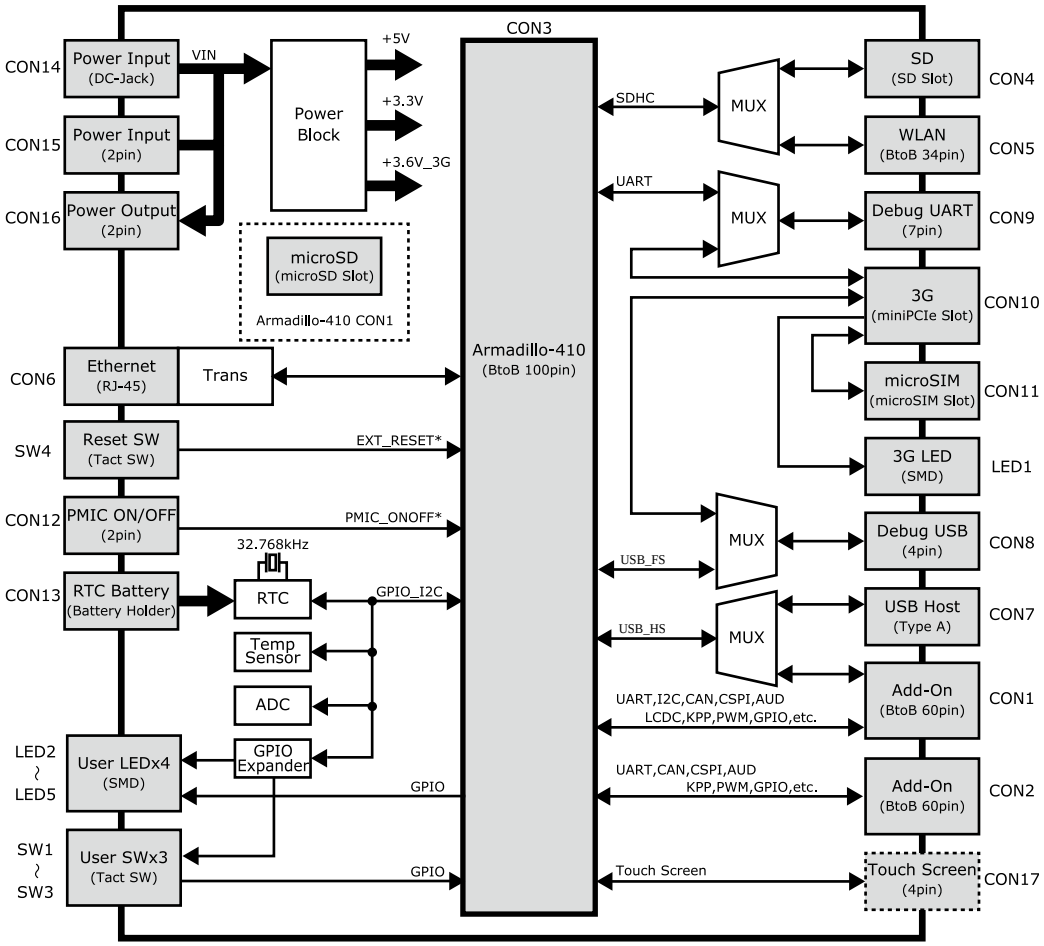


図 3.2 ブロック図

3.5. ソフトウェア構成

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

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

表 3.3 Armadillo-IoT 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 2.6.x 系の 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-IoT 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	26.75Mbyte	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

3.6. 製品ラインアップ

Armadillo-IoT の製品ラインアップは次の通りです。

表 3.5 Armadillo-IoT 製品ランナップ

名称	型番
Armadillo-IoT スタンダードモデル 開発セット	AG401-D00Z[a]
Armadillo-IoT ゲートウェイ スタンダードモデル 量産用 (3G 搭載)	AG401-C00Z
Armadillo-IoT ゲートウェイ スタンダードモデル 量産用 (3G 非搭載)	AG400-C00Z

[a]AG401-D01Z、AG401-D02Z は期間限定のキャンペーン品です。キャンペーン付属品以外の内容物は AG401-D00Z と同等です。

アドオンモジュールのラインナップは次の通りです。

表 3.6 アドオンモジュールラインナップ

名称	型番
Armadillo-IoT RS232C アドオンモジュール RS00	OP-AGA-RS00-00
Armadillo-IoT 絶縁 RS232C/422/485 アドオンモジュール RS01	OP-AGA-RS01-00[a]
Armadillo-IoT 絶縁デジタル入出力/アナログ入力アドオンモジュール DA00	OP-AGA-DA00-00[a]
Armadillo-IoT BLE アドオンモジュール BT00	OP-AGA-BT00-00[a]
Armadillo-IoT Wi-SUN アドオンモジュール WS00	OP-AGA-WS00-00[a]
Armadillo-IoT EnOcean アドオンモジュール EN00	OP-AGA-EN00-00[a]

[a]2015 年初頭発売予定

3.6.1. Armadillo-IoT スタンダードモデル 開発セット

Armadillo-IoT スタンダードモデル 開発セット(型番: AG401-D00Z)は、Armadillo-IoT を使った開発がすぐに開始できるように、開発に必要なものを一式含んだセットです。内蔵の RS232C アドオンモジュール RS01 以外のアドオンモジュールは別売です。

表 3.7 Armadillo-IoT スタンダードモデル 開発セットのセット内容

Armadillo-IoT スタンダードモデル (3G モジュール、無線 LAN モジュール内蔵、ケース入り)
RS232C アドオンモジュール RS01(Armadillo-IoT スタンダードモデルに内蔵)
3G モジュール用アンテナ
開発用 USB シリアル変換アダプタ
USB2.0 ケーブル(A-miniB タイプ)
AC アダプタ(12V)
開発用 DVD-ROM

3.6.2. Armadillo-IoT スタンダードモデル 量産用

Armadillo-IoT を使った製品の量産用モデルとして、Armadillo-IoT スタンダードモデル 量産用(3G 搭載)(型番: AG401-C00Z)と Armadillo-IoT スタンダードモデル 量産用(3G 非搭載)(型番: AG400-C00Z)をラインナップしています。アドオンモジュールや無線 LAN モジュール、その他付属品など、量産時に必要なものを同時に発注することができます。

また、シルク印刷の指定や、あるいはケース無しでの発注も可能です。詳細はお問い合わせください。

表 3.8 Armadillo-IoT スタンダードモデル 量産用(3G 搭載) 構成品

Armadillo-IoT スタンダードモデル (3G モジュール内蔵、ケース入り)

表 3.9 Armadillo-IoT スタンダードモデル 量産用(3G 非搭載) 構成品

Armadillo-IoT スタンダードモデル (ケース入り)

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 1 つ以上の USB ポートを持つ 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。
SD カード	SD 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。
microSIM(UIM カード)と APN 情報	3G の動作を確認する場合に利用します。通信事業者との契約が必要です。
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。Armadillo-IoT ゲートウェイ スタンダードモデルに対応している ATDE は、ATDE5 の v20141212 以降です。

ATDE5 は Debian GNU/Linux 7(コードネーム wheezy)をベースに、Armadillo-IoT ゲートウェイ スタンダードモデル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-IoT ゲートウェイ スタンダードモデル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-*[version]*-amd64.tar.xz	64-bit PC(「amd64」)アーキテクチャ用 Debian GNU/Linux 7
atde5-*[version]*-i386.tar.xz	32-bit PC(「i386」)アーキテクチャ用 Debian GNU/Linux 7

Armadillo-IoT ゲートウェイ スタンダードモデルに対応している ATDE5 のバージョンは v20141212 以降です。ATDE5 のバージョンと対応製品を次に示します。

バージョン	対応製品
v20141212 以降	・ Armadillo-IoT ゲートウェイ スタンダードモデル ・ Armadillo-840 ・ Armadillo-810
v20130710 ～ v20140131	・ Armadillo-840 ・ Armadillo-810
v20130319 以前	・ Armadillo-810

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。

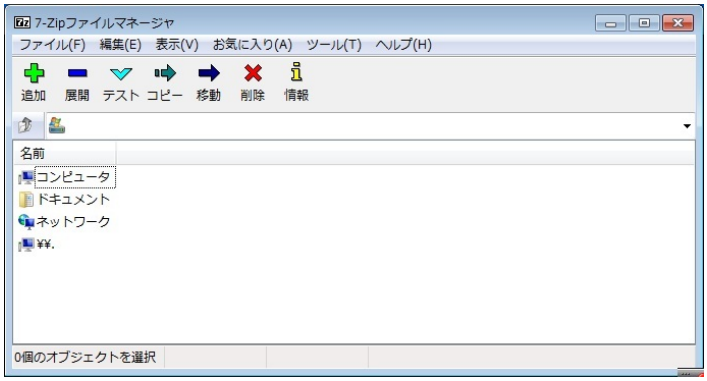
手順 4.1 Windows で ATDE5 のアーカイブ展開する

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。

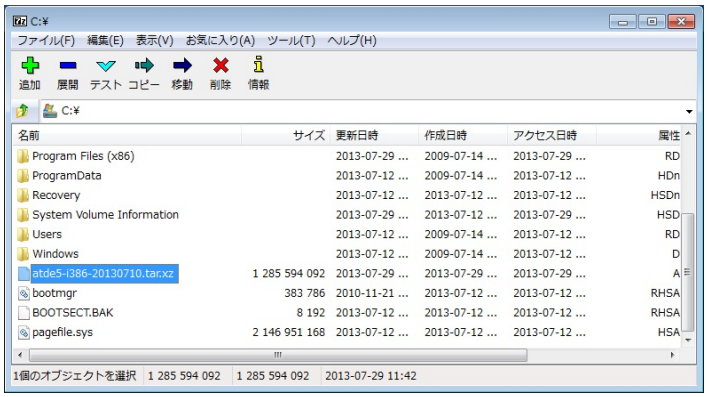
2. **7-Zip の起動**

   7-Zip を起動します。



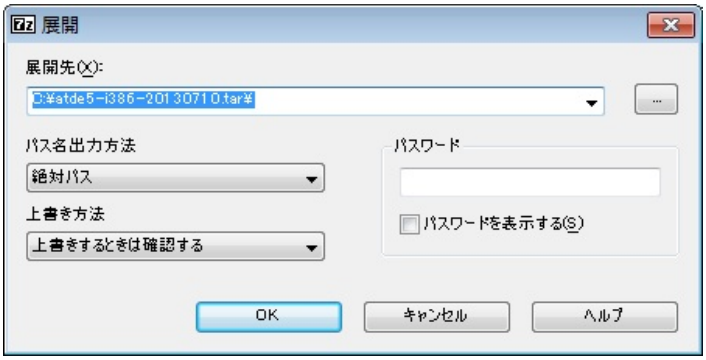
3. **xz 圧縮ファイルの選択**

   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。



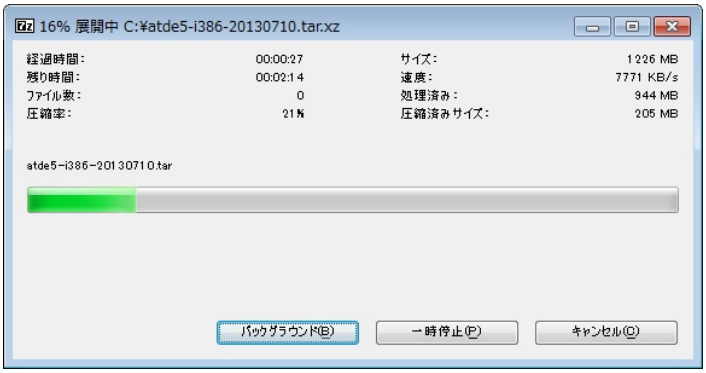
4. **xz 圧縮ファイルの展開先の指定**

「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**

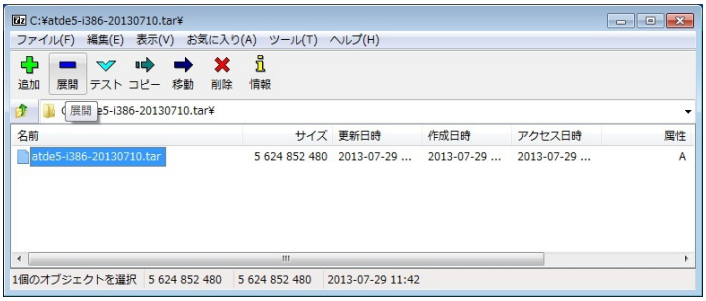
展開が始まります。



6. **tar アーカイブファイルの選択**

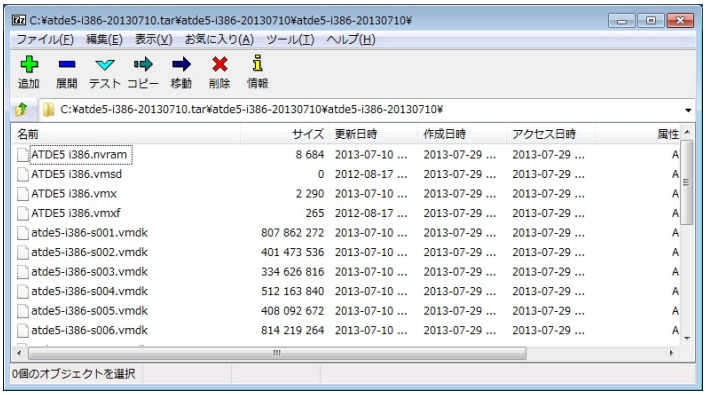
xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。

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-IoT 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
USB シリアル変換アダプタ	Future Devices FT232R USB UART
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

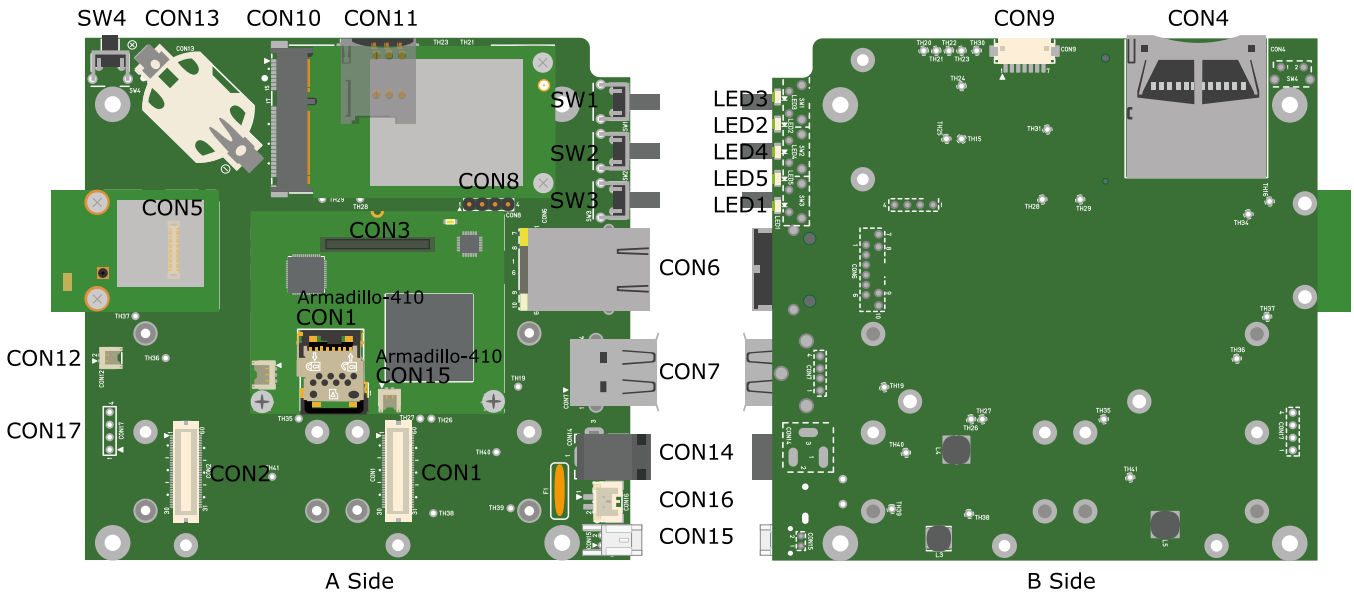


図 4.6 インターフェースレイアウト図

表 4.5 インターフェース内容(ベースボード)

部品番号	インターフェース名	形状	備考
CON1	アドオンインターフェース	BtoB コネクタ 60 ピン(0.5mm ピッチ)	挿抜寿命： 40 回[a]
CON2	アドオンインターフェース	BtoB コネクタ 60 ピン(0.5mm ピッチ)	挿抜寿命： 40 回[a]
CON3	Armadillo-410 インターフェース	BtoB コネクタ(0.4mm ピッチ)	挿抜寿命： 40 回[a]
CON4	SD インターフェース	SD スロット	
CON5	WLAN インターフェース	BtoB コネクタ 34 ピン(0.5mm ピッチ)	挿抜寿命： 40 回[a]
CON6	LAN インターフェース	RJ-45 コネクタ	
CON7	USB ホストインターフェース	Type A コネクタ	
CON8	デバッグ USB インターフェース	ピンヘッダ 4 ピン(2.54mm ピッチ)	
CON9	デバッグシリアルインターフェース	ピンヘッダ 7 ピン(1.25mm ピッチ)	挿抜寿命： 40 回[a]
CON10	3G インターフェース	PCI Express Mini Card スロット	
CON11	microSIM インターフェース	microSIM スロット	
CON12	PMIC ON/OFF インターフェース	ピンヘッダ 2 ピン(1.2mm ピッチ)	挿抜寿命： 20 回[a]
CON13	RTC 外部バックアップインターフェース	電池ボックス	対応電池: CR2032
CON14	電源入力インターフェース	DC ジャック	対応プラグ: 内径 2.1mm 外径 5.5mm
CON15	電源入力インターフェース	ピンヘッダ 2 ピン(2mm ピッチ)	
CON16	電源出力インターフェース	ピンヘッダ 2 ピン(2mm ピッチ)	
CON17	タッチスクリーンインターフェース	ピンヘッダ 4 ピン(2.54mm ピッチ)	
SW1	ユーザースイッチ	タクトスイッチ	
SW2			
SW3			
SW4	リセットスイッチ	タクトスイッチ	
LED1	3G LED	LED(緑色、面実装)	
LED2	ユーザー LED	LED(緑色、面実装)	
LED3			
LED4			
LED5			

[a]挿抜寿命は製品出荷時における目安であり、実際の挿抜可能な回数を保証するものではありません。

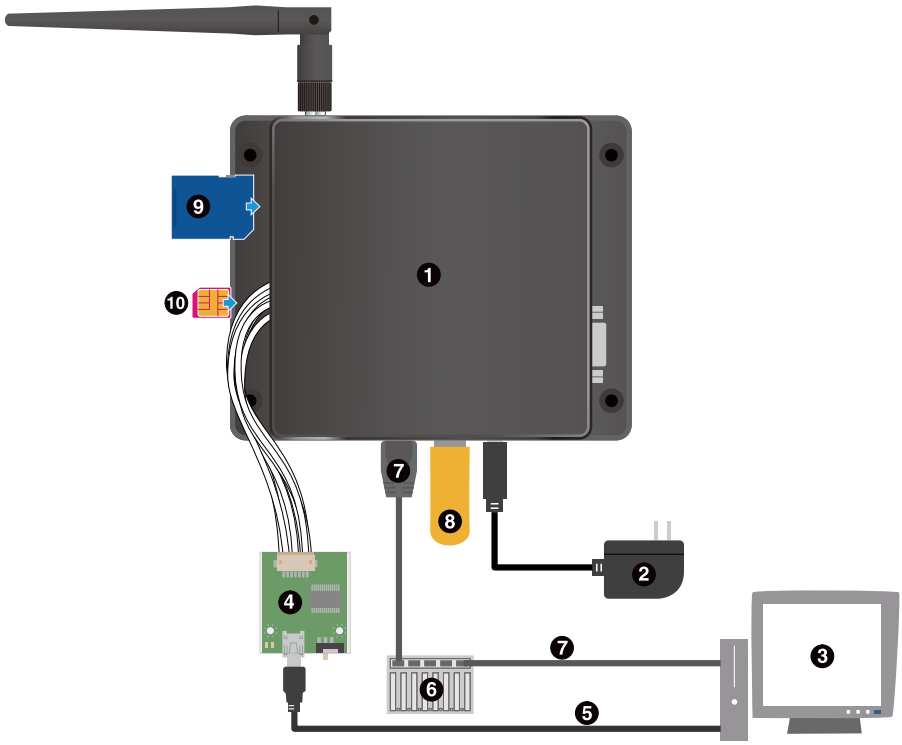
表 4.6 インターフェース内容(Armadillo-410)

部品番号	インターフェース名	形状	備考
CON1	microSD インターフェース	microSD スロット	
CON15	起動モード設定インターフェース	ピンヘッダ 2P(1.2mm ピッチ)	挿抜寿命: 20 回[a]

[a]挿抜寿命は製品出荷時における目安であり、実際の挿抜可能な回数を保証するものではありません。

4.4. 接続方法

Armadillo-IoT ゲートウェイ スタンダードモデルと周辺装置の接続例を次に示します。



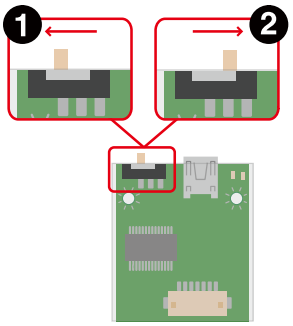
❶ Armadillo-IoT ゲートウェイ スタンダードモデル
❷ AC アダプタ(12V)[4]
❸ 作業用 PC
❹ USB シリアル変換アダプタ[4]
❺ USB2.0 ケーブル(A-miniB タイプ)[4]
❻ LAN HUB
❼ LAN ケーブル
❽ USB メモリ
❾ SD カード
❿ microSIM カード

図 4.7 Armadillo-IoT ゲートウェイ スタンダードモデルの接続例

[4]Armadillo-IoT ゲートウェイ スタンダードモデル開発セット付属品

4.5. スライドスイッチの設定について

USB シリアル変換アダプタのスライドスイッチを操作することで、シリアルの接続先を変更することができます。本書では、常に Armadillo-IoT のシリアルコンソールに接続して使用します。電源を投入する前にスライドスイッチの設定を確認してください。



❶　Armadillo-IoT のシリアルコンソール(i.MX257 の UART2)に接続します。ブートローダーは保守モード[5]になります。

❷　3G モジュールに接続します。

図 4.8 スライドスイッチの設定

Armadillo-IoT に USB シリアル変換アダプタを接続せずに電源を投入した場合、ブートローダーはオートブートモード[6]になります。

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 4.9 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)を行ないます。vi はコマンドモードの状態で起動します。

[5]ブートローダーのコマンドプロンプトが起動します。
[6]OS を自動起動します。

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 4.10. 入力モードに移行するコマンドの説明」に示します。

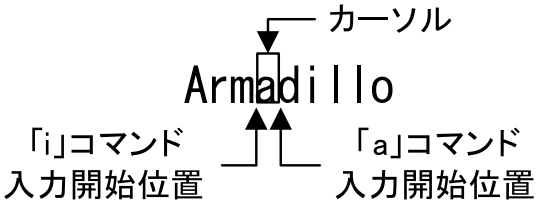


図 4.10 入力モードに移行するコマンドの説明

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

コマンド	動作
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 4.11. 文字を削除するコマンドの説明」に示します。

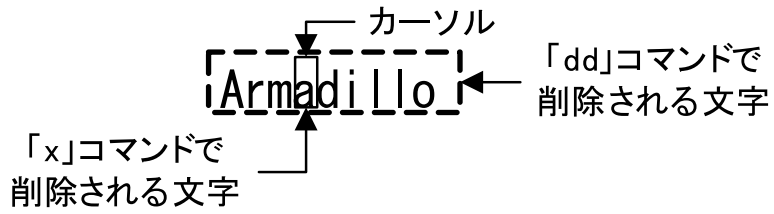


図 4.11 文字を削除するコマンドの説明

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-IoT 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 v3.3.0 (armadillo-iotg-std) compiled at 10:41:18, Dec 08 2014
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...........................................................
..........................................................done.
Uncompressing ramdisk...........................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
..............................................................................do
ne.
Linux version 2.6.26-at21 (2.6.26) (atmark@atde5) (gcc version 4.6.3 (Debian 4.6
.3-14atmark1) ) #1 PREEMPT Sun Dec 14 13:40:35 JST 2014
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
Machine: Armadillo-410
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 32512
Kernel command line: console=ttymxc1,115200
MXC IRQ initialized
```

```
PID hash table entries: 512 (order: 9, 2048 bytes)
MXC GPT timer initialized, rate = 133000000
Console: colour dummy device 80x30
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
Memory: 128MB = 128MB total
Memory: 72992KB available (3432K code, 235K data, 124K init)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
net_namespace: 624 bytes
NET: Registered protocol family 16
MXC WDOG1 Enabled
CPU is i.MX25 Revision 1.2
Clock input source is 24000000
MXC GPIO hardware
GPIO-56 autorequested
GPIO-6 autorequested
Using SDMA I.API
MXC DMA API initialized
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
i2c-gpio i2c-gpio.3: using pins 17 (SDA) and 18 (SCL, no clock stretching)
i2c-gpio i2c-gpio.4: using pins 66 (SDA) and 65 (SCL, no clock stretching)
MXC I2C driver
MC34704 regulator successfully probed
mc34704 0-0054: Loaded
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 4096 (order: 3, 32768 bytes)
TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP reno registered
NET: Registered protocol family 1
checking if image is initramfs...it isn't (bad gzip magic numbers); looks like a
n initrd
Freeing initrd memory: 52993K
usb: Host 2 host (serial) registered
usb: DR host (utmi) registered
msgmni has been set to 246
io scheduler noop registered
io scheduler cfq registered (default)
Serial: MXC Internal UART driver
mxcintuart.1: ttymxc1 at MMIO 0x43f94000 (irq = 32) is a Freescale MXC
console [ttymxc1] enabled
brd: module loaded
loop: module loaded
FEC Ethernet Driver
PPP generic driver version 2.4.2
usbcore: registered new interface driver asix
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 zaurus
usbcore: registered new interface driver sierra_net
Linux video capture interface: v2.00
usbcore: registered new interface driver uvcvideo
```

```
USB Video Class driver (v0.1.0)
Driver 'sd' needs updating - please use bus_type methods
armadillo-nor: Found 1 x16 devices at 0x0 in 16-bit bank
 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
armadillo-nor: use default partitions(4)
Creating 4 MTD partitions on "armadillo-nor":
0x00000000-0x00020000 : "nor.bootloader"
0x00020000-0x00420000 : "nor.kernel"
0x00420000-0x01f00000 : "nor.userland"
0x01f00000-0x02000000 : "nor.config"
fsl-ehci fsl-ehci.0: Freescale On-Chip EHCI Host Controller
fsl-ehci fsl-ehci.0: new USB bus registered, assigned bus number 1
fsl-ehci fsl-ehci.0: irq 35, io mem 0x53ff4400
fsl-ehci fsl-ehci.0: USB 2.0 started, EHCI 1.00, driver 10 Dec 2004
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
fsl-ehci fsl-ehci.1: Freescale On-Chip EHCI Host Controller
fsl-ehci fsl-ehci.1: new USB bus registered, assigned bus number 2
fsl-ehci fsl-ehci.1: irq 37, io mem 0x53ff4000
fsl-ehci fsl-ehci.1: USB 2.0 started, EHCI 1.00, driver 10 Dec 2004
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
usbcore: registered new interface driver usbserial
usbserial: USB Serial Driver core
usbserial: USB Serial support registered for Sierra USB modem
usbcore: registered new interface driver sierra
sierra: USB Driver for Sierra Wireless USB modems: v.1.7.40
input: gpio-keys as /devices/platform/gpio-keys.0/input/input0
input: gpio-keys-polled as /devices/virtual/input/input1
rtc-s35390a 3-0030: rtc core: registered rtc-s35390a as rtc0
i2c /dev entries driver
adc081c 3-0054: ADC081C021/027 driver probed
mxsdhci: MXC Secure Digital Host Controller Interface driver
mxsdhci: MXC SDHCI Controller Driver.
mmc0: SDHCI detect irq 159 irq 9 INTERNAL DMA
mxsdhci: MXC SDHCI Controller Driver.
mmc1: SDHCI detect irq 71 irq 8 INTERNAL DMA
Registered led device: led1
Registered led device: led2
Registered led device: led3
Registered led device: led4
Registered led device: yellow
usbcore: registered new interface driver usbhid
usbhid: v2.6:USB HID core driver
Advanced Linux Sound Architecture Driver Version 1.0.16.
usbcore: registered new interface driver snd-usb-audio
```

```
usbcore: registered new interface driver snd-usb-caiaq
ASoC version 0.13.2
ALSA device list:
  No soundcards found.
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP cubic registered
NET: Registered protocol family 10
NET: Registered protocol family 17
NET: Registered protocol family 15
Static Power Management for Freescale i.MX25
rtc-s35390a 3-0030: setting system clock to 2000-01-01 00:01:48 UTC (946684908)
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 52993KiB [1 disk] into ram disk... done.
usb 1-1: new full speed USB device using fsl-ehci and address 2
usb 1-1: not running at top speed; connect to a high speed hub
usb 1-1: config 1 has an invalid interface number: 7 but max is 5
usb 1-1: config 1 has no interface number 5
usb 1-1: configuration #1 chosen from 1 choice
sierra 1-1:1.0: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB0
sierra 1-1:1.1: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB1
sierra 1-1:1.2: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB2
sierra 1-1:1.3: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB3
sierra 1-1:1.4: Sierra USB modem converter detected
usb 1-1: Sierra USB modem converter now attached to ttyUSB4
usb0: register 'sierra_net' at usb-fsl-ehci.0-1, Sierra Wireless USB-Ethernet Mo
dem, 56:4d:db:fc:01:07
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 124K
Mounting proc: done
Starting fsck for root filesystem.
fsck 1.25 (20-Sep-2001)
/dev/ram0: clean, 2461/2968 files, 47898/52993 blocks
Checking root filesystem: done
Remounting root rw: done
Mounting usbfs: done
Mounting sysfs: done
Cleaning up system: done
Running local start scripts.
Starting udevd: done
Loading /etc/config: done
Changing file permissions: done
Configure /home/ftp: done
Mounting devpts: done
Starting syslogd: done
Starting klogd: done
Loading kernel module: awl13_sdio
awl13: Version 3.0.2 Load.
Starting basic firewall: done
Setting hostname: done
Configuring network interfaces: fec: PHY @ 0x0, ID 0x0007c0f1 -- LAN8720
udhcpc (v1.20.2) started
eth0: link up, 100Mbps, full-duplex
Sending discover...
Sending select for 192.0.2.100...
```

```
Lease of 192.0.2.100 obtained, lease time 86400
done
Starting inetd: done
Starting lighttpd: done
Creating avahi.services: done
Starting avahi.daemon: done
Mounting ramfs /home/ftp/pub: done
Running local start script (/etc/config/rc.local).

atmark-dist v1.36.0 (AtmarkTechno/Armadillo-IoTG-Std)
Linux 2.6.26-at21 [armv5tejl arch]

armadillo-iotg login:
```

図 5.2 起動ログ

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: done
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system halt
System halted.
```

図 5.3 終了方法

SD 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。

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-IoT は、複数の種類のネットワークに接続することができます。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	備考
有線 LAN	eth0	
無線 WLAN	awlan0	Armadillo-WLAN(AWL13) 搭載
3G	usb0	Sierra Wireless 製 MC8090 搭載

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
usb0	TCP/IP	DHCP	無効
awlan0	未設定	未設定	未設定

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

auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
iface usb0 inet dhcp
    pre-up 3g-connect
    post-down 3g-disconnect
```

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-IoT 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

項目	設定
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
iface usb0 inet dhcp
    pre-up 3g-connect
    post-down 3g-disconnect
```

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
iface usb0 inet dhcp
    pre-up 3g-connect
    post-down 3g-disconnect
```

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

awlan0 または usb0 を使用してネットワークに接続している場合、ネットワーク通信に eth0 が使用されない場合があります。確実に eth0 を使

用させる場合は、事前に eth0 以外のネットワークインターフェースを無効化してください。

6.2.4. 無線 LAN

ここでは、Armadillo-IoT に搭載されている無線 LAN モジュール「Armadillo-WLAN モジュール(AWL13)」の使用方法について説明します。

6.2.4.1. WLAN インターフェースの有効化

Armadillo-WLAN モジュール(AWL13)が接続されている WLAN インターフェース(ベースボード:CON5)と SD インターフェース(ベースボード:CON4)は、共通の信号が接続されています。工場出荷状態のソフトウェアでは、デフォルトで SD インターフェースが有効化されているため、無線 LAN モジュールを利用することができません。

WLAN インターフェースを有効化するためには、「図 6.7. 無線 LAN モジュールの有効化」のようにコマンドを実行します。事前に SD インターフェース(ベースボード:CON4)から SD カードを取り外しておく必要があります。

```
[armadillo ~]# sd-awlan-sel awlan
select to AWLAN
mmc1: new high speed SDIO card at address 02bd
awl13: RX Transmission mode SDINT HT
mmc1: registerd "awl13" device as awlan0
awl13: WID=0x5, STATUS CODE=0x0
awl13: disconnected!
awl13: device ready!
awl13: MAC is 00:1d:12:cf:28:ae
awl13: WID=0x5, STATUS CODE=0x1
awl13: WID=0x5, STATUS CODE=0x1
```

図 6.7 無線 LAN モジュールの有効化

WLAN インターフェースを有効化すると、Armadillo-WLAN モジュール(AWL13)の Linux カーネルモジュールおよびファームウェアのロードが自動的に行われます。

Armadillo-WLAN モジュール(AWL13)のファームウェアは、"STA"(ステーションおよびアドホックモード用)と"AP"(アクセスポイントモード用)の 2 種類が用意されています。工場出荷状態のソフトウェアでは、"STA"が自動的にロードされるよう設定されています。

"AP"を自動的にロードするように設定するには「図 6.7. 無線 LAN モジュールの有効化」を実行する前に、次のように設定ファイル/etc/config/awl13.conf を編集します。

```
[armadillo ~]# vi /etc/config/awl13.conf
#AWL13_MODE=STA
AWL13_MODE=AP
```

> 設定ファイルをコンフィグ領域に保存する方法については、「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。

6.2.4.2. 手動で無線 LAN インターフェースを有効化する

コマンドを入力して無線設定および無線 LAN インターフェースの有効化を行う方法について説明します。事前に「6.2.4.1. WLAN インターフェースの有効化」を参照して WLAN インターフェースを有効化しておく必要があります。

ここでは例として、WPA2-PSK(AES)のアクセスポイントに接続します。WPA2-PSK(AES)以外のアクセスポイントへの接続方法など、AWL13 のより詳細な情報については、「Armadillo-WLAN(AWL13) ソフトウェアマニュアル」を参照してください。

WPA2-PSK(AES)のアクセスポイントに接続する場合の設定例を次に示します。以降の説明では、アクセスポイントの ESSID を*[essid]*、パスフレーズを*[passphrase]*と表記します。

```
[armadillo ~]# iwconfig awlan0 essid [essid]
[armadillo ~]# iwpriv awlan0 set_psk [passphrase]
[armadillo ~]# iwpriv awlan0 set_cryptmode WPA2-AES
[armadillo ~]# iwconfig awlan0 mode managed
```

上記コマンドを実行すると、無線設定が完了します。無線 LAN インターフェースの IP アドレスを 192.0.2.1 に設定して有効化するには、次のようにコマンドを実行します。

```
[armadillo ~]# ifconfig awlan0 192.0.2.1 up
```

図 6.8 ネットワークインターフェース(awlan0)の IP アドレス設定と有効化

6.2.4.3. 自動で無線 LAN インターフェースを有効化する

Armadillo の起動時に、自動的に無線 LAN インターフェースの有効化を行う方法について説明します。無線 LAN の設定をコンフィグ領域に保存することにより、Armadillo を再起動するたびに設定を行う必要が無くなります。

Armadillo の起動時に、自動的に WLAN インターフェースを有効化するために/etc/config/rc.local に「sd-awlan-sel awlan」を追加します。

```
[armadillo ~]# vi /etc/config//etc/config/rc.local
        echo -n "Starting vinmonitor: "
        /etc/config/vinmonitor &
        check_status
fi

sd-awlan-sel awlan ❶
```

❶ WLAN インターフェースを有効化します

WPA2-PSK(AES)のアクセスポイントに接続する場合の/etc/config/interfaces の編集例を次に示します。

```
[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
iface usb0 inet dhcp
    pre-up 3g-connect
    post-down 3g-disconnect
iface awlan0 inet dhcp ❶
      pre-up iwpriv awlan0 set_psk [passphrase] ❷
      pre-up iwpriv awlan0 set_cryptmode WPA2-AES ❸
      pre-up iwconfig awlan0 essid [essid] ❹
      wireless-mode managed ❺
```

❶ awlan0 を DHCP に設定します

❷ パスフレーズを*[passphrase]*に設定します

❸ 暗号化方式を WPA2-PSK(AES)に設定します

❹ ESSID を*[essid]*に設定します

❺ 接続モードをインフラストラクチャモード(STA)に設定します

Armadillo の起動時に自動的に awlan0 が有効化されるようにするには、/etc/config/awl13-firmware-load.sh の最後の行に「ifup awlan0」を追加します。

```
[armadillo ~]# vi /etc/config/awl13-firmware-load.sh
[ -f /sys/module/awl13_usb/$WLAN/firmware ] && \
cat $FIRMWARE_USB > /sys/module/awl13_usb/$WLAN/firmware
iwpriv $WLAN fwload
iwpriv $WLAN fwsetup

ifup awlan0 ❶
```

❶ /etc/config/interfaces の設定で awlan0 を有効化します

追加後、次回起動時に設定が反映されるようにコンフィグ領域を保存します。

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

Armadillo を再起動すると、自動的に無線 LAN インターフェースが有効化されます。

Armadillo の再起動の前に、SD インターフェース(ベースボード:CON4)から SD カードを取り外しておく必要があります。

6.2.4.4. 無線 LAN の接続を確認する

無線 LAN で正常に通信が可能か確認します。

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 6.9 無線 LAN の PING 確認

eth0 または usb0 を使用してネットワークに接続している場合、ネットワーク通信に awlan0 が使用されない場合があります。確実に awlan0 を使用させる場合は、事前に awlan0 以外のネットワークインターフェースを無効化してください。

6.2.5. 3G

ここでは、Armadillo-IoT に搭載されている 3G モジュール「Sierra Wireless 製 MC8090」の使用方法について説明します。

6.2.5.1. 3G データ通信設定を行う前に

3G データ通信を利用するには、通信事業者との契約が必要です。契約時に通信事業者から貸与された microSIM(UIM カード)と APN 情報を準備します。

Armadillo-IoT の電源が切断されていることを確認してから microSIM(UIM カード)を取り付けてください。

microSIM(UIM カード)は、次のように Armadillo-IoT に取り付けます。

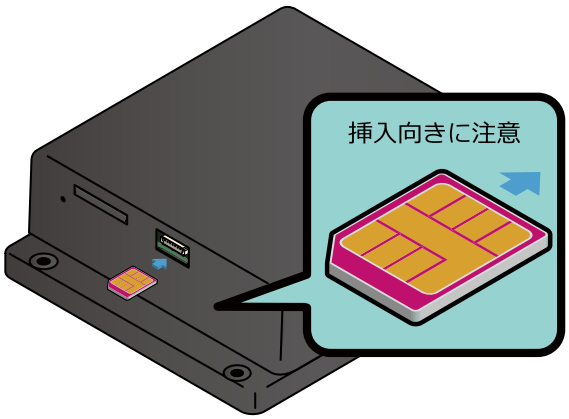


図 6.10 microSIM の取り付け

APN の設定を行うには、次に示す情報が必要です。

- APN
- ユーザー名
- パスワード
- 認証方式(PAP または CHAP)
- PDP Type(IP または PPP)

6.2.5.2. 3G モジュールを制御するソフトウェア

3G モジュールは、TTY デバイスファイル/dev/ttyUSB3 から制御を行うことができます。

/dev/ttyUSB3 から、対話形式での AT コマンドが利用できます。AT コマンドを利用すると、接続先やユーザ名、パスワードの設定などを行うことができます。

Armadillo-IoT のソフトウェアでは、AT コマンドを自動実行するシェルスクリプトがインストールされています。このスクリプトを使用して接続先設定とパケット接続を行うと、通常のネットワークインターフェースとして使用することができるようになります。

6.2.5.3. APN 設定方法

3G モジュールに APN 情報を設定します。APN 設定には 3g-set-ap コマンドを利用します。3g-set-ap コマンドのヘルプは次の通りです。

```
[armadillo ~]# 3g-set-ap
usage: /usr/bin/3g-set-ap [apn] [user] [passwd] [auth_type] [pdp_type]
   apn       ... access point name
   user      ... user name
   passwd    ... pass word
   auth_type ... NON/PAP/CHAP(default)
   pdp_type  ... IP(default)/PPP
```

図 6.11 3g-set-ap コマンドのヘルプ

「表 6.4. APN 情報設定例」の内容に設定する例を、「図 6.12. APN 設定例」に示します。

表 6.4 APN 情報設定例

項目	設定
APN	*[apn]*
ユーザー名	*[user]*
パスワード	*[password]*
認証方式	PAP
PDP Type	IP

```
[armadillo ~]# 3g-set-ap [apn] [user] [password] PAP IP
```

図 6.12 APN 設定例

APN 情報は、3G モジュール内の不揮発性メモリに保存されます。そのため Armadillo-IoT の電源を切断しても再設定を行う必要はありません。

6.2.5.4. 3G インターフェースの有効化、無効化

無効化されている 3G 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 usb0
```

図 6.13 ネットワークインターフェース(usb0)の有効化

有効化されている 3G 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 usb0
```

図 6.14 ネットワークインターフェース(usb0)の無効化

APN 情報が適切に設定されていない場合、3G インターフェースを有効化することができません。

```
[armadillo ~]# ifup usb0
3G connect
error
```

上記のように"error"と表示された場合は、APN の設定を確認してください。

6.2.5.5. 自動で 3G インターフェースを有効化する

Armadillo の起動時に、自動的に 3G インターフェースの有効化を行う方法について説明します。

/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 usb0 ❶
iface lo inet loopback
iface eth0 inet dhcp
iface usb0 inet dhcp
        pre-up 3g-connect
        post-down 3g-disconnect
```

❶ auto 節に usb0 を追加します。

追加後、次回起動時に設定が反映されるようにコンフィグ領域を保存します。

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

Armadillo を再起動すると、自動的に 3G インターフェースが有効化されます。

6.2.5.6. 3G の接続を確認する

3G で正常に通信が可能か確認します。

アットマークテクノの Web サーバーと PING 通信を行います。VPN 接続を利用するなどインターネットに接続できない場合は、ネットワーク内の通信機器に読み替えてください。

```
[armadillo ~]# ping www.atmark-techno.com
```

図 6.15 3G の PING 確認

eth0 または awlan0 を使用してネットワークに接続している場合、ネットワーク通信に usb0 が使用されない場合があります。確実に usb0 を使用させる場合は、事前に usb0 以外のネットワークインターフェースを無効化してください。

6.2.5.7. microSIM から電話番号を取得する

microSIM(UIM カード)から電話番号を取得するには、次のようにコマンドを実行します。

```
[armadillo ~]# 3g-phone-num
[number] ❶
```

❶ 11 桁の電話番号が表示されます。

図 6.16 microSIM からの電話番号取得

microSIM が適切に接続されていない場合、電話番号を取得することができません。

```
[armadillo ~]# 3g-phone-num
error
```

上記のように"error"と表示された場合は、「図 6.10. microSIM の取り付け」を参照して microSIM の接続を確認してください。

6.2.5.8. 3G モジュールの温度を取得する

3G モジュール内蔵の温度センサから温度を取得するには、次のようにコマンドを実行します。

```
[armadillo ~]# 3g-temp
30 ❶
```

❶ 温度は ℃ の単位で表示されます。この例では 30℃ を示しています。

図 6.17 3G モジュールからの温度取得

6.2.5.9. 3G モジュールの高度な設定

本書で紹介していない高度な設定を行うために、直接 AT コマンドを利用する方法について説明します。例として、ATI コマンドを実行し、3G モジュールの情報を表示する手順を次に示します。

手順 6.1 3G モジュールの情報を表示する

1. tip コマンドを実行して/dev/ttyUSB3 に接続します。ボーレートは 115200bps です。

```
[armadillo ~]$ tip -l /dev/ttyUSB3 -s 115200
Connected.
```

2. ATI コマンドを実行すると、3G モジュールの情報が表示されます。

```
ATI
Manufacturer: Sierra Wireless, Incorporated
Model: MC8090
Revision: P1_0_0_23AP R3272 CNSZXD00000132 2014/05/05 17:34:31
IMEI: 013087000546434
IMEI SV: 10
FSN: CEB0704032110
3GPP Release 6
+GCAP: +CGSM,+DS,+ES


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 AT コマンドについては Sierra Wireless 製ドキュメントを参照してください。ドキュメントのダウンロードには、ユーザー登録が必要です。

AirCard/AirPrime UMTS Supported AT Command Reference

http://developer.sierrawireless.com/Resources/Resources/AirPrime/Minicard/2130617_AC_AP_UMTS_Supported_AT_Command_Reference.aspx

AirPrime MC/SL-Series (UMTS/LTE) Extended AT Command Reference

http://developer.sierrawireless.com/Resources/Resources/AirPrime/Minicard/2130616_AP_MCSL_UMTS_LTE_Extended_AT_Command_Ref.aspx

MDM6200(MC8090 のチップセット)に対応した AT コマンドを実行することができます。

6.2.6. 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 6.18 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。

6.2.7. 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 6.19. iptables」のようにコマンド実行してください。

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

図 6.19 iptables

6.2.8. ネットワークアプリケーション

工場出荷イメージで利用することができるネットワークアプリケーションについて説明します。

ATDE と Armadillo のネットワーク設定がデフォルト状態であることを想定して記述しています。ネットワーク設定を変更している場合は適宜読み換えてください。

6.2.8.1. TELNET

ATDE などの PC からネットワーク経由でログインし、リモート操作することができます。ログイン可能なユーザを次に示します。

表 6.5 TELNET でログイン可能なユーザ

ユーザ名	パスワード
guest	(なし)

TELNET を使用して ATDE から Armadillo にリモートログインする場合の例を、次に示します。

```
[ATDE ~]$ telnet 192.0.2.10 ❶
Trying 192.0.2.10...
Connected to 192.0.2.10.
Escape character is '^]'.

atmark-dist v1.36.0 (AtmarkTechno/Armadillo-IoTG-Std)
Linux 2.6.26-at21 [armv5tejl arch]

armadillo-iotg login: guest ❷
[guest@armadillo ~]$
[guest@armadillo ~]$ su ❸
Password: ❹
[root@armadillo ~]#
[root@armadillo ~]# exit ❺
[guest@armadillo ~]$ exit ❻
Connection closed by foreign host.
[ATDE ~]$
```

❶ telnet の引数に Armadillo の IP アドレスを指定します。

❷ "guest"と入力するとログインすることができます。パスワードの入力は不要です。

❸ 特権ユーザーとなる場合には"su"コマンドを実行します。

❹ 特権ユーザーのデフォルトパスワードは"root"です。

❺ 特権トユーザーから guest ユーザーに戻る場合は、"exit"と入力します

❻ telnet を終了するにはもう一度"exit"を入力します

図 6.20 telnet でリモートログイン

6.2.8.2. FTP

ATDE などの PC からネットワーク経由でファイル転送することができます。次に示すユーザでログインすることができます。

表 6.6 ftp でログイン可能なユーザ

ユーザ名	パスワード
ftp	(なし)

ftp を使用して ATDE から Armadillo にファイルを転送する場合の例を、次に示します。

```
[ATDE ~]$ ls -l file
-rw-r--r-- 1 atmark atmark 1048576 Jan 1 12:00 file
[ATDE ~]$ ftp 192.0.2.10 ❶
Connected to 192.0.2.10.
220 localhost FTP server (GNU inetutils 1.4.1) ready.
Name (192.0.2.10:atmark): ftp
331 Guest login ok, type your name as password.
Password: ❷
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub ❸
250 CWD command successful.
ftp> put file ❹
local: file remote: file
200 PORT command sucessful.
150 Opening BINARY mode data connection for 'file'.
226 Transfer complete.
1048576 bytes sent in 0.14 secs (7399.5 kB/s)
ftp> quit ❺
221 Goodbye.
[ATDE ~]$
```

❶ ftp の引数に Armadillo の IP アドレスを指定します。

❷ ftp ユーザにパスワードが設定されていないため Enter キーを入力します。

❸ ファイル転送することができる pub ディレクトリに移動します。

❹ ファイルをアップロードします。ダウンロードする場合は"get"コマンドを使用します。

❺ ftp を終了する場合は"quit"と入力します。

図 6.21 ftp でファイル転送

ATDE から Armadillo にファイルをアップロードすると、/home/ftp/pub/ディレクトリ以下にファイルが作成されています。ダウンロードする場合も、同じディレクトリにファイルを配置してください。

```
[armadillo ~]# cd /home/ftp/pub/
[armadillo /home/ftp/pub]# ls
file
```

図 6.22 Armadillo 上でアップロードされたファイルを確認

6.2.8.3. HTTP サーバー

Armadillo では、HTTP サーバーが動作しています。ATDE などの PC の Web ブラウザから Armadillo の URL (http://[Armadillo の IP アドレス]/ [1] または、http://armadillo-iotg.local/)にアクセスすると、Armadillo のトップページ(index.html)が表示されます。

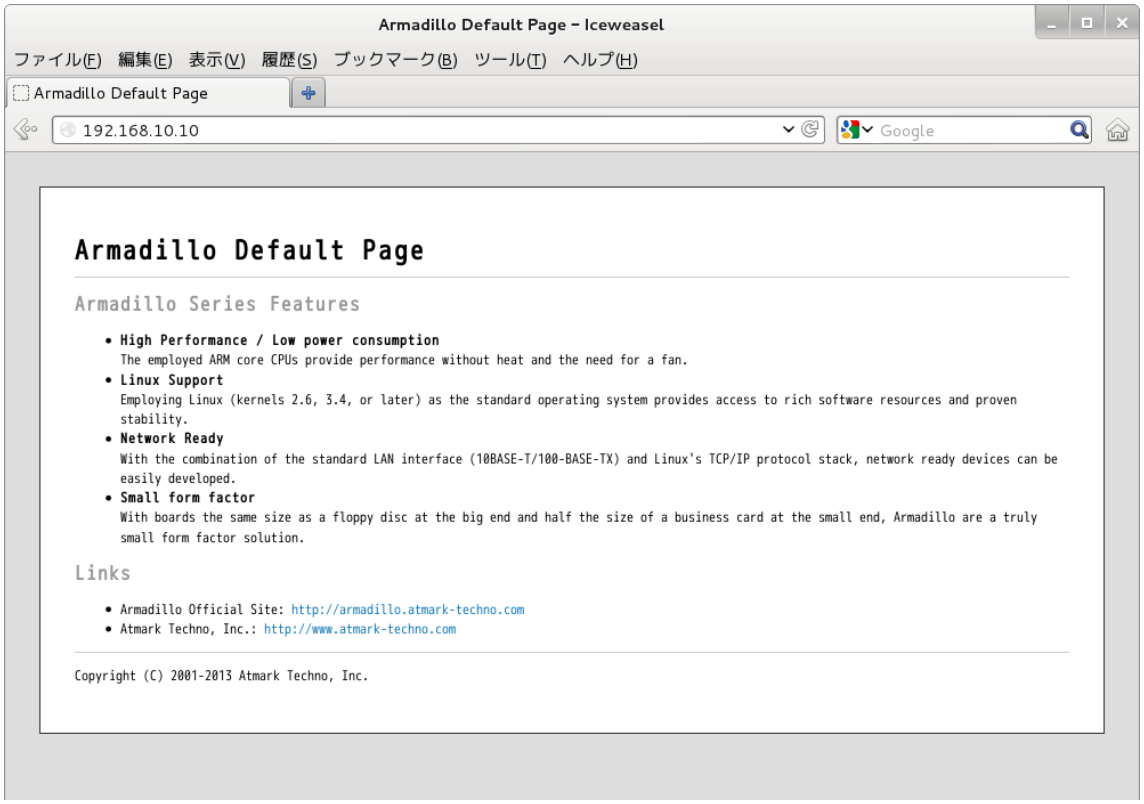


図 6.23 Armadillo トップページ

6.3. ストレージ

Armadillo-IoT 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 6.7 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
SD/SDHC/SDXC カード	/dev/mmcblk*[a]	/dev/mmcblk*p1	SD インターフェース(ベースボード:CON4)
microSD/microSDHC/ microSDXC カード	/dev/mmcblk*[b]	/dev/mmcblk*p1	SD インターフェース(Armadillo-410:CON1)
USB フラッシュメモリ	/dev/sd*[c]	/dev/sd*1	USB ホストインターフェース(ベースボード:CON7)

[a]microSD/microSDHC/microSDXC カードを接続した場合は、認識された順に mmcblk0 mmcblk1 となります。
[b]SD/SDHC/SDXC カード を接続した場合は、認識された順に mmcblk0 mmcblk1 となります。
[c]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、ベースボードに SDHC カードを接続した場合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、SD/SDHC/SDXC カードを SD カードと表記します。

[1] Armadillo の IP アドレスが 192.0.2.10 の場合、http://192.0.2.10/ となります。

SD インターフェース(ベースボード:CON4)と WLAN インターフェース(ベースボード:CON5)は、共通の信号が接続されています。工場出荷状態のソフトウェアでは、デフォルトで SD インターフェースが有効化されています。

「6.2.4.1. WLAN インターフェースの有効化」の手順を実行して WLAN インターフェースが有効化されている場合は、次のように SD インターフェースを有効化してください。

```
[armadillo ~]# sd-awlan-sel sd
select to SD
```

SDXC/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 SDXC/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 6.24 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[2]。FAT32 ファイルシステムの場合は vfat[3]、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。SD 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。

SD スロットに SDHC カードを挿入した状態で「図 6.25. 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 SDHC カードのファイルシステムをマウントします。SD 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。

[2]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。
[3]通常、購入したばかりの SDHC 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。

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

図 6.25 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 6.26 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 SDHC 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 SD カードのパーティションを、2 つに分割する例を「図 6.27. 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/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 6.27 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。SD 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 6.28 EXT3 ファイルシステムの構築

6.4. LED

Armadillo-IoT 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 6.8 LED クラスディレクトリと LED の対応

LED クラスディレクトリ	インターフェース	デフォルトトリガ
/sys/class/leds/led1/	ベースボード:LED3	default-on
/sys/class/leds/led2/	ベースボード:LED2	default-on
/sys/class/leds/led3/	ベースボード:LED4	none
/sys/class/leds/led4/	ベースボード:LED5	none

LED クラスディレクトリ	インターフェース	デフォルトトリガ
/sys/class/leds/yellow	Armadillo-410:LED5	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 6.29 LED を点灯させる

Armadillo-IoT 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 6.30 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 6.31 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 6.9 trigger の種類

設定	説明
none	トリガを設定しません。
mmc0	SD インターフェース(Armadillo-410:CON1)のアクセスランプにします。
mmc1	SD インターフェース(ベースボード:CON4)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。

設定	説明
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 6.32 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 6.33 LED のトリガを表示する

6.5. RTC

Armadillo-IoT には、カレンダ時計(Real Time Clock)が実装されています。電源を切断しても一定時間(平均 300 秒間、最小 60 秒間)時刻を保持することができます

電源が切断されても長時間時刻を保持させたい場合は、RTC 外部バックアップインターフェース(ベースボード:CON13)に外付けバッテリー(対応バッテリー例: CR2032)を接続することができます。

6.5.1. RTC に時刻を設定する

Linux の時刻には、Linux カーネルが管理するシステムクロックと、RTC が管理するハードウェアクロックの 2 種類があります。RTC に時刻を設定するためには、まずシステムクロックを設定します。その後に、ハードウェアクロックをシステムクロックと一致させる手順となります。

システムクロックは、date コマンドを用いて設定します。date コマンドの引数には、設定する時刻を[MMDDhhmmCCYY.ss]というフォーマットで指定します。時刻フォーマットの各フィールドの意味を次に示します。

表 6.10 時刻フォーマットのフィールド

フィールド	意味
MM	月
DD	日(月内通算)
hh	時
mm	分
CC	年の最初の 2 桁(省略可)
YY	年の最後の 2 桁(省略可)
ss	秒(省略可)

2014 年 12 月 19 日 12 時 34 分 56 秒に設定する例を次に示します。

```
[armadillo ~]# date ❶
Sat Jan  1 09:00:00 JST 2000
```

```
[armadillo ~]# date 121912342014.56 ❷
Fri Dec 19 12:34:56 JST 2014
[armadillo ~]# date ❸
Fri Dec 19 12:34:57 JST 2014
```

❶ 現在のシステムクロックを表示します。

❷ システムクロックを設定します。

❸ システムクロックが正しく設定されていることを確認します。

図 6.34 システムクロックを設定

Armadillo-IoT が接続しているネットワーク内にタイムサーバーがある場合は、NTP(Network Time Protocol)クライアントを利用してシステムクロックを設定することができます。

```
[armadillo ~]# ntpclient -h [NTP SERVER] -s
41976 29468.239   24954.0      0.8   14070.6      0.0         0
[armadillo ~]# date
Fri Dec 19 12:34:57 JST 2014
```

システムクロックを設定後、ハードウェアクロックを hwclock コマンドを用いて設定します。

```
[armadillo ~]# hwclock ❶
Sat Jan  1 00:00:00 2000  0.000000 seconds
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
Fri Dec 19 12:35:01 2014  -0.807757
```

❶ 現在のハードウェアクロックを表示します。

❷ ハードウェアクロックを協定世界時(UTC)で設定します。

❸ ハードウェアクロックが UTC で正しく設定されていることを確認します。

図 6.35 ハードウェアクロックを設定

6.6. GPIO

Armadillo-IoT の GPIO は、generic GPIO として実装されています。GPIO クラスディレクトリ以下のファイルによって GPIO の制御を行うことができます。

アドオンインターフェース(ベースボード:CON1, ベースボード:CON2)の GPIO と、GPIO クラスディレクトリの対応を次に示します。

表 6.11 アドオンインターフェースの GPIO ディレクトリ

ピン番号	GPIO クラスディレクトリ
CON1 3 ピン, CON2 24 ピン	/sys/class/gpio/gpio2
CON1 4 ピン, CON2 25 ピン	/sys/class/gpio/gpio3
CON1 5 ピン, CON2 33 ピン	/sys/class/gpio/gpio26
CON1 6 ピン, CON2 32 ピン	/sys/class/gpio/gpio78
CON1 7 ピン, CON2 41 ピン	/sys/class/gpio/gpio118
CON1 8 ピン, CON2 40 ピン	/sys/class/gpio/gpio119
CON1 9 ピン, CON2 7 ピン, CON2 39 ピン	/sys/class/gpio/gpio120
CON1 10 ピン, CON2 8 ピン, CON2 38 ピン	/sys/class/gpio/gpio121
CON1 11 ピン, CON2 50 ピン	/sys/class/gpio/gpio117
CON1 12 ピン, CON2 16 ピン, CON2 37 ピン	/sys/class/gpio/gpio27
CON1 13 ピン, CON2 17 ピン, CON2 36 ピン	/sys/class/gpio/gpio28
CON1 14 ピン, CON2 12 ピン, CON2 18 ピン, CON2 35 ピン	/sys/class/gpio/gpio29
CON1 15 ピン, CON2 13 ピン, CON2 19 ピン, CON2 34 ピン	/sys/class/gpio/gpio30
CON1 16 ピン, CON2 49 ピン	/sys/class/gpio/gpio61
CON1 17 ピン, CON2 48 ピン	/sys/class/gpio/gpio62
CON1 18 ピン, CON2 47 ピン	/sys/class/gpio/gpio63
CON1 19 ピン, CON2 46 ピン	/sys/class/gpio/gpio64
CON1 20 ピン, CON2 20 ピン, CON2 45 ピン	/sys/class/gpio/gpio65
CON1 21 ピン, CON2 21 ピン, CON2 44 ピン	/sys/class/gpio/gpio66
CON1 22 ピン, CON2 22 ピン, CON2 43 ピン	/sys/class/gpio/gpio67
CON1 23 ピン, CON2 23 ピン, CON2 42 ピン	/sys/class/gpio/gpio68
CON1 24 ピン	/sys/class/gpio/gpio0
CON1 25 ピン	/sys/class/gpio/gpio1
CON1 32 ピン	/sys/class/gpio/gpio5
CON1 33 ピン	/sys/class/gpio/gpio4
CON1 42 ピン	/sys/class/gpio/gpio21
CON1 43 ピン	/sys/class/gpio/gpio20
CON1 44 ピン	/sys/class/gpio/gpio19
CON1 45 ピン	/sys/class/gpio/gpio51
CON1 46 ピン	/sys/class/gpio/gpio50
CON1 47 ピン	/sys/class/gpio/gpio49
CON1 48 ピン	/sys/class/gpio/gpio48
CON1 49 ピン	/sys/class/gpio/gpio47
CON1 50 ピン	/sys/class/gpio/gpio25
CON1 51 ピン	/sys/class/gpio/gpio23
CON1 52 ピン	/sys/class/gpio/gpio22
CON1 53 ピン	/sys/class/gpio/gpio24

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

6.6.1. GPIO クラスディレクトリを作成する

/sys/class/gpio/export に GPIO 番号を書き込むことによって、GPIO クラスディレクトリを作成することができます。

アドオンインターフェース(ベースボード:CON1) 24 ピンに対応する GPIO クラスディレクトリを作成する例を次に示します。

```
[armadillo ~]# echo 0 > /sys/class/gpio/export
[armadillo ~]# ls /sys/class/gpio/gpio0/
direction  edge       power/     subsystem@ uevent     value
```

図 6.36 GPIO の入力レベルを取得する

作成済みの GPIO クラスディレクトリを削除するには、/sys/class/gpio/unexport に GPIO 番号を書き込みます。

```
[armadillo ~]# echo 0 > /sys/class/gpio/unexport
[armadillo ~]# ls /sys/class/gpio/gpio0/
ls: /sys/class/gpio/gpio0/: No such file or directory
```

6.6.2. 入出力方向を変更する

GPIO ディレクトリ以下の direction ファイルへ値を書き込むことによって、入出力方向を変更することができます。direction に書き込む有効な値を次に示します。

表 6.12 direction の設定

設定	説明
high	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。出力レベルは HIGH レベルになります。
out	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。出力レベルは LOW レベルになります。
low	out を設定した場合と同じです。
in	入出力方向を INPUT に設定します。入力レベルの取得を行うことができますが設定はできません。

```
[armadillo ~]# echo in > /sys/class/gpio/[GPIO]/direction
```

図 6.37 GPIO の入出力方向を設定する(INPUT に設定)

```
[armadillo ~]# echo out > /sys/class/gpio/[GPIO]/direction
```

図 6.38 GPIO の入出力方向を設定する(OUTPUT に設定)

6.6.3. 入力レベルを取得する

GPIO ディレクトリ以下の value ファイルから値を読み出すことによって、入力レベルを取得することができます。"0"は LOW レベル、"1"は HIGH レベルを表わします。入力レベルの取得は入出力方向が INPUT, OUTPUT のどちらでも行うことができます。入出力方向が OUTPUT の時に読み出される値は、GPIO ピンの状態ではなく、自分が value ファイルに書き込んだ値となります。

```
[armadillo ~]# cat /sys/class/gpio/[GPIO]/value
0
```

図 6.39 GPIO の入力レベルを取得する

6.6.4. 出力レベルを設定する

GPIO ディレクトリ以下の value ファイルへ値を書き込むことによって、出力レベルを設定することができます。"0"は LOW レベル、"0"以外は HIGH レベルを表わします。出力レベルの設定は入出力方向が OUTPUT でなければ行うことはできません。

```
[armadillo ~]# echo 1 > /sys/class/gpio/[GPIO]/value
```

図 6.40 GPIO の出力レベルを設定する

6.7. ユーザースイッチ

Armadillo-IoT 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 6.13 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
ベースボード:SW1	/dev/input/event0	2 (1)
ベースボード:SW2	/dev/input/event1	3 (2)
ベースボード:SW3		4 (3)

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。

6.7.1. 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/event1
Input driver version is 1.0.0
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys-polled"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 3 (2)
```

```
    Event code 4 (3)
Testing ... (interrupt to exit)
Event: time 946704238.665631, type 1 (Key), code 3 (2), value 1
Event: time 946704238.665651, -------------- Report Sync ------------ ❶
Event: time 946704238.785610, type 1 (Key), code 3 (2), value 0
Event: time 946704238.785623, -------------- 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 6.41 ユーザースイッチ: イベントの確認

6.8. 温度センサ

Armadillo-IoT には、温度センサが実装されています。基板周辺温度の取得や、温度変化を監視することができます。

6.8.1. 温度を取得する

/sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0048/temp1_input ファイルを値を読み出すことによって、現在の基板周辺温度を取得することができます。

```
[armadillo ~]# cat /sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0048/temp1_input
30000 ❶
```

❶ 温度はミリ°C の単位で表示されます。この例では 30°C を示しています。

図 6.42 基板周辺温度を取得する

6.8.2. 温度を監視する

thermaltrigger コマンドを利用して、指定した温度になった場合に任意のコマンドを実行させることができます。

thermaltrigger を複数起動することはできません。「9.3.2. thermalmonitor」に示す thermalmonitor コマンドも、内部的に thermaltrigger を起動しています。

thermaltrigger コマンドのヘルプは次の通りです。

```
[armadillo ~]# thermaltrigger
Usage: thermaltrigger -a|-b THRESHOLD COMMAND [ARGS]
Options:
  -a, --above=THRESHOLD
      Execute the program COMMAND when the detected temperature is equal
      to or above the THRESHOLD.
  -b, --below=THRESHOLD
      Execute the program COMMAND when the detected temperature is equal
      to or below the THRESHOLD.
TEMPERATURE: Range: -55000 - 125000
```

図 6.43 thermaltrigger コマンドのヘルプ

温度が 60000 ミリ℃(60℃)以上になった場合に、ベースボード:LED2 を点灯させる例を次に示します。

```
[armadillo ~]# thermaltrigger -a 60000 echo 1 > /sys/class/leds/led2/brightness
```

図 6.44 thermaltrigger コマンド例

thermaltrigger コマンドのログは/var/log/messages ファイルに出力されます。

```
[armadillo ~]# cat /var/log/messages
                :
Dec 19 12:34:59 (none) daemon.info thermaltrigger[1131]: Waiting for
60000 millidegrees Celsius or above. ❶
Dec 19 12:34:59 (none) daemon.info thermaltrigger[1131]: exceeded the
threshold. executing command. ❷
```

❶ 指定した温度(60000 ミリ℃)以上になることを待機します。

❷ 指定した温度に達したのでコマンドを実行します。

6.9. AD コンバーター

Armadillo-IoT には、AD コンバーターが実装されています。電源電圧の取得や、電圧の変化を監視することができます。

6.9.1. 電源電圧を取得する

電源電圧は、分圧されて AD コンバーターへ入力されています。電源電圧を取得するためには、まず AD コンバーターへの入力電圧を取得する必要があります。

/sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0054/value ファイルを値を読み出すことによって、現在の AD コンバーターへの入力電圧を取得することができます。

```
[armadillo ~]# cat /sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0054/value
1958 ❶
```

❶ 電圧は mV の単位で表示されます。この例では 1.958V を示しています。

図 6.45 AD コンバーターへの入力電圧を取得する

AD コンバーターへの入力電圧から、電源電圧を求める計算式を次に示します。

```
[電源電圧] = [AD コンバーターへの入力電圧] × (200 + 39) ÷ 39
```

図 6.46 電源電圧の計算式

「図 6.45. AD コンバーターへの入力電圧を取得する」を例にとると、AD コンバーターへの入力電圧 1.958V から、電源電圧が約 11.999V であることを求めることができます。

expr コマンドを利用して、次のように電源電圧を表示することができます。

```
[armadillo ~]# adc=`cat /sys/devices/platform/i2c-gpio.3/i2c-adapter/
i2c-3/3-0054/value`
[armadillo ~]# expr $adc \* \( 200 + 39 \) / 39
11999
```

6.9.2. 電源電圧を監視する

vintrigger コマンドを利用して、電源電圧が指定した電圧になった場合に任意のコマンドを実行させることができます。

vintrigger を複数起動することはできません。「9.3.3. vinmonitor」に示す vinmonitor コマンドも、内部的に vintrigger を起動しています。

vintrigger コマンドのヘルプは次の通りです。

```
[armadillo ~]# vintrigger
Usage: vintrigger -o|-u VOLTAGE COMMAND [ARGS]
Options:
  -o, --over=VOLTAGE
      Execute the program COMMAND when the detected voltage is equal
      to or over the VOLTAGE.
  -u, --under=VOLTAGE
      Execute the program COMMAND when the detected voltage is equal
      to or unver the VOLTAGE.
VOLTAGE: Range: 0 - 20223
```

図 6.47 vintrigger コマンドのヘルプ

電源電圧が 11000mV(11V)以下になった場合に、ベースボード:LED2 を点灯させる例を次に示します。

```
[armadillo ~]# vintrigger -u 11000 echo 1 > /sys/class/leds/led2/brightness
```

図 6.48 vintrigger コマンド例

vintrigger コマンドのログは/var/log/messages ファイルに出力されます。

```
[armadillo ~]# cat /var/log/messages
                :
Dec 19 12:34:57 (none) daemon.info vintrigger[1173]: waiting for an
under range alert (11000 mV). ❶
Dec 19 12:34:57 (none) daemon.info vintrigger[1173]: exceeded the limit.
executing command. ❷
```

❶ 指定した電圧(11000mV)以下になることを待機します。

❷ 指定した電圧に達したのでコマンドを実行します。

6.10. Armadillo-IoT RS232C アドオンモジュール RS00

Armadillo-IoT RS232C アドオンモジュール RS00(以降、RS232C アドオンモジュールと記載します)は RS232C レベルのシリアルポートが 1 ポート搭載されています。RS232C アドオンモジュールのシリアルポートのデバイスドライバは、TTY デバイスとして実装されているため TTY デバイスファイルから制御を行うことができます。

RS232C アドオンモジュールを接続するアドオンインターフェースと、TTY デバイスファイルの対応を次に示します。

表 6.14 アドオンインターフェースと TTY デバイスファイル

アドオンインターフェース	TTY デバイスファイル
ベースボード:CON1	/dev/ttymxc3
ベースボード:CON2	/dev/ttymxc0

工場出荷状態の開発セットは、ベースボード:CON1 に RS232C アドオンモジュールが接続されています。

RS232C アドオンモジュールが接続されているアドオンインターフェースは、Linux カーネルの起動ログで確認することができます。ベースボード:CON1 に接続されている場合は次のように出力されます。

```
Atmark Techno RS232C board detected at CON1(0x0001).
```

6.10.1. シリアルコンソールとして使用する

RS232C アドオンモジュールのシリアルインターフェースをシリアルコンソールとして使用します。ATDE のシリアル通信ソフトウェア(minicom)を用いることで、シリアル経由で Armadillo にログインすることができます。

アドオンインターフェース(ベースボード:CON1)に接続した RS232C アドオンモジュールのシリアルインターフェースをシリアルコンソールとして使用する手順を次に示します。

手順 6.2 シリアルコンソールとして使用

1. ATDE で minicom を起動します。シリアルデバイスには/dev/ttyS0 を指定します。

```
[ATDE ~]$ minicom -o -w -D /dev/ttyS0
```

2. Armadillo-IoT で getty を起動します。シリアルデバイスには ttymxc3[4]を指定します。/etc/inittab の設定を有効にするためには、プロセス ID が 1 である init プロセスに SIGHUP シグナルを送ります。

```
[armadillo ~]# echo ::respawn:/sbin/getty -L 115200 ttymxc3 >> /etc/inittab
[armadillo ~]# kill -SIGHUP 1
```

ATDE の minicom にログインプロンプトが表示されます。ユーザー「guest」でログインすることができます。

以下のように/etc/securetty に端末(シリアルデバイス)を登録すると、特権ユーザー「root」でログインすることが可能になります。

[4] /dev/を指定する必要はありません。

```
[armadillo ~]# echo ttymxc3 >> /etc/securetty
```

6.11. Armadillo-IoT BLE アドオンモジュール BT00

Armadillo-IoT BLE アドオンモジュール BT00(以降、BLE アドオンモジュールと記載します)は Microchip 製 RN4020 が搭載されています。Bluetooth(R) version 4.1 に対応しており、Bluetooth Low Energy 4.1 プロトコルスタックが内蔵されています。

BLE アドオンモジュールは、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。BLE アドオンモジュールを接続するアドオンインターフェースと、TTY デバイスファイルの対応を次に示します。

表 6.15 アドオンインターフェースと TTY デバイスファイル

アドオンインターフェース	TTY デバイスファイル
ベースボード:CON1	/dev/ttymxc3
ベースボード:CON2	/dev/ttymxc0

BLE アドオンモジュールが接続されているアドオンインターフェースは、Linux カーネルの起動ログで確認することができます。ベースボード:CON2 に接続されている場合は次のように出力されます。

```
Atmark Techno Bluetooth Low Energy board detected at CON2(0x0001).
```

6.11.1. 設定情報を取得する

BLE アドオンモジュールを制御する例として、RN4020 の設定情報の取得を行います。

アドオンインターフェース(ベースボード:CON2)に接続した BLE アドオンモジュールに搭載されている RN4020 の設定情報を取得する手順を次に示します。

手順 6.3 設定情報の取得

1. tip コマンドを実行して/dev/ttymxc0 に接続します。ボーレートは 115200bps です。

```
[armadillo ~]$ tip -l /dev/ttymxc0 -s 115200
Connected.
```

2. D (Dump configuration)コマンドを実行すると、RN4020 の設定情報が表示されます。

```
D
BTA=001EC01BBF7B
Name=RN4020_BF7B
Role=Peripheral
Connected=no
```

```
Bonded=no
Server Service=80000000
```

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、RN4020 の詳細な情報については Microchip 製ドキュメントを参照してください。

Data Sheets

http://www.microchip.com/TechDoc.aspx?type=datasheet&product=RN4020

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-IoT 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-IoT ゲートウェイ スタンダードモデル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-2.6.26-[VERSION].tar.gz)に含まれる `arch/arm/configs/armadillo_iotg_std_defconfig` です。

`armadillo_iotg_std_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
HIGH_RES_TIMERS	High Resolution Timer Support
PREEMPT	Preemptible Kernel
AEABI	Use the ARM EABI to compile the kernel

8.2. デフォルト起動オプション

工場出荷状態の Armadillo-IoT 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(ベースボード:CON9)を、ボーレートに 115200bps を指定します。

8.3. Linux ドライバ一覧

Armadillo-IoT 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-IoT ゲートウェイ スタンダードモデル

Armadillo-IoT ゲートウェイ スタンダードモデル 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。ユーザーオリジナルのアドオンボードを利用する場合などに変更を加えます。

関連するソースコード

arch/arm/mach-mx25/armadillo_iotg_std.c
arch/arm/mach-mx25/armadillo_iotg_std_addon.c
arch/arm/mach-mx25/armadillo_iotg_std_gpio.c
arch/arm/mach-mx25/clock.c

arch/arm/mach-mx25/cpu.c
arch/arm/mach-mx25/devices.c
arch/arm/mach-mx25/dma.c
arch/arm/mach-mx25/gpio.c
arch/arm/mach-mx25/mm.c
arch/arm/mach-mx25/pm.c
arch/arm/mach-mx25/serial.c
arch/arm/mach-mx25/system.c
arch/arm/mach-mx25/usb_dr.c
arch/arm/mach-mx25/usb_h2.c

カーネルコンフィギュレーション

```
System Type  --->
    ARM system type (Freescale MXC/iMX-based)  --->
    Freescale MXC Implementations  --->
        MXC/iMX System Type (MX25-based)  --->                          <ARCH_MX25>
        MX25 Options  --->
            [*] Support Atmark Techno Armadillo-410                <MACH_ARMADILLO410>
            Armadillo-400 Board options  --->
                Armadillo-410 CON2 extension board (Armadillo-IoTG Std Base board)  --->
                                                          <ARMADILLO410_CON2_AIOTG_STD>
                [*] Addon Module Auto Delect              <AIOTG_STD_ADDON_AUTO_DETECT>
                          :
```

ユーザーオリジナルのアドオンボードを利用する場合には、アットマークテクノ製アドオンボードの自動検出機能を示す「Addon Module Auto Delect」の選択を外します。

「Addon Module Auto Delect」の選択を外すと、アドオンインターフェースで実現することのできる機能を選択することができます。コンフィギュレーションで機能を明示的に割り当てない全てのピンには GPIO 機能が割り当てられます。

```
    [ ] Addon Module Auto Delect
    [ ]   Enable UART1 at CON1/CON2 (NEW)
    [ ]   Enable UART3 at CON1/CON2 (NEW)
    [ ]   Enable UART4 at CON1/CON2 KPP Pad (NEW)
    [ ]     Enable UART4 at CON1 LCD Pad (NEW)
    [ ]   Enable UART5 at CON1/CON2 (NEW)
    [ ]   Enable I2C2 at CON1/CON2 (NEW)
    [*]   Enable I2C3 at CON1_24/CON1_25 (NEW)
    [ ]   Enable SPI2 at CON1 (NEW)
    [ ]   Enable SPI3 at CON1/CON2 (NEW)
    [ ]   Enable CAN2 at CON1/CON2 (NEW)
    [ ]   Enable one wire at CON1_6/CON2_32 (NEW)
    [ ]   Enable PWM1 at CON1_5/CON2_33 (NEW)
    [ ]   Enable PWM4 at CON1_3/CON2_24 (NEW)
```

機能が割り当てられるピンを調べるには、各項目のヘルプを参照します。次に示す例では、「Enable UART1 at CON1/CON2」を選択した場合に、アドオンインターフェース(ベースボード:CON1) 7 ピンとアドオンインターフェース(ベースボード:CON2) 41 ピンに UART1 の RXD 信号が、アドオンインターフェース(ベースボード:CON1) 8 ピンと アドオンインターフェース(ベースボード:CON2) 40 ピンに UART1 の TXD 信号が割り当てられることが確認できます。

```
---------------------- Enable UART1 at CON1/CON2 ----------------------
CONFIG_AIOTG_STD_UART1:

Enable UART1 at CON1/CON2

CON1_7/CON2_41: UART1_RXD
CON1_8/CON2_40: UART1_TXD

Symbol: AIOTG_STD_UART1 [=n]
Prompt: Enable UART1 at CON1/CON2
  Defined at arch/arm/mach-mx25/Kconfig:1273
  Depends on: ARCH_MXC && ARCH_MX25 && (MACH_ARMADILLO410 || MACH_ARMAD
  Location:
    -> System Type
      -> Freescale MXC Implementations
        -> MX25 Options
          -> Armadillo-400 Board options
            -> Addon Module Auto Delect (AIOTG_STD_ADDON_AUTO_DETECT [=
  Selects: AIOTG_STD_CON1_7_CON2_41_UART1_RXD && AIOTG_STD_CON1_8_CON2_
--------------------------------------------------------------------------
                                < Exit >
```

8.3.2. フラッシュメモリ

Armadillo-IoT 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/mtdcore.c
drivers/mtd/mtdchar.c
drivers/mtd/mtdblock.c
drivers/mtd/mtdpart.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
drivers/mtd/maps/armadillo.c

デバイスファイル

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

カーネルコンフィギュレーション

```
Device Drivers  --->
  <*> Memory Technology Device (MTD) support  --->                              <MTD>
      -*-   MTD partitioning support                                      <MTD_PARTITIONS>
      [*]     Command line partition table parsing                      <MTD_CMDLINE_PARTS>
            *** User Modules And Translation Layers ***
      <*>   Direct char device access to MTD devices                          <MTD_CHAR>
      -*-   Common interface to block layer for MTD 'translation layers      <MTD_BLKDEVS>
      <*>   Caching block device access to MTD devices                        <MTD_BLOCK>
      RAM/ROM/Flash chip drivers  --->
          -*- Detect flash chips by Common Flash Interface (CFI) probe          <MTD_CFI>
          <*> Support for Intel/Sharp flash chips                        <MTD_CFI_INTELEXT>
      Mapping drivers for chip access  --->
          <*> CFI FLash device mapped on Armadillo board                  <MTD_ARMADILLO>


    <*> Memory Technology Device (MTD) support  --->                          <CONFIG_MTD>
        [*] Command line partition table parsing                  <CONFIG_MTD_CMDLINE_PARTS>
        [*] Read-only switching user interface support                  <CONFIG_MTD_RO_IF>
            *** User Modules And Translation Layers ***
        <*> Direct char device access to MTD devices                    <CONFIG_MTD_CHAR>
        -*- Common interface to block layer for MTD 'trnslation...     <CONFIG_MTD_BLKDEVS>
        <*> Caching block device access to MTD devices                  <CONFIG_MTD_BLOCK>
            RAM/ROM/Flash chip drivers  --->
            <*> Detect flash chips by Common Flash Interface...          <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-IoT のシリアルは、i.MX257 の UART(Universal Asynchronous Receiver/Transmitter)を利用しています。

Armadillo-IoT のシリアルは、最大 5 ポートを利用することができます。標準状態では、UART2(ベースボード:CON9)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[1]

関連するソースコード

drivers/serial/serial_core.c
drivers/serial/mxc_uart.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

[1]DMA 転送を有効にした場合の最大ボーレートです。標準状態では DMA 転送は無効になっています。

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Armadillo-400 Board options  --->
                [ ] Addon Module Auto Delect           <AIOTG_STD_ADDON_AUTO_DETECT> ❶
                [*]   Enable UART1 at CON1/CON2                  <AIOTG_STD_UART1> ❷
                [*]     Enable UART1 HW flow control at CON1/CON2
                                                      <AIOTG_STD_UART1_HW_FLOW> ❷
                [*]   Enable UART3 at CON1/CON2                  <AIOTG_STD_UART3> ❷
                [*]     Enable UART3 HW flow control at CON1/CON2
                                                      <AIOTG_STD_UART3_HW_FLOW> ❷
                [*]   Enable UART4 at CON1/CON2 KPP Pad        <AIOTG_STD_UART4_KPP> ❷
                [*]     Enable UART4 HW flow control at CON1/CON2 KPP Pad
                                                    <AIOTG_STD_UART4_HW_FLOW_KPP> ❷
                [*]   Enable UART5 at CON1/CON2                  <AIOTG_STD_UART5> ❷
                [*]     Enable UART5 HW flow control at CON1/CON2
                                                      <AIOTG_STD_UART5_HW_FLOW> ❷
Device Drivers  --->
    Character devices  --->
        Serial drivers  --->
            <*> MXC Internal serial port support                        <SERIAL_MXC>
            [*]   Support for console on a MXC/MX27/MX21 Internal serial port
                                                                  <SERIAL_MXC_CONSOLE>
```

❶ 標準状態では有効化されています。

❷ 標準状態では無効化されています。

8.3.4. Ethernet

Armadillo-IoT 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/mx25_fec.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 (10 or 100Mbit)  --->                           <NET_ETHERNET>
            -*-   Generic Media Independent Interface device support          <MII>
            <*>   FEC ethernet controller for i.MX25                    <MX25_FEC>
```

8.3.5. 3G

Armadillo-IoT には、Sierra Wireless 製 MC8090 が搭載されています。MC8090 は、USB Host2 に接続されています。

機能

リンク検出サポート

ネットワークデバイス

usb0[2]

デバイスファイル

/dev/ttyUSB0[3]
/dev/ttyUSB1[3]
/dev/ttyUSB2[3]
/dev/ttyUSB3[3]
/dev/ttyUSB4[3]

関連するソースコード

drivers/net/usb/sierra_net.c
drivers/net/usb/usbnet.c
drivers/usb/serial/sierra.c
drivers/usb/serial/usb-serial.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Network device support  --->                                    <NETDEVICES>
        USB Network Adapters  --->
            <*> Multi-purpose USB Networking Framework                  <USB_USBNET>
                <*>   USB-to-WWAN Driver for Sierra Wireless modems     <USB_SIERRA_NET>
    [*] USB support  --->                                               <USB_SUPPORT>
        <*>   USB Serial Converter support  --->                        <USB_SERIAL>
            <*>   USB Sierra Wireless Driver                <USB_SERIAL_SIERRAWIRELESS>
```

8.3.6. SD ホスト

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

[2]USB Ethernet などを接続している場合は、番号が異なる可能性があります。
[3]USB シリアルなどを接続している場合は、番号が異なる可能性があります。

Armadillo-IoT では、最大 2 ポートを利用することができます。SD インターフェース(Armadillo-410:CON1)が eSDHC1 を、SD インターフェース(ベースボード:CON4)が eSDHC2 を利用しています。

機能

カードタイプ(Armadillo-410:CON1): microSD/microSDHC/microSDXC
カードタイプ(ベースボード:CON4): SD/SDHC/SDXC
バス幅: 4bit
スピードモード: Default Speed(24MHz), High Speed(48MHz)
カードディテクトサポート(ベースボード:CON4 のみ)
ライトプロテクトサポート(ベースボード:CON4 のみ)

デバイスファイル

メモリカードの場合は、カードを認識した順番で/dev/mmcblkN (N は'0'または'1')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/host/mx_sdhci.c

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Armadillo-400 Board options  --->
                eSDHC2 select function  (SD)  --->
                (X) SD                                              <AIOTG_STD_ESDHC2_SD>
                ( ) AWLAN                                        <AIOTG_STD_ESDHC2_SD_AWLAN>
Device Drivers  --->
    <*> MMC/SD/SDIO card support  --->                                                  <MMC>
        <*>   MMC block device driver                                                <MMC_BLOCK>
        [*]     Use bounce buffer for simple hosts                                <MMC_BLOCK_BOUNCE>
        <*>   Freescale i.MX enhanced Secure Digital Host Controller Interface support
                                                                                    <MMC_IMX_ESDHCI>
```

SD インターフェース(ベースボード:CON4)と WLAN インターフェース(ベースボード:CON5)は、共通の信号が接続されています。「eSDHC2 select function」では、デフォルトでどちらの信号に接続するかを選択します。

8.3.7. USB ホスト

Armadillo-IoT 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-IoT では、USB ホストインターフェース(ベースボード:CON7)の OTG ポートのみを利用することができます。Host2 ポートは「8.3.5. 3G」に示す MC8090 に接続されています。

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(Host2): 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。I/O デバイスの場合は、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/usb/host/ehci-arc.c
drivers/usb/host/ehci-hcd.c
drivers/usb/host/ehci-hub.c
drivers/usb/host/ehci-q-iram.c
drivers/usb/host/ehci-mem-iram.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>
        [*]       Support for Freescale controller                        <USB_EHCI_ARC>
        [*]         Support for Host2 port on Freescale controller      <USB_EHCI_ARC_H2>
        [*]         Support for DR host port on Freescale controller   <USB_EHCI_ARC_OTG>
        [*]         Use IRAM for USB                                    <USB_STATIC_IRAM>
        (2048)        Size of 1 qTD's buffer                  <USB_STATIC_IRAM_TD_SIZE>
```

8.3.8. リアルタイムクロック

Armadillo-IoT には、セイコーインスツル(SII)製 S-35390A が搭載されています。 S-35390A は、I2C-GPIO3 (I2C ノード: 3-0030) に接続されています。

機能

アラーム割り込みサポート

デバイスファイル

/dev/rtc
/dev/rtc0

関連するソースコード

drivers/rtc/rtc-s35390a.c
drivers/rtc/class.c
drivers/rtc/rtc-dev.c
drivers/rtc/rtc-sysfs.c

カーネルコンフィギュレーション

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

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

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

```
[armadillo ~]# cat /proc/interrupts | grep rtc0 ❶
 79:          0    MXC_GPIO  rtc0
[armadillo ~]# echo +60 > /sys/class/rtc/rtc0/wakealarm ❷
[armadillo ~]# cat /sys/class/rtc/rtc0/wakealarm ❸
1418244060
[armadillo ~]# cat /sys/class/rtc/rtc0/since_epoch ❹
1418244061
[armadillo ~]# cat /proc/interrupts | grep rtc0 ❺
 79:          1    MXC_GPIO  rtc0
```

❶ アラーム割り込みの発生回数を確認します。この例では 0 回です。

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

❸ アラーム割り込みの発生時刻(UNIX エポックからの経過秒数)を確認します。この例では 1418244060 秒です。

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

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

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

8.3.9. 温度センサ

Armadillo-IoT には、NXP セミコンダクターズ製 LM75B が搭載されています。

LM75B は、I2C-GPIO3 (I2C ノード: 3-0048) に接続されています。LM75B の OS(Overtemperature Shutdown output) 信号は、GPIO(GPIO ディレクトリ:/sys/class/gpio/TEMP_ALERT_N/)に接続されているため、事前に指定した温度を超えたかどうか確認することができます。

機能

分解能: 0.125℃
測定範囲: -55℃ ～ +125℃

sysfs ディレクトリ

/sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0048/

関連するソースコード

drivers/i2c/chips/lm75b.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> I2C support  --->                                                   <I2C>
        Miscellaneous I2C Chip support  --->
            <*> NXP Semiconductors LM75B                                <SENSORS_LM75B>
```

8.3.10. AD コンバーター

Armadillo-IoT には、Texas Instruments 製 ADC081C021 が搭載されています。

ADC081C021 は、I2C-GPIO3 (I2C ノード: 3-0054) に接続されています。ADC081C021 の ALERT 信号は、GPIO(GPIO ディレクトリ:/sys/class/gpio/VIN_ALERT_N/)に接続されているため、事前に指定した電圧を超えたかどうか確認することができます。

機能

分解能: 8bit
測定範囲: 0V ～ 3.3V(ADC081C021 の電源電圧)

sysfs ディレクトリ

/sys/devices/platform/i2c-gpio.3/i2c-adapter/i2c-3/3-0054/

関連するソースコード

drivers/i2c/chips/ti-adc081c.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> I2C support  --->                                                   <I2C>
        Miscellaneous I2C Chip support  --->
            <*> Texas Instruments ADC081C021/027                        <TI_ADC081C>
```

8.3.11. LED

Armadillo-IoT 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。

Armadillo-410 には、LED5 が実装されています。 ベースボードには、LED2～LED5 が実装されています。 Linux カーネルでは、Armadillo-410 に実装された LED を色の名前(yellow)と命名して区別しています。

sysfs LED クラスディレクトリ

/sys/class/leds/led1
/sys/class/leds/led2
/sys/class/leds/led3
/sys/class/leds/led4
/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.c
drivers/leds/ledtrig-default-on.c
drivers/leds/ledtrig-heartbeat.c
drivers/leds/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>
```

8.3.12. ユーザースイッチ

Armadillo-IoT 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, gpio-keys-polled)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
ベースボード:SW1	KEY_1	2
ベースボード:SW2	KEY_1	3
ベースボード:SW3	KEY_1	4

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
ベースボード:SW1	gpio-keys
ベースボード:SW2	gpio-keys-polled
ベースボード:SW3	

SW2 と SW3 は、GPIO エクスパンダーに接続されています。Armadillo-IoT では、GPIO エクスパンダーの割り込み信号を利用していないため、イベント割り込みの対応が必須である gpio-keys を利用することができません。そのため、イベントをポーリングする gpio-keys-polled を利用しています。

デバイスファイル

ユーザースイッチ	デバイスファイル
ベースボード:SW1	/dev/input/event0[a]
ベースボード:SW2	/dev/input/event1[a]
ベースボード:SW3	

[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/input.c
drivers/input/input-polldev.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>
              *** Userland interfaces ***
        <*>   Event interface                                                <INPUT_EVDEV>
              *** Input Device Drivers ***
        [*]   Keyboards  --->                                              <INPUT_KEYBOARD>
            <*>   GPIO Buttons                                              <KEYBOARD_GPIO>
            <*>   Polled GPIO buttons                                <KEYBOARD_GPIO_POLLED>
```

8.3.13. I2C

Armadillo-IoT の I2C インターフェースは、i.MX257 の I2C(Inter IC Module) を利用します。 また、GPIO を利用した I2C バスドライバ(i2c-gpio)を利用することで、I2C バスを追加することができます。

Armadillo-IoT で利用している I2C バスと、接続される I2C デバイスを次に示します。

表 8.5 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
0(I2C1)	0x54	MC34704 マルチチャンネルパワーマネジメント IC
3(I2C-GPIO3)	0x30 (0x31〜0x37 も予約済み)	S-35390A リアルタイムクロック
	0x48	LM75B 温度センサ
	0x54	ADC081C021 コンバーター
	0x70	PCA9538 GPIO エクスパンダー
4(I2C-GPIO4)	0x50	M24C01-W EEPROM[a]
	0x51	M24C01-W EEPROM[b]

[a]アドオンインターフェース(ベースボード:CON1)にアドオンモジュールを接続した場合。
[b]アドオンインターフェース(ベースボード:CON2)にアドオンモジュールを接続した場合。

Armadillo-IoT の標準状態では、CONFIG_I2C_CHARDEV が有効となっているためユーザードライバで I2C デバイスを制御することができます。ユーザードライバを利用する場合は、Linux カーネルで I2C デバイスに対応するデバイスドライバを無効にする必要があります。

機能

最大転送レート: 400kbps (I2C1, I2C2, I2C3)

デバイスファイル

/dev/i2c-0 (I2C1)
/dev/i2c-3 (I2C-GPIO3)
/dev/i2c-4 (I2C-GPIO4)

関連するソースコード

drivers/i2c/busses/i2c-gpio.c
drivers/i2c/busses/mxc_i2c.c
drivers/i2c/i2c-core.c
drivers/i2c/i2c-dev.c
drivers/i2c/i2c-boardinfo.c

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Device options  --->
                [*] Enable I2C1 module                          <I2C_MXC_SELECT1>
                -*- Enable I2C2 module                          <I2C_MXC_SELECT2> ❶
                -*- Enable I2C3 module                          <I2C_MXC_SELECT3> ❶
            Armadillo-400 Board options  --->
                [ ] Addon Module Auto Delect          <AIOTG_STD_ADDON_AUTO_DETECT> ❷
                [*]   Enable I2C2 at CON1/CON2                  <AIOTG_STD_I2C2> ❶
                [ ]   Enable I2C3 at CON1_24/CON1_25     <AIOTG_STD_I2C3_CON1_24_25>
                [*]     Enable I2C3 at CON1_51/CON1_52   <AIOTG_STD_I2C3_CON1_51_52> ❶
Device Drivers  --->
    <*> I2C support  --->                                                  <I2C>
        <*>   I2C device interface                                  <I2C_CHARDEV>
        [*]   Autoselect pertinent helper modules                  <I2C_HELPER_AUTO>
            I2C Hardware Bus support  --->
                <*> GPIO-based bitbanging I2C                         <I2C_GPIO>
                <*> MXC I2C support                                    <I2C_MXC>
```

❶ 標準状態では無効化されています。

❷ 標準状態では有効化されています。

8.3.14. SPI

Armadillo-IoT の SPI インターフェースは、i.MX257 の CSPI(Configurable Serial Peripheral Interface)を利用します。

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

関連するソースコード

drivers/spi/mxc_spi.c
drivers/spi/spi.c
drivers/spi/spi_bitbang.c
drivers/spi/spidev.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] SPI support  --->                                                  <SPI>
            *** SPI Master Controller Drivers ***
        -*-   Bitbanging SPI master                                <SPI_BITBANG>
        <*>   MXC CSPI controller as SPI Master                       <SPI_MXC>
            *** SPI Protocol Masters ***
        < >   User mode SPI device driver support                   <SPI_SPIDEV>
```

8.3.15. ウォッチドッグタイマー

Armadillo-IoT 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。標準状態でのタイムアウト時間は 10 秒です。Linux カーネルでは、i.MX257 の GPT(General Purpose Timer)割り込みを利用して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

arch/arm/plat-mxc/wdog.c
arch/arm/plat-mxc/time.c

8.3.16. 1-Wire

Armadillo-IoT の 1-Wire は、i.MX257 の 1-Wire(1-Wire Module) を利用します。

Armadillo-IoT の標準状態では、1-Wire を利用することができません。1-Wire を利用するには、カーネルコンフィギュレーションしカーネルイメージを変更する必要があります。

関連するソースコード

drivers/w1/masters/mxc_w1.c

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Armadillo-400 Board options  --->
                [ ] Addon Module Auto Delect          <AIOTG_STD_ADDON_AUTO_DETECT> ❶
                [*]   Enable one wire at CON1_6/CON2_32          <AIOTG_STD_W1> ❷
Device Drivers  --->
    <*> Dallas's 1-wire support  --->                                          <W1> ❷
        1-wire Bus Masters  --->
            <*> Freescale MXC driver for 1-wire                  <W1_MASTER_MXC> ❷
```

❶ 標準状態では有効化されています。

❷ 標準状態では無効化されています。

8.3.17. PWM

Armadillo-IoT の PWM は、i.MX257 の PWM(Pulse-Width Modulator) を利用します。

Armadillo-IoT の標準状態では、PWM を利用することができません。PWM を利用するには、カーネルコンフィギュレーションしカーネルイメージを変更する必要があります。

関連するソースコード

drivers/mxc/pwm/mxc_pwm.c
drivers/mxc/pwm/mxc_pwm_class.c

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Armadillo-400 Board options  --->
                [ ] Addon Module Auto Delect          <AIOTG_STD_ADDON_AUTO_DETECT> ❶
                [*]   Enable PWM1 at CON1_5/CON2_33              <AIOTG_STD_PWM1> ❷
                [*]   Enable PWM2 at CON1_24                     <AIOTG_STD_PWM2> ❷
                [*]   Enable PWM3 at CON1_25                     <AIOTG_STD_PWM3> ❷
                [*]   Enable PWM4 at CON1_3/CON2_24              <AIOTG_STD_PWM4> ❷
Device Drivers  --->
    MXC support drivers  --->
        MXC PWM support  --->
            [*] Enable PWM driver                                       <MXC_PWM> ❷
            [*] MXC PWM class support                             <MXC_PWM_CLASS> ❷
```

❶ 標準状態では有効化されています。

❷ 標準状態では無効化されています。

8.3.18. CAN

Armadillo-IoT の CAN は、i.MX257 の FlexCAN(Controller Area Network) を利用します。

Armadillo-IoT の標準状態では、CAN を利用することができません。CAN を利用するには、カーネルコンフィギュレーションしカーネルイメージを変更する必要があります。

関連するソースコード

drivers/net/can/flexcan/

カーネルコンフィギュレーション

```
System Type  --->
    Freescale MXC Implementations  --->
        MX25 Options  --->
            Device options  --->
                -*- Enable FlexCAN1 module                          <FLEXCAN_SELECT1> ❶
                -*- Enable FlexCAN2 module                          <FLEXCAN_SELECT2> ❶
            Armadillo-400 Board options  --->
                [ ] Addon Module Auto Delect           <AIOTG_STD_ADDON_AUTO_DETECT> ❷
                [*]   Enable CAN1 at CON1                            <AIOTG_STD_CAN1> ❶
                [*]   Enable CAN2 at CON1/CON2                       <AIOTG_STD_CAN2> ❶
Networking  --->
    <*>   CAN bus subsystem support  --->                                      <CAN> ❶
        <*>   Raw CAN Protocol (raw access with CAN-ID filtering)          <CAN_RAW> ❶
        <*>   Broadcast Manager CAN Protocol (with content filtering)      <CAN_BCM> ❶
        CAN Device Drivers  --->
            <*> Freescale FlexCAN                                      <CAN_FLEXCAN> ❶
```

❶ 標準状態では無効化されています。

❷ 標準状態では有効化されています。

9. ユーザーランド仕様

本章では、工場出荷状態の Armadillo-IoT 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-IoT 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-IoT 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-IoT 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-IoT 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-IoT 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-IoT 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
S03udevd	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
S06checkftp	FTP が利用するファイルやライブラリの配置、パーミッションの設定をします
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,S60inetd	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
S70lighttpd, S71avahi	ネットワークデーモンを起動します
S99misc	各種設定や初期化を行います
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。

Armadill-IoT では、システム起動時に自動的に各種状態監視アプリケーションを起動するために利用しています。出荷状態では状態監視アプリケーションを起動しない設定になっています。/etc/config/rc.local を編集することで、自動起動するように設定を行うことができます。

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

#
# Starting a state monitoring applications
#
START_IFPLUGD=n ❶
if [ "${START_IFPLUGD}" = "y" ]; then
    echo -n "Starting ifplugd: "
    ifplugd -i usb0 -p -q
    check_status
fi

START_THERMALMONITOR=n ❷
if [ "${START_THERMALMONITOR}" = "y" ] && [ "${START_IFPLUGD}" = "y" ]; then
    echo -n "Starting thermalmonitor: "
    /etc/config/thermalmonitor &
    check_status
fi

START_VINMONITOR=n ❸
if [ "${START_VINMONITOR}" = "y" ]; then
    echo -n "Starting vinmonitor: "
    /etc/config/vinmonitor &
    check_status
fi
```

❶ "n"から"y"に設定を変更してコンフィグ領域を保存すると、次回起動時に ifplugd が自動起動されるようになります

❷ ifplugd を自動起動に設定し、"n"から"y"に設定を変更してコンフィグ領域を保存すると、次回起動時に thermalmonitor が自動起動されるようになります

❸ "n"から"y"に設定を変更してコンフィグ領域を保存すると、次回起動時に vinmonitor が自動起動されるようになります

図 9.3 デフォルト状態の/etc/config/rc.local

9.3. 状態監視アプリケーション

/etc/config/rc.local から起動させることのできる状態監視アプリケーションについて説明します。

9.3.1. ifplugd

ifplugd コマンドを使用して 3G の接続状態を監視します。

3G 接続が切断されると、/etc/ifplugd/ifplugd.action 内に記載された処理が実行され、3G の再接続を行います。

9.3.2. thermalmonitor

thermalmonitor は Armadillo-IoT の筐体内温度を監視するアプリケーションです。thermalmonitor の実行中は thermaltrigger コマンドを実行することができません。 thermaltrigger コマンドについては、「6.8.2. 温度を監視する」を参照してください。

Armadillo-IoT の筐体内温度が 75℃以上になると、故障等を避けるため 3G モジュールを機内モードに設定し温度上昇を抑えます。その後、筐体内温度が 70℃以下まで下がると 3G の再接続を行います。

thermalmonitor の設定はコンフィグ領域に保存された/etc/config/thermalmonitor を編集すると変更することができます。変更後、設定を保存したい場合はコンフィグ領域を保存してください。

```
#!/bin/sh

HOT_TEMP_MDEG=75000 ❶
HOT_TEMP_CMD='/etc/config/hot_temp_action' ❷
HOT_TEMP_CMD_ARGS='' ❸

PASSIVE_TEMP_MDEG=70000 ❹
PASSIVE_TEMP_CMD='/etc/config/passive_temp_action' ❺
PASSIVE_TEMP_CMD_ARGS='' ❻

while true
do
    thermaltrigger -a $HOT_TEMP_MDEG $HOT_TEMP_CMD $HOT_TEMP_CMD_ARGS
    thermaltrigger -b $PASSIVE_TEMP_MDEG $PASSIVE_TEMP_CMD $PASSIVE_TEMP_CMD_ARGS
done
```

❶ 危険温度のしきい値をミリ℃単位で設定します

❷ 危険温度のしきい値以上になった時に実行するコマンドを記載します

❸ 実行するコマンドの引数を記載します

❹ 安全温度のしきい値をミリ℃単位で設定します

❺ 安全温度のしきい値以下になった時に実行するコマンドを記載します

❻ 実行するコマンドの引数を記載します

図 9.4 デフォルト状態の/etc/config/thermalmonitor

9.3.3. vinmonitor

vinmonitor は Armadillo-IoT の電源電圧を監視するアプリケーションです。主に Armadillo-IoT をバッテリー駆動させた場合を想定しています。vinmonitor の実行中は vintrigger コマンドを実行することができません。vintrigger コマンドについては、「6.9.2. 電源電圧を監視する」を参照してください。

Armadillo-IoT の電源電圧が 7V 以下になると、突然の動作停止による保存データ破壊等を避けるためシステムをシャットダウンします。

vinmonitor の設定はコンフィグ領域に保存された/etc/config/vinmonitor を編集すると変更することができます。変更後、設定を保存したい場合はコンフィグ領域を保存してください。

```
#!/bin/sh

CRITICAL_VOLTAGE_MV=7000 ❶
CRITICAL_VOLTAGE_CMD='/etc/config/critical_voltage_action' ❷
CRITICAL_VOLTAGE_CMD_ARGS='' ❸

vintrigger -u $CRITICAL_VOLTAGE_MV $CRITICAL_VOLTAGE_CMD $CRITICAL_VOLTAGE_CMD_ARGS
```

❶ 危険電圧のしきい値を mV 単位で設定します

❷ 危険電圧のしきい値以下になった時に実行するコマンドを記載します

❸ 実行するコマンドの引数を記載します

図 9.5 デフォルト状態の/etc/config/vinmonitor

9.4. 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        ip            more          setarch
adduser       echo          ipaddr        mount         setserial
amixer        ed            ipcalc        mountpoint    sh
aplay         egrep         iplink        mpstat        sleep
arecord       ethtool       iproute       mt            ssh
ash           evtest        iprule        mv            ssh-keygen
base64        expect        iptables      netflash      stat
busybox       false         iptunnel      netstat       stty
cat           fdflush       java          nice          su
catv          fgrep         keytool       ntpclient     swmgr
chattr        flatfsd       kill          pidof         sync
chgrp         fsck          linux32       ping          tar
chmod         fsck.ext2     linux64       ping6         tftp
chown         fsync         ln            pipe_progress tip
conspy        ftp           login         powertop      touch
cp            ftpd          lrz           printenv      true
cpio          getopt        ls            ps            tune2fs
cttyhack      grep          lsattr        pwd           umount
date          gunzip        lsz           reformime     uname
```

```
dd              gzip            lzop            rev             usleep
delgroup        hostname        mail            rm              vi
deluser         htpasswd        makemime        rmdir           watch
df              hush            mkdir           rpm             wget
dmesg           hwclock         mke2fs          run-parts       zcat
dnsdomainname   ionice          mknod           scriptreplay
dumpkmap        iostat          mktemp          sed
```

- /usr/bin

```
3g-connect      env             lspci           rpm2cpio        time
3g-disconnect   envdir          lsusb           rtcwake         timeout
3g-phone-num    envuidgid       lua             ruby            top
3g-set-ap       ether-wake      luac            runsv           tr
3g-temp         expand          lzcat           runsvdir        traceroute
[               expr            lzma            rx              traceroute6
[[              fdformat        lzopcat         script          ts_calibrate
add-shell       fgconsole       md5sum          sd-awlan-sel    tty
ar              find            mesg            seq             ttysize
arping          flock           microcom        setblank        udevinfo
awk             fold            mkfifo          setkeycodes     udpsvd
basename        free            mkpasswd        setsid          unexpand
beep            ftpget          mosquitto_pub   setuidgid       uniq
bunzip2         ftpput          mosquitto_sub   sha1sum         unix2dos
bzcat           fuser           nc              sha256sum       unlzma
bzip2           get_device      nmeter          sha512sum       unlzop
cal             get_driver      nohup           showkey         unxz
chat            get_module      nslookup        smemcap         unzip
chpst           groups          od              softlimit       uptime
chrt            hd              openvt          sort            users
chvt            head            passwd          spawn-fcgi      uudecode
cksum           hexdump         patch           split           uuencode
clear           hostid          pgrep           strings         vi
cmp             id              pkill           sudo            vintrigger
comm            ifplugd         pmap            sudoedit        vlock
crontab         install         printf          sum             volname
cryptpw         ipcrm           pscan           sv              wall
curl            ipcs            pstree          systool         wc
cut             kbd_mode        pwdx            tac             wget
dc              killall         rackup          tail            which
deallocvt       killall5        readahead       tcpsvd          who
diff            last            readlink        tee             whoami
dirname         less            realpath        telnet          whois
dos2unix        logger          remove-shell    test            xargs
dpkg-deb        logname         renice          tftp            xz
du              lpq             reset           tftpd           xzcat
dumpleases      lpr             resize          thermaltrigger  yes
eject           lsof            restclient      tilt
```

- /sbin

```
acpid           fsck.vfat       mke2fs          runlevel
adjtimex        getty           mkfs.ext2       setconsole
arp             halt            mkfs.minix      slattach
avahi-daemon    hdparm          mkfs.msdos      sshd
blkid           hwclock         mkfs.vfat       start-stop-daemon
blockdev        ifconfig        mkswap          sulogin
```

```
bootchartd      ifdown          modinfo         swapoff
chat            ifenslave       modprobe        swapon
depmod          ifup            nameif          switch_root
devmem          init            nanddump        sysctl
dosfsck         insmod          nandwrite       syslogd
fbsplash        iwconfig        nftl_format     tunctl
fdisk           iwlist          nftldump        tune2fs
findfs          iwpriv          pivot_root      udevcontrol
flash_erase     klogd           poweroff        udevd
flash_eraseall  loadkmap        pppd            udevsettle
flash_info      logread         pppdump         udevtrigger
flash_lock      losetup         pppoe-discovery udhcpc
flash_unlock    lsmod           pppstats        vconfig
freeramdisk     makedevs        raidautorun     watchdog
fsck            man             reboot          zcip
fsck.minix      mdev            rmmod
fsck.msdos      mkdosfs         route
```

・ /usr/sbin

```
brctl                   lighttpd                setlogcons
chpasswd                loadfont                svlogd
chroot                  lpd                     telnetd
crond                   nanddump                ubiattach
dhcprelay               nandwrite               ubidetach
dnsd                    nbd-client              ubimkvol
fakeidentd              ntpd                    ubirmvol
fbset                   popmaildir              ubirsvol
ftpd                    rdate                   ubiupdatevol
get-board-info          rdev                    udevmonitor
get-board-info-aiotg-std readprofile            udhcpd
httpd                   sendmail                visudo
inetd                   setfont
```

9.5. 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
Ruby	オブジェクト指向スクリプト言語です。
Java	オブジェクト指向プログラミング言語です。Armadillo-IoT では Oracle Java が使用可能です。
Lua	C 言語等のホストプログラムに組み込まれることを目的に設計されたスクリプト言語です。高速な動作と、高い移植性、組み込みの容易さが特徴です。
Wget	ウェブサーバーからコンテンツを取得するコマンドラインツールです。`wget` コマンドにて実行が可能です。
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-IoT では、`curl` コマンドにて実行が可能です。
Mosquitto	MQTT ブローカー/クライアントです。Armadillo-IoT では、`mosquitto_pub` コマンド、`mosquitto_sub` コマンドをプリインストールしています。

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、USB シリアル変換アダプタのスライドスイッチの状態により、2 つのモードのどちらかに遷移します。USB シリアル変換アダプタのスライドスイッチの詳細については、「4.5. スライドスイッチの設定について」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	スライドスイッチ	説明
保守モード	外側	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	内側	電源投入後、自動的に Linux カーネルを起動させます。この場合、コンソールが使用できません。

USB シリアル変換アダプタが未接続の場合オートブートモードとなり、Linux カーネルが起動します。

10.1.1. Linux でコンソールを使用する

オートブートモードで起動するとコンソールが使用できません。Linux でコンソールを使用するには、保守モードで起動してから `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は、「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

コマンド	説明
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	オートブートモード時のログ出力先	保守モードプロンプト出力先[a]
ttymxc1	デバッグシリアルインターフェース(ベースボード:CON9)	デバッグシリアルインターフェース(ベースボード:CON9)
none	なし	デバッグシリアルインターフェース(ベースボード:CON9)
その他(ttymxc0 等)	指定するコンソール[b]	アドオンインターフェース(ベースボード:CON1 or ベースボード:CON2)

[a]ブートローダーの再起動後に反映されます

[b]ブートローダーのログは出力されません

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、SD カード内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(Armadillo-410:CON1 に接続)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
mmcblk1p1	SD カード(ベースボード:CON4 に接続)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-IoT 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`
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。 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-IoT ゲートウェイ スタンダードモデル ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-iot/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-IoT ゲートウェイ スタンダードモデルでは、「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」、「AWL13 デバイスドライバ」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
awl13-[version].tar.gz        linux-3.4-[version].tar.gz
[ATDE ~]$ tar zxf atmark-dist-[version].tar.gz
```

```
[ATDE ~]$ tar zxf awl13-[version].tar.gz
[ATDE ~]$ tar zxf ejdk-[version].tar.gz
[ATDE ~]$ tar zxf linux-2.6.26-at[version].tar.gz
[ATDE ~]$ ls
atmark-dist-[version]         awl13-[version].tar.gz  linux-3.4-[version]
atmark-dist-[version].tar.gz  ejdk[version]           linux-3.4-[version].tar.gz
awl13-[version]               ejdk-[version].tar.gz
```

2. **シンボリックリンクの作成**

Atmark Dist に、AWL13、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 ../awl13-[version] awl13
[ATDE ~/atmark-dist-[version]]$ ln -s ../linux-2.6.26-at-[version] linux-2.6.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 v1.36.0 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-IoTG-Std" を選択します。

```
 atmark-dist v1.36.0 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-IoTG-Std) 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 v1.36.0 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 v1.36.0 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-IoT 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-IoT ゲートウェイ スタンダードモデル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_iotg_std_defconfig を指定します。UART 起動用イメージを作成する場合は、armadillo_iotg_std_boot_defconfig を指定してください。

```
[ATDE ~]$ cd hermit-at
[ATDE ~/hermit-at]$ make armadillo_iotg_std_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-iotg-std/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-iotg-std/loader-armadillo-iotg-std-*.bin
src/target/armadillo-iotg-std/loader-armadillo-iotg-std-[version].bin
```

12. フラッシュメモリの書き換え方法

本章では、Armadillo-IoT 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-iotg-std-*[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-aiotg-std-*[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-aiotg-std-*[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-aiotg-std-[version].bin.gz
[ATDE ~]$ cp linux-aiotg-std-[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-aiotg-std-*[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-aiotg-
std-[version].bin.gz
.....................................................................
.....................................................................
(省略)
.....................................
netflash: got "http://192.0.2.1/linux-aiotg-std-[version].bin.gz", length=2564696
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(SD 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 SD 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 SD カード上のイメージファイルを書き込む

1. SD 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-aiotg-std-[version].img.gz
```

2. SD カード上のイメージファイルのパス(/mnt/romfs-aiotg-std-*[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-aiotg-std-
[version].img.gz
.....................................................................
.....................................................................
```

```
（省略）
.....................................................
netflash: got "/mnt/romfs-aiotg-std-[version].img.gz", length=10316650
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. SD 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-iotg-std-[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-iotg-std-[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-IoT の電源が切断されていることを確認します。
2. 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-IoT 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-3.3/loader-armadillo-iotg-std-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-iotg-std-boot-[version].bin: 51456
bytes (51456 bytes buffer)
/dev/null: 0 bytes (0 bytes buffer)
Waiting for target - press Wakeup now.
```

3. プロセッサ(i.MX257)を UART ブートモードに設定します。起動モード設定インターフェース(Armadillo-410:CON15)の 1-2 ピンをショートしてください。

金属製の工具(M2 のマイナスドライバー等)で 起動モード設定インターフェース(Armadillo-410:CON15) の 1-2 ピン間を

ショートして、UART ブートモードに設定することも可能です。その際、周囲のコネクタ等に工具が接触しないようご注意ください。

4. Armadillo に電源を投入します。電源投入後に、起動モード設定インターフェース(Armadillo-410:CON15)の 1-2 ピンをオープンします。

```
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-3.3/loader-armadillo-iotg-std-boot-[version].bin:
- start: 0x80800000 size: 0x0000c900 last: 0x8080c8ff
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-IoT 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-IoT 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-iotg-std-[version].bin --region bootloader --force-locked --port /dev/ttyUSB0 ⏎
serial: completed 0x0000a92c (43308) 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-IoT 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-IoT では armel というアーキテクチャが使われています。Armadillo-IoT 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-IoT 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 armhf 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-IoT 上で動作するように main.c をクロスコンパイル

Armadillo-IoT に実行ファイルを転送して動作の確認を行います。ここではファイル転送に FTP を利用します。次の例では、Armadillo-IoT の IP アドレスが「192.0.2.10」であることを想定しています。

```
[ATDE ~/hello]$ ftp 192.0.2.10
Connected to 192.0.2.10.
220 localhost FTP server (GNU inetutils 1.4.1) ready.
Name (192.0.2.10:atmark): ftp
331 Guest login ok, type your name as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub
250 CWD command successful.
ftp> put hello
local: hello remote: hello
200 PORT command sucessful.
150 Opening BINARY mode data connection for 'hello'.
226 Transfer complete.
5087 bytes sent in 0.00 secs (112903.9 kB/s)
ftp> quit
221 Goodbye.
```

図 13.5 Armadillo に FTP で hello を転送

minicom などを利用して Armadillo-IoT にログインすると/home/ftp/pub に hello が転送されています。転送され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 ~]# cd /home/ftp/pub/
[armadillo ~/home/ftp/pub]# ls
hello
[armadillo ~/home/ftp/pub]# chmod +x hello
[armadillo ~/home/ftp/pub]# ./hello
Hello World!
```

図 13.6 Armadillo-IoT 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-IoT 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 13.7 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 13.8 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 13.7. 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 13.9 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-IoT 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-IoTG-Std/ vendors/AtmarkTechno/my-
product
[ATDE ~/atmark-dist]$ cp -r ../hello/ vendors/AtmarkTechno/my-product/
```

図 13.10 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。

```
20 comma := ,
30 empty :=
31 space := $(empty) $(empty)
32
33 SUBDIR_y =
34 SUBDIR_y += hello/
35 SUBDIR_$(CONFIG_VENDOR_SWMGR_SWMGR)        += swmgr/
36 SUBDIR_$(CONFIG_VENDOR_THERMALTRIGGER_THERMALTRIGGER)    += thermaltrigger/
37 SUBDIR_$(CONFIG_VENDOR_VINTRIGGER_VINTRIGGER)    += vintrigger/
38 SUBDIR_$(CONFIG_VENDOR_AWL12_AERIAL)        += awl12/
39 SUBDIR_$(CONFIG_VENDOR_AWL13_AWL13)        += awl13/
```

図 13.11 オリジナルプロダクト(my-product)に hello を登録

「図 13.7. 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 13.12 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-IoTG-Std"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 13.13 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 Armaidllo-IoT 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 v1.36.0 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 v1.36.0 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-2.6.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 v1.36.0 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 v1.36.0 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 v1.36.0 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
               [*] swmgr

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

                   <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
               [*] swmgr

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

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 v1.36.0 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-2.6.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-2.6.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

14. ハードウェア仕様

14.1. インターフェース仕様

Armadillo-IoT ゲートウェイ スタンダードモデルのインターフェース仕様について説明します。

14.1.1. インターフェースレイアウト

Armadillo-IoT ゲートウェイ スタンダードモデルは、Armadillo-IoT ゲートウェイ ベースボードと Armadillo-410 で構成されます。

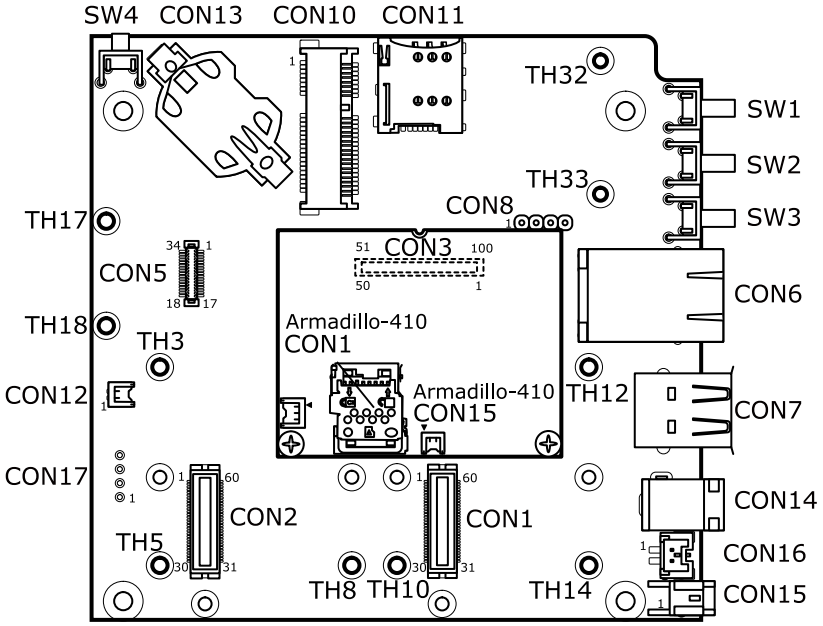


図 14.1 インターフェースレイアウト(A 面)

表 14.1 搭載コネクタ、スイッチ型番一覧(A 面)[a]

部品番号	インターフェース名	型番	メーカー
CON1	アドオンインターフェース	DF17(4.0)-60DS-0.5V(57)	HIROSE ELECTRIC
CON2	アドオンインターフェース		
CON3	Armadillo-410 インターフェース	DF40HC(3.0)-100DS-0.4V(51)	HIROSE ELECTRIC
CON5	WLAN インターフェース	AXK6F34347YG-E	Panasonic
CON6	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse
CON7	USB ホストインターフェース	UBA-4R-D14T-4D	J.S.T. Mfg.
CON8	デバッグ USB インターフェース	A2-4PA-2.54DSA(71)	HIROSE ELECTRIC
CON10	3G インターフェース	MM60-52B1-E1	JAE
CON11	microSIM インターフェース	CIM-J78	MITSUMI
CON12	PMIC ON/OFF インターフェース	BM02B-ACHSS-GAN-ETF	J.S.T. Mfg.
CON13	RTC 外部バックアップインターフェース	SMTU2032-LF	RENATA
CON14	電源入力インターフェース	PJ-102AH	CUI
CON15	電源入力インターフェース	S02B-PASK-2	J.S.T. Mfg.
CON16	電源出力インターフェース	BM02B-PASS	J.S.T. Mfg.

部品番号	インターフェース名	型番	メーカー
CON17	タッチスクリーンインターフェース	A2-4PA-2.54DSA(71)	HIROSE ELECTRIC
SW1	ユーザースイッチ	SKHHLRA010	ALPS ELECTRIC
SW2			
SW3			
SW4	リセットスイッチ	SKHHLUA010	ALPS ELECTRIC
TH3	アドオンモジュール用スタッド	TH-1.6-8.0-M2	MAC8
TH5			
TH8			
TH10			
TH12			
TH14			
TH17	WLAN モジュール用スタッド	TH-1.6-1.5-M2	MAC8
TH18			
TH32	3G モジュール用スタッド	NT4	JAE
TH33			

[a]色のついたセルの部品は実装していません。実装例を記載しています。

表 14.2 搭載コネクタ型番(Armadillo-410)

部品番号	インターフェース名	型番	メーカー
CON1	microSD インターフェース	SDHK-8BNS-K-303-TB(HF)	J.S.T. Mfg.
CON15	起動モード設定インターフェース	BM02B-ACHSS-GAN-ETF	J.S.T. Mfg.

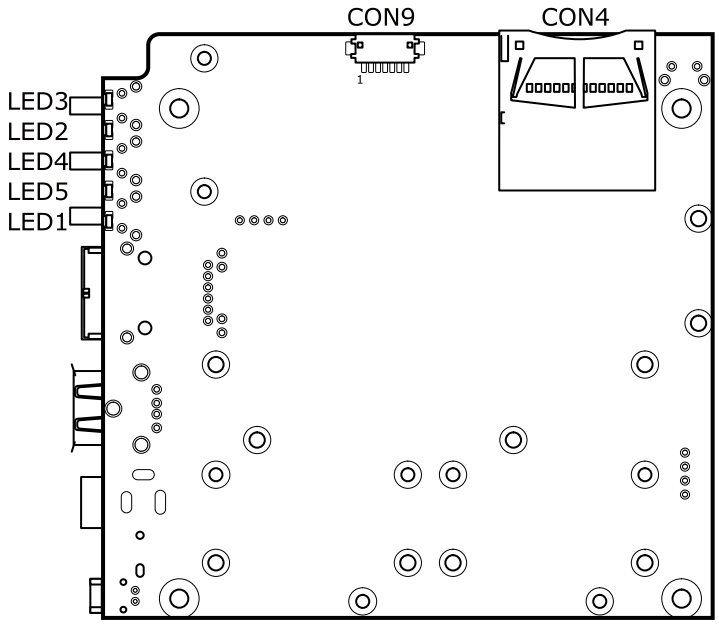


図 14.2 インターフェースレイアウト(B 面)

表 14.3 搭載コネクタ、スイッチ、LED 型番一覧(B 面)

部品番号	インターフェース名	型番	メーカー
CON4	SD インターフェース	SCDA9A0400	ALPS ELECTRIC
CON9	デバッグシリアルインターフェース	DF13A-7P-1.25H(51)	HIROSE ELECTRIC
LED1	3G LED	SML-A12P8T	ROHM

部品番号	インターフェース名	型番	メーカー
LED2	ユーザー LED	SML-A12P8T	ROHM
LED3			
LED4			
LED5			

14.1.2. CON1 アドオンインターフェース

・ 許容電流: 0.3A(端子 1 本あたり)

表 14.4 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	EXTIO22	In/Out	拡張入出力、i.MX257 の GPIO_C ピンに接続
4	EXTIO23	In/Out	拡張入出力、i.MX257 の GPIO_D ピンに接続
5	PWMO1	In/Out	拡張入出力、i.MX257 の PWM ピンに接続
6	EXTIO1	In/Out	拡張入出力、i.MX257 の RTCK ピンに接続
7	EXTIO35	In/Out	拡張入出力、i.MX257 の UART1_RXD ピンに接続
8	EXTIO36	In/Out	拡張入出力、i.MX257 の UART1_TXD ピンに接続
9	EXTIO37	In/Out	拡張入出力、i.MX257 の UART1_RTS ピンに接続
10	EXTIO38	In/Out	拡張入出力、i.MX257 の UART1_CTS ピンに接続
11	EXTIO12	In/Out	拡張入出力、i.MX257 の CSI_D9 ピンに接続
12	EXTIO3	In/Out	拡張入出力、i.MX257 の CSI_D2 ピンに接続
13	EXTIO5	In/Out	拡張入出力、i.MX257 の CSI_D3 ピンに接続
14	EXTIO7	In/Out	拡張入出力、i.MX257 の CSI_D4 ピンに接続
15	EXTIO9	In/Out	拡張入出力、i.MX257 の CSI_D5 ピンに接続
16	EXTIO25	In/Out	拡張入出力、i.MX257 の KPP_ROW0 ピンに接続
17	EXTIO26	In/Out	拡張入出力、i.MX257 の KPP_ROW1 ピンに接続
18	EXTIO27	In/Out	拡張入出力、i.MX257 の KPP_ROW2 ピンに接続
19	EXTIO28	In/Out	拡張入出力、i.MX257 の KPP_ROW3 ピンに接続
20	EXTIO29	In/Out	拡張入出力、i.MX257 の KPP_COL0 ピンに接続
21	EXTIO30	In/Out	拡張入出力、i.MX257 の KPP_COL1 ピンに接続
22	EXTIO31	In/Out	拡張入出力、i.MX257 の KPP_COL2 ピンに接続
23	EXTIO32	In/Out	拡張入出力、i.MX257 の KPP_COL3 ピンに接続
24	EXTIO33	In/Out	拡張入出力、i.MX257 の GPIO_A ピンに接続
25	EXTIO34	In/Out	拡張入出力、i.MX257 の GPIO_B ピンに接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	+3.3V	Power	電源出力(+3.3V)
30	+5V	Power	電源出力(+5V)
31	DETECT_CON1	Out	1kΩ でプルダウン
32	LCD_LD17	In/Out	拡張入出力、i.MX257 の GPIO_F ピンに接続
33	LCD_LD16	In/Out	拡張入出力、i.MX257 の GPIO_E ピンに接続
34	LCD_LD15	In/Out	拡張入出力、i.MX257 の LD15 ピンに接続
35	LCD_LD14	In/Out	拡張入出力、i.MX257 の LD14 ピンに接続
36	LCD_LD13	In/Out	拡張入出力、i.MX257 の LD13 ピンに接続
37	LCD_LD12	In/Out	拡張入出力、i.MX257 の LD12 ピンに接続
38	LCD_LD11	In/Out	拡張入出力、i.MX257 の LD11 ピンに接続
39	LCD_LD10	In/Out	拡張入出力、i.MX257 の LD10 ピンに接続
40	LCD_LD9	In/Out	拡張入出力、i.MX257 の LD9 ピンに接続
41	LCD_LD8	In/Out	拡張入出力、i.MX257 の LD8 ピンに接続

ピン番号	ピン名	I/O	説明
42	LCD_LD7	In/Out	拡張入出力、i.MX257 の LD7 ピンに接続
43	LCD_LD6	In/Out	拡張入出力、i.MX257 の LD6 ピンに接続
44	LCD_LD5	In/Out	拡張入出力、i.MX257 の LD5 ピンに接続
45	LCD_LD4	In/Out	拡張入出力、i.MX257 の LD4 ピンに接続
46	LCD_LD3	In/Out	拡張入出力、i.MX257 の LD3 ピンに接続
47	LCD_LD2	In/Out	拡張入出力、i.MX257 の LD2 ピンに接続
48	LCD_LD1	In/Out	拡張入出力、i.MX257 の LD1 ピンに接続
49	LCD_LD0	In/Out	拡張入出力、i.MX257 の LD0 ピンに接続
50	LCD_OE_ACD	In/Out	拡張入出力、i.MX257 の OE_ACD ピンに接続
51	LCD_VSYN	In/Out	拡張入出力、i.MX257 の VSYNC ピンに接続
52	LCD_HSYN	In/Out	拡張入出力、i.MX257 の HSYNC ピンに接続
53	LCD_LSCLK	In/Out	拡張入出力、i.MX257 の LSCLK ピンに接続
54	GND	Power	電源(GND)
55	PMIC_ONOFF	In	パワーマネジメント IC の ON/OFF 用信号、CON12 の 2 ピンに接続
56	USB_VBUS	Power	電源(VBUS)
57	USB_VBUS	Power	電源(VBUS)
58	GND	Power	電源(GND)
59	EXT_USB_HS_DP	In/Out	USB1 のプラス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DP ピンに接続
60	EXT_USB_HS_DM	In/Out	USB1 のマイナス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DM ピンに接続

14.1.3. CON2 アドオンインターフェース

- 許容電流: 0.3A(端子 1 本あたり)

表 14.5 CON2 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	NC	-	未接続
4	NC	-	未接続
5	NC	-	未接続
6	NC	-	未接続
7	EXTIO37	In/Out	拡張入出力、i.MX257 の UART1_RTS ピンに接続
8	EXTIO38	In/Out	拡張入出力、i.MX257 の UART1_CTS ピンに接続
9	NC	-	未接続
10	NC	-	未接続
11	NC	-	未接続
12	EXTIO7	In/Out	拡張入出力、i.MX257 の CSI_D4 ピンに接続
13	EXTIO9	In/Out	拡張入出力、i.MX257 の CSI_D5 ピンに接続
14	NC	-	未接続
15	NC	-	未接続
16	EXTIO3	In/Out	拡張入出力、i.MX257 の CSI_D2 ピンに接続
17	EXTIO5	In/Out	拡張入出力、i.MX257 の CSI_D3 ピンに接続
18	EXTIO7	In/Out	拡張入出力、i.MX257 の CSI_D4 ピンに接続
19	EXTIO9	In/Out	拡張入出力、i.MX257 の CSI_D5 ピンに接続
20	EXTIO29	In/Out	拡張入出力、i.MX257 の KPP_COL0 ピンに接続
21	EXTIO30	In/Out	拡張入出力、i.MX257 の KPP_COL1 ピンに接続
22	EXTIO31	In/Out	拡張入出力、i.MX257 の KPP_COL2 ピンに接続
23	EXTIO32	In/Out	拡張入出力、i.MX257 の KPP_COL3 ピンに接続
24	EXTIO22	In/Out	拡張入出力、i.MX257 の GPIO_C ピンに接続
25	EXTIO23	In/Out	拡張入出力、i.MX257 の GPIO_D ピンに接続

ピン番号	ピン名	I/O	説明
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	+3.3V	Power	電源出力(+3.3V)
30	+5V	Power	電源出力(+5V)
31	DETECT_CON2	Out	+3.3V_IO で 1kΩ プルアップ
32	EXTIO1	In/Out	拡張入出力、i.MX257 の RTCK ピンに接続
33	PWMO1	In/Out	拡張入出力、i.MX257 の PWM ピンに接続
34	EXTIO9	In/Out	拡張入出力、i.MX257 の CSI_D5 ピンに接続
35	EXTIO7	In/Out	拡張入出力、i.MX257 の CSI_D4 ピンに接続
36	EXTIO5	In/Out	拡張入出力、i.MX257 の CSI_D3 ピンに接続
37	EXTIO3	In/Out	拡張入出力、i.MX257 の CSI_D2 ピンに接続
38	EXTIO38	In/Out	拡張入出力、i.MX257 の UART1_CTS ピンに接続
39	EXTIO37	In/Out	拡張入出力、i.MX257 の UART1_RTS ピンに接続
40	EXTIO36	In/Out	拡張入出力、i.MX257 の UART1_TXD ピンに接続
41	EXTIO35	In/Out	拡張入出力、i.MX257 の UART1_RXD ピンに接続
42	EXTIO32	In/Out	拡張入出力、i.MX257 の KPP_COL3 ピンに接続
43	EXTIO31	In/Out	拡張入出力、i.MX257 の KPP_COL2 ピンに接続
44	EXTIO30	In/Out	拡張入出力、i.MX257 の KPP_COL1 ピンに接続
45	EXTIO29	In/Out	拡張入出力、i.MX257 の KPP_COL0 ピンに接続
46	EXTIO28	In/Out	拡張入出力、i.MX257 の KPP_ROW3 ピンに接続
47	EXTIO27	In/Out	拡張入出力、i.MX257 の KPP_ROW2 ピンに接続
48	EXTIO26	In/Out	拡張入出力、i.MX257 の KPP_ROW1 ピンに接続
49	EXTIO25	In/Out	拡張入出力、i.MX257 の KPP_ROW0 ピンに接続
50	EXTIO12	In/Out	拡張入出力、i.MX257 の CSI_D9 ピンに接続
51	NC	-	未接続
52	NC	-	未接続
53	NC	-	未接続
54	GND	Power	電源(GND)
55	PMIC_ONOFF	In	パワーマネジメント IC の ON/OFF 用信号、CON12 の 2 ピンに接続
56	NC	-	未接続
57	NC	-	未接続
58	GND	Power	電源(GND)
59	NC	-	未接続
60	NC	-	未接続

14.1.4. CON3 Armadillo-410 インターフェース

CON3 には Armadillo-410 が接続されます。Armadillo-410 のピンアサインにつきましては、『Armadillo-410 ハードウェアマニュアル』をご参照ください。

14.1.5. CON4 SD インターフェース

表 14.6 CON4 信号配列

ピン番号	ピン名	I/O	説明
1	CD/DAT3	In/Out	データバス(bit3)、マルチプレクサを経由して i.MX257 の CSI_PIXCLK ピンに接続
2	CMD	In/Out	SD コマンド/レスポンス、マルチプレクサを経由して i.MX257 の CSI_D6 ピンに接続
3	VSS	Power	電源(GND)
4	VDD	Power	電源(+3.3V)
5	CLK	Out	SD クロック、マルチプレクサを経由して i.MX257 の CSI_D7 ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	データバス(bit0)、マルチプレクサを経由して i.MX257 の CSI_MCLK ピンに接続
8	DAT1	In/Out	データバス(bit1)、マルチプレクサを経由して i.MX257 の CSI_VSYNC ピンに接続

ピン番号	ピン名	I/O	説明
9	DAT2	In/Out	データバス(bit2)、マルチプレクサを経由して i.MX257 の CSI_HSYNC ピンに接続

表 14.7 CON4 カード検出、ライトプロテクト

項目	説明
カード検出	マルチプレクサを経由して i.MX257 の CSI_D8 ピンに接続 (Low: カード挿入、High: カード未挿入)
ライトプロテクト検出	マルチプレクサを経由して i.MX257 の CLKO ピンに接続 (Low: 書き込み可能、High: 書き込み不可能)

14.1.6. CON5 WLAN インターフェース

表 14.8 CON5 信号配列

ピン番号	ピン名	I/O	説明
1	SDDATA1	In/Out	SDIO データ(bit1)、マルチプレクサを経由して i.MX257 の CSI_VSYNC ピンに接続
2	SDDATA0	In/Out	SDIO データ(bit0)、マルチプレクサを経由して i.MX257 の CSI_MCLK ピンに接続
3	GND	Power	電源(GND)
4	GND	Power	電源(GND)
5	USB_DM	-	未接続
6	USB_DP	-	未接続
7	SDCLK	Out	SDIO クロック、マルチプレクサを経由して i.MX257 の CSI_D7 ピンに接続
8	VCC	Power	電源(+3.3V)
9	NC	-	未接続
10	SDCMD	In/Out	SDIO コマンド、マルチプレクサを経由して i.MX257 の CSI_D6 ピンに接続
11	SDDATA3	In/Out	SDIO データ(bit3)、マルチプレクサを経由して i.MX257 の CSI_PIXCLK ピンに接続
12	SDDATA2	In/Out	SDIO データ(bit2)、マルチプレクサを経由して i.MX257 の CSI_HSYNC ピンに接続
13	UART_RXD	-	未接続
14	UART_TXD	-	未接続
15	BOOT_SEL1	Out	起動モード設定、SDIO モードに設定
16	BOOT_SEL0	Out	
17	HOST_SEL	Out	
18	FLASH_RXD	-	未接続
19	FLASH_CSB	-	未接続
20	FLASH_CLK	-	未接続
21	FLASH_TXD	Out	GND に 47kΩ プルダウン
22	FLASH_SEL	-	未接続
23	GPIO0	-	未接続
24	GPIO1	-	未接続
25	M_ANA	-	未接続
26	GPIO2	-	未接続
27	GPIO6	-	未接続
28	HRST	Out	+3.3V に接続
29	PRST	-	未接続
30	TMS	-	未接続
31	TCK	-	未接続
32	TDI	-	未接続
33	TDO	-	未接続
34	TRSTB	-	未接続

14.1.7. CON6 LAN インターフェース

表 14.9 CON6 信号配列

ピン番号	ピン名	I/O	説明
1	TX+	In/Out	送信出力 差動ペア(+)
2	TX-	In/Out	送信出力 差動ペア(−)
3	RX+	In/Out	受信入力 差動ペア(+)
4	-	-	CON6 5 ピンと接続後に 75Ω 終端
5	-	-	CON6 4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	受信入力 差動ペア(−)
7	-	-	CON6 8 ピンと接続後に 75Ω 終端
8	-	-	CON6 7 ピンと接続後に 75Ω 終端

14.1.8. CON7 USB ホストインターフェース

表 14.10 CON7 信号配列

ピン番号	ピン名	I/O	説明
1	USB_VBUS	Power	USB 電源(VBUS)
2	D-	In/Out	USB マイナス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DM ピンに接続
3	D+	In/Out	USB プラス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DP ピンに接続
4	GND	Power	電源(GND)

14.1.9. CON8 デバッグ USB インターフェース

表 14.11 CON8 信号配列

ピン番号	信号名	I/O	機能
1	USB_FS_SEL	In	3G モジュールの USB 信号の接続先の切替用信号、マルチプレクサのセレクトピンに接続 (High: Armadillo-410、Low: CON8)
2	GND	Power	電源(GND)
3	EXT_3G_USB_DP	In/Out	3G モジュールの USB プラス側信号、マルチプレクサを経由して 3G モジュールに接続
4	EXT_3G_USB_DM	In/Out	3G モジュールの USB マイナス側信号、マルチプレクサを経由して 3G モジュールに接続

14.1.10. CON9 デバッグシリアルインターフェース

表 14.12 CON9 信号配列

ピン番号	信号名	I/O	機能
1	DEBUG_UART_RXD	In	受信データ、マルチプレクサ等を経由して i.MX257 の UART2_RXD ピン、CON10 の 51 ピンに接続
2	GND	Power	電源(GND)
3	DEBUG_UART_TXD	Out	送信データ、マルチプレクサ等を経由して i.MX257 の UART2_TXD ピン、CON10 の 49 ピンに接続
4	+3.3V_CPU	Power	電源出力(+3.3V_CPU)
5	DEBUG_UART_CTS	In	送信可能、マルチプレクサ等を経由して i.MX257 の UART2_RTS ピン、CON10 の 47 ピンに接続
6	A410/3G_SEL	In	マルチプレクサのセレクトピン、i.MX257 の NF_CE0 ピンに接続 (Low: Armadillo-410 保守モード、High: 3G)

ピン番号	信号名	I/O	機能
7	DEBUG_UART_RTS	Out	送信要求、マルチプクレサ等を経由して i.MX257 の UART2_CTS ピン、CON10 の 45 ピンに接続

14.1.11. CON10 3G インターフェース

表 14.13 CON10 信号配列

ピン番号	ピン名	I/O	説明
1	NC	-	未接続
2	VCC	Power	電源(+3.6V_3G)
3	NC	-	未接続
4	GND	Power	電源(GND)
5	NC	-	未接続
6	NC	-	未接続
7	NC	-	未接続
8	EXT_VREG_USIM	-	電源、CON11 の 2 ピンに接続
9	GND	Power	電源(GND)
10	EXT_USIM_DATA	In/Out	USIM 入出力信号、CON11 の 6 ピンに接続
11	VREF_1V8	Power	電源(+1.8V)
12	EXT_USIM_CLK	In	USIM クロック、CON11 の 4 ピンに接続
13	NC	-	未接続
14	EXT_USIM_RESET	In	USIM リセット、CON11 の 3 ピンに接続
15	GND	Power	電源(GND)
16	NC	-	未接続
17	NC	-	未接続
18	GND	Power	電源(GND)
19	NC	-	未接続
20	W_DISABLE_N	Out	無線動作の有効/無効信号、レベル変換 IC を経由して i.MX257 の CSPI1_SS0 ピンに接続
21	GND(CD)	Power	電源(GND)
22	NC	-	未接続
23	NC	-	未接続
24	VCC	Power	電源(+3.6V_3G)
25	NC	-	未接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	NC	-	未接続
29	GND	Power	電源(GND)
30	NC	-	未接続
31	NC	-	未接続
32	WAKE_N	In/Out	ホストインターフェース WAKE 信号、レベル変換 IC を経由して i.MX257 の CSPI1_MOSI ピンに接続
33	SYSTEM_RESET_N	Out	リセット信号、レベル変換 IC を経由して i.MX257 の EXT_ARMCLK ピンに接続
34	GND	Power	電源(GND)
35	GND	Power	電源(GND)
36	USB_D-	In/Out	USB マイナス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DM ピンに接続
37	GND	Power	電源(GND)
38	USB_D+	In/Out	USB プラス側信号、マルチプレクサを経由して i.MX257 の USBPHY1_DP ピンに接続
39	VCC	Power	電源(+3.6V_3G)
40	GND	Power	電源(GND)
41	VCC	Power	電源(+3.6V_3G)

ピン番号	ピン名	I/O	説明
42	LED_FLASH	In	LED1 に接続
43	GND	Power	電源(GND)
44	GPIO_1	-	未接続
45	UART1_CTS_N	In	送信可能、レベル変換 IC、マルチプレクサを経由して CON9 の 7 ピンに接続
46	GPIO_3	-	未接続
47	UART1_RTS_N	Out	送信要求、レベル変換 IC、マルチプレクサを経由して CON9 の 5 ピンに接続
48	GPIO_2	-	未接続
49	UART1_RXD	In	受信データ、レベル変換 IC、マルチプレクサを経由して CON9 の 3 ピンに接続
50	GND	Power	電源(GND)
51	UART1_TXD	Out	送信データ、レベル変換 IC、マルチプレクサを経由して CON9 の 1 ピンに接続
52	VCC	Power	電源(+3.6V_3G)

14.1.12. CON11 microSIM インターフェース

表 14.14 CON11 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	VCC	Power	電源、CON10 の 8 ピンに接続
3	RST	Out	リセット、CON10 の 14 ピンに接続
4	CLK	Out	クロック、CON10 の 12 ピンに接続
5	VPP	-	未接続
6	I/O	In	入出力信号、CON10 の 10 ピンに接続

14.1.13. CON12 PMIC ON/OFF インターフェース

表 14.15 CON12 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	PMIC_ONOFF	Out	パワーマネジメント IC の ON/OFF 用信号、CON1 の 55 ピン、CON2 の 55 ピンに接続

14.1.14. CON13 RTC 外部バックアップインターフェース

表 14.16 CON13 信号配列

ピン番号	ピン名	I/O	機能
1	BAT	Power	リアルタイムクロックの外部バックアップ用電源入力
2	GND	Power	電源(GND)

14.1.15. CON14 電源入力インターフェース

表 14.17 CON14 信号配列

ピン番号	ピン名	I/O	機能
1	VIN	Power	電源入力(VIN)
2	GND	Power	電源(GND)
3	GND	Power	電源(GND)

図 14.3 AC アダプターの極性マーク

14.1.16. CON15 電源入力インターフェース

表 14.18 CON15 信号配列

ピン番号	ピン名	I/O	機能
1	VIN	Power	電源入力(VIN)
2	GND	Power	電源(GND)

14.1.17. CON16 電源出力インターフェース

表 14.19 CON16 信号配列

ピン番号	ピン名	I/O	機能
1	VOUT	Power	電源出力(VOUT)
2	GND	Power	電源(GND)

14.1.18. CON17 タッチスクリーンインターフェース

表 14.20 CON17 信号配列

ピン番号	ピン名	I/O	機能
1	TOUCH_XP	In/Out	i.MX257 の XP ピンに接続
2	TOUCH_XN	In/Out	i.MX257 の XN ピンに接続
3	TOUCH_YP	In/Out	i.MX257 の YP ピンに接続
4	TOUCH_YN	In/Out	i.MX257 の YN ピンに接続

14.1.19. SW1～SW3 ユーザースイッチ

表 14.21 ユーザースイッチの接続

部品番号	説明
SW1	i.MX257 の NFWP_B ピンに接続(ON: Low、OFF: High)
SW2	GPIO エクスパンダーに接続(ON: Low、OFF: High)
SW3	GPIO エクスパンダーに接続(ON: Low、OFF: High)

14.1.20. SW4 リセットスイッチ

表 14.22 リセットスイッチの接続

部品番号	説明
SW4	外部リセット (ON: リセット状態、OFF： リセット解除)

14.1.21. LED1 3G LED

表 14.23 3G LED の接続

部品番号	説明
LED1	CON10 の 42 ピンに接続(Low: 点灯、High: 消灯)

14.1.22. LED2～LED5 ユーザー LED

表 14.24 ユーザー LED の接続

部品番号	説明
LED2	i.MX257 の NFALE ピンに接続(Low: 消灯、High: 点灯)

部品番号	説明
LED3	i.MX257 の NFCLE ピンに接続(Low: 消灯、High: 点灯)
LED4	GPIO エクスパンダーに接続(Low: 消灯、High: 点灯)
LED5	GPIO エクスパンダーに接続(Low: 消灯、High: 点灯)

14.2. 電気的仕様

14.2.1. 絶対最大定格

表 14.25 絶対最大定格

Parameter	Symbol	Min	Max	Units	Conditions
Power Supply Voltage Range	VIN	-0.3	17	V	
Input Voltage Range	VI	-0.5	OVDD+0.3	V	OVDD=+3.3V_IO、+3.3V @CON1、CON2
Operating Temperature Range	Topr	-10	60	℃	結露なきこと

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。

14.2.2. 推奨動作条件

表 14.26 推奨動作条件

Parameter	Symbol	Min	Typ	Max	Units	Conditions
Power Supply Voltage Range	VIN	8	12	17	V	
Operating Ambient Temperature Range	Ta	-10	25	60	℃	結露なきこと

14.2.3. 入出力インターフェースの電気的仕様

表 14.27 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	+5V	4.75	5	5.25	V	
	+3.6V	3.42	3.6	3.78	V	
	+3.3V_CPU	3.135	3.3	3.465	V	
	+3.3V_IO	3.135	3.3	3.465	V	
	+3.3V	3.135	3.3	3.465	V	

表 14.28 入出力インターフェースの電気的仕様(OVDD = +3.3V_CPU)

Symbol	Parameter	Min	Max	Units	Conditions
VIH	CMOS High-Level Input Voltage	0.7×OVDD	OVDD	V	
VIL	CMOS Low-Level Input Voltage	-0.3	0.3×OVDD	V	
VOH	CMOS High-Level Output Voltage	OVDD-0.15	-	V	IOH = -1mA
		0.8×OVDD	-	V	IOH = Specified Drive
VOL	CMOS Low-Level Output Voltage	-	0.15	V	IOL = 1mA
		-	0.2×OVDD	V	IOL = Specified Drive

Symbol	Parameter	Min	Max	Units	Conditions
IOH_S	High-Level Output Current, Slow Slew Rate	-2.0	-	mA	VOH = 0.8×OVDD, Std Drive
		-4.0	-	mA	VOH = 0.8×OVDD, High Drive
		-8.0	-	mA	VOH = 0.8×OVDD, Max Drive
IOH_F	High-Level Output Current, Fast Slew Rate	-4.0	-	mA	VOH = 0.8×OVDD, Std Drive
		-6.0	-	mA	VOH = 0.8×OVDD, High Drive
		-8.0	-	mA	VOH = 0.8×OVDD, Max Drive
IOL_S	Low-Level Output Current, Slow Slew Rate	2.0	-	mA	VOL = 0.2×OVDD, Std Drive
		4.0	-	mA	VOL = 0.2×OVDD, High Drive
		8.0	-	mA	VOL = 0.2×OVDD, Max Drive
IOL_F	Low-Level Output Current, Fast Slew Rate	4.0	-	mA	VOL = 0.2×OVDD, Std Drive
		6.0	-	mA	VOL = 0.2×OVDD, High Drive
		8.0	-	mA	VOL = 0.2×OVDD, Max Drive
IIN	Input Current (no PU/PD[a])	-	0.1	μA	VI = 0
		-	0.06	μA	VI = OVDD
	Input Current (22kΩPU)	117	184	μA	VI = 0
		0.0001	0.0001	μA	VI = OVDD
	Input Current (47kΩPU)	54	88	μA	VI = 0
		0.0001	0.0001	μA	VI = OVDD
	Input Current (100kΩPU)	25	42	μA	VI = 0
		0.0001	0.0001	μA	VI = OVDD
	Input Current (100kΩPD)	0.0001	0.0001	μA	VI = 0
		25	42	μA	VI = OVDD
ICC	High-impedance Supply Current	-	1.2	μA	VI = 0
		-	1.2	μA	VI = OVDD

[a]PU=Pull Up, PD=Pull Down

14.3. 電源回路の構成

Armadillo-IoT ゲートウェイ ベースボードの電源回路の構成は次のとおりです。CON14 もしくは CON15 からの入力電圧を電源 IC で各電圧に変換し、内部回路および各インターフェースに供給しています。デバイスの電流容量の制限を超えないように、外部機器の接続、供給電源の設計を行ってください。

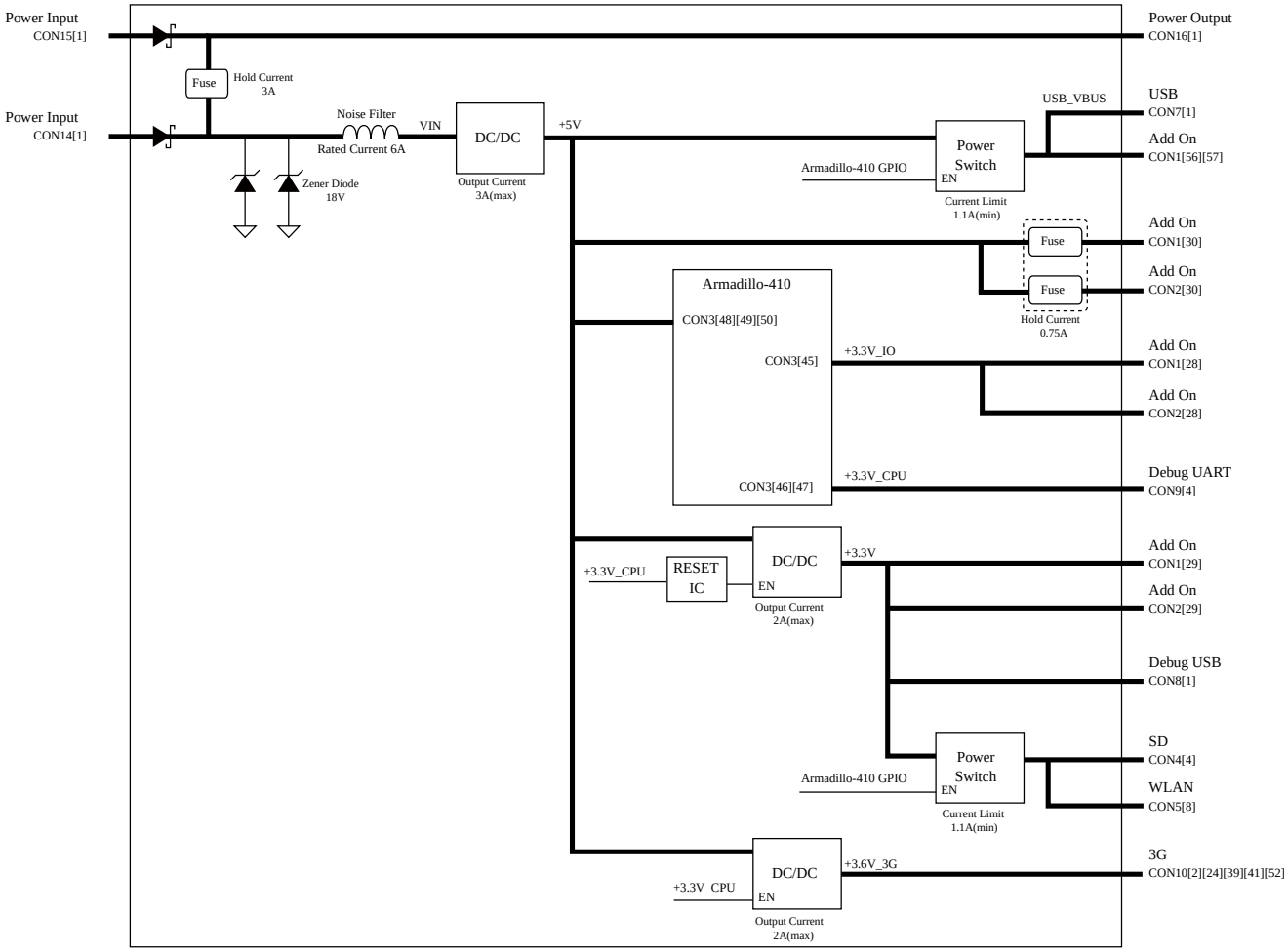
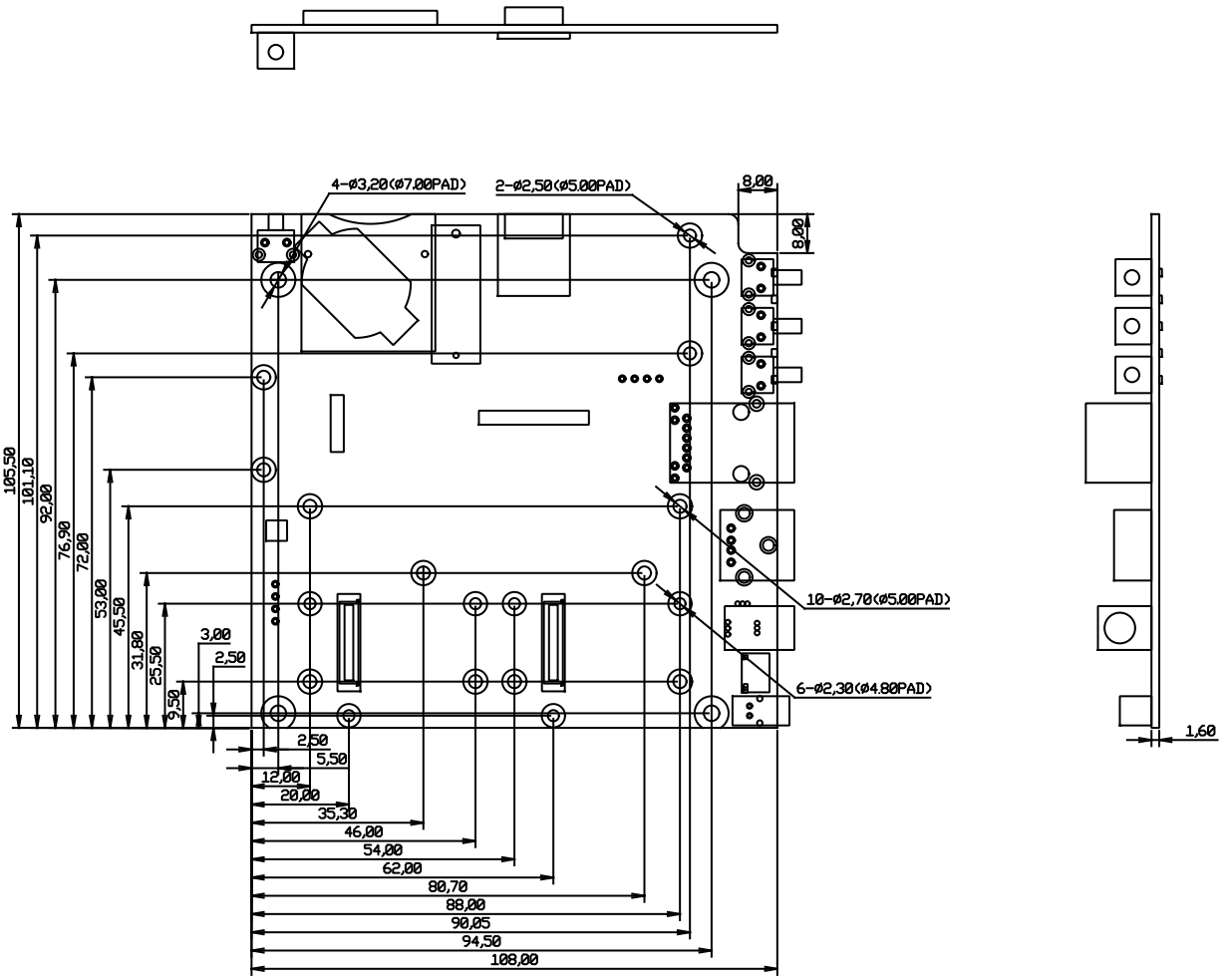


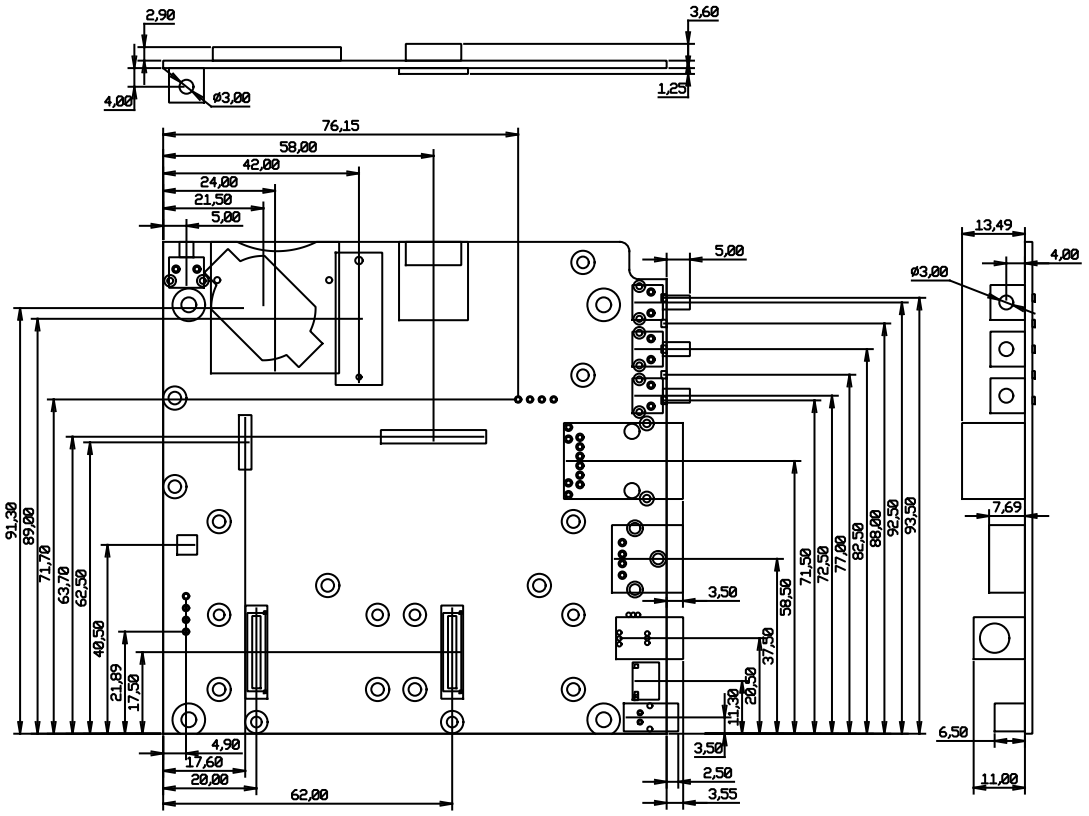
図 14.4 電源回路の構成

14.4. 形状図



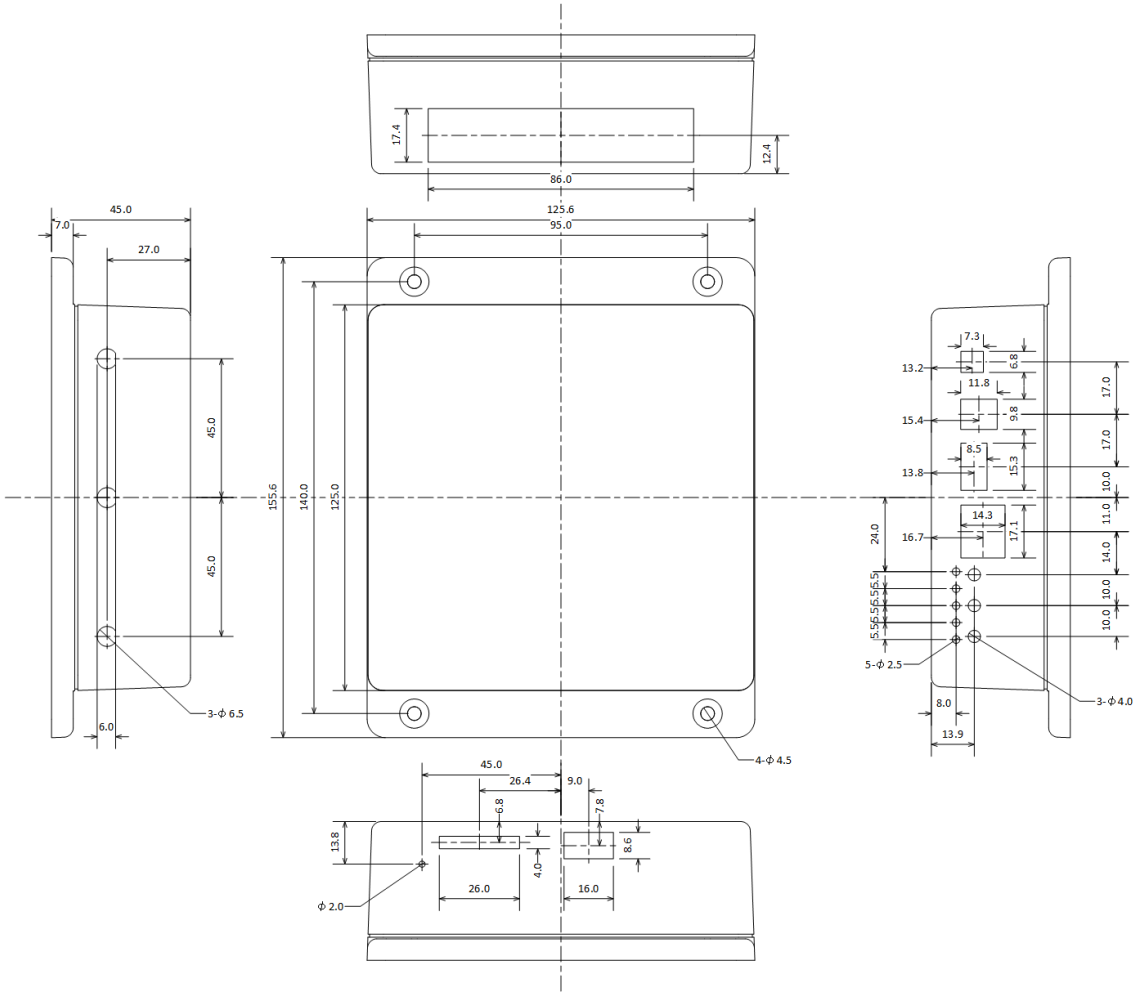
[Unit : mm]

図 14.5 基板形状および固定穴寸法



[Unit : mm]

図 14.6 コネクタ中心寸法



[Unit : mm]

図 14.7 ケース寸法

15. アドオンモジュール

本章では、Armadillo-IoT ゲートウェイのアドオンモジュールについて説明します。アドオンモジュールのラインアップは「表 15.1. Armadillo-IoT ゲートウェイ アドオンモジュール」のとおりです。

表 15.1 Armadillo-IoT ゲートウェイ アドオンモジュール

名称	型番	備考
Armadillo-IoT RS232C アドオンモジュール RS00	OP-AGA-RS00-00	
Armadillo-IoT 絶縁 RS232C/422/485 アドオンモジュール RS01	OP-AGA-RS01-00	開発中
Armadillo-IoT BLE アドオンモジュール BT00	OP-AGA-BT00-00	開発中
Armadillo-IoT EnOcean アドオンモジュール EN00	OP-AGA-EN00-00	開発中
Armadillo-IoT Wi-SUN アドオンモジュール WS00	OP-AGA-WS00-00	開発中

アドオンモジュールの回路図/部品表は「アットマークテクノ ユーザーズサイト」で「購入者向けの限定公開データ」としてダウンロード可能です。

開発中の製品は、仕様が変更になる可能性があります。

15.1. Armadillo-IoT RS232C アドオンモジュール RS00

15.1.1. 概要

Armadillo-IoT RS232C アドオンモジュール RS00(以降、RS232C アドオンモジュールと記載します)は、RS232C レベルのシリアルを 1 ポート追加することができます。また、ベースボードのアドオンインターフェース(CON1、CON2)に実装されている 0.5mm ピッチのコネクタを 2.54 ピッチに変換するテストインターフェースを備えています。

RS232C アドオンモジュールの仕様は次のとおりです。

表 15.2 RS232C アドオンモジュールの仕様

シリアル(UART)	Texas Instruments 製 MAX3243E 搭載 最大データ転送レート: 1Mbps フロー制御ピンあり(CTS、RTS、DTR、DSR、DCD、RI)
電源電圧	DC 3.3V±5%
基板サイズ	40 x 60mm(突起部を除く)

15.1.2. ブロック図

RS232C アドオンモジュールのブロック図は次のとおりです。

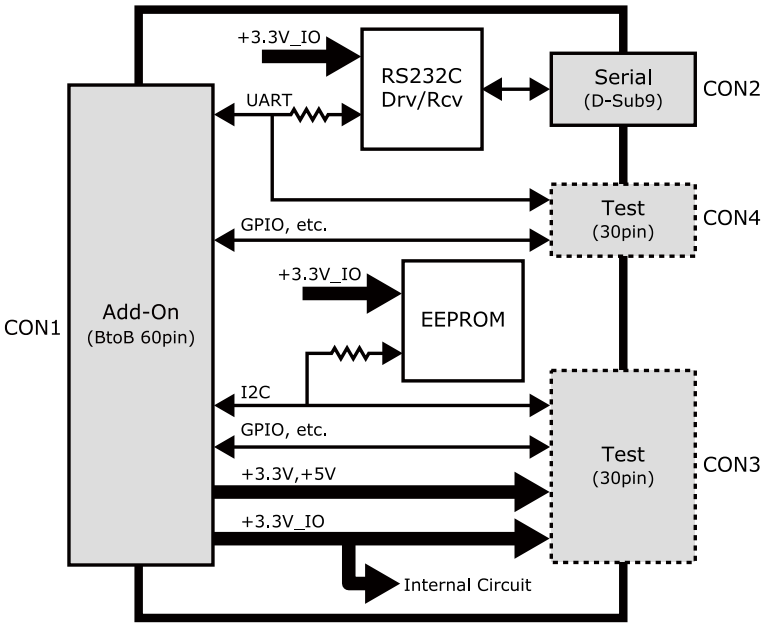


図 15.1 RS232C アドオンモジュール ブロック図

抵抗を取り外すことにより、RS232C レベル変換 IC、EEPROM への配線を切り離すことが可能です。詳細につきましては、回路図をご参照ください。

15.1.3. インターフェース仕様

RS232C アドオンモジュールのインターフェース仕様について説明します。

15.1.3.1. RS232C アドオンモジュール インターフェースレイアウト

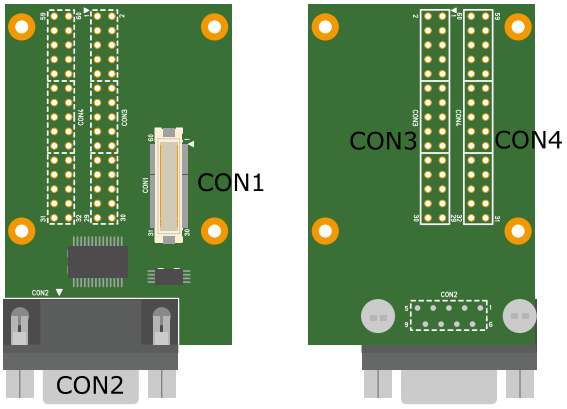


図 15.2 RS232C アドオンモジュール インターフェースレイアウト

表 15.3 搭載コネクタ、スイッチ型番一覧[a]

部品番号	インターフェース名	型番	メーカー
CON1	アドオンインターフェース	DF17(4.0)-60DP-0.5V(57)	HIROSE ELECTRIC
CON2	シリアル(UART)インターフェース	XM2C-0942-132L	OMRON
CON3	テストインターフェース	A1-30PA-2.54DSA(71)	HIROSE ELECTRIC
CON4	テストインターフェース		

[a]色のついたセルの部品は実装していません。実装例を記載しています。

CON3、CON4 は開発用途でご使用ください。

15.1.3.2. CON1 アドオンインターフェース

・ 許容電流: 0.3A(端子 1 本あたり)

表 15.4 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	ADDIO3	In/Out	拡張入出力、CON3 の 3 ピンに接続
4	ADDIO4	In/Out	拡張入出力、CON3 の 4 ピンに接続
5	ADDIO5	In/Out	拡張入出力、CON3 の 5 ピンに接続
6	ADDIO6	In/Out	拡張入出力、CON3 の 6 ピンに接続
7	ADDIO7	In/Out	拡張入出力、CON3 の 7 ピンに接続
8	ADDIO8	In/Out	拡張入出力、CON3 の 8 ピンに接続
9	ADDIO9	In/Out	拡張入出力、CON3 の 9 ピンに接続
10	ADDIO10	In/Out	拡張入出力、CON3 の 10 ピンに接続
11	ADDIO11	In/Out	拡張入出力、CON3 の 11 ピンに接続
12	ADDIO12	In/Out	拡張入出力、CON3 の 12 ピンに接続
13	ADDIO13	In/Out	拡張入出力、CON3 の 13 ピンに接続
14	ADDIO14	In/Out	拡張入出力、CON3 の 14 ピンに接続
15	ADDIO15	In/Out	拡張入出力、CON3 の 15 ピンに接続
16	ADDIO16	In/Out	拡張入出力、CON3 の 16 ピンに接続
17	ADDIO17	In/Out	拡張入出力、CON3 の 17 ピンに接続
18	ADDIO18	In/Out	拡張入出力、CON3 の 18 ピンに接続
19	ADDIO19	In/Out	拡張入出力、CON3 の 19 ピンに接続
20	ADDIO20	In/Out	拡張入出力、CON3 の 20 ピン、EEPROM の SCL ピンに接続
21	ADDIO21	In/Out	拡張入出力、CON3 の 21 ピン、EEPROM の SDA ピンに接続
22	ADDIO22	In/Out	拡張入出力、CON3 の 22 ピンに接続
23	ADDIO23	In/Out	拡張入出力、CON3 の 23 ピンに接続
24	ADDIO24	In/Out	拡張入出力、CON3 の 24 ピンに接続
25	ADDIO25	In/Out	拡張入出力、CON3 の 25 ピンに接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	+3.3V	Power	電源出力(+3.3V)
30	+5V	Power	電源出力(+5V)
31	DETECT	In	EEPROM のアドレスピン、CON4 の 29 ピンに接続
32	ADDIO32	In/Out	拡張入出力、CON4 の 32 ピンに接続
33	ADDIO33	In/Out	拡張入出力、CON4 の 33 ピンに接続

ピン番号	ピン名	I/O	説明
34	ADDIO34	In/Out	拡張入出力、CON4 の 34 ピンに接続
35	ADDIO35	In/Out	拡張入出力、CON4 の 35 ピンに接続
36	ADDIO36	In/Out	拡張入出力、CON4 の 36 ピンに接続
37	ADDIO37	In/Out	拡張入出力、CON4 の 37 ピンに接続
38	ADDIO38	In/Out	送信要求 CON4 の 38 ピン、レベル変換 IC を経由して CON2 の 7 ピンに接続
39	ADDIO39	In/Out	送信可能 CON4 の 39 ピン、レベル変換 IC を経由して CON2 の 8 ピンに接続
40	ADDIO40	In/Out	送信データ CON4 の 40 ピン、レベル変換 IC を経由して CON2 の 3 ピンに接続
41	ADDIO41	In/Out	受信データ CON4 の 41 ピン、レベル変換 IC を経由して CON2 の 2 ピンに接続
42	ADDIO42	In/Out	拡張入出力、CON4 の 42 ピンに接続
43	ADDIO43	In/Out	拡張入出力、CON4 の 43 ピンに接続
44	ADDIO44	In/Out	拡張入出力、CON4 の 44 ピンに接続
45	ADDIO45	In/Out	拡張入出力、CON4 の 45 ピンに接続
46	ADDIO46	In/Out	被呼表示 CON4 の 46 ピン、レベル変換 IC を経由して CON2 の 9 ピンに接続
47	ADDIO47	In/Out	キャリア検出 CON4 の 47 ピン、レベル変換 IC を経由して CON2 の 1 ピンに接続
48	ADDIO48	In/Out	データセットレディ CON4 の 48 ピン、レベル変換 IC を経由して CON2 の 6 ピンに接続
49	ADDIO49	In/Out	データ端末レディ CON4 の 49 ピン、レベル変換 IC を経由して CON2 の 4 ピンに接続
50	ADDIO50	In/Out	拡張入出力、CON4 の 50 ピンに接続
51	ADDIO51	In/Out	拡張入出力、CON4 の 51 ピンに接続
52	ADDIO52	In/Out	拡張入出力、CON4 の 52 ピンに接続
53	ADDIO53	In/Out	拡張入出力、CON4 の 53 ピンに接続
54	GND	Power	電源(GND)
55	PMIC_ONOFF	Out	パワーマネジメント IC の ON/OFF 用信号、CON4 の 55 ピンに接続
56	USB_VBUS	Power	USB 電源、CON4 の 56 ピンに接続
57	USB_VBUS	Power	USB 電源、CON4 の 57 ピンに接続
58	GND	Power	電源(GND)
59	EXT_USB_HS_DP	In/Out	USB プラス側信号、CON4 の 59 ピンに接続
60	EXT_USB_HS_DM	In/Out	USB マイナス側信号、CON4 の 60 ピンに接続

15.1.3.3. CON2 シリアルインターフェース

表 15.5 CON2 信号配列

ピン番号	信号名	I/O	機能
1	DCD	In	キャリア検出、レベル変換 IC を経由して CON1 の 47 ピンに接続
2	RXD	In	受信データ、レベル変換 IC を経由して CON1 の 41 ピンに接続
3	TXD	Out	送信データ、レベル変換 IC を経由して CON1 の 40 ピンに接続
4	DTR	Out	データ端末レディ、レベル変換 IC を経由して CON1 の 49 ピンに接続
5	GND	Power	電源(GND)
6	DSR	In	データセットレディ、レベル変換 IC を経由して CON1 の 48 ピンに接続
7	RTS	Out	送信要求、レベル変換 IC を経由して CON1 の 38 ピンに接続
8	CTS	In	送信可能、レベル変換 IC を経由して CON1 の 39 ピンに接続
9	RI	In	被呼表示、レベル変換 IC を経由して CON1 の 46 ピンに接続

15.1.3.4. CON3 テストインターフェース

表 15.6 CON3 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	ADDIO3	In/Out	CON1 の 3 ピンに接続
4	ADDIO4	In/Out	CON1 の 4 ピンに接続
5	ADDIO5	In/Out	CON1 の 5 ピンに接続
6	ADDIO6	In/Out	CON1 の 6 ピンに接続
7	ADDIO7	In/Out	CON1 の 7 ピンに接続
8	ADDIO8	In/Out	CON1 の 8 ピンに接続
9	ADDIO9	In/Out	CON1 の 9 ピンに接続
10	ADDIO10	In/Out	CON1 の 10 ピンに接続
11	ADDIO11	In/Out	CON1 の 11 ピンに接続
12	ADDIO12	In/Out	CON1 の 12 ピンに接続
13	ADDIO13	In/Out	CON1 の 13 ピンに接続
14	ADDIO14	In/Out	CON1 の 14 ピンに接続
15	ADDIO15	In/Out	CON1 の 15 ピンに接続
16	ADDIO16	In/Out	CON1 の 16 ピンに接続
17	ADDIO17	In/Out	CON1 の 17 ピンに接続
18	ADDIO18	In/Out	CON1 の 18 ピンに接続
19	ADDIO19	In/Out	CON1 の 19 ピンに接続
20	ADDIO20	In/Out	CON1 の 20 ピンに接続
21	ADDIO21	In/Out	CON1 の 21 ピンに接続
22	ADDIO22	In/Out	CON1 の 22 ピンに接続
23	ADDIO23	In/Out	CON1 の 23 ピンに接続
24	ADDIO24	In/Out	CON1 の 24 ピンに接続
25	ADDIO25	In/Out	CON1 の 25 ピンに接続
26	ADDIO26	In/Out	CON1 の 26 ピンに接続
27	GND	Power	電源(GND)
28	GND	Power	電源(GND)
29	+3.3V	Power	電源(+3.3V)
30	+3.3V_IO	Power	電源(+3.3V_IO)

15.1.3.5. CON4 テストインターフェース

表 15.7 CON4 信号配列

ピン番号	ピン名	I/O	説明
31	DETECT	In/Out	CON1 の 31 ピンに接続
32	ADDIO32	In/Out	CON1 の 32 ピンに接続
33	ADDIO33	In/Out	CON1 の 33 ピンに接続
34	ADDIO34	In/Out	CON1 の 34 ピンに接続
35	ADDIO35	In/Out	CON1 の 35 ピンに接続
36	ADDIO36	In/Out	CON1 の 36 ピンに接続
37	ADDIO37	In/Out	CON1 の 37 ピンに接続
38	ADDIIO38	In/Out	CON1 の 38 ピンに接続
39	ADDIO39	In/Out	CON1 の 39 ピンに接続
40	ADDIO40	In/Out	CON1 の 40 ピンに接続
41	ADDIO41	In/Out	CON1 の 41 ピンに接続
42	ADDIO42	In/Out	CON1 の 42 ピンに接続
43	ADDIO43	In/Out	CON1 の 43 ピンに接続
44	ADDIO44	In/Out	CON1 の 44 ピンに接続

ピン番号	ピン名	I/O	説明
45	ADDIO45	In/Out	CON1 の 45 ピンに接続
46	ADDIO46	In/Out	CON1 の 46 ピンに接続
47	ADDIO47	In/Out	CON1 の 47 ピンに接続
48	ADDIO48	In/Out	CON1 の 48 ピンに接続
49	ADDIO49	In/Out	CON1 の 49 ピンに接続
50	ADDIO50	In/Out	CON1 の 50 ピンに接続
51	ADDIO51	In/Out	CON1 の 51 ピンに接続
52	ADDIO50	In/Out	CON1 の 52 ピンに接続
53	ADDIO53	In/Out	CON1 の 53 ピンに接続
54	GND	Power	電源(GND)
55	PMIC_ONOFF	In/Out	CON1 の 55 ピンに接続
56	USB_VBUS	Power	CON1 の 56 ピンに接続
57	USB_VBUS	Power	CON1 の 57 ピンに接続
58	GND	Power	電源(GND)
59	EXT_USB_HS_DP	In/Out	CON1 の 59 ピンに接続
60	EXT_USB_HS_DM	In/Out	CON1 の 60 ピンに接続

15.2. Armadillo-IoT 絶縁 RS232C/RS422/RS485 アドオンモジュール RS01

15.2.1. 概要

Armadillo-IoT 絶縁 RS232C/RS422/RS485 アドオンモジュール RS01(以降、RS485 アドオンモジュールと記載します)は、電気的に絶縁された RS232C/RS422/RS485 のシリアルを 1 ポート追加することができます。

RS485 アドオンモジュールの仕様は次のとおりです。

表 15.8 RS485 アドオンモジュールの仕様

シリアル(UART)	Exar 製 XR3160E 搭載 RS232C/RS422/RS485 レベル x 1 最大データ転送レート: 1Mbps
スイッチ	RS232C/RS422/RS485 切替用ディップスイッチ
絶縁耐圧	2kV
電源電圧	DC 3.3V±5%
基板サイズ	40 x 60mm(突起部を除く)

15.2.2. ブロック図

RS485 アドオンモジュールのブロック図は次のとおりです。

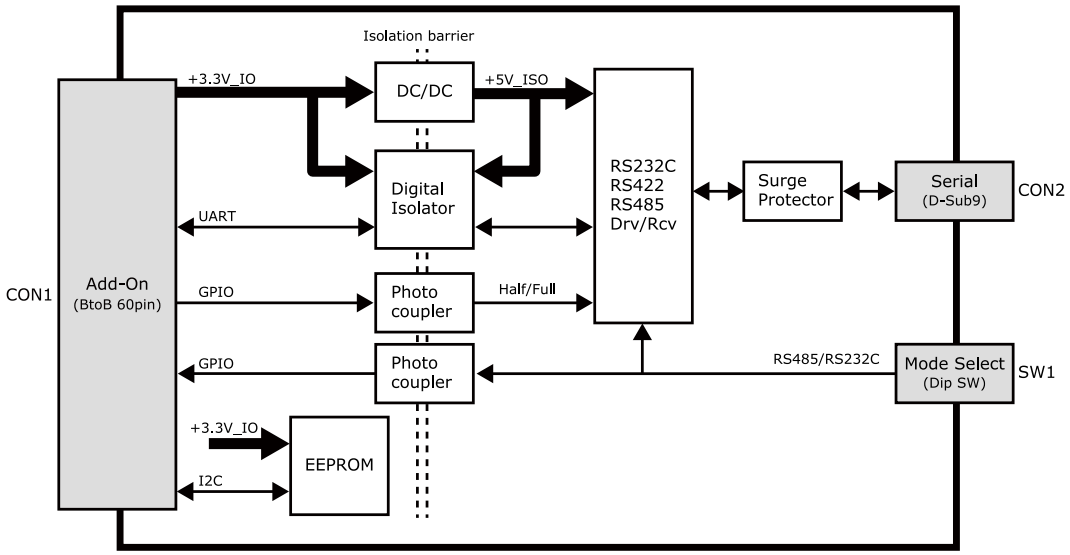


図 15.3 RS485 アドオンモジュール ブロック図

15.2.3. インターフェース仕様

RS485 アドオンモジュールのインターフェース仕様について説明します。

15.2.3.1. インターフェースレイアウト

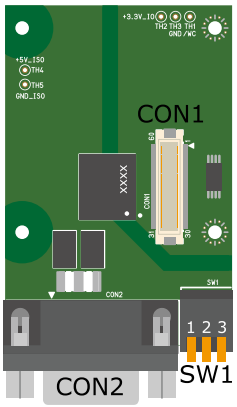


図 15.4 RS485 アドオンモジュール インターフェースレイアウト

表 15.9 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
CON1	アドオンインターフェース	DF17(4.0)-60DP-0.5V(57)	HIROSE ELECTRIC
CON2	シリアル(UART)インターフェース	XM2C-0942-132L	OMRON
SW1	RS232C/RS422/RS485 切替スイッチ	AE6ER-3104	OMRON

15.2.3.2. CON1 アドオンインターフェース

- 許容電流: 0.3A(端子 1 本あたり)

表 15.10 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	NC	-	未接続
4	NC	-	未接続
5	NC	-	未接続
6	NC	-	未接続
7	NC	-	未接続
8	NC	-	未接続
9	NC	-	未接続
10	NC	-	未接続
11	NC	-	未接続
12	NC	-	未接続
13	NC	-	未接続
14	NC	-	未接続
15	NC	-	未接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	NC	-	未接続
20	EEPROM_SCL	In/Out	EEPROM の SCL ピンに接続
21	EEPROM_SDA	In/Out	EEPROM の SDA ピンに接続
22	NC	-	未接続
23	NC	-	未接続
24	NC	-	未接続
25	NC	-	未接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	NC	-	未接続
30	NC	-	未接続
31	DETECT	In	EEPROM のアドレスピンに接続
32	GPIO0	In	半二重/全二重通信の切替信号入力 (High: 半二重、Low: 全二重)
33	NC	-	未接続
34	NC	-	未接続
35	NC	-	未接続
36	NC	-	未接続
37	NC	-	未接続
38	UART_RTS	In	デジタルアイソレータ、レベル変換 IC を経由して CON2 の 7 ピンに接続
39	UART_CTS	Out	デジタルアイソレータ、レベル変換 IC を経由して CON2 の 8 ピンに接続
40	UART_TXD	In	デジタルアイソレータ、レベル変換 IC を経由して CON2 の 3 ピンに接続
41	UART_RXD	Out	デジタルアイソレータ、レベル変換 IC を経由して CON2 の 2 ピンに接続
42	GPIO2	Out	RS232C/RS422/RS485 切替の信号出力、フォトカプラを経由して SW1 に接続 (High: RS422/RS485、Low: RS232C)
43	GPIO3	In	デジタルアイソレータのイネーブルピンに接続
44	NC	-	未接続
45	NC	-	未接続
46	NC	-	未接続
47	NC	-	未接続
48	NC	-	未接続
49	NC	-	未接続
50	NC	-	未接続

ピン番号	ピン名	I/O	説明
51	NC	-	未接続
52	NC	-	未接続
53	NC	-	未接続
54	GND	Power	電源(GND)
55	NC	-	未接続
56	NC	-	未接続
57	NC	-	未接続
58	GND	Power	電源(GND)
59	NC	-	未接続
60	NC	-	未接続

15.2.3.3. CON2 シリアルインターフェース

表 15.11 CON2 信号配列(RS232C に設定時)

ピン番号	信号名	I/O	機能
1	NC	-	未接続
2	RXD	In	受信データ レベル変換 IC、デジタルアイソレータを経由して CON1 の 41 ピンに接続
3	TXD	Out	送信データ レベル変換 IC、デジタルアイソレータを経由して CON1 の 40 ピンに接続
4	NC	-	未接続
5	GND	Power	電源(GND)
6	NC	-	未接続
7	RTS	Out	送信要求 レベル変換 IC、デジタルアイソレータを経由して CON1 の 38 ピンに接続
8	CTS	In	送信可能 レベル変換 IC、デジタルアイソレータを経由して CON1 の 39 ピンに接続
9	NC	-	未接続

表 15.12 CON2 信号配列(RS422/RS485 全二重に設定時)

ピン番号	信号名	I/O	機能
1	NC	-	未接続
2	RX +	In	受信データ(+) レベル変換 IC、デジタルアイソレータを経由して CON1 の 41 ピンに接続
3	TX−	Out	送信データ(−) レベル変換 IC、デジタルアイソレータを経由して CON1 の 40 ピンに接続
4	NC	-	未接続
5	GND	Power	電源(GND)
6	NC	-	未接続
7	TX +	Out	送信データ(+) レベル変換 IC、デジタルアイソレータを経由して CON1 の 38 ピンに接続
8	RX−	In	受信データ(−) レベル変換 IC、デジタルアイソレータを経由して CON1 の 39 ピンに接続
9	NC	-	未接続

表 15.13 CON2 信号配列(RS422/RS485 半二重に設定時)

ピン番号	信号名	I/O	機能
1	NC	-	未接続
2	-	-	
3	DATA−	In/Out	送受信データ(−) レベル変換 IC、デジタルアイソレータを経由して CON1 の 40 ピンに接続
4	NC	-	未接続
5	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
6	NC	-	未接続
7	DATA +	In/Out	送受信データ(+) レベル変換 IC、デジタルアイソレータを経由して CON1 の 38 ピンに接続
8	-	-	
9	NC	-	未接続

15.2.3.4. SW1 RS232C/RS422/RS485 切替スイッチ

表 15.14 SW1 機能

SW1	ON	OFF
1	RS232C	RS422/RS485
2	TX Termination 120Ω ON	TX Termination 120Ω OFF
3	RX Termination 120Ω ON	RX Termination 120Ω OFF

15.3. Armadillo-IoT BLE アドオンモジュール BT00

15.3.1. 概要

Armadillo-IoT BLE アドオンモジュール BT00(以降、BLE アドオンモジュールと記載します)は、Microchip Technology 製の RN4020 を搭載した Bluetooth モジュールです。

BLE アドオンモジュールの仕様は次のとおりです。

表 15.15 BLE アドオンモジュールの仕様

Bluetooth	Microchip Technology 製 RN4020 搭載 Bluetooth 4.1/LE
電源電圧	DC 3.3V±5%
基板サイズ	40 x 50mm(突起部を除く)

15.3.2. ブロック図

BLE アドオンモジュールのブロック図は次のとおりです。

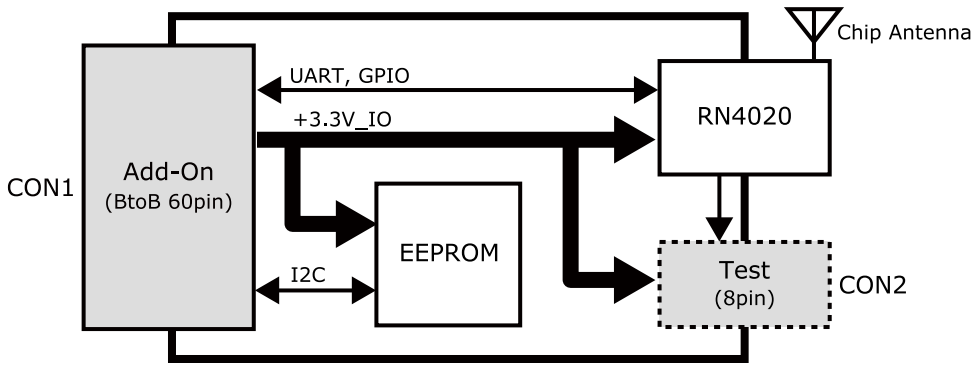


図 15.5 BLE アドオンモジュール ブロック図

15.3.3. インターフェース仕様

BLE アドオンモジュールのインターフェース仕様について説明します。

15.3.3.1. BLE アドオンモジュール インターフェースレイアウト

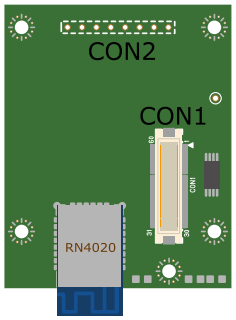


図 15.6 BLE アドオンモジュール インターフェースレイアウト

表 15.16 搭載コネクタ、スイッチ型番一覧[a]

部品番号	インターフェース名	型番	メーカー
CON1	アドオンインターフェース	DF17(4.0)-60DP-0.5V(57)	HIROSE ELECTRIC
CON2	テストインターフェース	A2-8PA-2.54DSA(71)	HIROSE ELECTRIC

[a]色のついたセルの部品は実装していません。実装例を記載しています。

CON2 は開発用途でご使用ください。

15.3.3.2. CON1 アドオンインターフェース

- 許容電流: 0.3A(端子 1 本あたり)

表 15.17 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	NC	-	未接続
4	NC	-	未接続
5	NC	-	未接続
6	NC	-	未接続
7	NC	-	未接続
8	NC	-	未接続
9	NC	-	未接続
10	NC	-	未接続
11	NC	-	未接続
12	NC	-	未接続
13	NC	-	未接続
14	NC	-	未接続
15	NC	-	未接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	NC	-	未接続
20	EEPROM_SCL	In/Out	EEPROM の SCL ピンに接続

ピン番号	ピン名	I/O	説明
21	EEPROM_SDA	In/Out	EEPROM の SDA ピンに接続
22	NC	-	未接続
23	NC	-	未接続
24	NC	-	未接続
25	NC	-	未接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	NC	-	未接続
30	NC	-	未接続
31	DETECT	In	EEPROM のアドレスピンに接続
32	NC	-	未接続
33	NC	-	未接続
34	NC	-	未接続
35	NC	-	未接続
36	NC	-	未接続
37	NC	-	未接続
38	UART_RTS	In	RN4020 の 14 ピンに接続
39	UART_CTS	Out	RN4020 の 18 ピンに接続
40	UART_TXD	In	RN4020 の 6 ピンに接続
41	UART_RXD	Out	RN4020 の 5 ピンに接続
42	GPIO2	Out	RN4020 の 15 ピンに接続
43	GPIO3	Out	RN4020 の 7 ピンに接続
44	NC	-	未接続
45	NC	-	未接続
46	GPIO6	Out	RN4020 の 8 ピンに接続
47	NC	-	未接続
48	NC	-	未接続
49	NC	-	未接続
50	NC	-	未接続
51	NC	-	未接続
52	NC	-	未接続
53	NC	-	未接続
54	GND	Power	電源(GND)
55	NC	-	未接続
56	NC	-	未接続
57	NC	-	未接続
58	GND	Power	電源(GND)
59	NC	-	未接続
60	NC	-	未接続

15.3.3.3. CON2 テストインターフェース

表 15.18 CON2 信号配列

ピン番号	信号名	I/O	機能
1	SPI_MODE	In/Out	RN4020 の 17 ピンに接続
2	+3.3_IO	Power	電源(+3.3V_IO)
3	GND	Power	電源(GND)
4	LED1_PIO1_SCK	In/Out	RN4020 の 10 ピンに接続
5	LED2_PIO2_SS	In/Out	RN4020 の 11 ピンに接続
6	LED3_PIO3_MOSI	In/Out	RN4020 の 12 ピンに接続
7	PIO4_MISO	In/Out	RN4020 の 13 ピンに接続
8	AIO0	In/Out	RN4020 の 4 ピンに接続

15.4. Armadillo-IoT EnOcean アドオンモジュール EN00

15.4.1. 概要

Armadillo-IoT EnOcean アドオンモジュール EN00(以降、EnOcean アドオンモジュールと記載します)は、ROHM 製の BP35A3 を搭載した EnOcean モジュールです。

EnOcean アドオンモジュールの仕様は次のとおりです。

表 15.19 EnOcean アドオンモジュールの仕様

EnOcean	ROHM 製 BP35A3 搭載
電源電圧	DC 3.3V±5%
基板サイズ	40 x 50mm(突起部を除く)

15.4.2. ブロック図

EnOcean アドオンモジュールのブロック図は次のとおりです。

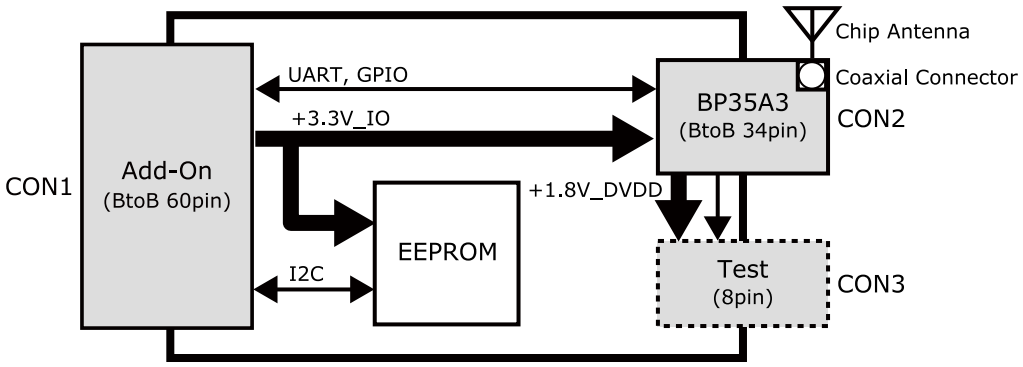


図 15.7 EnOcean アドオンモジュール ブロック図

15.4.3. インターフェース仕様

EnOcean アドオンモジュールのインターフェース仕様について説明します。

15.4.3.1. EnOcean アドオンモジュール インターフェースレイアウト

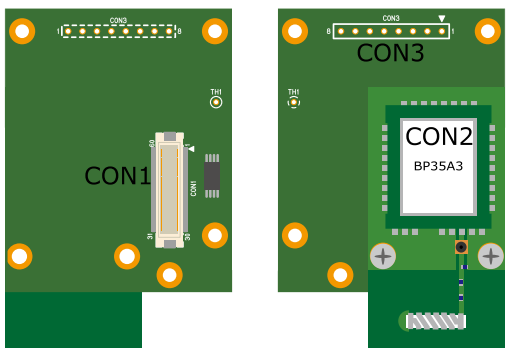


図 15.8 EnOcean アドオンモジュール インターフェースレイアウト

表 15.20 搭載コネクタ、スイッチ型番一覧[a]

部品番号	インターフェース名	型番	メーカー
CON1	アドオンインターフェース	DF17(4.0)-60DP-0.5V(57)	HIROSE ELECTRIC
CON2	EnOcean モジュールインターフェース	AXK6F34347YG-E	Panasonic
CON3	テストインターフェース	A2-8PA-2.54DSA(71)	HIROSE ELECTRIC

[a]色のついたセルの部品は実装していません。実装例を記載しています。

CON3 は開発用途でご使用ください。

15.4.3.2. CON1 アドオンインターフェース

- 許容電流: 0.3A(端子 1 本あたり)

表 15.21 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	NC	-	未接続
4	NC	-	未接続
5	NC	-	未接続
6	NC	-	未接続
7	NC	-	未接続
8	NC	-	未接続
9	NC	-	未接続
10	NC	-	未接続
11	NC	-	未接続
12	NC	-	未接続
13	NC	-	未接続
14	NC	-	未接続
15	NC	-	未接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	NC	-	未接続
20	EEPROM_SCL	In/Out	EEPROM の SCL ピンに接続
21	EEPROM_SDA	In/Out	EEPROM の SDA ピンに接続
22	NC	-	未接続
23	NC	-	未接続
24	NC	-	未接続
25	NC	-	未接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	NC	-	未接続
30	NC	-	未接続
31	DETECT	In	EEPROM のアドレスピンに接続
32	NC	-	未接続
33	NC	-	未接続
34	NC	-	未接続

ピン番号	ピン名	I/O	説明
35	NC	-	未接続
36	NC	-	未接続
37	NC	-	未接続
38	NC	-	未接続
39	NC	-	未接続
40	UART_TXD	In	BP35A3 の 17 ピンに接続
41	UART_RXD	Out	BP35A3 の 16 ピンに接続
42	GPIO2	Out	BP35A3 の 5 ピンに接続
43	NC	-	未接続
44	NC	-	未接続
45	NC	-	未接続
46	NC	-	未接続
47	NC	-	未接続
48	NC	-	未接続
49	NC	-	未接続
50	NC	-	未接続
51	NC	-	未接続
52	NC	-	未接続
53	NC	-	未接続
54	GND	Power	電源(GND)
55	NC	-	未接続
56	NC	-	未接続
57	NC	-	未接続
58	GND	Power	電源(GND)
59	NC	-	未接続
60	NC	-	未接続

15.4.3.3. CON3 テストインターフェース

表 15.22 CON3 信号配列

ピン番号	信号名	I/O	機能
1	WXIDIO	In/Out	BP35A3 の 2 ピンに接続
2	WXODIO	In/Out	BP35A3 の 1 ピンに接続
3	GND	Power	電源(GND)
4	PROG_EN	In	BP35A3 の 15 ピンに接続
5	SCSEDIO0	In/Out	BP35A3 の 14 ピンに接続
6	SCLKDIO1	In/Out	BP35A3 の 13 ピンに接続
7	WSDADIO2	In/Out	BP35A3 の 12 ピンに接続
8	RSDADIO3	In/Out	BP35A3 の 11 ピンに接続

15.5. Armadillo-IoT Wi-SUN アドオンモジュール WS00

15.5.1. 概要

Armadillo-IoT Wi-SUN アドオンモジュール WS00(以降、Wi-SUN アドオンモジュールと記載します)は、ROHM 製の BP35A1 を搭載した Wi-SUN モジュールです。

Wi-SUN アドオンモジュールの仕様は次のとおりです。

表 15.23 Wi-SUN アドオンモジュールの仕様

Wi-SUN	ROHM 製 BP35A1 搭載
電源電圧	DC 3.3V±5%

基板サイズ	40 x 49mm(突起部を除く)

15.5.2. ブロック図

Wi-SUN アドオンモジュールのブロック図は次のとおりです。

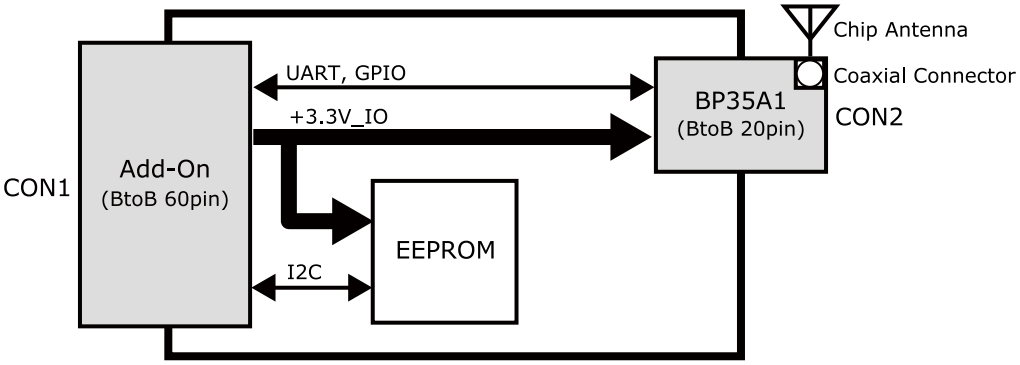


図 15.9 Wi-SUN アドオンモジュール ブロック図

15.5.3. インターフェース仕様

Wi-SUN アドオンモジュールのインターフェース仕様について説明します。

15.5.3.1. Wi-SUN アドオンモジュール インターフェースレイアウト

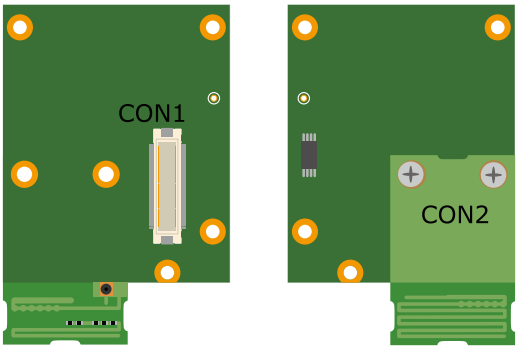


図 15.10 Wi-SUN アドオンモジュール インターフェースレイアウト

表 15.24 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
CON1	アドオンインターフェース	DF17(4.0)-60DP-0.5V(57)	HIROSE ELECTRIC
CON2	Wi-SUN モジュールインターフェース	20P3.0-JMCS-G-B-TF(N)	J.S.T. Mfg.

15.5.3.2. CON1 アドオンインターフェース

- 許容電流: 0.3A(端子 1 本あたり)

表 15.25 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	NC	-	未接続

ピン番号	ピン名	I/O	説明
4	NC	-	未接続
5	NC	-	未接続
6	NC	-	未接続
7	NC	-	未接続
8	NC	-	未接続
9	NC	-	未接続
10	NC	-	未接続
11	NC	-	未接続
12	NC	-	未接続
13	NC	-	未接続
14	NC	-	未接続
15	NC	-	未接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	NC	-	未接続
20	EEPROM_SCL	In/Out	EEPROM の SCL ピンに接続
21	EEPROM_SDA	In/Out	EEPROM の SDA ピンに接続
22	NC	-	未接続
23	NC	-	未接続
24	NC	-	未接続
25	NC	-	未接続
26	GND	Power	電源(GND)
27	GND	Power	電源(GND)
28	+3.3V_IO	Power	電源出力(+3.3V_IO)
29	NC	-	未接続
30	NC	-	未接続
31	DETECT	In	EEPROM のアドレスピンに接続
32	NC	-	未接続
33	NC	-	未接続
34	NC	-	未接続
35	NC	-	未接続
36	NC	-	未接続
37	NC	-	未接続
38	UART_RTS	In	BP35A1 の 14 ピンに接続
39	UART_CTS	Out	BP35A1 の 15 ピンに接続
40	UART_TXD	In	BP35A1 の 4 ピンに接続
41	UART_RXD	Out	BP35A1 の 3 ピンに接続
42	GPIO2	Out	BP35A1 の 6 ピンに接続
43	GPIO3	Out	BP35A1 の 5 ピンに接続
44	NC	-	未接続
45	NC	-	未接続
46	NC	-	未接続
47	NC	-	未接続
48	NC	-	未接続
49	NC	-	未接続
50	NC	-	未接続
51	NC	-	未接続
52	NC	-	未接続
53	NC	-	未接続
54	GND	Power	電源(GND)
55	NC	-	未接続
56	NC	-	未接続
57	NC	-	未接続

ピン番号	ピン名	I/O	説明
58	GND	Power	電源(GND)
59	NC	-	未接続
60	NC	-	未接続

16. オプション品

本章では、Armadillo-IoT 関連のオプション品について説明します。

表 16.1 Armadillo-IoT 関連のオプション品

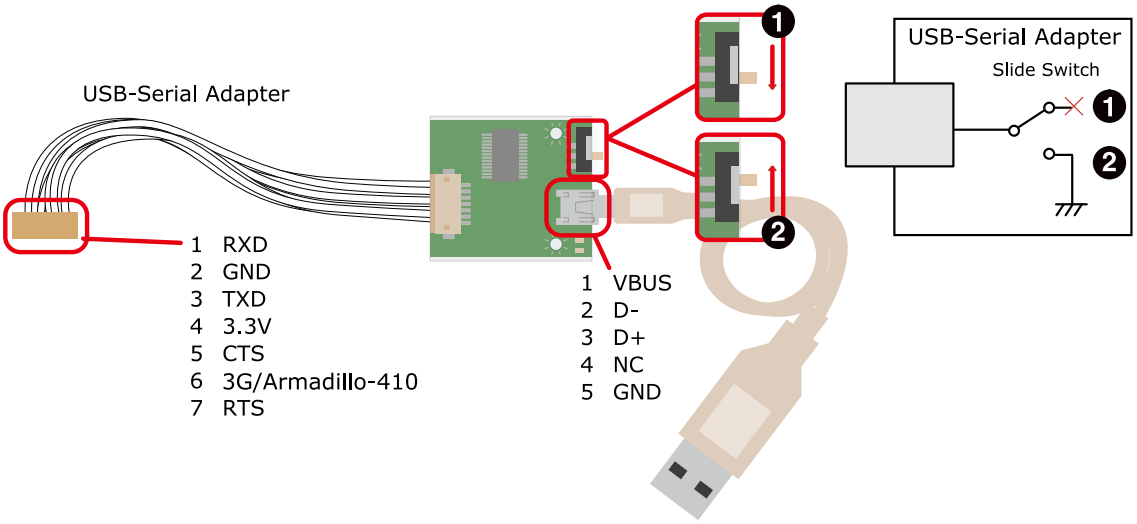
名称	型番
USB シリアル変換アダプタ	SA-SCUSB-00
Armadillo-WLAN(AWL13)	AWL13-U00Z
Armadillo-WLAN 外付けアンテナセット	OP-AWL-ANT-01
AC アダプタ(12V/2.0A φ2.1mm)標準品	OP-AC12V2-00
AC アダプタ(12V/2.0A φ2.1mm)温度拡張品	OP-AC12V3-00

USB シリアル変換アダプタは、試作・開発用の製品です。外観や仕様を予告なく変更する場合があります。

Armadillo-WLAN(AWL13)の詳細につきましては、Armadillo-WLAN 製品ページ [http://armadillo.atmark-techno.com/armadillo-wlan/awl13]をご参照ください。

16.1. USB シリアル変換アダプタ

USB シリアル変換アダプタは、FT232RL を搭載した USB-シリアル変換アダプタです。シリアルの信号レベルは 3.3V CMOS です。デバッグシリアルインターフェース(CON9)に接続して使用することが可能です。スライドスイッチが実装されており、信号線の接続先を切替することができます。



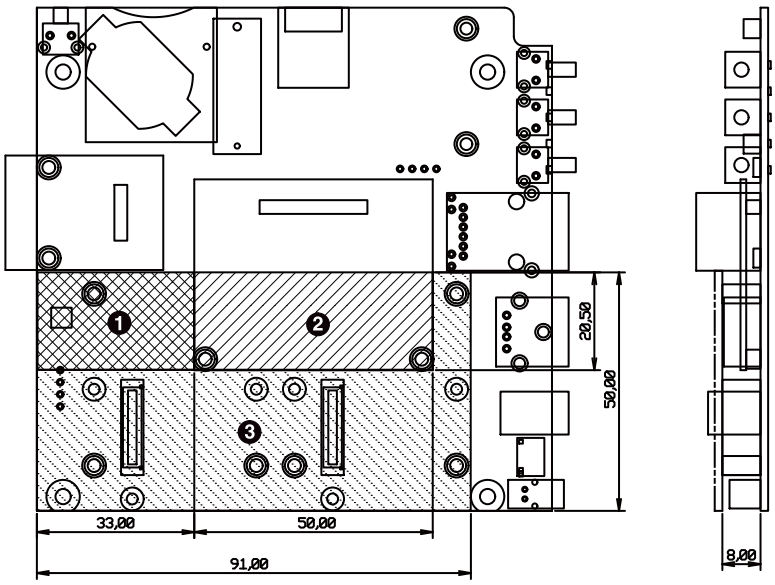
❶ 3G

❷ Armadillo-410 保守モード

図 16.1 USB シリアル変換アダプタの配線

17. 設計情報

アドオンボードを設計する際、Armadillo-410 の高さ、LAN コネクタの位置にご注意ください。



[Unit : mm]

❶ 最大部品高さ 2.0mm

❷ 最大部品高さ 6.7mm

❸ 部品非搭載(アドオンインターフェース搭載コネクタおよびスペーサを除く)

図 17.1 ベースボードの部品高さ

18. Howto

本章では、Armadillo-IoT のソフトウェアをカスタマイズをする方法などについて説明します。

18.1. イメージをカスタマイズする

コンフィギュレーションを変更して Linux カーネル、ユーザーランドイメージをカスタマイズする方法を説明します。

Atmark Dist には様々なアプリケーションやフォントなどが含まれており、コンフィギュレーションによってそれらをイメージに含めたり、外したりすることができます。また、Linux カーネルのコンフィギュレーションの変更を行うこともできます。

手順 18.1 イメージをカスタマイズ

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
awl13-[version].tar.gz        linux-3.4-[version].tar.gz
[ATDE ~]$ tar zxf atmark-dist-[version].tar.gz
[ATDE ~]$ tar zxf awl13-[version].tar.gz
[ATDE ~]$ tar zxf ejdk-[version].tar.gz
[ATDE ~]$ tar zxf linux-2.6.26-at[version].tar.gz
[ATDE ~]$ ls
atmark-dist-[version]         awl13-[version].tar.gz  linux-3.4-[version]
atmark-dist-[version].tar.gz  ejdk[version]           linux-3.4-[version].tar.gz
awl13-[version]               ejdk-[version].tar.gz
```

2. **シンボリックリンクの作成**

   Atmark Dist に、AWL13、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 ../awl13-[version] awl13
[ATDE ~/atmark-dist-[version]]$ ln -s ../linux-2.6.26-at-[version] linux-2.6.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 v1.36.0 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-IoTG-Std" を選択します。

```
 atmark-dist v1.36.0 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 ❶
               --- Select the Product you wish to target
               (Armadillo-IoTG-Std) AtmarkTechno Products ❷


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

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

❶ "AtmarkTechno"を選択します

❷ "Armadillo-IoTG-Std"を選択します

5. **コンフィギュレーション変更対象の指定**

カーネル、ユーザーランドのそれぞれで、コンフィギュレーションの変更を行うかどうかを指定します。

カーネルコンフィギュレーションを変更するには、「Customize Kernel Settings」を選択します。ユーザーランドコンフィギュレーションを変更するには「Customize Vendor/User

Settings」を選択します。その後、"Exit"を選択して「Do you wish to save your new kernel configuration?」で"Yes"とします。

```
 atmark-dist v1.36.0 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)
             [*] Customize Kernel Settings  ❶
             [*] Customize Vendor/User Settings  ❷
             [ ] Update Default Vendor Settings

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

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

❶ カーネルコンフィギュレーションを変更する場合に選択します

❷ ユーザーランドコンフィギュレーションを変更する場合に選択します

6. カーネルコンフィギュレーションの変更

「Customize Kernel Settings」を選択した場合は、Linux Kernel Configuration メニューが表示されます。カーネルコンフィギュレーションを変更後、"Exit"を選択して「Do you wish to save your new kernel configuration？<ESC><ESC> to continue.」で"Yes"とし、カーネルコンフィギュレーションを確定します。

```
 .config - Linux Kernel v2.6.26-at21 Configuration
------------------------------------------------------------------------------
                Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
  ---------------------------------------------------------------------

        General setup  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        Floating point emulation  --->
        Userspace binary formats  --->
        Power management options  --->
  ---------------------------------------------------------------------

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

Linux Kernel Configuration メニューで"/"キーを押下すると、カーネルコンフィギュレーションの検索を行うことができます。カーネルコンフィギュレーションのシンボル名(の一部)を入力して"Ok"を選択すると、部分一致するシンボル名を持つカーネルコンフィギュレーションの情報が一覧されます。

7. ユーザーランドコンフィギュレーションの変更

「Customize Vendor/User Settings」を選択した場合は、Userland Configuration メニューが表示されます。アプリケーションのユーザーランドコンフィギュレーションを変更後、"Exit"を選択して「Do you wish to save your new kernel configuration?」で"Yes"とし、ユーザーランドコンフィギュレーションを確定します。

```
 atmark-dist v1.36.0 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 >
```

8. ビルド

コンフィギュレーションの確定後にビルドを行います。ビルドは"make"コマンドを実行します。

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

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カスタマイズされたイメージファイルが作成されています。Armadillo-IoT 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
```

19. 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/

19.1. 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[1]」をダウンロードすることができるようになります。

Armadillo-IoT 購入製品登録

https://users.atmark-techno.com/armadillo-iot/register

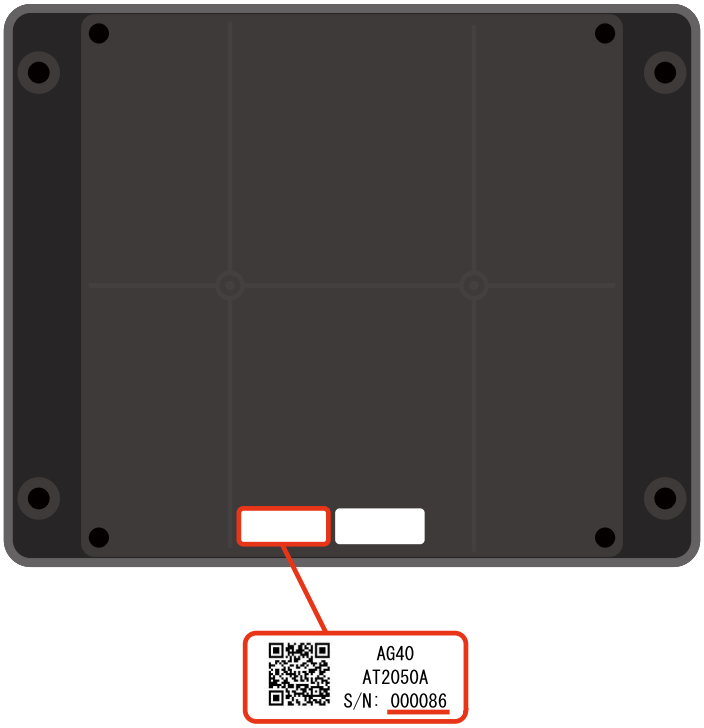
Armadillo-IoT の購入製品登録を行うには、ユーザーズサイトで「シリアル番号」の入力および「正規認証ファイル」のアップロードを行う必要があります

Armadillo-IoT のシリアル番号の確認方法を「19.1.1. シリアル番号を確認する方法」に、Armadillo-IoT から正規認証ファイル(board-info.txt)を取り出す手順を「19.1.2. 正規認証ファイルを取り出す手順」に示します。

19.1.1. 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「000086」であることが確認できます。

[1]アドオンモジュールの回路図データなど



シリアル番号を「Armadillo-IoT 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。

19.1.2. 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。そのファイルをお使いの Web ブラウザを使ってダウンロードしてください。

1. ATDE で minicom を立ち上げて、Armadillo-IoT 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

armadillo-iotg login: root
Password:
[root@armadillo-iotg (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@armadillo-iotg (ttymxc1) ~]# get-board-info
[root@armadillo-iotg (ttymxc1) ~]# ls
board-info.txt
[root@armadillo-iotg (ttymxc1) ~]#
```

3. Armadillo 上で動いている WEB サーバーがアクセスできる場所に、正規認証ファイルを移動し、アクセス権限を変更します。

```
[root@armadillo-iotg (ttymxc1) ~]# mv board-info.txt /home/www-data/
[root@armadillo-iotg (ttymxc1) ~]# chmod a+r /home/www-data/board-info.txt
```

4. minicom を終了させ、お使いの Web ブラウザから、Armadillo の URL にアクセスしてください。下記どちらかの指定方法でアクセス可能です。

   - http://armadillo-iotg.local/board-info.txt
   - http://[Armadillo の IP アドレス]/board-info.txt [2]

   取り出した正規認証ファイルを「Armadillo-IoT 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。

[2] Armadillo の IP アドレスが 192.0.2.10 の場合、http://192.0.2.10/board-info.txt となります。

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2014/12/16	・初版発行

Armadillo-IoT ゲートウェイスタンダードモデル製品マニュアル
Version 1.0.0
2014/12/16

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