Armadillo-840
製品マニュアル

A8400-U00Z
A8400-D00Z
A8401-D00Z

Version 1.3.0
2014/07/31

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-840 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 1.3.0
2014/07/31

目次

図目次

表目次

1. はじめに

Armadillo-840 をお使いいただき、ありがとうございます。

Armadillo-840 は、ルネサスエレクトロニクス製 Cortex-A9 プロセッサ「R-Mobile A1」、DDR3 SDRAM、フラッシュメモリを中心に、HDMI、USB 2.0 ホストポート、Ethernet ポート、SD カードスロットなどを搭載し、且つ、拡張用コネクタには USB 2.0 ホスト/デバイスインターフェース、LCD インターフェース、カメラインターフェース、SD/SDIO インターフェース、SPI、GPIO などといった組み込みシステムに求められる機能を備える小型 CPU ボードです。

Armadillo-840 は、画面出力・表示機能に特化した組み込みプラットフォームとして利用することを想定して設計されています。ネットワークから受けたデータを Full HD で HDMI ディスプレイに出力したり、Qt という GUI フレームワークを使ってユーザーインターフェースを構築することができます。開発セットには、Qt Creator という統合開発環境や開発に必要なソフトウェアが同梱されていますので、ご購入後すぐにシステム開発をスタートすることができます。

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. 本書で必要となる知識と想定する読者

本書は、Armadillo-840 を使って組み込み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 と Linux の組み合わせで、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 は組み込みプラットフォームですので、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

: C や C++がある程度書けるエンジニアを想定しています。また、Linux のコンソールでコマンドを入力し、結果を得ることができることを想定しています。

ハードウェアエンジニア

: 電子工学の基礎をもったエンジニアを想定しています。回路図や部品表を理解できることを想定しています。

1.2. 本書で扱うこと扱わないこと

1.2.1. 扱うこと

本書では、Armadillo-840 に関するソフトウェアとハードウェアの情報を扱っています。組み込み機器の開発時にはハードウェアとソフトウェアが密に関連することが多いため、このマニュアルには両方の情報を合せて記載しました。Armadillo-840 をご利用いただく時の注意事項から、購入時のソフトウェアの状態、動作を確認する手順、設定の変更方法や保存方法、デフォルトで対応しているデバイスやデバイスドライバの情報、起動処理、ソフトウェアのビルド環境構築方法やビルドの方法、基本的な開発方法などを記載しています。Linux が初めての方でも、できるかぎり Armadillo の操作ができるように、コマンド例も記載してます。

また、Armadillo-840 は組み込みプラットフォームであり、最終製品に組み込まれることを前提としています。そのため、Armadillo-840 を組み込むために必要な、インターフェイスの電気的仕様やサイズ、電源やリセット回路の構成、絶対定格や推奨動作条件、基板形状などを扱っています。

さらに、Armadillo サイトやユーザーズサイトなど、Armadillo に関する情報提供サイトについても扱っています。

1.2.2. 扱わないこと

本書では、一般的な Linux のプログラミング、ツール、デバッグ方法、Qt やその他フレームワークの API など、一般的な情報や、すでに詳しい書籍があるものは扱いません。また、(Armadillo が組み込まれる)最終製品に固有な情報・知識も含まれていません。

Armadillo を使った製品化までの一連の開発方法についてもこのマニュアルでは扱いません。こちらに関しては、「実践ガイド [http://armadillo.atmark-techno.com/armadillo-guide]」を参照してください。

1.3. ユーザー限定コンテンツ

Armadillo-840 には、**ご購入ユーザーに限定して公開しているソフトウェアやハードウェア情報**があります。限定コンテンツを取得するには、「22. 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-840 ドキュメント・ダウンロード**
>
> http://armadillo.atmark-techno.com/armadillo-840/downloads

1.5. 本書の構成

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

◆ **はじめにお読みください。**

「1. はじめに」、「2. 注意事項」

◆ **Armadillo-840 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」、「14. Qt - GUI フレームワーク」、「15. AV コーデックミドルウェア」、「16. SD ブートの活用」、「17. JTAG ICE を利用する」

◆ **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。**

「18. ハードウェア仕様」、「19. 基板形状図」、「20. オプション品」

◆ **ソフトウェアのカスタマイズ方法や、様々な機能の使用方法を紹介します。**

「21. 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。**

「22. ユーザー登録」

1.6. 表記について

1.6.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

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

1.6.2. コマンド入力例

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

表 1.2 表示プロンプトと実行環境の関係

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

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

表 1.3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.6.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

1.7. 謝辞

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

2. 注意事項

2.1. 製品本体開封についてのご注意

製品本体を開封する前に、以下の事項をご確認ください。

- 本製品をご利用いただくには、あらかじめ「ソフトウェア使用許諾契約書」 (本製品に同梱されている資料「はじめにお読みください」に記載)に同意いただくことが必要です。はじめに「ソフトウェア使用許諾契約書」をご確認いただき、同意の上で開封してください。

2.2. 評価ボードについてのご注意

評価ボード（「評価セット」の本体ボード、または「開発セット」に評価・開発用として同梱されたボード）は、評価目的、技術開発またはデモンストレーション用途向けです。

以下の事項をご理解・ご了承いただいた上で、ご使用いただきますようお願いいたします。

- 評価ボードは、電子工学に関する技術知識と実務経験を有する技術者によって、良識ある技術的・実務的基準に従って取り扱われることを想定しています。
- 評価ボードは、一般消費者が利用する最終製品において通常要求されるような設計上、販売上、または製造上の保護的措置については未完成品です。
- 弊社は評価ボードについて、弊社の製品保証規定に従いご購入後 1 年間の交換保証のみを行うものとします。
- 弊社は評価ボードのご購入者に対し、上記の交換保証を除き、評価ボードが特定目的に合致することの保証を含む明示的・黙示的な保証、その他ありとあらゆる保証に関する一切の責任を負わないものとします。
- 評価ボードまたはその構成部品に不具合が発生した場合であっても、弊社はその原因の解析を行いません。

2.3. 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。

2.4. 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、FFC 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、HDMI、SD/SDIO、USB、マイク、ヘッドホン)以外へのコネクタ着脱は、絶対に行わないでください。

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

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

衝撃
落下や衝撃などの強い振動を与えないでください。

2.5. 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。

[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。

2.6. 書込み禁止領域について

EEPROM 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。

2.7. 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

2.8. 輸出について

本製品の開発・製造は、原則として日本国内での使用を想定して実施しています。本製品を輸出する際は、輸出者の責任において、輸出関連法令等を遵守し、必要な手続きを行ってください。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。本製品および関連技術は、大量破壊兵器の開発目的、軍事利用その他軍事用途の目的、その他国内外の法令および規則により製造・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。

2.9. 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。

- HDMI、HDMI ロゴ、High-Definition Multimedia Interface は HDMI Licensing, LLC の登録商標です。

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 対応の組み込み機器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-840 とは

Armadillo-840 は、ルネサスエレクトロニクス製 Cortex-A9 プロセッサ「R-Mobile A1」、DDR3 SDRAM、フラッシュメモリを中心に構成された低消費電力なマルチメディア向け小型 CPU ボードです。

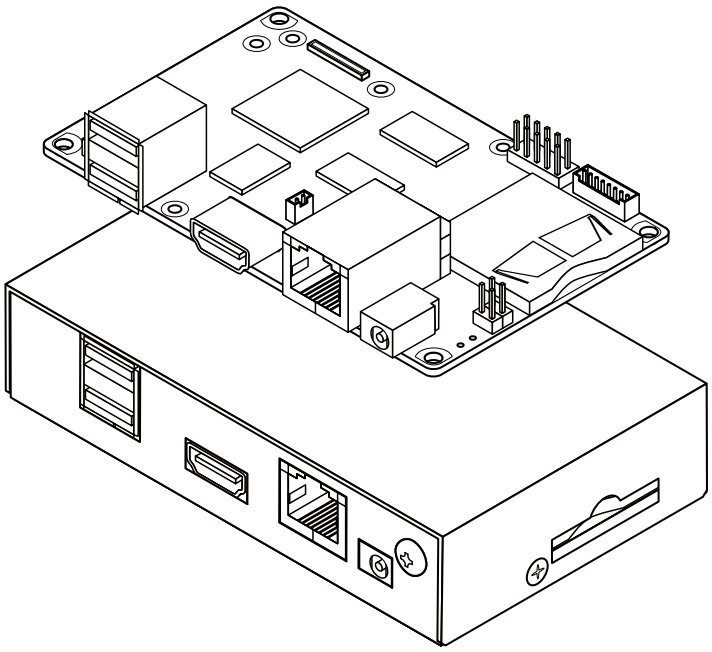


図 3.1 Armadillo-840

◆ 小型・省電力、-20℃〜＋ 70℃まで動作

高機能・高性能・省電力を手のひらサイズで実現した、組み込み CPU ボードです。動作温度範囲は−20℃〜＋ 70℃まで対応しており、使用環境を選びません。また、試作から量産まで安心して使うことができます。

◆ 画面出力と表示機能に特化

HDMI 対応で、Full HD サイズ(1920×1080 ピクセル)の外部出力が可能です。大画面出力のデジタルサイネージなどにも利用することが可能です。また、基板の両面に 100 ピン・60 ピンの拡張インターフェースを搭載。LCD タッチパネルインターフェース(最大 WXGA+、1440×900 ピクセル)、カメラインターフェースにも対応しています。

◆ Full HD サイズで H.264 動画再生

H.264、AAC のエンコード・デコードに対応する「AV コーデックミドルウェア」をボード本体に標準でバンドルしています。Full HD サイズ(1920×1080 ピクセル)での H.264 動画再生などにも対応可能です[1]。 「AV コーデックミドルウェア」はボード本体とアプリケーションの間を補完する、マルチメディア機能に特化したミドルウェアです。R-Mobile A1 に搭載されたリアルタイム制御用のサブ CPU(SH-4A)やアクセラレータ (VCP1、SPU など)によるハードウェア支援を最大限に活用することで、メイン CPU(ARM Cortex-A9)に大きな負荷をかけずに動画再生などの機能を実現することができ、効率的なシステム設計に役立ちます。

AV コーデックミドルウェアは、「アットマークテクノユーザーズサイト [https://users.atmark-techno.com/]」にて、購入者向けに提供しています。AV コーデックミドルウェアをダウンロードするには、前述のユーザーズサイトでユーザーアカウントの作成および購入製品登録を行う必要があります。

[1]量産時は、使用条件によりライセンス料の請求対象となる場合があります。

◆ GUI フレームワーク「Qt (キュート)」対応

C++で実装されたオープンソースの GUI 向けアプリケーション開発フレームワーク「Qt (キュート)」に対応しています。Qt はマルチプラットフォームの GUI ツールキットで、開発資産を有効に活用することが可能です。Armadillo-840 では Qt5 を採用しています。

◆ 統合開発環境

Armadillo-840 用の ATDE には、Qt Creator という統合開発環境が標準で準備されています。これにより、複雑なマルチメディアの処理や、優れたユーザーエクスペリエンスを提供するための GUI 開発を、より簡単に行えるようになっています。また、開発に必要なソフトウェアもすべて同梱されていますので、ご購入後すぐにシステム開発をスタートすることができます。

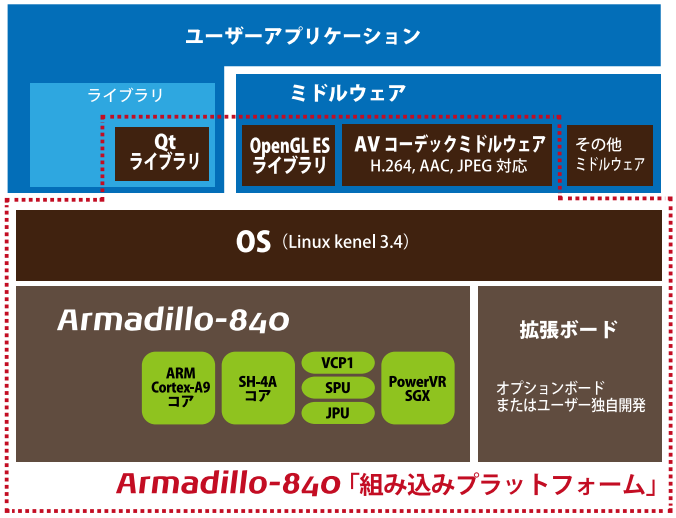


図 3.2 Armadillo-840 の特長

3.2. 仕様

Armadillo-840 の主な仕様は次の通りです。

表 3.1 仕様

プロセッサ	ルネサスエレクトロニクス R-Mobile A1(R8A77404DBA)
CPU コア	メイン: ARM Cortex-A9 - 命令/データキャッシュ 32kByte/32kByte - L2 キャッシュ 256kByte - メディアプロセッシングエンジン(NEON)搭載 - 浮動小数点コプロセッサ(VFPv3)搭載 リアルタイム制御用: SH-4A
システムクロック	CPU コアクロック(ARM Cortex-A9): 792MHz CPU コアクロック(SH-4A): 594MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz 24MHz
RAM	DDR3 SDRAM: 512MByte バス幅 32bit (DDR3-800) Micron Technology MT41K128M16JT-125 IT:K もしくは同等品
フラッシュメモリ	NOR フラッシュメモリ: 128MByte バス幅 16bit Micron Technology PC28F00AP33BFA もしくは同等品
LAN(Ethernet)	RJ-45 x 1 10BASE-T/100BASE-TX AUTO-MDIX 対応

USB	USB Type A コネクタ x 2	
SD/MMC	SD スロット x 1	
HDMI	HDMI Type A コネクタ x 1 解像度最大 1920 x 1080 ピクセル(Full HD) リニア PCM 音声 CEC 対応	
シリアル(UART)	3.3V CMOS x 1 フロー制御ピンあり(CTS、RTS) 最大データ転送レート：1Mbps	
拡張インターフェース 1(C コネクタ)	LCD x 1、カメラ x 1、コンポジットビデオ x 1、SD/MMC x 1、eMMC x 1、USB(Host/Device) x 1、UART x 7、SPI x 2、I2S x 1、I2C x 1、PWM x 4、GPIO x 76、キースキャン x 1[a]	
拡張インターフェース 2(D コネクタ)	カメラ x 2、UART x 5、I2C x 1、PWM x 3、GPIO x 36[a]	
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	
スイッチ	リセットスイッチ	
JTAG	10 ピン(2.54mm ピッチ)[b]	
LED	黄色(面実装) x 2	
電源電圧	DC 5V±5%	
消費電力[c]	アイドル時	約 1.6W
	通常動作時	約 1.8W
	GPU(PowerVR SGX540)、AV コーデックミドルウェア動作時	約 2.6W
使用周囲温度	-20～70℃(ただし結露なきこと)	
基板サイズ	98 x 60mm(突起部を除く)	

[a]各々のチャンネル数は R-Mobile A1 のマルチプレクス機能で、他の機能を無効化して優先的に設定した場合のチャンネル数となります。

[b]オプション品の「8 ピン JTAG 変換ケーブル(Armadillo-400/800 シリーズ対応)」(OP-JC8P25-00)を使用して ARM 標準 20 ピンに変換することが可能です。

[c]LAN、USB、SD、HDMI、シリアルコネクタにケーブル、デバイスを接続した状態での消費電力となります。外部接続機器の消費分は含みません。

3.3. ブロック図

Armadillo-840 のブロック図は次の通りです。

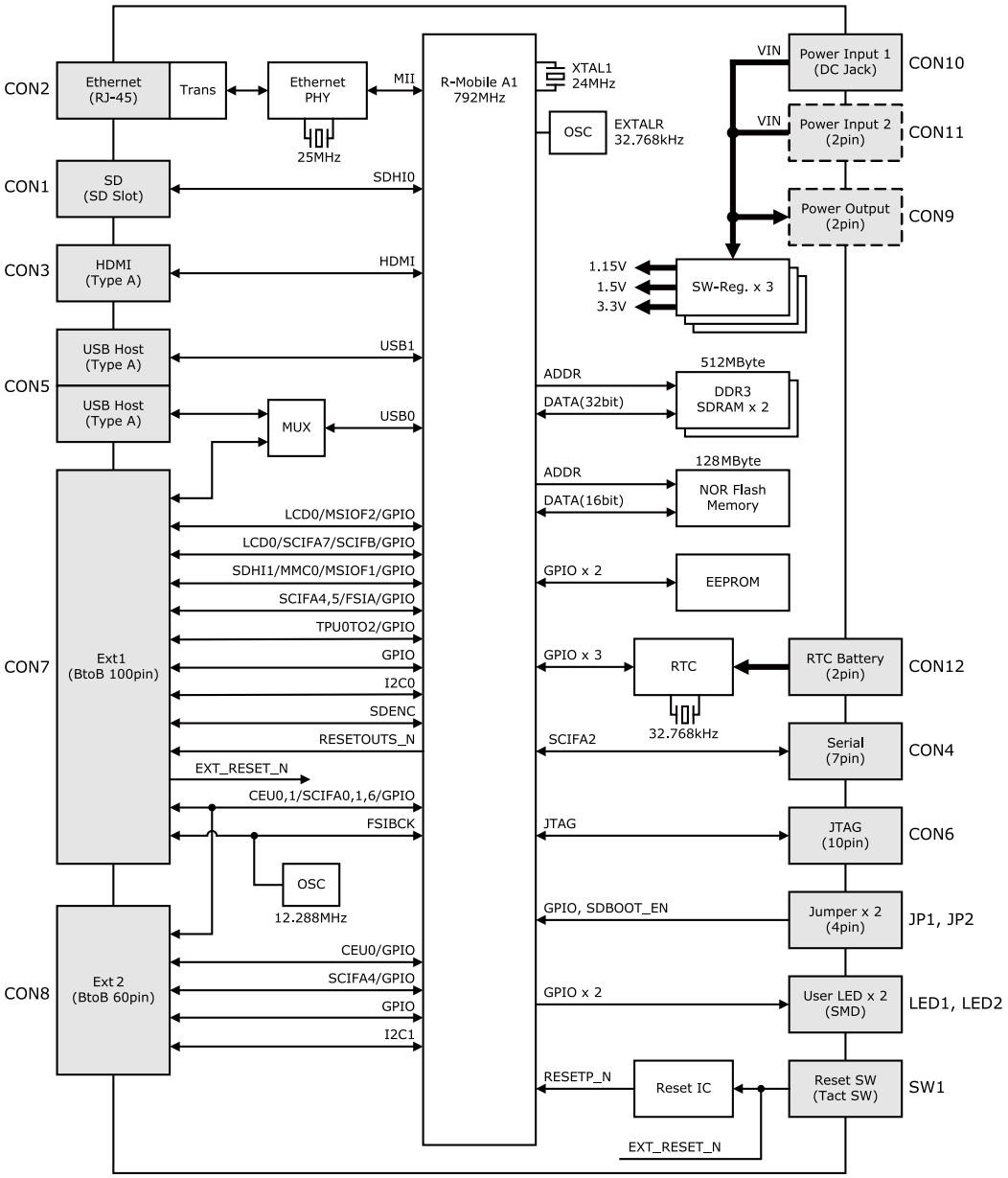


図 3.3 ブロック図

3.4. ソフトウェア構成

Armadillo-840 で動作するソフトウェアの構成について説明します。

Armadillo-840 で利用可能なソフトウェアを「表 3.2. Armadillo-840 で利用可能なソフトウェア」に示します。

表 3.2 Armadillo-840 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が、プロセッサ(R-Mobile A1)の機能により SD カードに配置することもできます。

ソフトウェア	説明
Linux カーネル	バージョン 3.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 SD 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、SD カードなどのストレージに配置することもできます。
Debian GNU/Linux	Debian Project によって作成された Linux ディストリビューションです。パッケージ管理システムを備えているため、Debian Project が提供する豊富なソフトウェアパッケージを簡単に追加することができます。利用する場合は、SD カードなどのストレージデバイスに構築する必要があります。
armhf アーキテクチャ用 OpenGL ES2 ライブラリ	armhf アーキテクチャ用の OpenGL ES2 ライブラリです。GPU(PowerVR SGX540)を利用するために必要です。
AV コーデックミドルウェア	H.264/AVC、AAC デコード及び H.264/AVC、AAC、JPEG エンコードに対応したミドルウェアです。

Armadillo-840 のフラッシュメモリのメモリマップを「表 3.3. フラッシュメモリ メモリマップ」に示します。

表 3.3 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
0x04000000 | 0x0403FFFF	bootloader	256kByte	Hermit-At ブートローダーイメージ
0x04040000 | 0x0407FFFF	config	256kByte	アプリケーションの設定情報など
0x04080000 | 0x040BFFFF	license	256kByte	AV コーデックミドルウェアライセンス
0x040C0000 | 0x044BFFFF	firmware	4MByte	armhf アーキテクチャ用 OpenGL ES2 ライブラリ AV コーデックミドルウェア
0x044C0000 | 0x048BFFFF	kernel	4MByte	Linux カーネルイメージ
0x048C0000 | 0x0BFFFFFF	userland	119.25Mbyte	Atmark Dist ユーザーランドイメージ

3.5. 製品ラインアップ

Armadillo-840 の製品ラインアップは次の通りです。

表 3.4 型番と名称

型番	名称
A8400-D00Z	Armadillo-840 ベーシックモデル開発セット
A8401-D00Z	Armadillo-840 液晶モデル開発セット

3.5.1. Armadillo-840 ベーシックモデル開発セット ベーシックモデル

Armadillo-840 ベーシックモデル開発セットは、Armadillo-840 の基本機能をお使いになるお客様用に作られています。拡張コネクタを使わず、Armadillo-840 単体で HDMI 出力やビデオデコードなどが必要な場合を想定しています。また、ベーシックモデルには、アルミ製の Armadillo-840 専用ケースを付属しています。拡張コネクタを使わずそのまま製品化したい場合にもご利用ください。

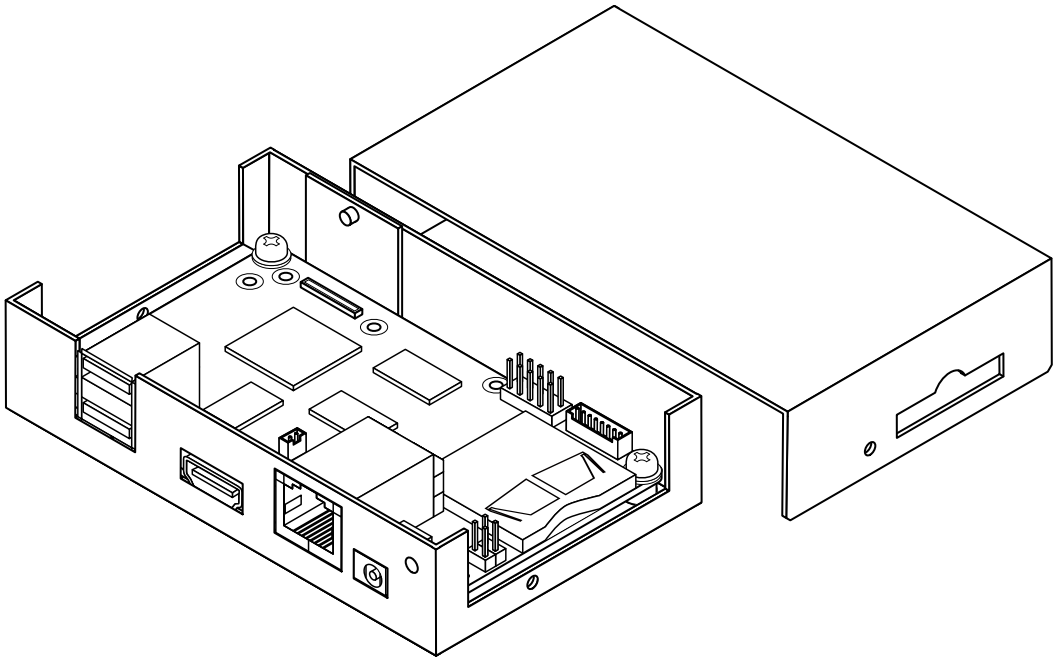


図 3.4 Armadillo-840 ベーシックモデル

表 3.5 Armadillo-840 ベーシックモデル開発セットのセット内容

Armadillo-840
Armadillo-840 オプションケース(金属製)
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)
HDMI ケーブル
AC アダプタ(5V/2.0A、EIAJ#2)
AC アダプタ抜け防止結束バンド類
ジャンパソケット x 2
各種ねじ類
開発用 DVD-ROM

3.5.2. Armadillo-840 液晶モデル開発セット 液晶モデル

Armadillo-840 液晶モデル開発セットは、Armadillo-840 の拡張コネクタに繋がる拡張ボードが付属しています。この拡張ボードは、マルチタッチ対応のタッチパネル LCD やオーディオ入出力、ビデオ出力機能を実現しています。拡張ボードの回路図は、ユーザー限定コンテンツとして、ユーザーズサイトで公開しています。Armadillo-840 拡張ボードを新規開発する際のリファレンスとしてご利用ください。

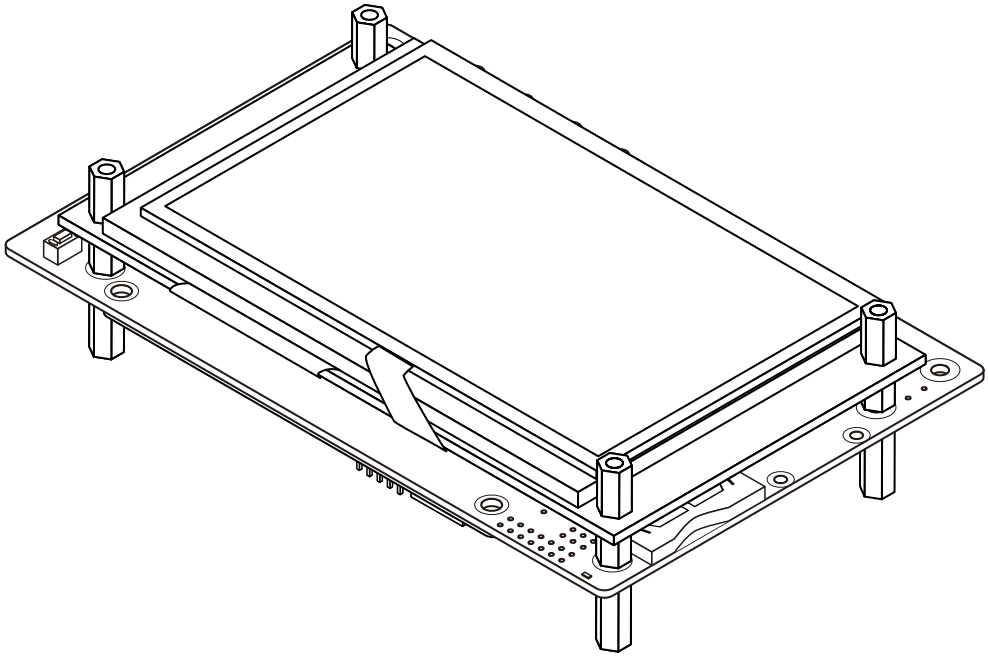


図 3.5 Armadillo-840 液晶モデル

表 3.6 Armadillo-840 液晶モデル開発セットのセット内容

Armadillo-840
Armadillo-840 拡張ボード 01(C コネクタ用)
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) x 2
HDMI ケーブル
AC アダプタ(5V/2.0A、EIAJ#2)
ジャンパソケット x 4
各種ねじ類
開発用 DVD-ROM

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。
HDMI 対応ディスプレイ	HDMI の動作を確認する場合に利用します。
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 マウスと USB キーボード	USB ホストの動作を確認したり、HDMI 対応ディスプレイに表示されたアプリケーションを操作する場合などに利用します。
tar.xz 形式のファイルを展開するソフトウェア	開発/動作確認環境を構築するために利用します。Linux では、tar[1]で展開できます。Windows では、7-Zip や Lhaz などが対応しています。7-Zip は、開発用 DVD に収録されています。
スピーカ又はヘッドホン	サウンドの再生を確認する場合に利用します。
マイク	サウンドの録音を確認する場合に利用します。

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-840 に対応している ATDE は、ATDE5 (ATDE バージョン 5)です。

[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。

ATDE5 は Debian GNU/Linux 7(コードネーム wheezy)をベースに、Armadillo-840 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-840 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。

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

作業用 PC の動作環境(ハードウェア、VMware、ATDE5 の種類など)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。

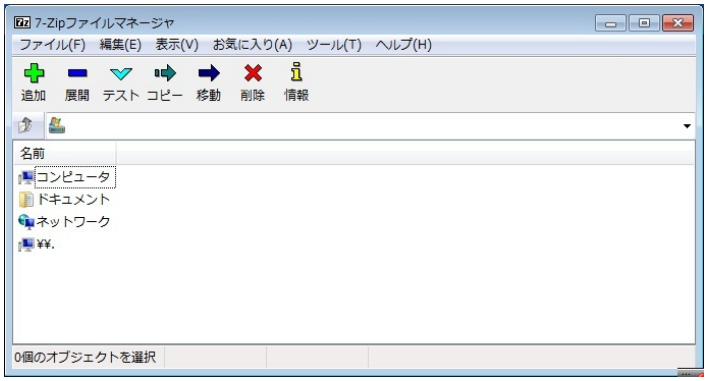
手順 4.1 Windows で ATDE5 のアーカイブ展開する

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。

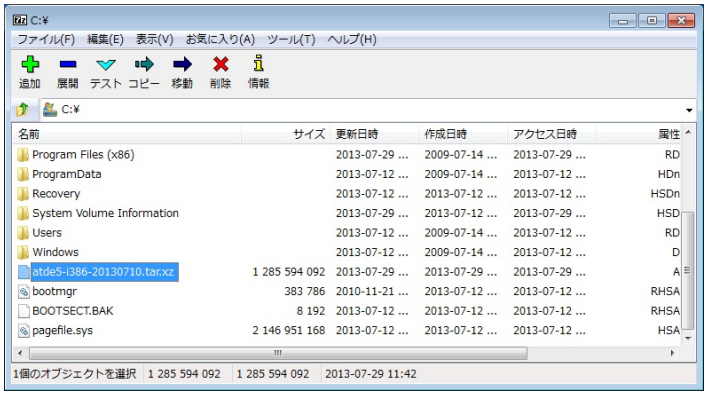
2. **7-Zip の起動**

   7-Zip を起動します。



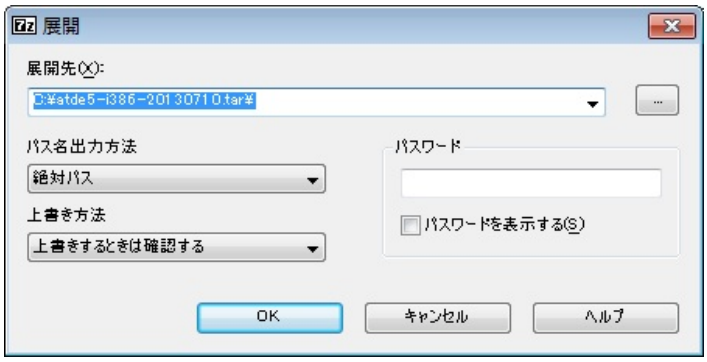
3. **xz 圧縮ファイルの選択**

   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。



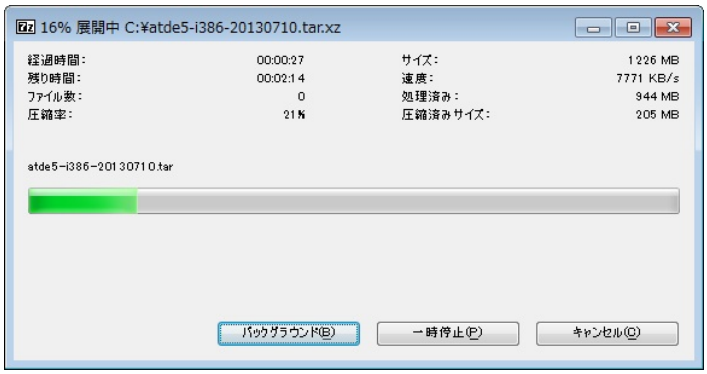
4. **xz 圧縮ファイルの展開先の指定**

   「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**

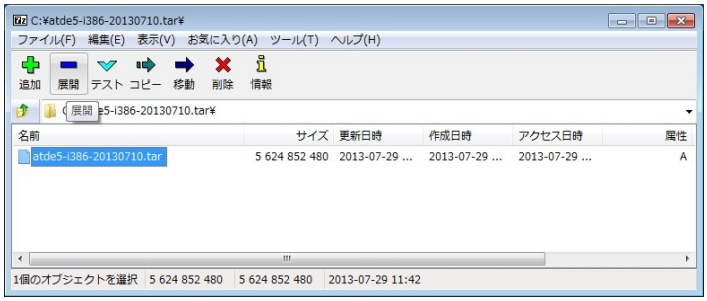
展開が始まります。



6. **tar アーカイブファイルの選択**

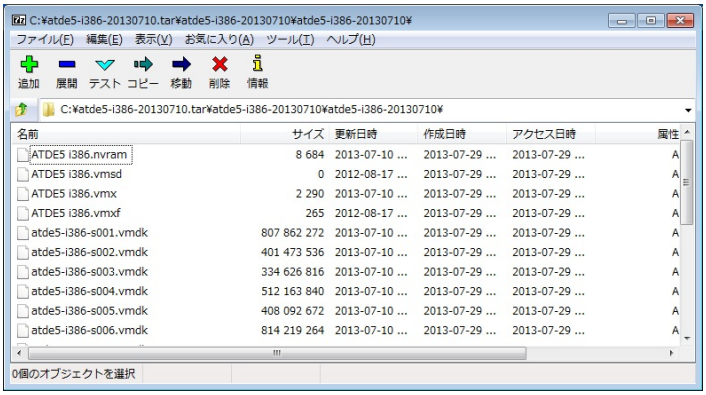
xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。

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-i386-20130710.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-840 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
開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)	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

4.3.1. Armadillo-840

Armadillo-840 のインターフェースレイアウトです。各インターフェースの配置場所等を確認してください。

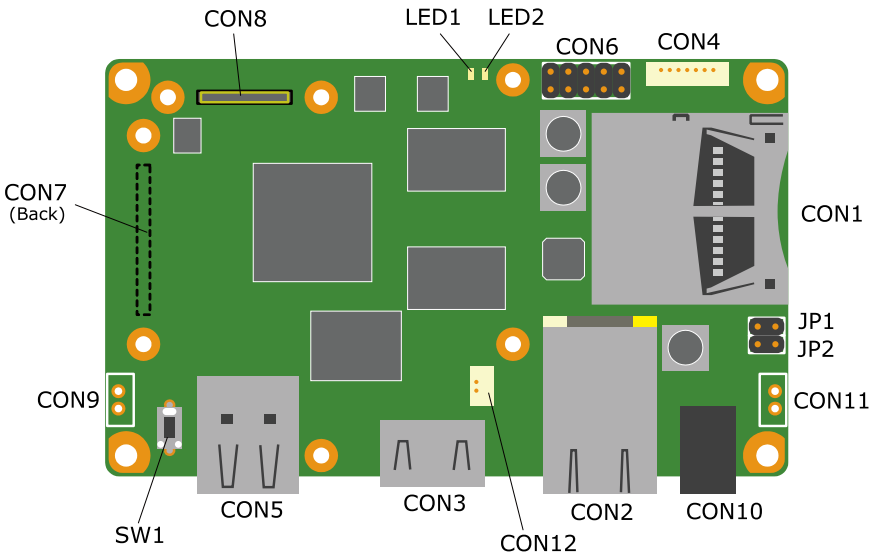


図 4.6 インターフェースレイアウト図

表 4.5 インターフェース内容

部品番号	インターフェース名	形状	備考
CON1	SD インターフェース	SD スロット	
CON2	LAN インターフェース	RJ-45 コネクタ	
CON3	HDMI インターフェース	HDMI Type-A コネクタ	
CON4	シリアルインターフェース	ピンヘッダ 7P(1.25mm ピッチ)	挿抜寿命: 50 回
CON5	USB インターフェース	USB Type-A コネクタ(2 段)	
CON6	JTAG インターフェース	ピンヘッダ 10P(2.54mm ピッチ)	
CON7	拡張インターフェース 1(C コネクタ)	BtoB コネクタ 100P(0.4mm ピッチ)	挿抜寿命: 30 回
CON8	拡張インターフェース 2(D コネクタ)	BtoB コネクタ 60P(0.4mm ピッチ)	挿抜寿命: 30 回
CON9	電源出力インターフェース	ピンヘッダ 2P(2.5mm ピッチ)	
CON10	電源入力インターフェース 1	DC ジャック	対応プラグ: EIAJ#2 ※CON11 と同時使用不可
CON11	電源入力インターフェース 2	ピンヘッダ 2P(2.5mm ピッチ)	※CON10 と電源ライン共通
CON12	RTC 外部バックアップ用電源入力インターフェース	ピンヘッダ 2P(1.25mm ピッチ)	挿抜寿命: 30 回
JP1	設定ジャンパ	ピンヘッダ 2P(2.54mm ピッチ)	オープン: OS 自動起動モード ショート: 保守モード
JP2			オープン: オンボードフラッシュメモリブート ショート: SD(CON1)ブート
LED1	ユーザー LED	LED(黄色、面実装)	
LED2			
SW1	リセットスイッチ	タクトスイッチ	

4.3.2. Armadillo-840 拡張ボード 01(C コネクタ用) 液晶モデル

Armadillo-840 拡張ボード 01(C コネクタ用)のインターフェースレイアウトです。各インターフェースの配置場所等を確認してください。

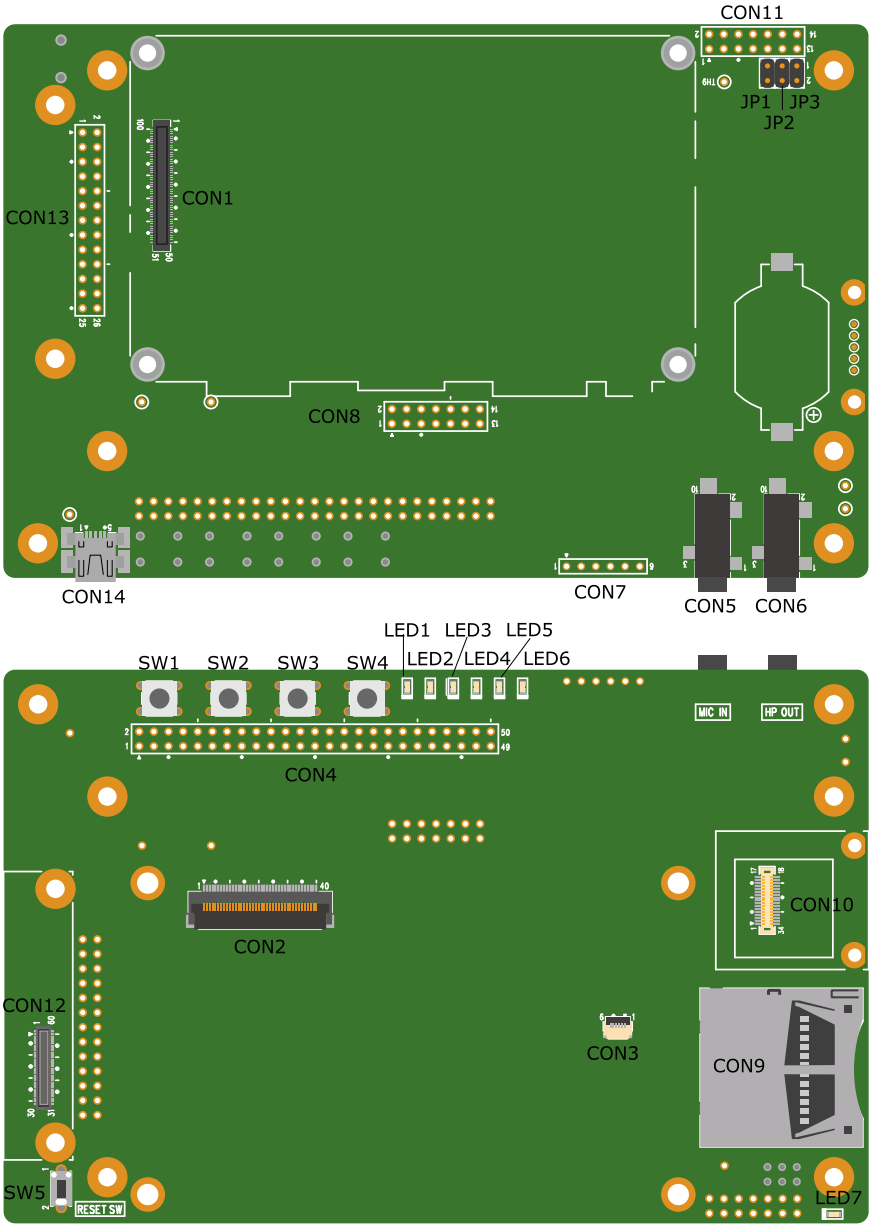


図 4.7 インターフェースレイアウト図

表 4.6 Armadillo-840 拡張ボード 01(C コネクタ用) インターフェース内容

部品番号	インターフェース名	形状	備考
CON1	Armadillo-840 インターフェース	BtoB コネクタ 100P(0.4mm ピッチ)	挿抜寿命: 30 回
CON2	LCD インターフェース	FFC コネクタ 40P(0.5mm ピッチ)	接続可能 LCD: SCF0500133GFR03/ DataImage 挿抜寿命: 20 回
CON3	タッチパネルインターフェース	FFC コネクタ 6P(0.5mm ピッチ)	接続可能 LCD: SCF0500133GFR03/ DataImage 挿抜寿命: 20 回
CON4	拡張インターフェース 1	ピンヘッダ 50P(2.54mm ピッチ)	コネクタ非搭載

部品番号	インターフェース名	形状	備考
CON5	マイク入力インターフェース	ミニジャック(φ3.5mm)	
CON6	ヘッドホン出力インターフェース	ミニジャック(φ3.5mm)	
CON7	オーディオライン/コンポジットビデオ出力インターフェース	ピンヘッダ 6P(2.54mm ピッチ)	コネクタ非搭載
CON8	拡張インターフェース 2	ピンヘッダ 14P(2.54mm ピッチ)	コネクタ非搭載
CON9	SD インターフェース	SD スロット	
CON10	WLAN インターフェース	BtoB コネクタ 34P(0.5mm ピッチ)	接続可能モジュール: AWL13-U00Z/アットマークテクノ 挿抜寿命： 50 回
CON11	拡張インターフェース 3	ピンヘッダ 14P(2.54mm ピッチ)	コネクタ非搭載
CON12	カメラインターフェース	BtoB コネクタ 60P(0.4mm ピッチ)	接続可能モジュール: OP-A810-CAM01-00/アットマークテクノ 挿抜寿命: 30 回
CON13	拡張インターフェース 4	ピンヘッダ 26P(2.54mm ピッチ)	コネクタ非搭載
CON14	USB インターフェース	USB mini B コネクタ	
JP1	ユーザージャンパ	ピンヘッダ 2P(2.54mm ピッチ)	
JP2	設定ジャンパ	ピンヘッダ 2P(2.54mm ピッチ)	
JP3			
SW1	ユーザースイッチ	タクトスイッチ	
SW2			
SW3			
SW4			
SW5	リセットスイッチ	タクトスイッチ	
LED1	ユーザー LED	LED(黄色、面実装)	
LED2			
LED3			
LED4			
LED5			
LED6			
LED7	リセット LED	LED(黄色、面実装)	

4.4. 接続方法

開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)の取扱い上の注意

USB シリアル変換アダプタには電源投入順序があります。Armadillo-840 に接続する際は、以下の手順に従ってご使用ください。接続手順に従わない場合は、USB シリアル変換アダプタが故障する可能性がありますのでご注意ください。

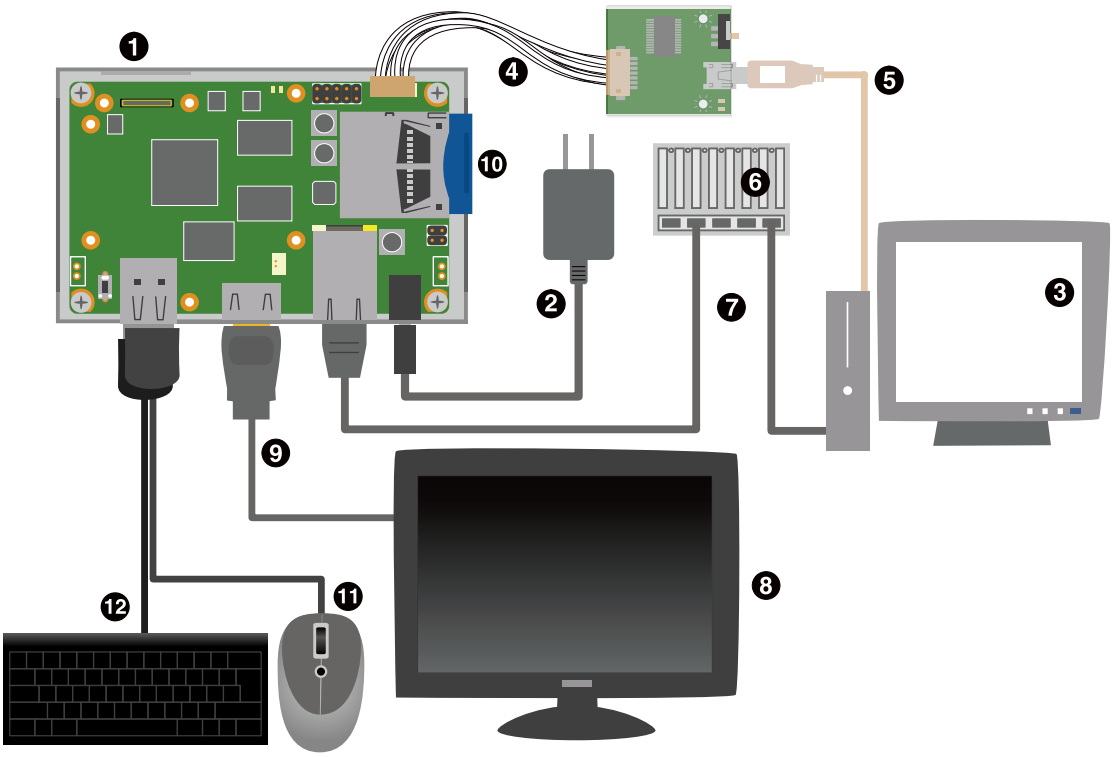
1. 起動中の作業用 PC と USB シリアル変換アダプタを USB2.0 ケーブルで接続します。
2. シリアルインターフェース(Armadillo-840: CON4)に USB シリアル変換アダプタを接続します。
3. 上記接続を確認後、Armadillo-840 に電源を投入します。

また、Armadillo-840 に USB シリアル変換アダプタを接続した状態のまま、作業用 PC または USB シリアル変換アダプタから USB2.0 ケーブル

を抜く場合や作業用 PC をシャットダウンする場合は、Armadillo-840 の電源が切断されていることを確認してから行ってください。

4.4.1. Armadillo-840 ベーシックモデルの接続方法 ベーシックモデル

Armadillo-840 ベーシックモデルと周辺装置の接続例を次に示します。



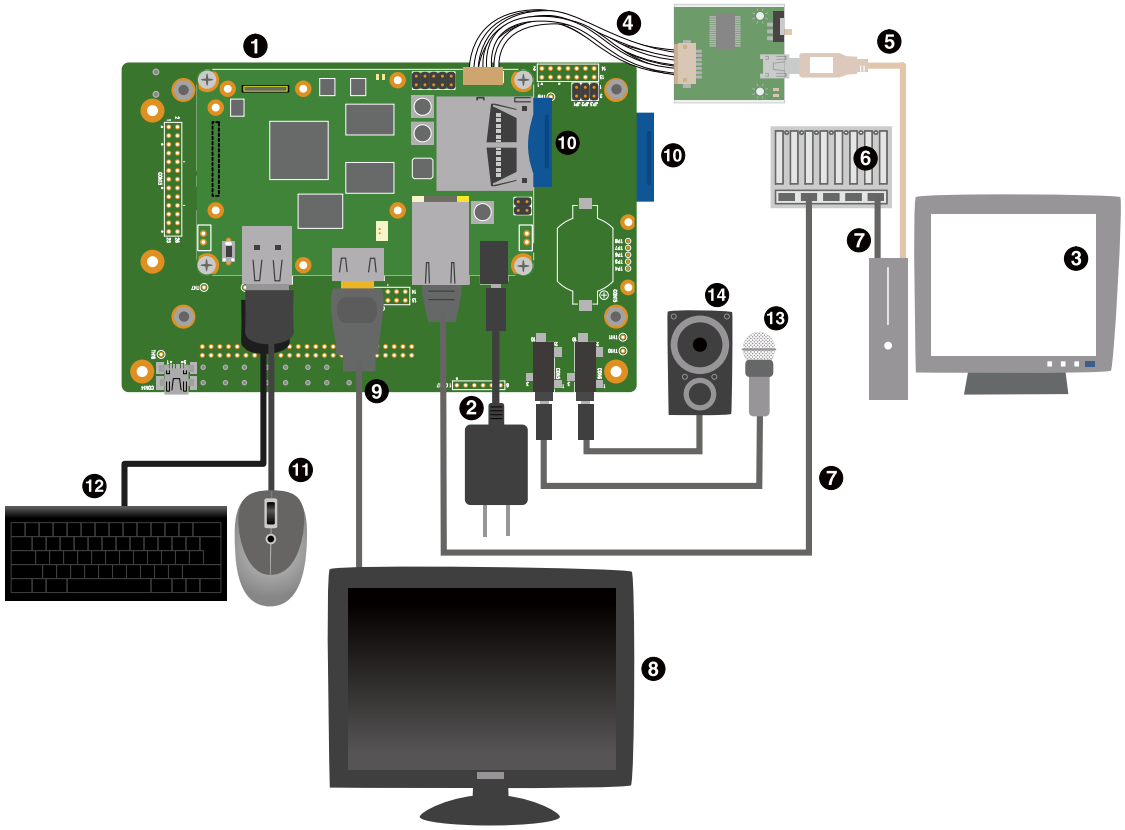
❶ Armadillo-840 ベーシックモデル
❷ AC アダプタ(5V/2.0A EIAJ#2)[4]
❸ 作業用 PC
❹ 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[4]
❺ USB2.0 ケーブル(A-miniB タイプ)[4]
❻ LAN HUB
❼ LAN ケーブル
❽ HDMI 対応ディスプレイ
❾ HDMI ケーブル(A-A タイプ)[4]
❿ SD カード
⓫ USB マウス
⓬ USB キーボード

図 4.8 Armadillo-840 ベーシックモデルの接続例

4.4.2. Armadillo-840 液晶モデルの接続方法 液晶モデル

Armadillo-840 液晶モデルと周辺装置の接続例を次に示します。

[4]Armadillo-840 ベーシックモデル開発セット付属品



❶ Armadillo-840 液晶モデル
❷ AC アダプタ(5V/2.0A EIAJ#2)[5]
❸ 作業用 PC
❹ 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[5]
❺ USB2.0 ケーブル(A-miniB タイプ)[5]
❻ LAN HUB
❼ LAN ケーブル
❽ HDMI 対応ディスプレイ
❾ HDMI ケーブル(A-A タイプ)[5]
❿ SD カード
⓫ USB マウス
⓬ USB キーボード
⓭ スピーカー又はヘッドホン
⓮ マイク

図 4.9 Armadillo-840 液晶モデルの接続例

[5]Armadillo-840 液晶モデル開発セット付属品

4.5. ジャンパピンの設定について

ジャンパの設定を変更することで、Armadillo-840 の動作を変更することができます。ジャンパの機能を「表 4.7. ジャンパの機能」に示します。

表 4.7 ジャンパの機能

ジャンパ	機能	動作
JP1	起動モード設定	オープン: OS を自動起動します。 ショート: ブートローダーを保守モードにします。
JP2	起動デバイス設定	オープン: オンボードフラッシュメモリのブートローダーを起動します。 ショート: SD カードのブートローダーを起動します。

各ジャンパは必要に応じて切り替えの指示があります。ここでは、全てのジャンパをオープンに設定しておきます。

ジャンパピンの位置は「図 4.6. インターフェースレイアウト図」で確認することができます。

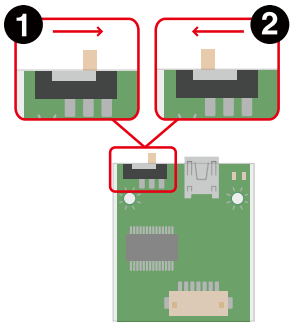
ジャンパのオープン、ショートとは

「オープン」とはジャンパピンにジャンパソケットを接続していない状態です。

「ショート」とはジャンパピンにジャンパソケットを接続している状態です。

4.6. スライドスイッチの設定について

開発用 USB シリアル変換アダプタ (Armadillo-800 シリーズ対応) のスライドスイッチには、Armadillo-840 の JP1 と同じ機能が割り当てられています。



❶　OS 自動起動モード

❷　保守モード

図 4.10 スライドスイッチの設定

4.7. 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。

4.7.1. 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 4.11 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。

4.7.2. 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 4.8. 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 4.8 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 4.12. 入力モードに移行するコマンドの説明」に示します。

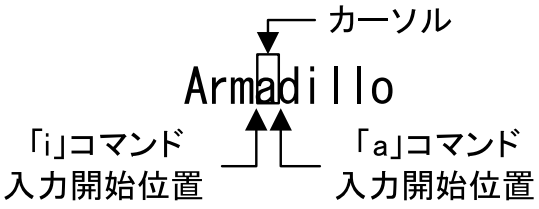


図 4.12 入力モードに移行するコマンドの説明

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、「4.7.4. 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。

4.7.3. 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 4.9. 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 4.9 カーソルの移動コマンド

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

4.7.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.10. 文字の削除コマンド」に示すコマンドを入力します。

表 4.10 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 4.13. 文字を削除するコマンドの説明」に示します。

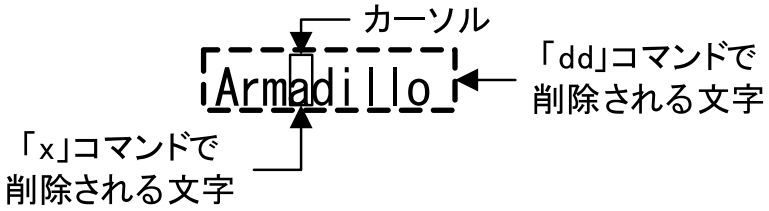


図 4.13 文字を削除するコマンドの説明

4.7.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.11. 保存・終了コマンド」に示します。

表 4.11 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 の電源を投入してください。次のように起動ログがシリアル通信ソフトウェアに表示されます。

```
Hermit-At v3.2.4 (Armadillo-840/nor) compiled at 21:52:47, Jan 27 2014
Uncompressing  kernel.....................................................
.....................................................................
.....................................done.
Uncompressing ramdisk.....................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................done.
```

```
Booting Linux on physical CPU 0
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Linux version 3.4-at6 (atmark@atde5) (gcc version 4.6.3 (Debian 4.6.3-14atmark1)
 ) #1 PREEMPT Mon Jan 27 23:10:37 JST 2014
CPU: ARMv7 Processor [412fc093] revision 3 (ARMv7), cr=10c53c7d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: armadillo840
cma: CMA: reserved 128 MiB at 50000000
Memory policy: ECC disabled, Data cache writeback
bootconsole [early_ttySC2] enabled
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 97536
Kernel command line: console=ttySC2,115200 earlyprintk=sh-sci.2,115200 mem=384M
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
allocated 786432 bytes of page_cgroup
please try 'cgroup_disable=memory' option if you don't want memory cgroups
Memory: 384MB = 384MB total
Memory: 146336k/146336k available, 246880k reserved, 0K highmem
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    vmalloc : 0xd8800000 - 0xff000000   ( 616 MB)
    lowmem  : 0xc0000000 - 0xd8000000   ( 384 MB)
    pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
    modules : 0xbf000000 - 0xbfe00000   (  14 MB)
      .text : 0xc0008000 - 0xc052b000   (5260 kB)
      .init : 0xc052b000 - 0xc0551000   ( 152 kB)
      .data : 0xc0552000 - 0xc058cb60   ( 235 kB)
       .bss : 0xc058cb84 - 0xc05d6bf4   ( 297 kB)
NR_IRQS:16 nr_irqs:16 16
sched_clock: 32 bits at 128 Hz, resolution 7812500ns, wraps every 3489660920ms
Console: colour dummy device 80x30
 sh_cmt_simple.10: used as clock source
 sh_cmt_simple.14: used for clock events
 sh_cmt_simple.14: used for periodic clock events
Calibrating delay loop... 1576.53 BogoMIPS (lpj=6156288)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 512
Initializing cgroup subsys cpuacct
Initializing cgroup subsys memory
Initializing cgroup subsys devices
Initializing cgroup subsys freezer
Initializing cgroup subsys blkio
CPU: Testing write buffer coherency: ok
hw perfevents: enabled with ARMv7 Cortex-A9 PMU driver, 7 counters available
Setting up static identity map for 0x403f3788 - 0x403f37bc
dummy:
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
pfc: r8a7740_pfc handling gpio 0 -> 858
gpiochip_add: registered GPIOs 0 to 858 on device: r8a7740_pfc
CON7: no extension board found.
L310 cache controller enabled
l2x0: 8 ways, CACHE_ID 0x410000c7, AUX_CTRL 0x42440000, Cache size: 262144 B
hw-breakpoint: found 5 (+1 reserved) breakpoint and 1 watchpoint registers.
hw-breakpoint: maximum watchpoint size is 4 bytes.
```

```
bio: create slab <bio-0> at 0
sdhi0: 3300 mV
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.2: using pins 106 (SDA) and 114 (SCL)
i2c-sh_mobile i2c-sh_mobile.0: Runtime PM disabled, clock forced on.
i2c-sh_mobile i2c-sh_mobile.0: I2C adapter 0 with bus speed 100000 Hz
i2c-sh_mobile i2c-sh_mobile.1: Runtime PM disabled, clock forced on.
i2c-sh_mobile i2c-sh_mobile.1: I2C adapter 1 with bus speed 100000 Hz
Linux video capture interface: v2.00
Advanced Linux Sound Architecture Driver Version 1.0.25.
Switching to clocksource sh_cmt_simple.10
 sh_cmt_simple.14: used for oneshot clock events
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP: reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (junk in compressed archive); looks like an initrd
Freeing initrd memory: 105280K
audit: initializing netlink socket (disabled)
type=2000 audit(1.023:1): initialized
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
squashfs: version 4.0 (2009/01/31) Phillip Lougher
NFS: Registering the id_resolver key type
nfs4filelayout_init: NFSv4 File Layout Driver Registering...
msgmni has been set to 747
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 253)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
sh-mobile-hdmi sh-mobile-hdmi: Detected HDMI controller 0x1:0xd5
sh_mobile_lcdc_fb sh_mobile_lcdc_fb.1: Runtime PM disabled, clock forced on.
sh_mobile_lcdc_fb sh_mobile_lcdc_fb.1: registered sh_mobile_lcdc_fb.1/mainlcd as
 1920x1080 32bpp.
sh-dma-engine sh-dma-engine.0: Runtime PM disabled, clock forced on.
sh-dma-engine sh-dma-engine.1: Runtime PM disabled, clock forced on.
sh-dma-engine sh-dma-engine.2: Runtime PM disabled, clock forced on.
sh-dma-engine sh-dma-engine.3: Runtime PM disabled, clock forced on.
SuperH SCI(F) driver initialized
sh-sci sh-sci.0: Runtime PM disabled, clock forced on.
sh-sci.0: ttySC0 at MMIO 0xe6c40000 (irq = 132) is a scifa
console [ttySC2] enabled, bootconsole disabled
console [ttySC2] enabled, bootconsole disabled
sh-sci sh-sci.1: Runtime PM disabled, clock forced on.
sh-sci.1: ttySC1 at MMIO 0xe6c50000 (irq = 133) is a scifa
```

```
sh-sci sh-sci.2: Runtime PM disabled, clock forced on.
sh-sci.2: ttySC2 at MMIO 0xe6c60000 (irq = 134) is a scifa
sh-sci sh-sci.3: Runtime PM disabled, clock forced on.
sh-sci.3: ttySC3 at MMIO 0xe6c70000 (irq = 135) is a scifa
sh-sci sh-sci.4: Runtime PM disabled, clock forced on.
sh-sci.4: ttySC4 at MMIO 0xe6c80000 (irq = 136) is a scifa
sh-sci sh-sci.5: Runtime PM disabled, clock forced on.
sh-sci.5: ttySC5 at MMIO 0xe6cb0000 (irq = 137) is a scifa
sh-sci sh-sci.6: Runtime PM disabled, clock forced on.
sh-sci.6: ttySC6 at MMIO 0xe6cc0000 (irq = 138) is a scifa
sh-sci sh-sci.7: Runtime PM disabled, clock forced on.
sh-sci.7: ttySC7 at MMIO 0xe6cd0000 (irq = 139) is a scifa
sh-sci sh-sci.8: Runtime PM disabled, clock forced on.
sh-sci.8: ttySC8 at MMIO 0xe6c30000 (irq = 140) is a scifb
brd: module loaded
loop: module loaded
r8a7740_cec r8a7740_cec.0: Runtime PM disabled, clock forced on.
physmap platform flash device: 08000000 at 04000000
physmap-flash.0: Found 1 x16 devices at 0x0 in 16-bit bank. Manufacturer ID 0x00
0089 Chip ID 0x008967
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Intel/Sharp Extended Query Table at 0x010A
Using buffer write method
Using auto-unlock on power-up/resume
cfi_cmdset_0001: Erase suspend on write enabled
Creating 6 MTD partitions on "physmap-flash.0":
0x000000000000-0x000000040000 : "bootloader"
0x000000040000-0x000000080000 : "config"
0x000000080000-0x0000000c0000 : "license"
0x0000000c0000-0x0000004c0000 : "firmware"
0x0000004c0000-0x0000008c0000 : "kernel"
0x0000008c0000-0x000008000000 : "userland"
sh-eth sh-eth: Runtime PM disabled, clock forced on.
sh_mii: probed
Base address at 0xe9a00000, 00:11:0c:16:00:d2, IRQ 142.
pegasus: v0.6.14 (2006/09/27), Pegasus/Pegasus II USB Ethernet driver
usbcore: registered new interface driver pegasus
usbcore: registered new interface driver asix
usbcore: registered new interface driver smsc95xx
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
rmobile-ehci-driver rmobile-ehci-driver: R-Mobile EHCI
rmobile-ehci-driver rmobile-ehci-driver: new USB bus registered, assigned bus nu
mber 1
rmobile-ehci-driver rmobile-ehci-driver: irq 266, io mem 0xc6701000
rmobile-ehci-driver rmobile-ehci-driver: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
rmobile-ohci-driver rmobile-ohci-driver: R-Mobile OHCI
rmobile-ohci-driver rmobile-ohci-driver: new USB bus registered, assigned bus nu
mber 2
rmobile-ohci-driver rmobile-ohci-driver: irq 266, io mem 0xc6700000
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
Initializing USB Mass Storage driver...
```

```
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mousedev: PS/2 mouse device common for all mice
rtc-s35390a 2-0030: rtc core: registered rtc-s35390a as rtc0
i2c /dev entries driver
uvcvideo: Unable to create debugfs directory
usbcore: registered new interface driver uvcvideo
USB Video Class driver (1.1.1)
sh_mobile_wdt sh_mobile_wdt.0: Runtime PM disabled, clock forced on.
device-mapper: ioctl: 4.22.0-ioctl (2011-10-19) initialised: dm-devel@redhat.com
sh_mobile_sdhi sh_mobile_sdhi.0: Runtime PM disabled, clock forced on.
sh_mobile_sdhi sh_mobile_sdhi.0: Platform OCR mask is ignored
sh_mobile_sdhi sh_mobile_sdhi.0: mmc0 base at 0xe6850000 clock rate 99 MHz
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
usbcore: registered new interface driver snd-usb-audio
fsi-pcm-audio sh_fsi2: Runtime PM disabled, clock forced on.
sh-mobile-hdmi sh-mobile-hdmi: SH Mobile HDMI Audio Codec
asoc: sh_mobile_hdmi-hifi <-> fsib-dai mapping ok
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP: cubic registered
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 3
registered taskstats version 1
rtc-s35390a 2-0030: setting system clock to 2000-01-11 11:21:52 UTC (947589712)
ALSA device list:
  #0: FSI2B-HDMI
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 105281KiB [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem) on device 1:0.
Freeing init memory: 152K
Mounting proc: done
Starting fsck for root filesystem.
fsck 1.25 (20-Sep-2001)
/dev/ram0: clean, 1569/1976 files, 94050/105281 blocks
Checking root filesystem: done
Remounting root rw: done
Mounting usbfs: done
Mounting sysfs: done
Mounting tmpfs on /dev: done
Cleaning up system: done
Running local start scripts.
Creating mtd devnode: done
Loading /etc/config: done
sh_hdmi_hotplug: 4 callbacks suppressed
Starting udevd: done
Mounting devpts: done
Changing file permissions: done
Configure /home/ftp: done
Starting syslogd: done
Starting klogd: done
Mounting firmware on /opt/firmware: done
Mounting license on /opt/license: done
Mounting tmpfs on /tmp, /var/tmp: done
Mounting ramfs on /home/ftp/pub: done
Creating decoder firmware symlink: done
Creating encoder firmware symlink: done
Setting hostname: done
```

```
Starting PVR Server: done
Starting basic firewall: done
Configuring network interfaces: net eth0: attached phy 0 to driver SMSC LAN8710/
LAN8720
udhcpc (v1.20.2) started
Sending discover...
sh_hdmi_hotplug: 1 callbacks suppressed
PHY: sh-eth-ffffffff:00 - Link is Up - 100/Full
Sending discover...
Sending select for 192.168.1.100...
Lease of 192.168.1.100 obtained, lease time 86400
done
Starting inetd: done
Creating avahi.services: done
Starting avahi.daemon: done
Starting lighttpd: done
Starting sshd: failed
  (sshd: you will be available to use after run '/etc/init.d/sshd keygen')
Running local start script (/etc/config/rc.local).
Starting photoviewer: done
load decoder firmware: done
acm_h264dec: H.264 Decoder of AV Codec Middleware
acm_aacdec: AAC Decoder of AV Codec Middleware

atmark-dist v1.33.0 (AtmarkTechno/Armadillo-840)
Linux 3.4-at6 [armv7l arch]

armadillo840-0 login:
```

図 5.1 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 5.2 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. 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。

6.1.1. 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 6.1 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	手動	無効

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

auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
iface usb0 inet manual
    up ifconfig usb0 up
    post-up zcip usb0 /etc/zcip.script > /dev/null
    down ifconfig usb0 down
```

図 6.1 デフォルト状態の/etc/config/interfaces

usb0 は USB ガジェットで利用することを想定した設定です。

6.1.2. ネットワークの有効化、無効化

有効化されていないインターフェースや一度無効化したインターフェースを再度有効化するには、以下のコマンドを使います。

```
[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

有効化されているインターフェースを無効化するには、以下のコマンドを使います。設定を変更する前には、かならず無効化してください。

```
[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

コマンドの eth0 を usb0 など他のインターフェース名に変更することで、指定したインターフェースの操作をすることが可能です。

6.1.3. ネットワーク設定の変更方法

Armadillo のネットワーク設定の変更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 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。

6.1.3.1. 固定 IP アドレスに設定する

「表 6.2. 固定 IP アドレス設定例」に示す内容に設定変更するには、vi エディタで/etc/config/interfaces を、「図 6.4. 固定 IP アドレス設定」のように編集します。

表 6.2 固定 IP アドレス設定例

項目	設定
IP アドレス	192.168.10.10
ネットマスク	255.255.255.0
ネットワークアドレス	192.168.10.0
ブロードキャストアドレス	192.168.10.255
デフォルトゲートウェイ	192.168.10.1

```
[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 192.168.10.10
    netmask 255.255.255.0
    network 192.168.10.0
    broadcast 192.168.10.255
    gateway 192.168.10.1
iface usb0 inet manual
    up ifconfig usb0 up
    post-up zcip usb0 /etc/zcip.script > /dev/null
    down ifconfig usb0 down
```

図 6.4 固定 IP アドレス設定

6.1.3.2. DHCP に設定する

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 manual
    up ifconfig usb0 up
    post-up zcip usb0 /etc/zcip.script > /dev/null
    down ifconfig usb0 down
```

図 6.5 DHCP 設定

6.1.3.3. DNS サーバーを指定する

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 192.168.10.1
```

図 6.6 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。

6.1.4. 接続を確認する

ここでは、変更した IP 設定で正常に通信が可能か確認します。設定を変更した後は、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。下記の例では、通信機器が「192.168.10.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping 192.168.10.20
```

図 6.7 PING 確認

6.1.5. ファイアーウォール

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

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

図 6.8 iptables

6.1.6. 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。

6.1.6.1. 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 6.3 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 192.168.10.10 ❶
Trying 192.168.10.10...
Connected to 192.168.10.10.
Escape character is '^]'.

atmark-dist v1.32.0 (AtmarkTechno/Armadillo-840)
Linux 3.4-at4 [armv7l arch]

armadillo840-0 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 6.9 telnet でリモートログイン

6.1.6.2. 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 6.4 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 192.168.10.10 ❶
Connected to 192.168.10.10.
220 localhost FTP server (GNU inetutils 1.4.1) ready.
Name (192.168.10.10: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 6.10 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 6.11 Armadillo 上でアップロードされたファイルを確認

6.1.6.3. 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://armadillo840-0.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。

[1] Armadillo の IP アドレスが 192.168.10.10 の場合、http://192.168.10.10/ となります。

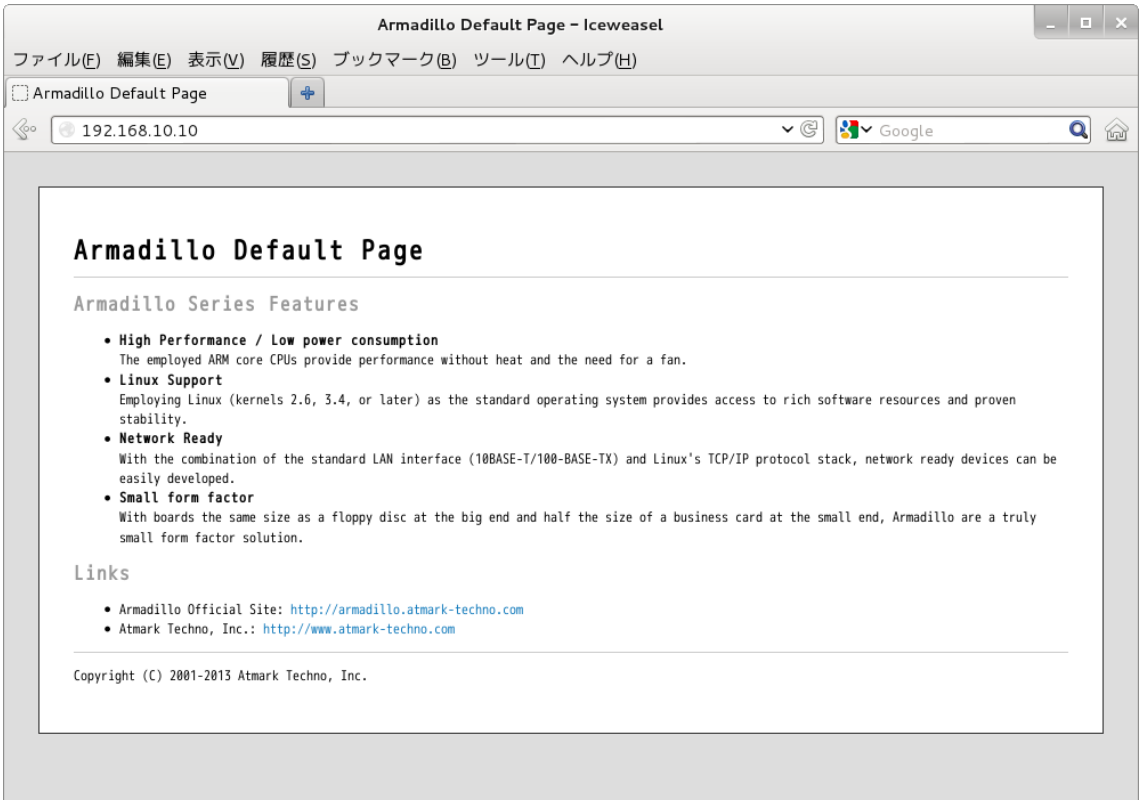


図 6.12 Armadillo トップページ

6.2. ビデオ

Armadillo-840 は画面出力インターフェースを搭載しています。 これらのインターフェースは、フレームバッファデバイス(fb)として扱うことができます。

次に、標準状態で利用可能なフレームバッファデバイスを示します。

フレームバッファデバイス - /dev/fb0

HDMI インターフェース(Armadillo-840: CON3)
解像度: 1920 x 1080[2]
カラーフォーマット: ARGB8888 (32bit)

フレームバッファデバイス - /dev/fb1

LCD インターフェース(拡張ボード 01: CON2)
解像度: 800 x 480
カラーフォーマット: ARGB8888 (32bit)

6.2.1. フレームバッファデバイスにテスト画像を出力

上述した利用可能なフレームバッファデバイスに、テスト画像を出力する方法について説明します。ここでは、テスト画像を生成するために GStreamer の「videotestsrc」を利用します。

[2]接続する HDMI 対応ディスプレイによって異なる場合があります。

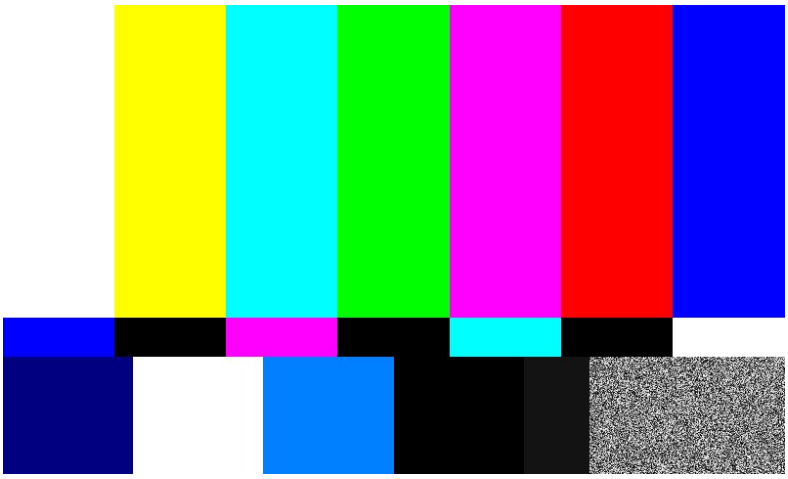


図 6.13 GStreamer のテスト画像

次のようにコマンドを実行するとテスト画像が表示されます。

```
[armadillo ~]# gst-launch-1.0 videotestsrc ! \
                        "video/x-raw,width=1920,height=1080" ! \ ❶
                        fbdevsink device=/dev/fb0 ❷
```

注）本来は一行のコマンドとして実行します。

❶ width, height パラメータには、画面の解像度を指定します。

❷ device パラメータには、出力するフレームバッファデバイスを指定します。

図 6.14 テスト画像を表示するコマンド

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 videotestsrc ! \
                        "video/x-raw-rgb,width=1920,height=1080" ! \
                        fbdevsink device=/dev/fb0
```

コマンドの違いは、インストールされている Gstreamer のバージョンによるものです。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では Gstreamer0.10 がインストールされていましたが、ユーザーランドイメージ romfs-a840-v1.02.img 以降(Atmark Dist v20140131 以降)では Gstreamer1.0 がインストールされています。

フレームバッファデバイスの解像度がわからない場合、次のように fbset コマンドを用いると現在設定されている解像度を表示することができます。

```
[armadillo ~]# fbset -fb /dev/fb0

mode "1920x1080-30"
        # D: 74.250 MHz, H: 33.750 kHz, V: 30.027 Hz
        geometry 1920 1080 1920 2160 32
        timings 13468 148 88 30 4 44 10
        accel false
        rgba 8/16,8/8,8/0,8/24
endmode
```

6.2.2. HDMI - フレームバッファデバイス /dev/fb0

Armadillo-840 の標準状態では、デフォルトアプリケーションが自動的に起動されるようになっています。 このデフォルトアプリケーションは、フレームバッファデバイス /dev/fb0 に描画を行います。そのため、HDMI インターフェース(Armadillo-840: CON3)に HDMI 対応ディスプレイ(本節では単に「ディスプレイ」と称します) を接続し Armadillo-840 を起動した場合には、次のような画面が表示されます。

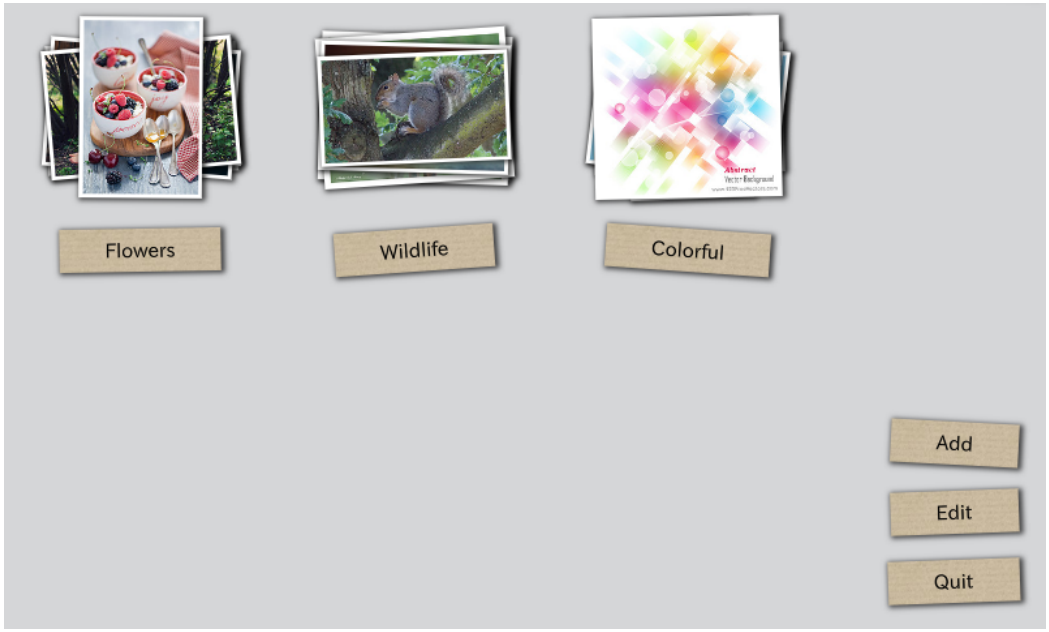


図 6.15 自動起動されるデフォルトアプリケーション画面

このデフォルトアプリケーションは、Qt を利用して作成された「Photo Viewer」というデモアプリケーションです。 指定したキーワードに対応する写真を「Flickr」という写真共有サイトから取得します。スタックされた写真をクリックすると、指定したキーワードの写真が画面に広がります。 インターネットに繋がっていない場合にはデータを取得できないため、 「図 6.15. 自動起動されるデフォルトアプリケーション画面」のように写真を表示することができません。 ネットワークの設定については、「6.1. ネットワーク」を確認してください。

Armadillo-840 では、ディスプレイによって自動的にビデオモードを変更する機能が搭載されています。 この機能によりフレームバッファデバイス /dev/fb0 の解像度は、 Armadillo-840 とディスプレイがサポートで

きる最大の解像度に設定されます。 ディスプレイが接続されていない場合は、フレームバッファデバイスの解像度は FullHD (1920 x 1080 px)に設定されます。 Armadillo-840 を起動した後に、FullHD に対応していないディスプレイを接続すると、 デフォルトアプリケーションが認識している解像度とフレームバッファデバイスに設定されている解像度が異なる場合があり、 正常に画面が表示できなくなることがあります。

正常に画面が表示されない場合は、デフォルトアプリケーション「Photo Viewer」を再起動させると解決することがあります。 以下のようにコマンドを実行すると、アプリケーションを再起動させることができます。

```
[armadillo ~]# killall qmlscene
[armadillo ~]# /etc/config/rc.local
Starting photoviewer: done
```

利用するディスプレイによっては、ビデオモードの自動設定が完了した後であっても画像が表示されない場合があります。 これは、ディスプレイが持つ表示可能なビデオモードを Armadillo-840 が再現できない場合があるためです。

画像が表示されない場合は、次のように該当箇所を変更してください。特定ビデオモードに対する排他処理機能を利用して、表示されないビデオモードを排除することができます。

```
[armadillo ~]# vi /etc/config/configure-fbmode.sh
PARAM=$3

MUST_VMODE_CHANGE=y
IGNORE_MODE_1='1920x1080p-60'
IGNORE_MODE_2='U:'  ❶

fbmode_reconfigure() {
    # p_: path
    # s_: strings
```

❶ 'U:'が含まれるビデオモードを排他します。

変更後、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
```

6.2.3. LCD - フレームバッファデバイス /dev/fb1 液晶モデル

Armadillo-840 の標準状態では、LCD インターフェース(拡張ボード 01: CON2)に対応するフレームバッファデバイス /dev/fb1 へ描画を行うアプリケーションが自動起動するように設定されていないため、LCD の画面は黒一色となります。

画面の出力を確認する場合は、「6.2.1. フレームバッファデバイスにテスト画像を出力」にも記載されている、次のようなコマンドを実行してください。

```
[armadillo ~]# gst-launch-1.0 videotestsrc ! \
                        "video/x-raw,width=800,height=480" ! \
                        fbdevsink device=/dev/fb1
```

図 6.16 LCD にテスト画像を表示するコマンド

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch videotestsrc ! \
                          "video/x-raw-rgb,width=800,height=480" ! \
                          fbdevsink device=/dev/fb1
```

コマンドの違いは、インストールされている Gstreamer のバージョンによるものです。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では Gstreamer0.10 がインストールされていましたが、ユーザーランドイメージ romfs-a840-v1.02.img 以降(Atmark Dist v20140131 以降)では Gstreamer1.0 がインストールされています。

6.2.3.1. バックライトの輝度調整

拡張ボード 01 に搭載された LCD のバックライトは、ソフトウェアで輝度を調整することができます。

LCD のバックライトは、バックライトクラスとして実装されています。 バックライトの輝度を変更するには、/sys/class/backlight/pwm-backlight.0/ディレクトリ以下の次の表に示す sysfs ファイルを使用します。

表 6.5 輝度設定に使用する sysfs ファイル

ファイル	説明
brightness	0(消灯) ～ max_brightness(最高輝度)までの数値を書き込むことで輝度を変更することができます。
max_brightness	brightness に書きこむ数値の最大値(最高輝度 = 255)が読み出せます。

次に、バックライトの輝度を調整する場合のコマンド例を示します。

最高輝度を取得する

```
[armadillo ~]# cat /sys/class/backlight/pwm-backlight.0/max_brightness
255
```

消灯させる

```
[armadillo ~]# echo 0 > /sys/class/backlight/pwm-backlight.0/brightness
```

任意の輝度に変更する (ここでは 「128」 に設定)

```
[armadillo ~]# echo 128 > /sys/class/backlight/pwm-backlight.0/brightness
```

6.3. オーディオ

ここでは、サウンドの再生および録音の方法について説明します。

Linux でオーディオ機能を実現するには、ALSA[3]と OSS[4]の 2 つの方法があります。デフォルト設定では、ALSA によるオーディオ機能を提供しています。

利用可能な ALSA デバイスを次に示します。

ALSA デバイス - hw:0

HDMI オーディオインターフェース(Armadillo-840: CON3)
サンプリング周波数: 48k Hz
チャンネル数: 2
フォーマット: Signed 16/24 bit, Little-endian

ALSA デバイス - hw:1

モノラルマイク入力インターフェース(拡張ボード 01: CON5)
ステレオヘッドホン出力インターフェース(拡張ボード 01: CON6)
サンプリング周波数: 48k, 44.1k, 32k, 16k, 8k Hz
チャンネル数: 1 or 2
フォーマット: Signed 16/24 bit, Little-endian

6.3.1. サウンドを再生する

ここでは、テストサウンドを再生する方法を示します。 テストサウンドには、Gstreamer の「audiotestsrc」を利用します。

次のようにコマンドを実行すると、ALSA デバイスに対応するオーディオ出力から正弦波(440Hz)の音が再生されます。

[3]Advanced Linux Sound Architecture http://alsa.sourceforge.net
[4]Open Sound System http://developer.opensound.com/

```
[armadillo ~]# gst-launch-1.0 audiotestsrc ! \
                        "audio/x-raw,channels=2,rate=48000,width=16" ! \ ❶
                        alsasink device=hw:0 ❷
```

注）本来は一行のコマンドとして実行します。

❶ rate パラメータには、サンプリング周波数を指定します。

❷ device パラメータには、ALSA デバイスを指定します。

図 6.17 テストサウンドの再生

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 audiotestsrc ! \
                "audio/x-raw-int,channels=2,rate=48000,width=16" ! \
                alsasink device=hw:0
```

コマンドの違いは、インストールされている Gstreamer のバージョンによるものです。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では Gstreamer0.10 がインストールされていましたが、ユーザーランドイメージ romfs-a840-v1.02.img 以降(Atmark Dist v20140131 以降)では Gstreamer1.0 がインストールされています。

6.3.2. サウンドを録音する 液晶モデル

モノラルマイク入力(拡張ボード 01: CON5)に接続されたマイクから入力された音声を録音する方法を示します。 ここでは、WAV (RIFF waveform Audio Format)ファイル形式で録音する例を示します。

```
[armadillo ~]# gst-launch-1.0 alsasrc device=hw:1 ! \ ❶
                        wavenc ! \ ❷
                        filesink location=sample.wav ❸
```

注）本来は一行のコマンドとして実行します。

❶ device パラメータには、ALSA デバイスを指定します。

❷ ソフトウェアエンコーダに「wavenc」を指定します。他のエンコーダを指定することも可能です。

❸ location パラメータには、保存するファイル名を指定します。

図 6.18 サウンドの録音

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 alsasrc device=hw:1 ! \
                           wavenc ! \
                           filesink location=sample.wav
```

録音したファイルを再生するには、次のようにコマンドを実行します。 ここでは、ステレオヘッドホン出力(拡張ボード 01: CON6) hw:1 に出力するように指定しています。

```
[armadillo ~]# gst-launch-1.0 filesrc location=sample.wav ! \
                       wavparse ! \
                       alsasink device=hw:1
```

図 6.19 録音したファイルを再生

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 filesrc location=sample.wav ! \
                           wavparse ! \
                           alsasink device=hw:1
```

6.4. ストレージ

Armadillo-840 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 6.6 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
USB フラッシュメモリ	/dev/sd*[a]	/dev/sd*1
SD/SDHC/SDXC カード	/dev/mmcblk*[b]	/dev/mmcblk*p1

[a]USB ハブを利用して複数の USB メモリを接続した場合は、認識された順に sda sdb sdc ... となります。
[b]拡張ボード 01 を接続して 2 つの 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。

6.4.1. ストレージの使用方法

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

SDXC カードを使用する場合は、事前に「6.4.2. 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 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 6.20 mount コマンド書式

-t オプションに続く device には、ファイルシステムタイプを指定します[5]。FAT32 ファイルシステムの場合は vfat[6]、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 6.21. 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。

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

図 6.21 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 6.22 ストレージのアンマウント

[5]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。
[6]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。

6.4.2. 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 6.23. 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 6.23 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 6.24 EXT3 ファイルシステムの構築

6.5. AV コーデックミドルウェア

AV コーデックミドルウェアを使い、H.264/AVC 及び AAC でエンコードされている動画を再生する方法について説明します。動画は MP4(MPEG-4 Part 14)コンテナに格納されているものを使用します。

AV コーデックミドルウェアには、Atmark Dist v20140131 以降(ユーザーランドイメージ romfs-a840-v1.02.img 以降)、Linux カーネル v3.4-at6 以降(カーネルイメージ linux-a840-v1.02.img.gz 以降)で対応しています。それ以前のものを使用されている場合、本節で説明する動作確認を行う前にイメージを対応バージョンに書き換えてください。

動作確認に利用する動画はサイズが大きいため、Armadillo サイトから取得し、ストレージに保存します。「6.1. ネットワーク」を参照してネットワーク設定を行い、Armadillo からインターネットに接続できる状態にしておいてください[7]。また、USB メモリや SD カード等のストレージを/mnt にマウントしているという前提で説明を行います。「6.4. ストレージ」を参照して適切なデバイスをマウントしておいてください。

表 6.7 サンプル動画

種類	ファイル名
Full HD サイズ 30 秒動画	big-buck-bunny-30sec-fullhd.mp4
800×480 サイズ 30 動画	big-buck-bunny-30sec-800x480.mp4

次のようにコマンドを実行し、Armadillo サイトから動画ファイルを取得してください。

```
[armadillo ~]# cd /mnt
[armadillo /mnt]# wget http://download.atmark-techno.com/sample/bbb/big-buck-bunny-30sec-
fullhd.mp4
[armadillo /mnt]# wget http://download.atmark-techno.com/sample/bbb/big-buck-
bunny-30sec-800x480.mp4
```

図 6.25 サンプル動画の取得

30 秒動画に使われている Big Buck Bunny は、Creative Commons Attribution 3.0 Unported License で提供されています。(c) copyright 2008, Blender Foundation / www.bigbuckbunny.org

[7]動画は開発セット付属の DVD にも収録されています。ネットワークに接続できない環境の場合、そちらをご利用ください。

6.5.1. HDMI ディスプレイへの表示

HDMI インターフェース(Armadillo-840: CON3)に接続した HDMI 対応ディスプレイに動画を表示します。本節で示すコマンド例をそのまま実行するためには、HDMI 対応ディスプレイは Full HD サイズの表示に対応している必要があります。対応するフレームバッファは/dev/fb0 です。

サウンドについても、HDMI 対応ディスプレイに出力します。対応する ALSA デバイスは hw:0 です。

下記コマンドを実行しデフォルトアプリケーション「Photo Viewer」を停止しておいてください。

```
[armadillo ~]# killall qmlscene
```

図 6.26 Photo Viewer の停止

次のようにコマンドを実行すると動画が再生されます。動画の再生を途中で停止する場合は、Ctrl+c を入力してください。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 \
 ! qtdemux name=demux0 \
 demux0.audio_0 ! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 \
 ! alsasink device=hw:0 \
 demux0.video_0 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb0
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

図 6.27 サンプル動画の再生(HDMI ディスプレイ)

このコマンドはとても長く間違えやすいので、コマンドを本マニュアルからシリアル通信プログラムにコピー&ペーストすることをお勧めします。コマンドプロンプト（ [armadillo ~]# ）はコピーしないよう注意してください。

HDMI 対応ディスプレイが Full HD サイズの表示に対応していない場合は、解像度を指定しなければ動画が再生されません。解像度が 1280 x 720 の場合のコマンド例を次に示します。

```
[armadillo ~]# gst-launch-1.0 filesrc \
 location=/mnt/big-buck-bunny-30sec-fullhd.mp4 \
 ! qtdemux name=demux0 \
 demux0.audio_0 ! queue ! acmaacdec ! audioresample \
```

```
! audio/x-raw,rate=48000,channels=2 \
! alsasink device=hw:0 \
demux0.video_0 ! queue ! acmh264dec ! \
video/x-raw,width=1280,height=720 \
! acmfbdevsink device=/dev/fb0
```

6.5.2. LCD への表示 液晶モデル

LCD インターフェース(拡張ボード 01: CON2)に接続された LCD に動画を表示します。対応するフレームバッファは/dev/fb1 です。

サウンドは、ステレオヘッドホン出力インターフェース(拡張ボード 01: CON6)に接続されたスピーカ又はヘッドホンに出力します。対応する ALSA デバイスは hw:1 です。

次のようにコマンドを実行すると動画が再生されます。動画の再生を途中で停止する場合は、Ctrl+c を入力してください。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-800x480.mp4 \
 ! qtdemux name=demux0 \
 demux0.audio_0 ! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 \
 ! alsasink device=hw:1 \
 demux0.video_0 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb1
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

図 6.28 サンプル動画の再生(拡張ボード 01)

このコマンドはとても長く間違えやすいので、コマンドを本マニュアルからシリアル通信プログラムにコピー&ペーストすることをお勧めします。コマンドプロンプト（ [armadillo ~]# ）はコピーしないよう注意してください。

6.6. LED

Armadillo-840 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 - Armadillo-840

LED クラスディレクトリ	説明	デフォルトトリガ
/sys/class/leds/LED1/	Armadillo-840: LED1	none
/sys/class/leds/LED2/	Armadillo-840: LED2	none

表 6.9 LED - 拡張ボード 01

LED クラスディレクトリ	説明	デフォルトトリガ
/sys/class/leds/EXT1/	拡張ボード 01: LED1	none
/sys/class/leds/EXT2/	拡張ボード 01: LED2	none
/sys/class/leds/EXT3/	拡張ボード 01: LED3	none
/sys/class/leds/EXT4/	拡張ボード 01: LED4	none
/sys/class/leds/EXT5/	拡張ボード 01: LED5	none
/sys/class/leds/EXT6/	拡張ボード 01: LED6	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。

6.6.1. 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-840 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

6.6.2. 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 6.10 trigger の種類

設定	説明
none	トリガを設定しません。
mmc0	SD カードのアクセスランプにします。
timer	任意のタイミングで点灯/消灯を行います。この設定にすることにより、LED クラスディレクトリ以下に delay_on, delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します。
heartbeat[a]	心拍のように点灯/消灯を行います。工場出荷イメージでは設定することができません。
default-on[a]	主にカーネルから使用します。起動時に LED が点灯します。工場出荷イメージでは設定することができません。

[a]カーネルコンフィギュレーションで該当トリガを有効にすると設定可能になります。

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 timer
```

図 6.33 LED のトリガを表示する

6.7. RTC

Armadillo-840 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 外部バックアップ用電源入力インターフェース(Armadillo-840: CON12)に外付けバッテリー(対応バッテリー例: CR2032 WK11)[8]を接続することができます。

6.7.1. 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 6.11 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)

[8]詳しくは、各 Armadillo 販売代理店にお問い合わせください。

フィールド	意味
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)

2013 年 1 月 23 日 4 時 56 分 00 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 012304562013.00 ❷
Wed Jan 23 04:56:00 JST 2013
[armadillo ~]# date ❸
Wed Jan 23 04:56:00 JST 2013
```

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

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
Wed Jan 23 04:56:10 2013  0.000000 seconds
```

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

6.8. GPIO

Armadillo-840 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。GPIO クラスディレクトリと GPIO の対応を次に示します。

表 6.12 拡張インターフェース 1(Armadillo-840: CON7)の GPIO ディレクトリ

ピン番号	GPIO ディレクトリ
CON7 2 ピン	/sys/class/gpio/gpio195
CON7 3 ピン	/sys/class/gpio/gpio196
CON7 4 ピン	/sys/class/gpio/gpio23

ピン番号	GPIO ディレクトリ
CON7 5 ピン	/sys/class/gpio/gpio21
CON7 6 ピン	/sys/class/gpio/gpio160
CON7 7 ピン	/sys/class/gpio/gpio197
CON7 8 ピン	/sys/class/gpio/gpio198
CON7 9 ピン	/sys/class/gpio/gpio194
CON7 10 ピン	/sys/class/gpio/gpio193
CON7 12 ピン	/sys/class/gpio/gpio62
CON7 13 ピン	/sys/class/gpio/gpio63
CON7 14 ピン	/sys/class/gpio/gpio64
CON7 15 ピン	/sys/class/gpio/gpio65
CON7 16 ピン	/sys/class/gpio/gpio61
CON7 17 ピン	/sys/class/gpio/gpio165
CON7 18 ピン	/sys/class/gpio/gpio164
CON7 19 ピン	/sys/class/gpio/gpio202
CON7 20 ピン	/sys/class/gpio/gpio102
CON7 21 ピン	/sys/class/gpio/gpio59
CON7 22 ピン	/sys/class/gpio/gpio60
CON7 25 ピン	/sys/class/gpio/gpio172
CON7 26 ピン	/sys/class/gpio/gpio173
CON7 27 ピン	/sys/class/gpio/gpio4
CON7 28 ピン	/sys/class/gpio/gpio3
CON7 29 ピン	/sys/class/gpio/gpio2
CON7 30 ピン	/sys/class/gpio/gpio0
CON7 31 ピン	/sys/class/gpio/gpio1
CON7 33 ピン	/sys/class/gpio/gpio66
CON7 34 ピン	/sys/class/gpio/gpio67
CON7 35 ピン	/sys/class/gpio/gpio68
CON7 36 ピン	/sys/class/gpio/gpio69
CON7 37 ピン	/sys/class/gpio/gpio70
CON7 38 ピン	/sys/class/gpio/gpio71
CON7 39 ピン	/sys/class/gpio/gpio72
CON7 40 ピン	/sys/class/gpio/gpio73
CON7 41 ピン	/sys/class/gpio/gpio74
CON7 42 ピン	/sys/class/gpio/gpio75
CON7 43 ピン	/sys/class/gpio/gpio97
CON7 44 ピン	/sys/class/gpio/gpio98
CON7 45 ピン	/sys/class/gpio/gpio99
CON7 46 ピン	/sys/class/gpio/gpio100
CON7 61 ピン	/sys/class/gpio/gpio13
CON7 62 ピン	/sys/class/gpio/gpio12
CON7 63 ピン	/sys/class/gpio/gpio9
CON7 64 ピン	/sys/class/gpio/gpio5
CON7 65 ピン	/sys/class/gpio/gpio20
CON7 66 ピン	/sys/class/gpio/gpio10
CON7 67 ピン	/sys/class/gpio/gpio8
CON7 68 ピン	/sys/class/gpio/gpio7
CON7 70 ピン	/sys/class/gpio/gpio40
CON7 71 ピン	/sys/class/gpio/gpio41
CON7 72 ピン	/sys/class/gpio/gpio42
CON7 73 ピン	/sys/class/gpio/gpio43
CON7 74 ピン	/sys/class/gpio/gpio44
CON7 75 ピン	/sys/class/gpio/gpio45
CON7 76 ピン	/sys/class/gpio/gpio46
CON7 77 ピン	/sys/class/gpio/gpio47

ピン番号	GPIO ディレクトリ
CON7 78 ピン	/sys/class/gpio/gpio48
CON7 79 ピン	/sys/class/gpio/gpio49
CON7 80 ピン	/sys/class/gpio/gpio50
CON7 81 ピン	/sys/class/gpio/gpio51
CON7 82 ピン	/sys/class/gpio/gpio52
CON7 83 ピン	/sys/class/gpio/gpio53
CON7 84 ピン	/sys/class/gpio/gpio54
CON7 85 ピン	/sys/class/gpio/gpio55
CON7 86 ピン	/sys/class/gpio/gpio56
CON7 87 ピン	/sys/class/gpio/gpio57
CON7 88 ピン	/sys/class/gpio/gpio58
CON7 90 ピン	/sys/class/gpio/gpio24
CON7 91 ピン	/sys/class/gpio/gpio25
CON7 92 ピン	/sys/class/gpio/gpio26
CON7 93 ピン	/sys/class/gpio/gpio178
CON7 94 ピン	/sys/class/gpio/gpio179
CON7 95 ピン	/sys/class/gpio/gpio180
CON7 96 ピン	/sys/class/gpio/gpio181
CON7 97 ピン	/sys/class/gpio/gpio182

表 6.13 拡張インターフェース 2(Armadillo-840: CON8)の GPIO ディレクトリ

ピン番号	GPIO ディレクトリ
CON8 4 ピン	/sys/class/gpio/gpio34
CON8 5 ピン	/sys/class/gpio/gpio33
CON8 6 ピン	/sys/class/gpio/gpio32
CON8 7 ピン	/sys/class/gpio/gpio31
CON8 8 ピン	/sys/class/gpio/gpio30
CON8 9 ピン	/sys/class/gpio/gpio29
CON8 10 ピン	/sys/class/gpio/gpio28
CON8 11 ピン	/sys/class/gpio/gpio27
CON8 13 ピン	/sys/class/gpio/gpio35
CON8 15 ピン	/sys/class/gpio/gpio38
CON8 16 ピン	/sys/class/gpio/gpio37
CON8 17 ピン	/sys/class/gpio/gpio39
CON8 19 ピン	/sys/class/gpio/gpio36
CON8 21 ピン	/sys/class/gpio/gpio158
CON8 22 ピン	/sys/class/gpio/gpio159
CON8 27 ピン	/sys/class/gpio/gpio199
CON8 28 ピン	/sys/class/gpio/gpio94
CON8 29 ピン	/sys/class/gpio/gpio93
CON8 30 ピン	/sys/class/gpio/gpio22
CON8 40 ピン	/sys/class/gpio/gpio195
CON8 41 ピン	/sys/class/gpio/gpio196
CON8 42 ピン	/sys/class/gpio/gpio23
CON8 44 ピン	/sys/class/gpio/gpio21
CON8 45 ピン	/sys/class/gpio/gpio160
CON8 46 ピン	/sys/class/gpio/gpio197
CON8 47 ピン	/sys/class/gpio/gpio198
CON8 48 ピン	/sys/class/gpio/gpio194
CON8 49 ピン	/sys/class/gpio/gpio193
CON8 50 ピン	/sys/class/gpio/gpio182
CON8 51 ピン	/sys/class/gpio/gpio181
CON8 52 ピン	/sys/class/gpio/gpio180

ピン番号	GPIO ディレクトリ
CON8 53 ピン	/sys/class/gpio/gpio179
CON8 54 ピン	/sys/class/gpio/gpio178
CON8 55 ピン	/sys/class/gpio/gpio26
CON8 56 ピン	/sys/class/gpio/gpio25
CON8 57 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。

6.8.1. 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 6.14 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。

6.8.2. 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 6.36 GPIO の入力レベルを取得する

6.8.3. 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 6.37 GPIO の出力レベルを設定する

6.8.4. ユーザージャンパを使用する 液晶モデル

拡張ボード 01 のジャンパピン JP1 はユーザージャンパとして使用できます。ユーザージャンパは generic GPIO として実装されており、GPIO と同様の操作でユーザージャンパの入力レベルを取得することができます。対応する GPIO ディレクトリは/sys/class/gpio/gpio75 です。

ユーザージャンパの状態と取得できる値の対応表を次に示します。

表 6.15 ユーザージャンパの状態と取得できる値の対応

ユーザージャンパの状態	取得できる値
ショート	0
オープン	1

6.8.4.1. 状態を取得する

ユーザージャンパの状態を取得するには、direction を in に設定する必要があります。 詳しい GPIO のアクセス方法については、「6.8.1. 入出力方向を変更する」および「6.8.2. 入力レベルを取得する」を参考にしてください。

ユーザージャンパの状態を取得する例を次に示します。

```
[armadillo ~]# echo in > /sys/class/gpio/gpio75/direction
[armadillo ~]# cat /sys/class/gpio/gpio75/value
1
```

図 6.38 ユーザージャンパの状態を取得する

ユーザージャンパを利用して、起動時に行う処理を変更することができます。ここでは例として、Qt サンプルアプリケーションの photoviewer の画面出力先を次のように変更してみます。

ユーザージャンパの状態	画面出力先
オープン	HDMI 対応ディスプレイ
ショート	LCD

/etc/config/rc.local を次のように編集します。

```
[armadillo ~]# vi /etc/config/rc.local

# First read /etc/profile
test -f /etc/profile && . /etc/profile

JP1=/sys/class/gpio/gpio75
echo in > ${JP1}/direction ❶
if [ $(cat ${JP1}/value) -eq 0 ]; then ❷
        export QT_QPA_EGLFS_DISPLAY=1 ❸
        export QT_QPA_EGLFS_WIDTH=800 ❹
        export QT_QPA_EGLFS_HEIGHT=480 ❺
fi

#
# Starting a default application
#
```

❶ 入出力方向を INPUT に設定します。

❷ 入力レベルを取得し、処理を分岐します。
❸ 画面出力先を LCD に設定します。
❹ 画面の幅を 800 ピクセルに設定します。
❺ 画面の高さを 480 ピクセルに設定します。

変更後、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
```

6.9. ユーザースイッチ 液晶モデル

拡張ボード 01 に搭載されているユーザースイッチ(SW1～SW4)のボタンプッシュ/リリースイベントを取得する方法について説明します。

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。 そのため、インプットデバイスのデバイスファイルから各イベントを取得することができます。

Armadillo-840 で利用可能な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 6.16 インプットデバイスファイルとイベントコード

ユーザースイッチ	インプットデバイスファイル	イベントコード
SW1	/dev/input/event1	116 (Power)
SW2		158 (Back)
SW3		139 (Menu)
SW4		102 (Home)

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。

6.9.1. 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/リリースイベントは、 インプットデバイスファイルから取得することができます。 ここでは、「evtest」コマンドを利用してイベントを確認します。 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 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
    Event code 102 (Home)
```

```
    Event code 116 (Power)
    Event code 139 (Menu)
    Event code 158 (Back)
Testing ... (interrupt to exit)
Event: time 946777243.784020, type 1 (Key), code 116 (Power), value 1 ❶
Event: time 946777243.784028, ---------- Report Sync ------------
Event: time 946777243.895528, type 1 (Key), code 116 (Power), value 0 ❷
Event: time 946777243.895534, ---------- Report Sync ------------
                :
[armadillo ~]#
```

❶ SW1 のボタンプッシュイベントを検出したときの表示。

❷ SW1 のボタンリリースイベントを検出したときの表示。

図 6.39 ユーザースイッチ: イベントの確認

6.10. タッチスクリーン 液晶モデル

拡張ボード 01 には、タッチパネル LCD が搭載されています。 ソフトウェアでタッチイベントを取得することができます。

タッチスクリーンドライバは、インプットデバイスとして実装されています。 そのため、インプットデバイスのデバイスファイルからタッチイベントを取得することができます。 デバイスファイルは、/dev/input/event0 です。

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。

6.10.1. イベントを確認する

タッチイベントは、インプットデバイスファイルから取得することができます。 ここでは、「evtest」コマンドを利用してイベントを確認します。 evtest を停止するには、Ctrl+c を入力してください。

```
[armadillo ~]# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "st1232-touchscreen"
Supported events:
  Event type 0 (Sync)
  Event type 1 (Key)
  Event type 3 (Absolute)
    Event code 48 (Touch Major)
      Value      0
      Min        0
      Max      255
    Event code 53 (Position X)
      Value      0
      Min        0
      Max      799
```

```
    Event code 54 (Position Y)
      Value      0
      Min        0
      Max      479
Testing ... (interrupt to exit)
Event: time 946699216.584437, type 3 (Absolute), code 48 (Touch Major), value 73 ❶
Event: time 946699216.584446, type 3 (Absolute), code 53 (Position X), value 476 ❷
Event: time 946699216.584451, type 3 (Absolute), code 54 (Position Y), value 251 ❸
Event: time 946699216.584456, ---------- Config Sync ------------
Event: time 946699216.584460, ---------- Report Sync ------------
                :
[armadillo ~]#
```

❶ タッチされている楕円の直径を表すイベントの表示。

❷ タッチされているポイントの X 座標を表すイベントの表示。

❸ タッチされているポイントの Y 座標を表すイベントの表示。

図 6.40 タッチスクリーン: イベントの確認

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-840 の Linux カーネルの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 カーネルイメージをビルドする場合には、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-840 用のデフォルトコンフィギュレーションが記載されているファイルは、Linux カーネルソースファイル(linux-3.4-[VERSION].tar.gz)に含まれる arch/arm/configs/armadillo840_defconfig です。

armadillo840_defconfig は、「ベーシックモデル」と「液晶モデル」の 2 つのモデルに対応したイメージを作成することができます。Armadillo-840 に「Armadillo-840 拡張ボード 01(C コネクタ用)」を接続して Linux を起動した場合は、液晶モデルで利用するデバイスの登録、ピンマルチプレクスの設定が行われ、拡張ボード 01 が未接続の場合はベーシックモデルに対応した設定が自動的に行われます。

armadillo840_defconfig で有効になっている主要な設定を「表 8.1. Linux カーネル主要設定」に示します。

表 8.1 Linux カーネル主要設定

コンフィグ	説明
NO_HZ	Tickless System (Dynamic Ticks)
HIGH_RES_TIMERS	High Resolution Timer Support
PREEMPT	Preemptible Kernel (Low-Latency Desktop)
AEABI	Use the ARM EABI to compile the kernel
VFP	VFP-format floating point maths
NEON	Advanced SIMD (NEON) Extension support
BINFMT_ELF	Kernel support for ELF binaries

8.2. デフォルト起動オプション

工場出荷状態の Armadillo-840 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=ttySC2,115200	起動ログなどが出力されるイニシャルコンソールに ttySC2(Armadillo-840:CON4)を、ボーレートに 115200bps を指定します。
earlyprintk=sh-sci.2,115200	可能な限り早い段階で起動ログを出力するデバイスとして sh-sci.2(Armadillo-840:CON4)を、ボーレートに 115200bps を指定します。Linux カーネルが起動しないような不具合のデバッグに役立ちます。
mem=384M	Linux カーネルが利用可能なメモリを 384MByte に制限します。AV コーデックミドルウェアを使用する場合に必須の設定です。

Linux カーネルイメージ linux-a840-v1.02.bin.gz 以降(linux-3.4-at6 以降)のデフォルト起動オプションです。linux-a840-v1.01.bin.gz 以前(linux-3.4-at5 以前)では、AV コーデックミドルウェアが使用できないため次のように設定されていました。

```
console=ttySC2,115200 earlyprintk=sh-sci.2,115200
```

8.3. Linux ドライバ一覧

Armadillo-840 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-840

Armadillo-840 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。 新規にデバイスを追加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-shmobile/Kconfig.armadillo800
arch/arm/mach-shmobile/board-armadillo840.c
arch/arm/mach-shmobile/setup-r8a7740.c
arch/arm/mach-shmobile/pfc-r8a7740.c
arch/arm/mach-shmobile/intc-r8a7740.c
arch/arm/mach-shmobile/clock-r8a7740.c

カーネルコンフィギュレーション

```
System Type  --->
    ARM system type (Renesas SH-Mobile / R-Mobile)  --->          <CONFIG_ARCH_SHMOBILE>
[*] R-Mobile A1 (R8A77400)                                         <CONFIG_ARCH_R8A7740>
    *** SH-Mobile Board Type ***
[*] Armadillo-840 board                                        <CONFIG_MACH_ARMADILLO840>
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
        *** Extension Board select ***
        CON7 extension board (LCD)  --->                    <CONFIG_ARMADILLO840_CON7EB_LCD>
        CON8 extension board (Custom)  --->              <CONFIG_ARMADILLO840_CON8EB_CUSTOM>
        *** Interface and Device select ***
    :
```

ユーザーオリジナルの拡張ボードを利用する場合には、「CON7/8 extension board」には「Custom」を選択します。 Custom を選択した場合には、全てのピンマルチプレクスを変更することができます。

「Armadillo-840 System Configuration --->」のメニュー内では、Armadillo-840 の拡張インターフェースで実現することができるピンマルチプレクスを選択することができます。

ピンマルチプレクスは、1 つの信号に 1 つのファンクションを割り当てることができます。 利用しようとする信号が既に選択されている場合には、選択項目自体が表示されない場合があります。

次に示す例では、「CEU0 Upper [CLKs, SYNCs and D15-8]」を選択すると、「CEU1 [CLKs, SYNCs and D7-0]」の項目自体が非表示となっています。これは、CEU0 のデータバス上位 8bit と CEU1 のデータバス 8bit が同様の信号のため、同時利用ができないために項目が非表示となります。

- 「use CEU0 Upper [CLKs, SYNCs and D15-8]」を選択する前の表示

```
    *** Interface and Device select ***
[ ] use CEU0 Lower [CLKs, SYNCs and D7-0]
[ ] use CEU0 Upper [CLKs, SYNCs and D15-8]
[ ] use CEU1 [CLKs, SYNCs and D7-0]
-*- use FSIA as Slave
-*-   AUDIO: WM8978 codec
```

- 「use CEU0 Upper [CLKs, SYNCs and D15-8]」を選択した後の表示

```
    *** Interface and Device select ***
[ ] use CEU0 Lower [CLKs, SYNCs and D7-0]
[*] use CEU0 Upper [CLKs, SYNCs and D15-8]
[ ]   CAMERA: KBCR-iC01VG (NEW)
-*- use FSIA as Slave
-*-   AUDIO: WM8978 codec
```

8.3.2. タイマー

Armadillo-840 では、カーネル内部のクロックソースに R-Mobile A1 の CMT0(Compare Match Timer0) を利用しています。

関連するソースコード

drivers/clocksource/sh_cmt_simple.c

カーネルコンフィギュレーション

```
System Type  --->
    *** SH-Mobile System Configuration ***
    Timer and clock configuration  --->
    [*] CMT simple timer driver                          <CONFIG_SH_TIMER_CMT_SIMPLE>
```

8.3.3. フラッシュメモリ

Armadillo-840 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/chips/cfi_cmdset_0001.c
drivers/mtd/maps/physmap.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/flash/bootloader		
/dev/mtdblock0	ブロック	
/dev/flashblk/bootloader		
/dev/mtd1	キャラクタ	config
/dev/flash/config		
/dev/mtdblock1	ブロック	
/dev/flashblk/config		
/dev/mtd2	キャラクタ	license
/dev/flash/license		
/dev/mtdblock2	ブロック	
/dev/flashblk/license		
/dev/mtd3	キャラクタ	firmware
/dev/flash/firmware		
/dev/mtdblock3	ブロック	
/dev/flashblk/firmware		
/dev/mtd4	キャラクタ	kernel
/dev/flash/kernel		
/dev/mtdblock4	ブロック	
/dev/flashblk/kernel		
/dev/mtd5	キャラクタ	userland
/dev/flash/userland		
/dev/mtdblock5	ブロック	
/dev/flashblk/userland		

カーネルコンフィギュレーション

```
Device Drivers  --->
<*> Memory Technology Device (MTD) support  --->                        <CONFIG_MTD>
    [*] Command line partition table parsing                <CONFIG_MTD_CMDLINE_PARTS>
    [*] 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>
```

8.3.4. UART

Armadillo-840 のシリアルは、R-Mobile A1 の SCIFA (Serial Communications Interface with FIFO A) 及び SCIFB (Serial Communications Interface with FIFO B) を利用しています。

Armadillo-840 では、最大 9 ポートを利用することができます。 Armadillo-840 の標準状態で利用可能なポートは、SCIFA2 (Armadillo-840: CON4) のみとなっています。 SCIFA2 以外のポートを利用する場合には、カーネルをコンフィギュレーションする必要があります。

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: 1Mbps

関連するソースコード

drivers/tty/serial/serial_core.c
drivers/tty/serial/sh-sci.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
SCIFA0	/dev/ttySC0
SCIFA1	/dev/ttySC1
SCIFA2	/dev/ttySC2
SCIFA3	/dev/ttySC3
SCIFA4	/dev/ttySC4
SCIFA5	/dev/ttySC5
SCIFA6	/dev/ttySC6
SCIFA7	/dev/ttySC7
SCIFB	/dev/ttySC8

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [ ] use SCIFA0                                      <CONFIG_ARMADILLO840_SCIFA0>
        [ ]   have RTS/CTS                     <CONFIG_ARMADILLO840_SCIFA0_HAVE_RTSCTS>
        [ ] use SCIFA1                                      <CONFIG_ARMADILLO840_SCIFA1>
        [ ]   have RTS/CTS                     <CONFIG_ARMADILLO840_SCIFA1_HAVE_RTSCTS>
        [ ] use SCIFA3                                      <CONFIG_ARMADILLO840_SCIFA3>
        [ ] use SCIFA4 [RX:PORT12, TX:PORT13]        <CONFIG_ARMADILLO840_SCIFA4_12_13>
        [ ] use SCIFA4 [RX:PORT94, TX:PORT93]        <CONFIG_ARMADILLO840_SCIFA4_94_93>
        [ ] use SCIFA5                                      <CONFIG_ARMADILLO840_SCIFA5>
        [ ] use SCIFA6                                      <CONFIG_ARMADILLO840_SCIFA6>
        [ ] use SCIFA7                                      <CONFIG_ARMADILLO840_SCIFA7>
        [ ] use SCIFB                                        <CONFIG_ARMADILLO840_SCIFB>
        [ ]   have RTS/CTS                      <CONFIG_ARMADILLO840_SCIFB_HAVE_RTSCTS>
Device Drivers  --->
    Character devices  --->
        Serial drivers  --->
            *** Non-8250 serial port support ***
            <*> SuperH SCI(F) serial port support                <CONFIG_SERIAL_SH_SCI>
            (9)   Maximum number of SCI(F) serial ports
            [*]   Support for console on SuperH SCI(F)          <SERIAL_SH_SCI_CONSOLE>
```

8.3.5. Ethernet

Armadillo-840 の Ethernet(LAN)は、R-Mobile A1 の GETHER(Gigabit 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

関連するソースコード

drivers/net/ethernet/renesas/sh_eth.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  --->                                    <CONFIG_NETDEVICES>
    -*-   Network core driver support                                 <CONFIG_NET_CORE>
    -*-     Generic Media Independent Interface device support             <CONFIG_MII>
    [*]   Ethernet driver support  --->                              <CONFIG_ETHERNET>
        <*>   Renesas SuperH Ethernet support                          <CONFIG_SH_ETH>
    -*-   PHY Device support and infrastructure  --->                 <CONFIG_PHYLIB>
        <*>   Drivers for SMSC PHYs                                   <CONFIG_SMSC_PHY>
        -*-   Support for bitbanged MDIO buses                      <CONFIG_MDIO_BITBANG>
```

8.3.6. SD ホスト

Armadillo-840 の SD ホストは、R-Mobile A1 の SDHI(SD card host interface)を利用しています。

Armadillo-840 では、最大 2 ポートを利用することができます。 Armadillo-840 の標準状態で利用可能なポートは、SDHI0 (Armadillo-840: CON1) 及び、 拡張ボード 01 を接続している場合には、SDHI1 (拡張ボード 01: CON9 または 拡張ボード 01: CON10)となっています。

機能

カードタイプ: SD / SDHC / SDXC / SDIO
バス幅: 1bit or 4bit
スピードモード[1]: Default Speed (25MHz), High Speed (50MHz)
カードディテクトサポート
ライトプロテクトサポート

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/sh_mobile_sdhi.c
drivers/mmc/host/tmio_mmc_dma.c
drivers/mmc/host/tmio_mmc_pio.c

[1]Ultra High Speed (UHS)には対応していません

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [ ] use SDHI1                                       <CONFIG_ARMADILLO840_SDHI1>
        [ ]   have CD/WP [CD:PORT72...      <CONFIG_ARMADILLO840_SDHI1_HAVE_CDWP_72_73>
        [ ]   have Power-Switch [EN...    <CONFIG_ARMADILLO840_SDHI1_HAVE_PWRSW_PORT74>
Device Drivers  --->
[*] Voltage and Current Regulator Support  --->                    <CONFIG_REGULATOR>
    <*>   Fixed voltage regulator support             <CONFIG_REGULATOR_FIXED_VOLTAGE>
<*> MMC/SD/SDIO card support  --->                                      <CONFIG_MMC>
          *** MMC/SD/SDIO Card Drivers ***
    <*>   MMC block device driver                                  <CONFIG_MMC_BLOCK>
    (8)     Number of minors per block device                <CONFIG_MMC_BLOCK_MINORS>
    [*]     Use bounce buffer for simple hosts               <CONFIG_MMC_BLOCK_BOUNCE>
          *** MMC/SD/SDIO Host Controller Drivers ***
    <*>   SH-Mobile SDHI SD/SDIO controller support                  <CONFIG_MMC_SDHI>
```

8.3.7. USB ホスト

Armadillo-840 の USB ホストは、R-Mobile A1 の USB 2.0 Host Controller を利用しています。

Armadillo-840 では、USB ホストを 2 ポートまたは、USB ホストを 1 ポートと USB ファンクションを 1 ポートの 2 つの組み合わせから利用形態に応じて構成を選択することができます。 Armadillo-840 の標準状態では、USB ホスト 2 ポートを USB Type-A コネクタ(Armadillo-840: CON5) で利用することができます。拡張ボード 01 で USB ホストまたは USB ファンクションを利用する場合は、 カーネルをコンフィギュレーションする必要があります。 その場合には、USB Type-A コネクタ(Armadillo-840: CON5)の上段は利用不可となります。

機能

Universal Serial Bus Specification Revision 2.0 準拠
Open Host Controller Interface (OHCI) Specification for USB Rev 1.0a 準拠
Enhanced Host Controller Interface (EHCI) Specification for USB Rev 1.0a 準拠
転送レート: USB2.0 High-Speed (480Mbps), 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-rmobile.c
drivers/usb/host/ohci-rmobile.c
drivers/usb/host/ehci-hcd.c
drivers/usb/host/ohci-hcd.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
            USB1 selection (CON5 - Host)  --->
            (X) CON5 - Host                          <CONFIG_ARMADILLO840_USB1_CON5_HOST>
            ( ) CON7 - Host                          <CONFIG_ARMADILLO840_USB1_CON7_HOST>
Device Drivers  --->
[*] USB support  --->                                                  <CONFIG_USB_SUPPORT>
    <*>   Support for Host-side USB                                          <CONFIG_USB>
            *** Miscellaneous USB options ***
    [*]     USB device filesystem                                  <CONFIG_USB_DEVICEFS>
    [*]     USB device class-devices                             <CONFIG_USB_DEVICE_CLASS>
          *** USB Host Controller Drivers ***
    <*>   EHCI HCD (USB 2.0) support                             <CONFIG_USB_EHCI_HCD>
    [*]     Improved Transaction Translator scheduling    <CONFIG_USB_EHCI_TT_NEWSCHED>
    <*>   OHCI HCD support                                       <CONFIG_USB_OHCI_HCD>
```

8.3.8. USB ファンクション

Armadillo-840 の USB ファンクションは、R-Mobile A1 の USBF(USB 2.0 Function Module)及び USBHSF0-DMAC(Dedicated DMAC for High-Speed USB Function Operation)を利用しています。

Armadillo-840 の標準状態では、USB ファンクションを利用することができません。 USB ファンクションを利用するには、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。 USB インターフェースの接続変更、USBF ドライバを有効化、ガジェットドライバの選択などです。詳しい手順については、「21.4. USB ガジェットを使用する 液晶モデル」を参照してください。

Armadillo-840 とホスト機器を接続する場合は、USB Mini-B コネクタ(拡張ボード 01: CON14)に USB ケーブルを接続して利用します。

Linux カーネルに用意されている代表的なガジェットドライバを次に示します。

表 8.3 代表的なガジェットドライバ

ガジェット	機能
Ethernet Gadget	接続されるとホスト機器間でネットワーク通信させることができます
Mass Storage Gadget	接続されるとホスト機器上で USB メモリとして扱うことができます
Serial Gadget	接続されるとホスト機器間でシリアル通信させることができます

関連するソースコード

drivers/usb/renesas_usbhs/mod.c
drivers/usb/renesas_usbhs/mod_host.c
drivers/usb/renesas_usbhs/mod_gadget.c
drivers/usb/renesas_usbhs/pipe.c
drivers/usb/renesas_usbhs/fifo.c
drivers/usb/renesas_usbhs/common.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
            USB1 selection (CON7 - Device)  --->
            (X) CON7 - Device                    <CONFIG_ARMADILLO840_USB1_CON7_DEVICE>
Device Drivers  --->
[*] USB support  --->                                          <CONFIG_USB_SUPPORT>
    <*>   Renesas USBHS controller                          <CONFIG_USB_RENESAS_USBHS>
          *** USB Miscellaneous drivers ***
    <*>   USB Gadget Support  --->                           <CONFIG_USB_GADGET>
        <*>   USB Peripheral Controller (Renesas USBHS controller)  --->
            (X) Renesas USBHS controller                <CONFIG_USB_RENESAS_USBHS_UDC>
        <*>   USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support)
            (X) Ethernet Gadget (with CDC Ethernet support)      <CONFIG_USB_ETH>
            ( ) Mass Storage Gadget                        <CONFIG_USB_MASS_STORAGE>
            ( ) Serial Gadget (with CDC ACM and CDC OBEX support) <CONFIG_USB_G_SERIAL>
                :
```

8.3.9. HDMI

Armadillo-840 の HDMI は、R-Mobile A1 の LCDC1(LCD Controller 1) 及び HDMI(High-Definition Multimedia Interface)、FSI2(FIFO-Buffered Serial Interface 2)を利用しています。

Armadillo-840 では、HDMI 対応ディスプレイへの画像出力及び、音声出力をサポートしています。Linux では、それぞれフレームバッファデバイス、オーディオデバイスとして利用することができます。

機能(フレームバッファ)

最大解像度[2]: 1920 x 1080 px
カラーフォーマット: ARGB8888 (32bit) or RGB565 (16bit)
ダブルバッファサポート

機能(オーディオ)

サンプリング周波数: 48000
チャンネル数: 2
フォーマット: Signed 16/24 bit, Little-endian

デバイスファイル

フレームバッファデバイス: /dev/fb0
オーディオデバイス: hw:0

関連するソースコード

drivers/video/sh_mobile_lcdcfb.c
drivers/video/sh_mobile_hdmi.c
sound/soc/sh/fsi.c
sound/soc/generic/simple-card.c

[2]ドットクロックが 75MHz を超えるビデオモードには対応できません

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
            Primary framebuffer (LCDC1)  --->
            ( ) LCDC0                                   <ARMADILLO840_PRIMARY_FB_LCDC0>
            (X) LCDC1                                   <ARMADILLO840_PRIMARY_FB_LCDC1>
Device Drivers  --->
    Graphics support  --->
        <*> Support for frame buffer devices  --->                      <CONFIG_FB>
        -*-   Enable Video Mode Handling Helpers          <CONFIG_FB_MODE_HELPERS>
              *** Frame buffer hardware drivers ***
        <*>   SuperH Mobile LCDC framebuffer support     <CONFIG_FB_SH_MOBILE_LCDC>
        <*>     SuperH Mobile HDMI controller support    <CONFIG_FB_SH_MOBILE_HDMI>
```

"Primary framebuffer"の選択は、linux-3.4-at6 で対応しました。
linux-3.4-at5 以前では選択することができません。

8.3.9.1. ビデオモードの変更

Armadillo-840 では、sysfs 経由で HDMI のビデオモードを変更することができます。 関連する sysfs ファイルを次に示します。

表 8.4 ビデオモード変更に利用する sysfs ファイル

ファイル	機能
/sys/class/graphics/fb0/modes	利用可能なビデオモードの一覧を取得することができます
/sys/class/graphics/fb0/mode	現在のビデオモードの取得、及びビデオモードの変更するする場合に利用します

利用可能なビデオモードは、HDMI 対応ディスプレイから取得したビデオモードの内、 Armadillo-840 で設定可能なビデオモードを抽出した[3]リストとなっています。 設定可能なビデオモードを取得するには、次のように/sys/class/graphics/fb0/modes ファイルから読み出します。

```
[armadillo ~]# cat /sys/class/graphics/fb0/modes
D:720x480p-59
D:1280x720p-60
D:1920x1080i-60
V:640x480p-75
V:640x480p-72
U:640x480p-67
V:640x480p-60
D:1920x1080p-60 ❶
```

❶　ドットクロックが 75MHz を超えてしまうため設定不可

[3]ドットクロックが 75MHz を越えるビデオモードもリストアップされます

ビデオモードを変更する場合は、/sys/class/graphics/fb0/mode ファイルに設定したいビデオモードを書き込みます。 ここでは「D:1280x720p-60」を設定する例を示します。

```
[armadillo ~]# echo D:1280x720p-60 > /sys/class/graphics/fb0/mode
```

8.3.9.2. ビデオモード自動調整機能

Armadillo-840 の標準状態では、ディスプレイの接続時に自動的に最適なビデオモードとなる機能 (以降、本書ではビデオモード自動調整機能と称します)が組み込まれています。

ビデオモード自動調整機能の仕組み

HDMI はディスプレイから取得できる EDID の情報から、設定可能なビデオモードを自動生成しています。 EDID を取得してビデオモードのリストが更新された場合、uevent により更新されたことをユーザー空間に通知します。 Armadillo-840 では、対応する uevent を受信した場合に /etc/config/configure-fbmode.sh を実行して、 適切なビデオモードに設定されるように実装されています。

8.3.9.3. ビデオモード自動調整機能の無効化

任意のビデオモードに固定して利用する場合は、ビデオモード自動調整機能を無効にする必要があります。 無効化するには次のように/etc/config/configure-fbmode.sh のパーミッションを変更します。

```
[armadillo ~]# chmod -x /etc/config/configure-fbmode.sh
```

次回起動時にもビデオモード自動調整機能を無効化したい場合は、コンフィグ領域を保存しておきます。

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

8.3.10. LCD

Armadillo-840 の LCD インターフェースは、R-Mobile A1 の LCDC0(LCD Controller 0)を利用しています。拡張ボード 01 のタッチパネル LCD では、バックライトの輝度調整及び、タッチスクリーンに対応しています。Linux では、それぞれ PWM 制御用バックライトドライバ(pwm_bl)、ST1232 タッチスクリーンドライバを利用しています。

機能

対応可能 LCD: TFT カラー液晶 (最大解像度: WXGA+)
カラーフォーマット: ARGB8888 (32bit) or RGB565 (16bit)
ダブルバッファサポート
256 段階バックライト輝度調整
マルチタッチサポート (ST1232)

デバイスファイル

フレームバッファデバイス: /dev/fb1
タッチスクリーンデバイス: /dev/input/event0[4]
バックライト輝度調整用 sysfs ファイル: /sys/class/backlight/pwm-backlight.0/brightness

関連するソースコード

drivers/video/sh_mobile_lcdcfb.c
drivers/video/backlight/backlight.c
drivers/video/backlight/pwm_bl.c
drivers/misc/rmob-tpu-pwm.c
drivers/input/touchscreen/st1232.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [*] TOUCHSCREEN: ST1232                <CONFIG_ARMADILLO840_I2C0_DEVICE_ST1232>
        [*] use LCDC0                                        <CONFIG_ARMADILLO840_LCDC0>
        [*]   have 24bit width [D23-18]        <CONFIG_ARMADILLO840_LCDC0_HAVE_D18_D23>
        [ ]   have LCLK0 [PORT165]       <CONFIG_ARMADILLO840_LCDC0_HAVE_LCLK0_PORT165>
              LCD (LCD: SCF0500)  --->
            (X) LCD: SCF0500                 <CONFIG_ARMADILLO840_LCDC0_DEVICE_SCF0500>
          Primary framebuffer (LCDC1)  --->
          ( ) LCDC0                                     <ARMADILLO840_PRIMARY_FB_LCDC0>
          (X) LCDC1                                     <ARMADILLO840_PRIMARY_FB_LCDC1>
        [*] use LED-Driver TB62752             <CONFIG_ARMADILLO840_LED_DRIVER_TB62752>
        [*] use TPU0 Output2 [PORT202]            <CONFIG_ARMADILLO840_TPU0TO2_PORT202>
        [*]   PWM: Generic PWM based Backlight
                            <CONFIG_ARMADILLO840_TPU0TO2_PORT202_DEVICE_PWM_BACKLIGHT>
Device Drivers  --->
    Misc devices  --->
        <*> R-Mobile TPU PWM driver                                <CONFIG_RMOB_TPU_PWM>
    Input device support  --->
        -*- Generic input layer (needed for keyboard, mouse, ...)        <CONFIG_INPUT>
        <*>   Event interface                                     <CONFIG_INPUT_EVDEV>
              *** Input Device Drivers ***
        [*]   Touchscreens  --->                               <CONFIG_INPUT_TOUCHSCREEN>
            <*>   Sitronix ST1232 touchscreen controllers   <CONFIG_TOUCHSCREEN_ST1232>
            [ ]     Enable single touch event for ST1232
                                              <CONFIG_TOUCHSCREEN_ST1232_SINGLETOUCH>
    Graphics support  --->
        <*> Support for frame buffer devices  --->                            <CONFIG_FB>
        -*-   Enable Video Mode Handling Helpers                <CONFIG_FB_MODE_HELPERS>
              *** Frame buffer hardware drivers ***
        <*>   SuperH Mobile LCDC framebuffer support         <CONFIG_FB_SH_MOBILE_LCDC>
        -*- Backlight & LCD device support  --->         <CONFIG_BACKLIGHT_LCD_SUPPORT>
            -*-   Lowlevel Backlight controls           <CONFIG_BACKLIGHT_CLASS_DEVICE>
            <*>     Generic PWM based Backlight Driver           <CONFIG_BACKLIGHT_PWM>
```

[4]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

"Primary framebuffer"の選択は、linux-3.4-at6 で対応しました。
linux-3.4-at5 以前では選択することができません。

8.3.11. アナログオーディオ

Armadillo-840 のアナログオーディオは、R-Mobile A1 の FSI2(FIFO-Buffered Serial Interface 2)を利用しています。拡張ボード 01 のオーディオコーデックには、Wolfson WM8978 が採用されています。WM8978 は I2C0 (I2C ノード: 0-001a) に接続されています。

機能

サンプリング周波数: 8000, 11025, 16000, 22050, 32000, 44100, 48000
チャンネル数: 1 or 2
フォーマット: Signed 16/24 bit, Little-endian
再生(Playback), 録音(Capture)サポート
ミキサーサポート

オーディオデバイス

hw:1

関連するソースコード

sound/soc/sh/fsi.c
sound/soc/generic/simple-card.c
sound/soc/codecs/wm8978.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [*] use FSIA as Slave                          <CONFIG_ARMADILLO840_FSIA_SLAVE>
        [*]   AUDIO: WM8978 codec        <CONFIG_ARMADILLO840_FSIA_SLAVE_DEVICE_WM8978>
Device Drivers  --->
    -*- Sound card support  --->                                          <CONFIG_SOUND>
        -*-   Advanced Linux Sound Architecture  --->                       <CONFIG_SND>
            [*]   Support old ALSA API                      <CONFIG_SND_SUPPORT_OLD_API>
            [*]   Verbose procfs contents                    <CONFIG_SND_VERBOSE_PROCFS>
            -*-   ALSA for SoC audio support  --->                        <CONFIG_SND_SOC>
                      SoC Audio support for SuperH  --->
                    <*> SH4 FSI support                          <CONFIG_SND_SOC_SH4_FSI>
                -*-   ASoC Simple sound card support             <CONFIG_SND_SIMPLE_CARD>
```

8.3.12. カメラ

Armadillo-840 のカメラインターフェースは、R-Mobile A1 の CEU(Capture Engine Unit)を利用しています。

拡張インターフェース 2(Armadillo-840: CON8)及び、カメラインターフェース(拡張ボード 01: CON12)は、 Armadillo-810 カメラモジュール 01 (B コネクタ用)(以降、カメラモジュール 01 と記載します)を接続して利用することができます。 標準状態ではカメラ機能が無効のため、利用する場合はカーネルをコンフィギュレーションする必要があります。

表 8.5 カメラモジュール 01 を利用する場合のカーネルコンフィギュレーション(ピンマルチプレクス)

コネクタ	対応するピンマルチプレクス設定
Armadillo-840: CON8	CONFIG_ARMADILLO840_CEU0_UPPER CONFIG_ARMADILLO840_CEU0_DEVICE_KBCRIC01VG 又は CONFIG_ARMADILLO840_CON8EB_KBCRIC01VG
拡張ボード 01: CON12	CONFIG_ARMADILLO840_CEU1 CONFIG_ARMADILLO840_CEU1_DEVICE_KBCRIC01VG

Armadillo-810 カメラモジュール 01 (B コネクタ用) (KBCR-iC01VG)に搭載されている CMOS イメージセンサーは、OmniVision 製 OV7725 です。

関連するソースコード

drivers/media/video/sh_mobile_ceu_camera.c
drivers/media/video/soc_camera.c
drivers/media/video/videobuf-core.c
drivers/media/video/v4l2-device.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Extension Board select ***
            CON8 extension board (KBCR-iC01VG)  --->
            (X) KBCR-iC01VG                     <CONFIG_ARMADILLO840_CON8EB_KBCRIC01VG>
            *** Interface and Device select ***
        [ ] use CEU0 Lower [CLKs, SYNCs and D7-0]      <CONFIG_ARMADILLO840_CEU0_LOWER>
        [ ] use CEU0 Upper [CLKs, SYNCs and D15-8]     <CONFIG_ARMADILLO840_CEU0_UPPER>
        [ ]   CAMERA: KBCR-iC01VG          <CONFIG_ARMADILLO840_CEU0_DEVICE_KBCRIC01VG>
        [ ] use CEU1 [CLKs, SYNCs and D7-0]                  <CONFIG_ARMADILLO840_CEU1>
        [ ]   CAMERA: KBCR-iC01VG          <CONFIG_ARMADILLO840_CEU1_DEVICE_KBCRIC01VG>
Device Drivers  --->
<*> Multimedia support  --->                                  <CONFIG_MEDIA_SUPPORT>
          *** Multimedia core support ***
    <*>   Video For Linux                                        <CONFIG_VIDEO_DEV>
          *** Multimedia drivers ***
    [*]   Video capture adapters  --->                  <CONFIG_VIDEO_CAPTURE_DRIVERS>
        [*]   V4L platform devices  --->                 <CONFIG_V4L_PLATFORM_DRIVERS>
            <*>   SoC camera support                               <CONFIG_SOC_CAMERA>
            <*>     ov772x camera support                <CONFIG_SOC_CAMERA_OV772X>
            <*>   SuperH Mobile CEU Interface driver      <CONFIG_VIDEO_SH_MOBILE_CEU>
```

8.3.13. GPU

Armadillo-840 を Linux で利用する場合、R-Mobile A1 の GPU(PowerVR SGX540)を利用することができます。 GPU 用のドライバは、Linux カーネルのソースコードにマージされています。

関連するソースコード

drivers/gpu/eurasia_km/

カーネルコンフィギュレーション

```
Device Drivers  --->
[*] GPU Support  --->
    <*>   SGX540 driver support for R8A7740                <CONFIG_GPU_EURASIA_SGX540>
```

8.3.14. AV コーデックミドルウェア

Armadillo-840 のマルチメディア機能(H.264/AVC 動画、AAC 音声、JPEG 画像の変換機能)は、SH-4A との通信によって実現しています。

次の R-Mobile A1 に搭載されたマルチメディア処理専用プロセッサは、SH-4A から制御するため、AV コーデックミドルウェアドライバでは直接制御しません。

VCP1(Video Multi Codec)
VIO6(Video I/O 6)
SPU2(Sound Processing Unit2)
JPU(JPEG Processing Unit)

対応フォーマット

デコーダー: H.264/AVC, AAC
エンコーダー: H.264/AVC, AAC, JPEG

デバイスファイル(デコーダー)

デバイスファイル	機能(デコーダー有効時)	機能(エンコーダー有効時)
/dev/video0[a]	H.264/AVC デコード	H.264/AVC エンコード
/dev/video1[a]	AAC デコード	AAC エンコード
/dev/video2[a]	-	JPEG エンコード

[a]UVC カメラなどを接続して V4L2 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/media/video/acm/acm_aacdec.c
drivers/media/video/acm/acm_aacenc.c
drivers/media/video/acm/acm_cert.c
drivers/media/video/acm/acm_drv.c
drivers/media/video/acm/acm_fw.c
drivers/media/video/acm/acm_h264dec.c
drivers/media/video/acm/acm_h264enc.c
drivers/media/video/acm/acm_jpegenc.c
drivers/media/video/acm/rto.c
drivers/media/video/v4l2-mem2mem-async.c

カーネルコンフィギュレーション

```
Device Drivers  --->
<*> Multimedia support  --->                                    <CONFIG_MEDIA_SUPPORT>
   [*]   Video capture adapters  --->                        <CONFIG_VIDEO_CAPTURE_DRIVERS>
         Encoders, decoders, sensors and other helper chips  --->
           <*> Armadillo AV Codec Middleware Driver                  <CONFIG_VIDEO_ACM>
```

8.3.15. リアルタイムクロック

Armadillo-840 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-GPIO2 (I2C ノード: 2-0030) 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  --->                                       <CONFIG_RTC_CLASS>
    [*]   Set system time from RTC on startup and resume      <CONFIG_RTC_HCTOSYS>
    (rtc0)  RTC used to set the system time              <CONFIG_RTC_HCTOSYS_DEVICE>
          *** RTC interfaces ***
    [*]   /sys/class/rtc/rtcN (sysfs)                     <CONFIG_RTC_INTF_SYSFS>
    [*]   /proc/driver/rtc (procfs for rtc0)              <CONFIG_RTC_INTF_PROC>
    [*]   /dev/rtcN (character devices)                   <CONFIG_RTC_INTF_DEV>
          *** I2C RTC drivers ***
    <*>   Seiko Instruments S-35390A                    <CONFIG_RTC_DRV_S35390A>
```

8.3.16. LED

Armadillo-840 に搭載されている制御可能な LED 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-840 には、LED1 及び LED2 が実装されています。 拡張ボード 01 には、LED1～LED6 が実装されています。 Linux カーネルでは、拡張ボード 01 に実装された LED を「EXTn」(n は 1～6 となります)のように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/EXT1
/sys/class/leds/EXT2
/sys/class/leds/EXT3
/sys/class/leds/EXT4
/sys/class/leds/EXT5
/sys/class/leds/EXT6

関連するソースコード

drivers/leds/leds-gpio.c
drivers/leds/led-class.c
drivers/leds/led-triggers.c
drivers/leds/ledtrig-timer.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [*] use External LED                                <CONFIG_ARMADILLO840_LED_EXT>
        [*]   LED: EXT1 [LED:102]                <CONFIG_ARMADILLO840_LED_EXT1_PORT102>
        [*]   LED: EXT2 [LED:59]                  <CONFIG_ARMADILLO840_LED_EXT2_PORT59>
        [*]   LED: EXT3 [LED:60]                  <CONFIG_ARMADILLO840_LED_EXT3_PORT60>
        [*]   LED: EXT4 [LED:10]                  <CONFIG_ARMADILLO840_LED_EXT4_PORT10>
        [*]   LED: EXT5 [LED:8]                    <CONFIG_ARMADILLO840_LED_EXT5_PORT8>
        [*]   LED: EXT6 [LED:7]                    <CONFIG_ARMADILLO840_LED_EXT6_PORT7>
Device Drivers  --->
[*] LED Support  --->                                                  <CONFIG_NEW_LEDS>
    <*>   LED Class Support                                          <CONFIG_LEDS_CLASS>
          *** LED drivers ***
    <*>   LED Support for GPIO connected LEDs                          <CONFIG_LEDS_GPIO>
    [*]   LED Trigger support                                       <CONFIG_LEDS_TRIGGERS>
          *** LED Triggers ***
    <*>   LED Timer Trigger                                  <CONFIG_LEDS_TRIGGER_TIMER>
```

8.3.17. ユーザースイッチ

Armadillo-840 に拡張ボード 01 を接続すると、ユーザースイッチを利用することができます。 各ユーザースイッチは、GPIO が接続されユーザー空間でイベント(Press/Release)を検出することができます。 Linux では、GPIO 接続用キーボードドライバ(gpio_keys)で制御されます。

拡張ボード 01 に搭載されたユーザースイッチには、それぞれにキーコードが割り当てられています。

表 8.6 キーコード

ユーザースイッチ	キーコード	イベントコード
SW1	KEY_POWER	116
SW2	KEY_BACK	158
SW3	KEY_MENU	139

ユーザースイッチ	キーコード	イベントコード
SW4	KEY_HOME	102

デバイスファイル

/dev/input/event1[5]

関連するソースコード

drivers/input/keyboard/gpio_keys.c
drivers/input/input.c
drivers/input/evdev.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [*] use GPIO-KEY                                    <CONFIG_ARMADILLO840_GPIO_KEY>
        [*]   KEY: Power [KEY:PORT97]       <CONFIG_ARMADILLO840_GPIO_KEY_POWER_PORT97>
        [*]   KEY: Back [KEY:PORT98]         <CONFIG_ARMADILLO840_GPIO_KEY_BACK_PORT98>
        [*]   KEY: Menu [KEY:PORT99]         <CONFIG_ARMADILLO840_GPIO_KEY_MENU_PORT99>
        [*]   KEY: Home [KEY:PORT100]       <CONFIG_ARMADILLO840_GPIO_KEY_HOME_PORT100>
    Input device support  --->
        -*- Generic input layer (needed for keyboard, mouse, ...)        <CONFIG_INPUT>
        <*>   Event interface                                          <CONFIG_INPUT_EVDEV>
            *** Input Device Drivers ***
        [*]   Keyboards  --->                                       <CONFIG_INPUT_KEYBOARD>
            <*>   GPIO Buttons                                       <CONFIG_KEYBOARD_GPIO>
```

8.3.18. I2C

Armadillo-840 の I2C インターフェースは、R-Mobile A1 の IIC (I2C Bus Interface)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-840 及び拡張ボード 01 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 8.7 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
I2C0	0x1a	WM8978 オーディオコーデック[a]
I2C0	0x55	ST1232 タッチスクリーンコントローラ[a]
I2C1	0x21	OV7725 COMS イメージセンサー[b]
I2C-GPIO2	0x30 (0x31～0x37 も予約済み)	S-35390A リアルタイムクロック
I2C-GPIO3	0x21	OV7725 COMS イメージセンサー[c]

[a]拡張ボード 01 を接続した場合
[b]拡張インターフェース 2(Armadillo-840: CON8)に Armadillo-810 カメラモジュール 01 (B コネクタ用)を接続した場合
[c]カメラインターフェース(拡張ボード 01: CON12)に Armadillo-810 カメラモジュール 01 (B コネクタ用)を接続した場合

[5]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

Armadillo-840 では標準状態では、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 (I2C0, I2C1)

デバイスファイル

/dev/i2c-0 (I2C0)
/dev/i2c-1 (I2C1)
/dev/i2c-2 (I2C-GPIO2)
/dev/i2c-3 (I2C-GPIO3)

関連するソースコード

drivers/i2c/busses/i2c-sh_mobile.c
drivers/i2c/busses/i2c-gpio.c
drivers/i2c/i2c-core.c
drivers/i2c/i2c-dev.c

カーネルコンフィギュレーション

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [ ] use I2C-GPIO3 [SCL:PORT194, SDA:PORT193]
                                              <CONFIG_ARMADILLO840_I2C_GPIO3_194_193>
Device Drivers  --->
<*> I2C support  --->                                                  <CONFIG_I2C>
    <*>   I2C device interface                                  <CONFIG_I2C_CHARDEV>
    [*]   Autoselect pertinent helper modules                 <CONFIG_I2C_HELPER_AUTO>
          I2C Hardware Bus support  --->
        <*> GPIO-based bitbanging I2C                               <CONFIG_I2C_GPIO>
        <*> SuperH Mobile I2C Controller                        <CONFIG_I2C_SH_MOBILE>
```

8.3.19. SPI

Armadillo-840 の SPI インターフェースは、R-Mobile A1 の MSIOF (Clock-Synchronized Serial Interface with FIFO)を利用します。

Armadillo-840 の標準状態では、SPI インターフェースを利用することができません。 SPI インターフェースを利用するには、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。

また、CONFIG_SPI_SPIDEV を有効にすると、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/spi-sh-msiof.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

```
System Type  --->
    *** Armadillo System Configuration ***
    Armadillo-840 System Configuration  --->
            *** Interface and Device select ***
        [ ] use MSIOF1                                      <CONFIG_ARMADILLO840_MSIOF1>
        [ ]   SPI: User mode SPI device [CS:PORT73]
                                        <CONFIG_ARMADILLO840_MSIOF1_DEVICE_SPIDEV_PORT73>
Device Drivers  --->
[ ] SPI support  --->                                                       <CONFIG_SPI>
          *** SPI Master Controller Drivers ***
    < >   Utilities for Bitbanging SPI masters                        <CONFIG_SPI_GPIO>
    < >   SuperH MSIOF SPI controller                               <CONFIG_SPI_SH_MSIOF>
          *** SPI Protocol Masters ***
    < >   User mode SPI device driver support                       <CONFIG_SPI_SPIDEV>
```

9. ユーザーランド仕様

本章では、工場出荷状態の Armadilloo-840 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. 起動処理

Armadillo-840 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

9.1.1. 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-840 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 ttySC2 vt102

::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-840 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

9.1.2. /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。

9.1.3. /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-840 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
S01mtd	フラッシュメモリのパーティション名に従って MTD のデバイスファイルへのシンボリックリンクを作成します
S03flatfsd	flatfsd を使いコンフィグ領域(/etc/config/)を復元します
S05udevd	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
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
S20checkroot	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
S21checkftp	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
S30syslogd, S31klogd	ログデーモンを起動します
S40mount	その他のファイルシステムをマウントします
S45firmware	/opt/firmware に配置されたファームウェアファイルから/lib/firmware/ディレクトリにシンボリックリンクを作成します
S45module-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
S50hostname	hostname を設定します
S50pvrsrv	PowerVR SGX540 の初期化を行います
S55firewall, S56networking, S57inetd	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
S60avahi, S60lighttpd, S60sshd	ネットワークデーモンを起動します
S90rc.local	コンフィグ領域(/etc/config/)に保存された rc.local を実行します

9.1.4. /etc/config/rc.local

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

Armadillo-840 では、システム起動時に自動的に Qt サンプルアプリケーションの photoviewer を起動させたり、AV コーデックミドルウェアのファームウェアをロードするために利用しています。photoviewer は自動起動させないように設定したり、AV コーデックミドルウェアを使わない場合はファームウェアをロードしないように設定できます。

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

# First read /etc/profile
test -f /etc/profile && . /etc/profile

#
# Starting a default application
#
START_PHOTOVIEWER_WITH_QMLSCENE=y ❶
if [ "${START_PHOTOVIEWER_WITH_QMLSCENE}" = "y" ]; then
        echo -n "Starting photoviewer: "
        qmlscene /usr/share/qt5/photoviewer/photoviewer.qml >/dev/null 2>&1 &
        check_status
fi

#
# for AV Codec Midleware
#   - load firmware
#
ACM_CODEC=decoder ❷
if [ "${ACM_CODEC}" = "encoder" -o "${ACM_CODEC}" = "decoder" ]; then
    echo -n "load ${ACM_CODEC} firmware: "
    echo "${ACM_CODEC}" > /sys/devices/platform/acm.0/codec
    for i in 1 2 3 4 5; do
        sleep 1
        grep "\[${ACM_CODEC}\]" /sys/devices/platform/acm.0/codec > /dev/null
        if [ $? -eq 0 ]; then
            break
        else
            false
        fi
    done
    check_status
fi
```

❶ "y"から"n"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 photoviewer が自動起動されないようになります

❷ "encoder"を指定しコンフィグ領域を保存すると、次回起動時にエンコーダー用ファームウェアをロードするようになります。また、"encoder"か"decoder"以外の文字列を指定した場合は、ファームウェアをロードしません。

図 9.3 デフォルト状態の/etc/config/rc.local

AV コーデックミドルウェアのファームウェアロード処理は、romfs-a840-v1.02.img(Atmark Dist v20140131)で追加されました。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)で Armadillo-840 を起動したことがある場合は、次のようにコンフィグ領域を初期化すると次回起動時からファームウェアロード処理が行われるようになります。

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

9.2. 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          expect            linux32           pwd
adduser           false             linux64           reformime
amixer            fdflush           ln                rev
aplay             fgrep             login             rm
arecord           flatfsd           lrz               rmdir
ash               fsck              ls                rpm
base64            fsck.ext2         lsattr            run-parts
busybox           fsync             lsz               scriptreplay
cat               ftp               lzop              sed
catv              ftpd              mail              setarch
chattr            gdbserver         makemime          setserial
chgrp             getopt            mkdir             sh
chmod             grep              mke2fs            sleep
chown             gst-inspect-1.0   mknod             ssh
conspy            gst-launch-1.0    mktemp            ssh-keygen
cp                gst-typefind-1.0  more              stat
cpio              gunzip            mount             stty
cttyhack          gzip              mountpoint        su
date              hostname          mpstat            sync
dd                htpasswd          mt                tar
delgroup          hush              mv                tftp
deluser           ionice            netflash          tip
df                iostat            netstat           touch
dmesg             ip                nice              true
dnsdomainname     ipaddr            ntpclient         tune2fs
dumpkmap          ipcalc            pidof             umount
e2fsck            iplink            ping              uname
echo              iproute           ping6             usleep
ed                iprule            pipe_progress     vi
egrep             iptables          powertop          watch
ethtool           iptunnel          printenv          wget
evtest            kill              ps                zcat
```

・/usr/bin

```
[               envuidgid       lzcat           runsv           traceroute6
[[              ether-wake      lzma            runsvdir        tty
add-shell       expand          lzopcat         rx              ttysize
ar              expr            md5sum          script          udevinfo
arping          fdformat        mesg            seq             udpsvd
awk             fgconsole       microcom        setkeycodes     unexpand
basename        find            mjpg_streamer   setsid          uniq
beep            flock           mkfifo          setuidgid       unix2dos
bunzip2         fold            mkpasswd        sha1sum         unlzma
bzcat           free            mksquashfs      sha256sum       unlzop
```

```
bzip2           ftpget          nc              sha512sum       unsquashfs
cal             ftpput          nmeter          showkey         unxz
chat            fuser           nohup           smemcap         unzip
chpst           groups          nslookup        softlimit       uptime
chrt            hd              od              sort            users
chvt            head            openvt          spawn-fcgi      uudecode
cksum           hexdump         passwd          split           uuencode
clear           hostid          patch           strings         vi
cmp             id              pgrep           sudo            vlock
comm            ifplugd         pkill           sudoedit        volname
convert_pubkey  install         pmap            sum             wall
crontab         ipcrm           printf          sv              wc
cryptpw         ipcs            pscan           tac             wget
cut             kbd_mode        pstree          tail            which
dc              killall         pwdx            tcpsvd          who
deallocvt       killall5        qmlscene        tee             whoami
diff            last            readahead       telnet          whois
dirname         less            readlink        test            xargs
dos2unix        logger          realpath        tftp            xz
dpkg-deb        logname         remove-shell    tftpd           xzcat
du              lpq             renice          time            yes
dumpleases      lpr             reset           timeout
eject           lsof            resize          top
env             lspci           rpm2cpio        tr
envdir          lsusb           rtcwake         traceroute
```

・/sbin

```
acpid           fsck.msdos      mkdosfs         route
adjtimex        fsck.vfat       mke2fs          runlevel
arp             getty           mkfs.ext2       setconsole
avahi-daemon    halt            mkfs.minix      slattach
blkid           hdparm          mkfs.msdos      sshd
blockdev        hwclock         mkfs.vfat       start-stop-daemon
bootchartd      ifconfig        mkswap          sulogin
chat            ifdown          modinfo         swapoff
depmod          ifenslave       modprobe        swapon
devmem          ifup            nameif          switch_root
dosfsck         init            nanddump        sysctl
dosfslabel      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
```

・/usr/sbin

```
brctl           i2cset          setfont
chpasswd        inetd           setlogcons
chroot          lighttpd        svlogd
```

```
crond                loadfont            telnetd
dhcprelay            lpd                 ubiattach
dnsd                 nanddump            ubidetach
fakeidentd           nandwrite           ubimkvol
fbset                nbd-client          ubirmvol
ftpd                 ntpd                ubirsvol
get-board-info-a840  popmaildir          ubiupdatevol
httpd                rdate               udevmonitor
i2cdetect            rdev                udhcpd
i2cdump              readprofile         visudo
i2cget               sendmail
```

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. ブートローダーイメージの選択

電源投入時の"SDBOOT_EN"ピンの状態によりブートローダーイメージを選択することができます。"SDBOOT_EN"ピンの状態が Low であればフラッシュメモリの bootloader パーティションに書き込まれているブートローダーが起動し、"SDBOOT_EN"ピンの状態が High であれば SD カードの第 1 パーティションのブートローダーイメージ(/sdboot.bin)が起動します。Armadillo-840 のデフォルト状態では、"SDBOOT_EN"ピンは Low(GND に 10kΩ プルダウン)となっており、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ます。

表 10.1 SDBOOT_EN ピンとブートローダーイメージの対応

SDBOOT_EN	ブートローダーイメージ
Low(0V)	フラッシュメモリの bootloader パーティション
High(3.3V)	SD カードの第 1 パーティションの/sdboot.bin

"SDBOOT_EN"ピンは Armadillo-840 の JP2 に接続されており、ジャンパーのオープン/ショートによりブートローダーイメージを選択することができます。

表 10.2 Armadillo-840 の JP2 によるブートローダーイメージの選択

JP2	ブートローダーイメージ
オープン	フラッシュメモリの bootloader パーティション
ショート	SD カードの第 1 パーティションの/sdboot.bin

10.2. 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"HERMIT_EN_N"ピンの状態により 2 つのモードのどちらかに遷移します。

表 10.3 ブートローダー起動モード

起動モードの種別	HERMIT_EN_N	説明
OS 自動起動モード	High(3.3V)	電源投入後、自動的に Linux カーネルを起動させます。
保守モード	Low(0V)	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。

"HERMIT_EN_N"ピンは Armadillo-840 の JP1 と開発用 USB シリアル変換アダプタに接続されています。JP1 をオープンすると"HERMIT_EN_N"ピンの状態は High、ショートとすると"HERMIT_EN_N"ピンの状態は Low となります。開発用 USB シリアル変換アダプタではスライドスイッチに接続されており、基板内側にスライドさせると"HERMIT_EN_N"ピンの状態は High、外側にスライドさせると"HERMIT_EN_N"ピンの状態は Low となります。

JP1 と開発用 USB シリアル変換アダプタのスライドスイッチの組み合わせにより、どの起動モードとなるかを「表 10.4. ブートローダー起動モードスイッチ」に示します。

表 10.4 ブートローダー起動モードスイッチ

Armadillo-840 JP1	開発用 USB シリアル変換アダプタ スライドスイッチ	起動モード
オープン	内側	OS 自動起動モード

Armadillo-840 JP1	開発用 USB シリアル変換アダプタ スライドスイッチ	起動モード
オープン	外側	保守モード
ショート	内側	保守モード
ショート	外側	保守モード

10.3. 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 10.5. 保守モードコマンド一覧」に示します。

表 10.5 保守モードコマンド一覧

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

各コマンドのヘルプを表示するには「図 10.1. 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 10.1 hermit コマンドのヘルプを表示

tftpdl および tftpboot 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。

10.3.1. 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 10.6. コンソール指定子とログ出力先」に示します。

表 10.6 コンソール指定子とログ出力先

コンソール指定子	OS 自動起動モード時のログ出力先	保守モードプロンプト出力先[a]
ttySC2	Armadillo-840: CON4	Armadillo-840: CON4
none	なし	Armadillo-840: CON4
その他(tty1 等)	指定するコンソール[b]	Armadillo-840: CON3

[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

10.3.2. 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、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 10.7. 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 10.7 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	SD カードのパーティション 1 に保存されている/boot/linux.bin.gz ファイル "p1"はパーティションを示しており、"p2"とするとパーティション 2 のファイルを指定可能

10.3.3. 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-840 で使用することができる、代表的な起動オプションを「表 10.8. Linux カーネルの起動オプションの一例」に紹介します。

表 10.8 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 ttySC2 を、ボーレートに 115200 を指定しています。 `console=ttySC2,115200`

オプション指定子	説明
root=	ルートファイルシステムが構築されているデバイスを指定します。 デバイスには Linux カーネルが認識した場合のデバイスを指定します。 次の例では、デバイスに SD 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、ソースコードから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-840 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-840/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、ソースコードディストリビューションである「Atmark Dist」と、「Linux カーネル」のソースコードから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. **ソースコードの準備**

   ソースコードを準備します。Atmark Dist と Linux カーネルのソースコードアーカイブを準備し展開します。展開後、Atmark Dist に Linux カーネルのソースコードを登録するために、シンボリックリンクを作成します。

```
[ATDE ~]$ ls
atmark-dist.tar.gz  linux-3.4-at.tar.gz
[ATDE ~]$ tar zxf atmark-dist.tar.gz
[ATDE ~]$ tar zxf linux-3.4-at.tar.gz
[ATDE ~]$ ls
atmark-dist  atmark-dist.tar.gz  linux-3.4-at  linux-3.4-at.tar.gz
[ATDE ~]$ ln -s ../linux-3.4-at atmark-dist/linux-3.x ❶
```

❶ シンボリックリンク名は常に linux-3.x である必要があります。

2. コンフィギュレーションの開始

Atmark Dist ディレクトリに入り、コンフィギュレーションを行います。ここでは、menuconfig を利用します。

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

```
 atmark-dist v1.32.0 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 >
```

3. 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-840" を選択します。

```
 atmark-dist v1.32.0 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-840) AtmarkTechno Products


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

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

4. **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 v1.32.0 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 >
```

5. **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。

6. **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 v1.32.0 Configuration
--------------------------------------------------------------------------------


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

                            < Yes >      <  No  >

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

7. **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。

8. **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-840 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
```

11.1.1. ツールチェーンを変更するには

Armadillo-840 では、ARM の 2 つのアーキテクチャに対応しています。 "armhf" (デフォルト) では、浮動小数点演算に VFP コプロセッサを利用します。 "armel"では、浮動小数点演算に専用のソフトウェアライブラリを利用します。 基本的には"armhf"の方が性能が高く、特に"armel"でなければならない場合以外は"armhf"を利用してください。

"armel"アーキテクチャを利用する場合は、SGX540 ライブラリを利用することができません。そのため、Qt などの SGX540 ライブラリを必要とする機能を利用することができなくなります。

ATDE には、上記 2 つのアーキテクチャ用のツールチェーン(コンパイラやリンカ、クロスライブラリなど)を用意してあります。

Linux カーネル及びユーザーランドのアーキテクチャを変更するには、Atmark Dist のコンフィギュレーション時に、"Cross-dev"に利用したいアーキテクチャを選択します。次の例では、"armel"を指定している状態となります。"default"となっている場合は、Armadillo-840 の場合では"armhf"が選択されます。

```
 atmark-dist v1.32.0 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
            (armel) Cross-dev
            (None) Libc Version
            [*] Default all settings (lose changes) (NEW)
            [ ] Customize Kernel Settings (NEW)
            [ ] Customize Vendor/User Settings (NEW)
            [ ] Customize Vendor/User Settings (NEW)

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

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

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.tar.gz
   [ATDE ~]$ ls
   hermit-at  hermit-at.tar.gz
   [ATDE ~]$ cd hermit-at
   ```

2. **デフォルトコンフィギュレーションの適用**

   Armadillo-840 用にコンフィギュレーションを行います。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"armadillo840_nor_defconfig"を指定します。SD カード起動用イメージを作成する場合は、"armadillo840_mmcsd_defconfig"を指定してください。

   ```
   [ATDE ~/hermit-at]$ make armadillo840_nor_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/armadillo8x0/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/armadillo8x0/loader-armadillo840*.bin
src/target/armadillo8x0/loader-armadillo840-nor-[version].bin
```

11.2.1. ツールチェーンを変更するには

Linux カーネルとユーザーランドのアーキテクチャを変更するのと同様に、ブートローダーもアーキテクチャを変更することができます。ただし、特に動作に影響を与えないため、変更する必要はありません。

ブートローダーのビルド時にアーキテクチャを変更するには、CROSS_COMPILE オプションを利用します。"armel"を指定する場合は、ビルド時に "CROSS_COMPILE=arm-linux-gnueabi-"をつけてビルドしてください。

```
[ATDE ~/hermit-at]$ make CROSS_COMPILE=arm-linux-gnueabi-
```

12. フラッシュメモリの書き換え方法

本章では、Armadillo-840 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 2 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、「16. SD ブートの活用」を参照して SD カードからソフトウェアを起動させてください。

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-armadillo840-nor-*[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。
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。
license	あり	なし	有償ミドルウェアなどのライセンスファイルを配置するパーティションです。
firmware	あり	squashfs-a840-firmware-*[version]*.img	有償ミドルウェアなどのファームウェアを配置するパーティションです。
kernel	なし	linux-a840-*[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-a840-*[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。

license パーティションにはボードごとに固有なイメージが出荷時に書き込まれています。本パーティションを書き換えてライセンスファイルが消えてしまった場合、AV コーデックミドルウェアを使用できなくなります。特別な理由がない限り、license パーティションは書き換えないでください。万一、ライセンスファイルが消えてしまった場合、弊社営業部へご相談ください。

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 カーネルおよびユーザーランドイメージファイルは Armadillo サイトから、ファームウェア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、パーティションの書き込み制限をコマンドで外すことが可能です。Sysfs の MTD クラスディレクトリ以下にある"ro"というファイルに 0 を書き込むことで制限を外すことが可能です。逆に 1 を書き込めば、パーティションへの書き込みを制限する事が可能です。

MTD クラスディレクトリとパーティションの対応については、「表 12.3. パーティションと MTD クラスディレクトリの対応」を参照してください。

表 12.3 パーティションと MTD クラスディレクトリの対応

パーティション	MTD クラスディレクトリ
bootloader	/sys/class/mtd/mtd0
config	/sys/class/mtd/mtd1
license	/sys/class/mtd/mtd2
firmware	/sys/class/mtd/mtd3
kernel	/sys/class/mtd/mtd4
userland	/sys/class/mtd/mtd5

以降の説明では、任意のパーティションを示す MTD クラスディレクトリを"/sys/class/mtd/[MTD]"のように表記します。

書き込み制限を外すには、ro ファイルに 0 を書き込みます。

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

図 12.1 書き込み制限を外す

書き込みを制限するには、ro ファイルに 1 を書き込みます。

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

図 12.2 書き込みを制限する

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 12.3 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 12.4 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
bootloader[a]	/dev/flash/bootloader
config	/dev/flash/config
license[a]	/dev/flash/license
firmware[a]	/dev/flash/firmware
kernel	/dev/flash/kernel
userland	/dev/flash/userland

[a]書き込みが制限されています。詳細については、「12.1. フラッシュメモリのパーティションについて」を参照してください。

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-a840-[version].bin.gz
[ATDE ~]$ cp linux-a840-[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-a840-*[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「192.168.10.1」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://192.168.10.1/linux-a840-[version].bin.gz
.....................................................................
.....................................................................
(省略)
.....................................
netflash: got "http://192.168.10.1/linux-a840-[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-a840-[version].img.gz
```

2. SD カード上のイメージファイルのパス(/mnt/romfs-a840-*[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-a840-[version].img.gz
.....................................................................
.....................................................................
```

```
（省略）
.....................................................
netflash: got "/mnt/romfs-a840-[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 12.4 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、「10.2. 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-armadillo840-nor-v3.2.0.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-armadillo840-nor-v3.2.0.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 12.5. tftpdl コマンド例」は、Armadillo の IP アドレスを 192.168.10.10 に設定し、IP アドレスが 192.168.10.1 の TFTP サーバー上にある、romfs.img.gz を userland パーティションにを書き込む例です。

```
hermit> tftpdl 192.168.10.10 192.168.10.1 --blksize=1024 --userland=romfs.img.gz
```

図 12.5 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 12.5 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
config	--config

パーティション	オプション
license	--license
firmware	--firmware
kernel	--kernel
userland	--userland

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、SD ブート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. SD ブートを行うためのブートディスクを作成します。ブートディスクの作成方法については「16. SD ブートの活用」を参照してください。

2. Armadillo にブートディスクを接続し、ブートローダーが SD カードから起動し、且つ保守モードとなるように設定します。Armadillo-840 の JP1 および JP2 をショートに設定してください。

3. 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
```

4. 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>
```

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

6. bootloader パーティションと書き込むイメージファイル(loader-armadillo840-nor-[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 erase --region bootloader download --input-file loader-armadillo840-nor-[version].bin --region bootloader --force-locked --port /dev/ttyUSB0
serial: completed 0x0000a92c (43308) bytes.
```

7. 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 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-840 では armhf というアーキテクチャが使われています。Armadillo-840 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-840 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-gnueabihf-gcc)を利用します。

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

図 13.4 Armadillo-840 上で動作するように main.c をクロスコンパイル

Armadillo-840 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 の IP アドレスが「192.168.10.10」であることを想定しています。

```
[ATDE ~/hello]$ ftp 192.168.10.10
Connected to 192.168.10.10.
220 localhost FTP server (GNU inetutils 1.4.1) ready.
Name (192.168.10.10: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 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 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 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-gnueabihf-
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-gnueabihf-gcc -Wall -Wextra -O3 -c -o main.o main.c
arm-linux-gnueabihf-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-840 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。

```
[ATDE ~/atmark-dist]$ cp -r vendors/AtmarkTechno/Armadillo-840/ 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 27 行目を追加します。

```
22 comma := ,
23 empty :=
24 space := $(empty) $(empty)
25
26 SUBDIR_y =
27 SUBDIR_y += hello/
28 SUBDIR_$(CONFIG_VENDOR_AWL12_AERIAL)         += awl12/
29 SUBDIR_$(CONFIG_VENDOR_AWL13_AWL13)          += awl13/
30
31 all:
32      for i in $(SUBDIR_y) ; do $(MAKE) -C $$i || exit $? ; done
```

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-840"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-840 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 v1.32.0 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 v1.32.0 Configuration
--------------------------------------------------------------------------------
                  Kernel/Library/Defaults Selection
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ------------------------------------------------------------------------
            --- Kernel is linux-3.x
            (default) Cross-dev
            (None) Libc Version
            [ ] Default all settings (lose changes) (NEW)
            [ ] Customize Kernel Settings (NEW)
            [*] Customize Vendor/User Settings (NEW)
            [ ] Update Default Vendor Settings (NEW)

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

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

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

```
 atmark-dist v1.32.0 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「gstreamer」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「Multimedia tools」に gstreamer の項目があります。

```
 atmark-dist v1.32.0 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
  ------------------------------------------------------------------
               --- Multimedia tools
               [*] gstreamer
               [*]   gst-feedback
               [*]   gst-inspect
               [*]   gst-launch
               [*]   gst-typefind
               [ ]   gst-xmlinspect
               [ ]   gst-xmllaunch
                     plugins  --->
               --- Audio tools
  ------------------------------------------------------------------

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

5. gstreamer にカーソルを合わせて選択を解除して、"Exit"を 2 回して「Do you wish to save your new kernel configuration?」で"Yes"とすることで選択を解除することができます。

```
  ----------------------------------------------------------------
              --- Multimedia tools
              [ ] gstreamer
              --- Audio tools
```

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 v1.32.0 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
              --- Kernel modules
              [ ] Armadillo-WLAN

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

                  <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
              --- Kernel modules
              [ ] Armadillo-WLAN

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

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 v1.32.0 Configuration
--------------------------------------------------------------------------
                    Kernel/Library/Defaults Selection
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ----------------------------------------------------------------------
            --- Kernel is linux-3.x
            (default) Cross-dev
            (None) Libc Version
            [ ] Default all settings (lose changes) (NEW)
            [ ] Customize Kernel Settings (NEW)
            [ ] Customize Vendor/User Settings (NEW)
            [*] Update Default Vendor Settings (NEW)

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

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

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

表 13.1 デフォルトコンフィグファイル

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

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

14. Qt - GUI フレームワーク

Qt とは、ファイル操作やデータベースアクセス、XML 解析、ネットワークサポートなどの機能を備えたアプリケーションを開発するためのフレームワークです。Armadillo では、主に GUI ツールキットとして利用します。

Armadillo-840 で採用している Qt5 は、従来からの Qt C++ UI フレームワークに加え、Qt Quick と呼ばれるいまどきの UI を簡単に作成するためのフレームワークを持っています。Qt Quick は、OpenGL をベースに、3D 空間のオブジェクトを表現する「シーングラフ(Scene Graph)」、画面クリック時に波紋を生成するような「パーティクル・システム(Particle System)」、さらに画像処理に力を発揮する「シェーダー・エフェクト(Shader Effects)」機能を持ったフレームワークです。Qt5 では、このような高い機能を、QML という Javascript に似た言語によって表現することで、とても簡単に使うことができるようになっています。

Qt には、様々な支援ツールが存在しています。統合開発環境の Qt Creator、UI デザインツールの Qt Designer、翻訳支援ツールの Linguist などが用意されています。ATDE には、これらのツールキットが標準でインストールされており、すぐにアプリケーション開発を始めることができます。

14.1. ライセンス

Qt は複数のライセンスのもとに頒布されています。ATDE にインストールされている Qt は、「LGPL v2.1」向けのオープンソース版です。

Qt Licensing [Qt Project サイト内]

http://qt-project.org/products/licensing

オープンソース版: LGPL v2.1

・GNU Lesser General Public License, version 2.1 (LGPL v2.1)[1]および Digia Qt LGPL Exception version 1.1[2]に準拠して開発する必要があります。
・LGPL v2.1 として開発をスタートした後に、商用版に移行することはできません。

オープンソース版: GPL v3

・GNU General Public License, version 3 (GPLv 3)[3]に準拠して開発する必要があります。
・GPL v3 として開発をスタートした後に、商用版に移行することはできません。

商用版: Qt Commercial License (Digia 社から提供)

・商用版を使用して開発するために、有償の開発者ライセンス(Developer License)が必要です。
・商用版を組み込んだ機器の量産時に、有償のランタイムライセンス(Runtime Distribution License)が必要です。
・オープンソースライセンス(LGPL/GPL)由来の制限等は受けません。
・オープンソース版にはない追加機能やツールが存在します。

[1]詳細については、ATDE5 にインストールされている/usr/share/common-license/LGPL-2.1 を参照してください。
[2]詳細については、ATDE5 にインストールされている/usr/share/doc/qtbase5-dev/LGPL_EXCEPTION.txt を参照してください。
[3]詳細については、ATDE5 にインストールされている/usr/share/common-license/GPL-3 を参照してください。

オープンソース版で開発する前に

一度オープンソース版(LGPL v2.1/GPL v3)で開発をスタートすると、後から商用版(Qt Commercial License)に切り替えることはできませんのでご注意ください。

商用版ライセンスを使用したい場合、弊社窓口までお問い合わせください。

アットマークテクノ製品に関するお問い合わせウェブフォーム

https://www.atmark-techno.com/contactinfo/form_product

14.2. Qt on Armadillo

Armadillo で利用する場合、QPA (Qt Platform Abstraction) は、デフォルトでは 「eglfs」 を利用します。デフォルト状態では、プライマリフレームバッファ(fb0)の HDMI に画面が表示されますが、環境変数の QT_QPA_EGLFS_DISPLAY を設定することで、fb1 の LCD にも画面を表示させることができます。

eglfs 用の環境変数を次に示します。

表 14.1 eglfs 用の環境変数

環境変数	用途	設定値
QT_QPA_EGLFS_DISPLAY	画面表示先を変更します	0: HDMI(fb0)に画面を表示します 1: LCD(fb1)に画面を表示します
QT_QPA_EGLFS_WIDTH	有効な画面の幅を指定します	ピクセル数を指定します
QT_QPA_EGLFS_HEIGHT	有効な画面の高さを指定します	ピクセル数を指定します
QT_QPA_EGLFS_HIDECURSOR	マウスカーソルを隠します	0: マウスカーソルを表示します 1: マウスカーソルを表示しません

14.2.1. Armadillo 用に準備されているモジュール

Qt には様々なモジュールが存在しますが、Armadillo 用に準備されているものは(執筆時点では)限られています。次に示すモジュールが準備されています。

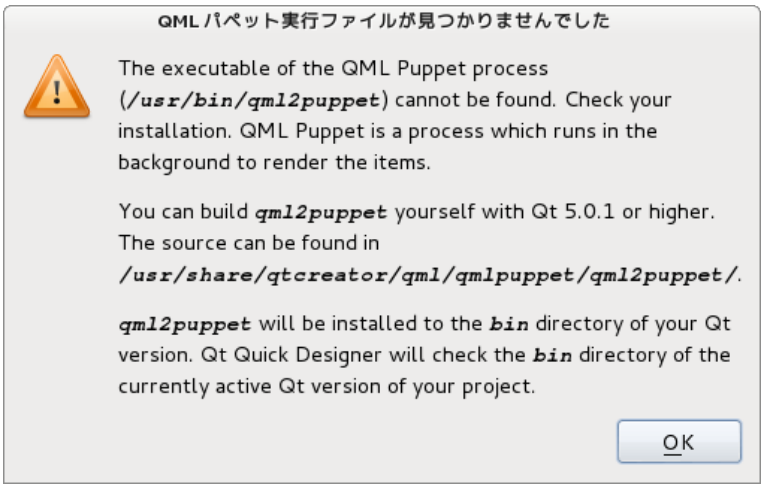
Qt5 Core module
Qt5 OpenGL module
Qt5 Network module
Qt5 SQL module
Qt5 Concurrent module
Qt5 QML module
Qt5 GUI module
Qt5 Widgets module
Qt5 D-Bus module
Qt5 Print support module
Qt5 JavaScript backend module
Qt5 XML module

14.2.2. 制限事項

執筆時点で判明している、不具合事項は次の通りです。

- メインウィンドウ以外にサブウィンドウを作成する場合、作成に失敗する場合があります。例えば、メッセージボックスやファイルダイアログなどを作ろうとする場合に、EGL_BAD_ALLOC エラーとなる場合があります。

・Qt Creator で QML のデザイナーが利用できません。次のようにダイアログが表示されます。



・QML 言語で記述したアプリケーションの画面を拡張ボード 01 の LCD に表示させた場合に、HDMI(プライマリフレームバッファ)の画面解像度が使用されてしまいます。LCD の画面解像度で表示させるためには、次のようにアプリケーションを実行する必要があります。

```
[armadillo ~]# QT_QPA_EGLFS_DISPLAY=1 QT_QPA_EGLFS_WIDTH=800 QT_QPA_EGLFS_HEIGHT=480 qml_app
```

14.3. Qt Creator

Qt Creator は、ユーザーインターフェース(UI)のデザインやプログラムのビルド・デバッグなどを行うことができる統合開発環境です。

デスクトップの左上の「アプリケーション -> プログラミング -> Qt Creator」で起動させることができます。

図 14.1 Qt Creator

14.3.1. 新規プロジェクトを作成する

本節では、新規にプロジェクトを作成する手順について説明します。 新規にプロジェクトを作成すると、スケルトンと言われる単純アプリケーションの ソースコードが自動的に生成されます。 このスケルトンをベースにアプリケーションを開発していきます。

新規プロジェクトを作成するには、Qt Creator のメニューから 「ファイル -> ファイル/プロジェクトの新規作成」を選択します。

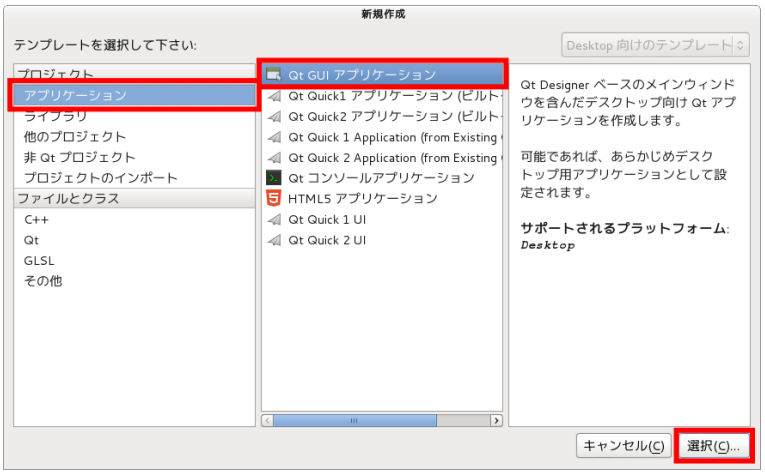


図 14.2 新規作成 - Qt GUI アプリケーション

新規作成画面では、Qt GUI アプリケーションを選択します。

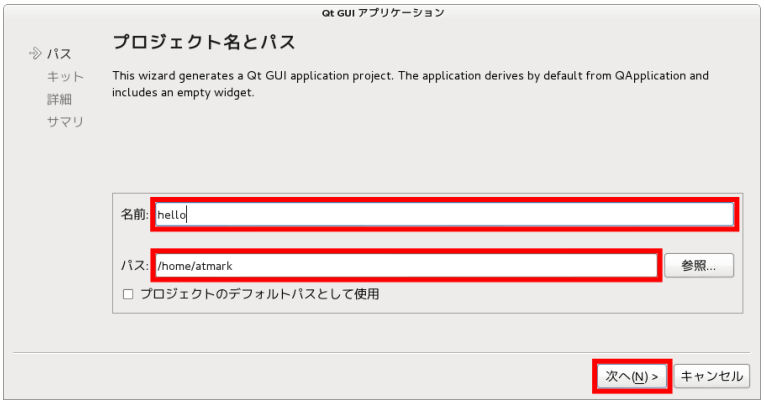


図 14.3 Qt GUI アプリケーション - プロジェクト名とパス

次に、Qt GUI アプリケーションの設定を行います。初めにプロジェクト名とパスを設定します。ここでは、名前に「hello」、パスに「/home/atmark」を指定します。このように設定すると、/home/atmark/hello/ディレクトリ以下に自動的にファイル(プロジェクトファイルやソースコードなど)が作成されます。

図 14.4 Qt GUI アプリケーション - キットの選択

次に、キットの選択を行います。

キットとは、特定のプラットフォーム用のアプリケーションをビルドする環境や実行する環境、その他の必要なツール類を指定する設定の総称です。ATDE では、「デスクトップ」と「Armadillo(armhf)」の 2 種類をあらかじめ用意してあります。

キット「Armadillo(armhf)」は、atmark ユーザーのみ使用可能です。ATDE に新規に追加したユーザーで「Armadillo(armhf)」を使用する場合は、 次のようにコマンドを実行してください。

```
[ATDE ~]$ mkdir -p ~/.config
[ATDE ~]$ cp -r /home/atmark/.config/QtProject ~/.config/
```

「デスクトップ」は、ATDE 上で動作するアプリケーション用の設定です。

「Armadillo(armhf)」は、Armadillo 上で動作するアプリケーション用の設定です。ビルド環境には、クロスコンパイラやクロスライブラリのパスなどが設定されています。Armadillo 上にファイルを転送するための設定やリモート実行する設定などがテンプレートとして指定されています。これらの制御はネットワークを経由して行われます。

通常は、設定を変更する必要はありません。

図 14.5 Qt GUI アプリケーション - クラス情報

次に、自動生成されるソースコードの基底クラスなどの情報を指定します。ダイアログベースのアプリケーションを作成する場合などに、基底クラスを変更します。

本節では、設定を変更する必要はありません。

図 14.6 Qt GUI アプリケーション - プロジェクト管理

次に、プロジェクトのバージョン管理方法を指定します。git などでバージョン管理を行う場合に設定を行います。

本節では、設定を変更する必要はありません。

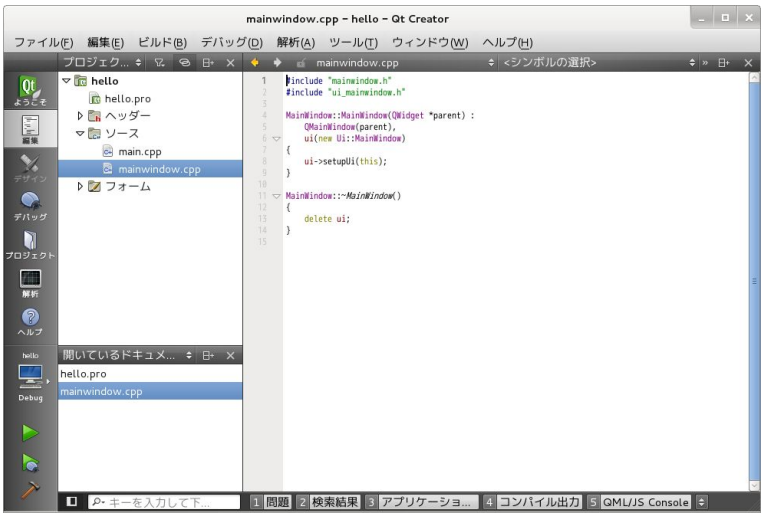


図 14.7 新規プロジェクトの作成が完了後の画面

これで新規プロジェクトの作成が完了しました。パスで設定した/home/atmark/hello/ディレクトリ以下にスケルトンが生成されています。

14.3.2. Hello World

新規プロジェクトで生成されたスケルトンのソースコードは、実行すると単純にウィンドウが表示されるだけのものです。

本節では、プログラミングのファーストステップである「Hello World」を作成してみましょう。作成されたスケルトンをベースに「Hello, World!」が表示されるようにソースコードを変更します。

まずは、プロジェクトの設定に一部情報を追加します。

リモート実行時に Armadillo 上の/tmp にファイルが転送され実行できるように、プロジェクトファイル(hello.pro)にインストールパスを設定します。

```
INSTALLS += target
target.path = /tmp
```

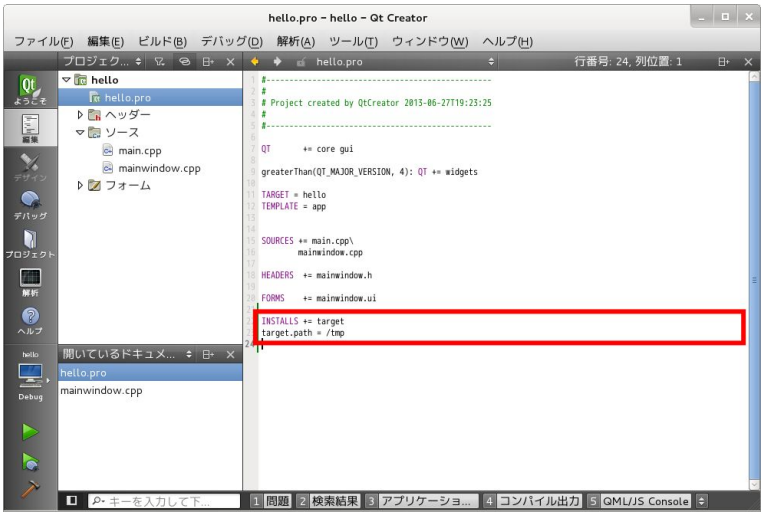


図 14.8 インストールパスを設定後の画面

続いて、「Hello World!」が表示されるように mainwindow.cpp を変更します。

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QLabel> ❶

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    //ui->setupUi(this); ❷
    QLabel *label = new QLabel(tr("Hello, World!")); ❸
    setCentralWidget(label); ❹
}
```

❶ QLabel を利用するため、インクルードを追加

❷ コメントアウト (必須ではありません)

❸ ラベルを定義しデフォルトの文字列を指定

❹ 定義したラベルをセントラルウィジェットに設定

図 14.9 mainwindow.cpp の変更箇所 (一部抜粋)

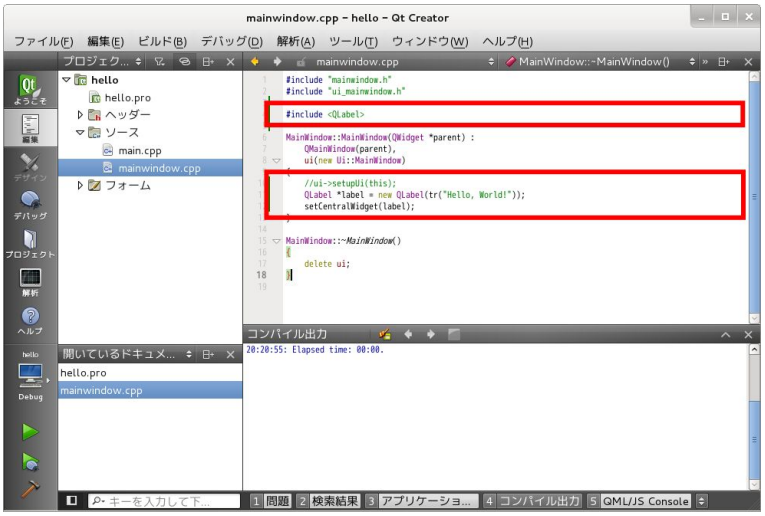


図 14.10 mainwindow.cpp の変更後の画面

14.3.3. Hello World をデスクトップ上で実行

作成したソースコードをビルドして実行してみましょう。まずは、ATDE 上のデスクトップで実行してみます。

キットには、デスクトップを選択します。画面左中央のプロジェクトタブを選択して、「デスクトップのビルド」を選択します。シャドウビルドのチェックは外しておきます。

図 14.11 デスクトップのビルド設定

ビルドを行うには、Qt Creator のメニューの「ビルド -> プロジェクト "hello" をビルド」を選択します。ビルドエラーとならない場合は、「ビルド -> 実行」を選択すると実行されます。 実行すると次のようにウィンドウが表示されます。

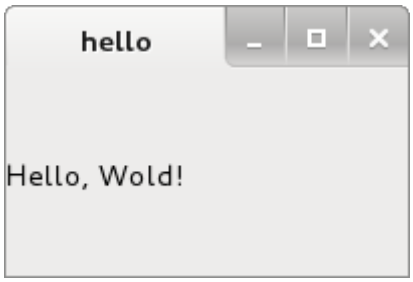


図 14.12 Hello World ウィンドウ

14.3.4. Hello World を Armadillo 上で実行

デスクトップ上で実行確認ができたので、次は Armadillo 上で実行してみましょう。 デスクトップ時の操作とほとんどかわりありませんが、標準状態の ATDE に設定されている「Armadillo(armhf)」の設定では、ターゲットの Armadillo の IP アドレスが設定されていないため、ファイル転送やリモート制御することができない状態となっています。また、工場出荷状態の Armadillo では、ファイル転送やリモート制御に利用する SSH が利用できない状態となっています。 まずは、Armadillo 側で SSH を利用できるように設定します。

14.3.4.1. Armadillo 上で SSH を設定

Armadillo の SSH の設定は、専用のスクリプトが用意されているため簡単です。 Armadillo を起動させてログイン後、次のようにコマンドを実行してください。

```
[armadillo ~]# /etc/init.d/sshd keygen ❶
generate rsa1 key ...done
generate dsa key ...done
generate rsa key ...done
[armadillo ~]# /etc/init.d/sshd ❷
Starting sshd: done
[armadillo ~]#
```

❶ SSH の鍵を生成

❷ SSH サーバーを起動

SSH の鍵を生成すると自動的にフラッシュメモリに保存され、Armadillo の次回起動時には自動的に SSH サーバーが起動します。

続いて、Armadillo の IP アドレスを確認しておきましょう。

```
[armadillo ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:11:0C:XX:XX:XX
          inet addr:192.168.1.123  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10240 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1048576 (1.0 MiB)  TX bytes:1048576 (1.0 MiB)
          Interrupt:142 DMA chan:ff
```

14.3.4.2. Armadillo(armhf)の設定

リモートターゲットの設定を行います。 画面左の「プロジェクト」タブをクリックします。

図 14.13 プロジェクト - Armadillo(armhf) - ビルド

左上にある「キットを管理」をクリックします。

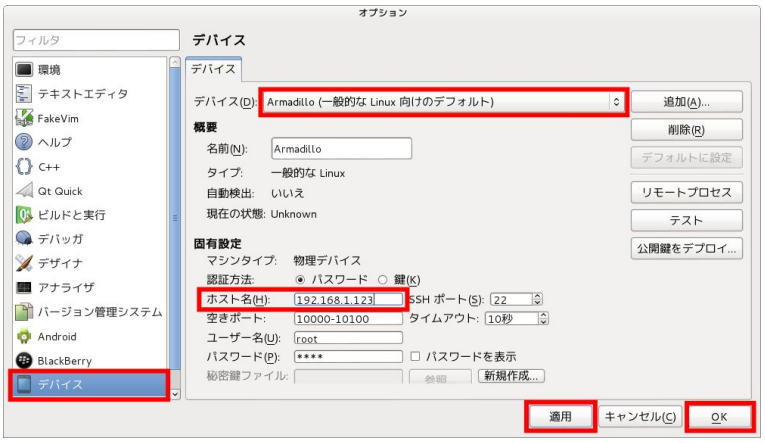


図 14.14 オプション - デバイス

フィルタリストの「デバイス」をクリックします。Armadillo デバイスの設定項目が表示されるので、ホスト名欄に利用する Armadillo の IP アドレスを入力します。

続いて、リモート実行時の設定を修正します。 画面上部の Armadillo(armhf)の「実行」タブをクリックします。

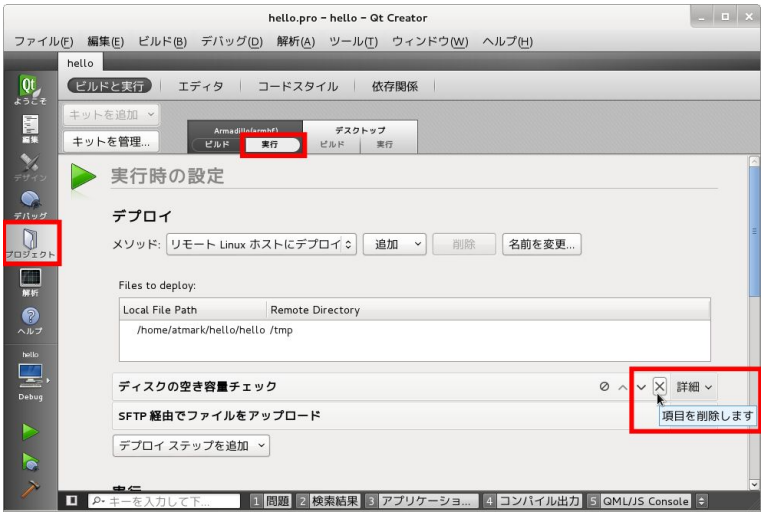


図 14.15 プロジェクト - Armadillo(armhf) - 実行

デプロイ項目の「ディスクの空き容量チェック」を削除します。

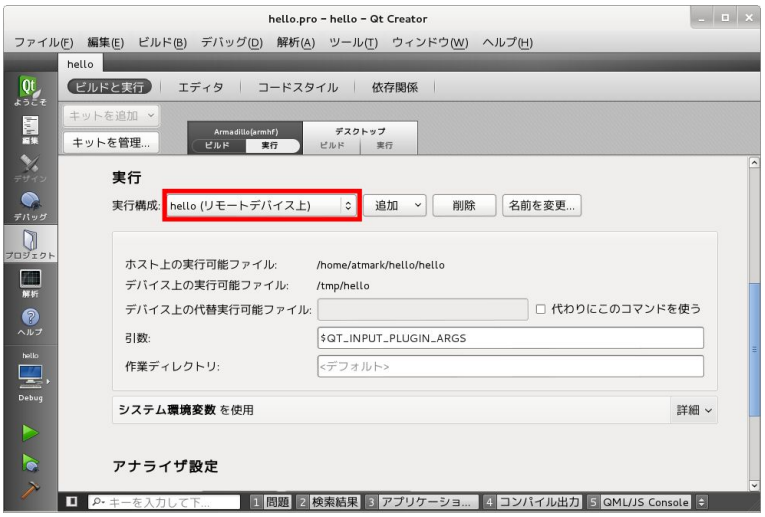


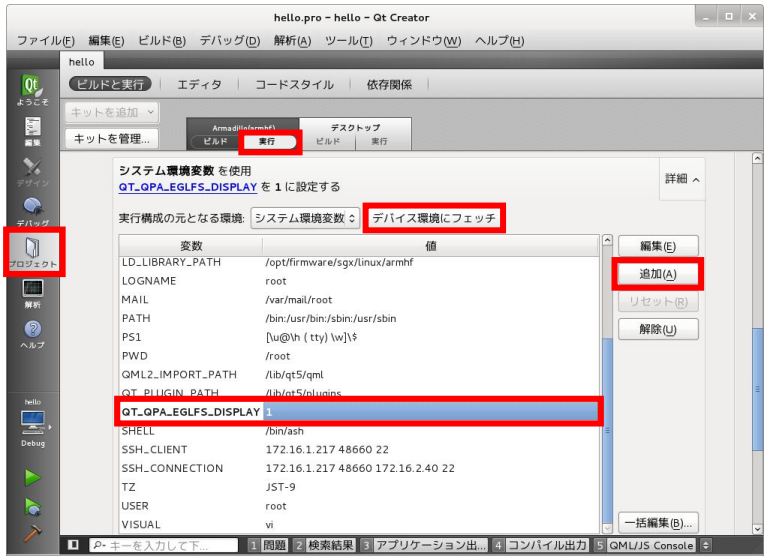
図 14.16 プロジェクト - Armadillo(armhf) - 実行

続いて、実行項目の実行構成を「hello (リモートデバイス上)」に設定します。

これで Armadillo 上でリモート実行する準備が整いました。

本節の構成どおりに作業を行った場合、hello プロジェクトはデスクトップ用にビルドされています。Armadillo 用にビルドしなおすために、メニューの「ビルド -> プロジェクト "hello" をリビルド」を選択します。ビルドエラーとならなければ、実行可能な状態となります。

拡張ボード 01 の LCD に画面を表示するには、環境変数「QT_QPA_EGLFS_DISPLAY」に値「1」を設定する必要があります。



14.4. Qt Linguist

Qt Linguist は、翻訳支援ツールです。多国語に対応したアプリケーションを作成する場合に利用します。本節では、「Hello World!」を「こんにちは!」に変換する手順を例に説明します。

Hello World のソースコードを見るとわかりますが、文字列は tr()によって括られています。この tr()で括られた文字列を任意の言語に変換して表示することができます。

大まかな手順は次の通りです。

1. プロジェクトファイルに TS ファイル[4]の指定を追加
2. プログラムで QM ファイル[5]を読み込むように変更
3. TS ファイル、QM ファイルを作成

まずは、プロジェクトファイル(hello.pro)に TS ファイルの指定を行いましょう。 次のように記述を追加します。

```
TRANSLATIONS = hello_ja.ts
```

[4]TS ファイルとは、翻訳対象の文字列と翻訳が記載された XML ファイルです。
[5]QM ファイルとは、TS ファイルを Qt アプリが解釈できるバイナリ形式に変換したファイルです。

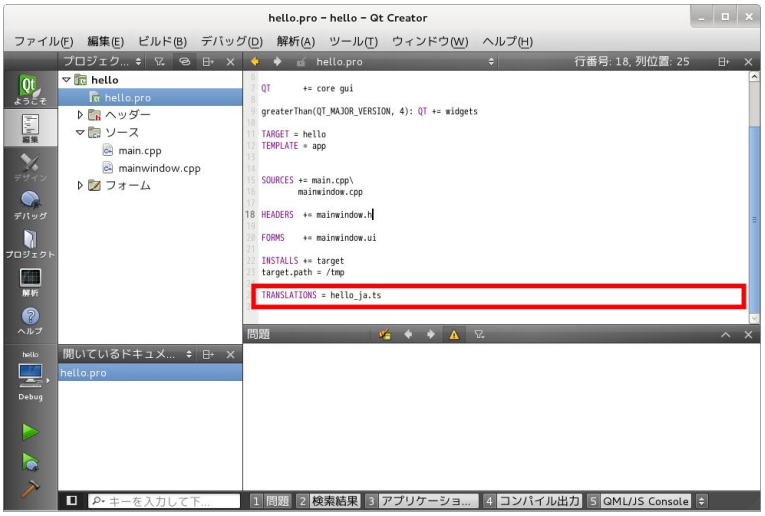


図 14.17 hello.pro に TRANSLATIONS を追加

続いてプログラムが QM ファイルを読み込むように、main.cpp を次のように変更します。

```
#include ”mainwindow.h”
#include <QApplication>
#include <QTranslator> ❶

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QTranslator translator; ❷
    translator.load(QLatin1String(”:hello_”) + QLocale::system().name()); ❸
    a.installTranslator(&translator); ❹
    MainWindow w;
    w.show();

    return a.exec();
}
```

❶ QTranslator を利用するため、インクルードを追加

❷ QTranslator オブジェクトを定義

❸ QM ファイルをロード

❹ 翻訳設定をアプリケーションにインストール

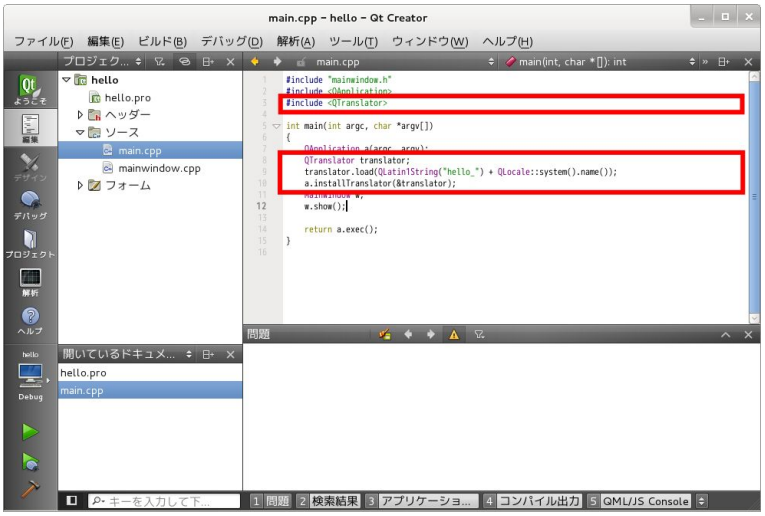


図 14.18 QM ファイルに対応

続いて、TS ファイルの作成を行います。この作業は Qt Creator では行えないため、 ターミナル上で作業します。次のコマンドを実行することにより、プロジェクト内の 翻訳対象の文字列から自動的に TS ファイル(hello_ja.ts)を作成します。

```
[ATDE ~]$ cd /home/atmark/hello/
[ATDE ~/hello]$ lupdate-qt4 ./hello.pro
```

生成された hello_ja.ts を Qt Linguist で開きます。

```
[ATDE ~/hello]$ linguist-qt4 hello_ja.ts
```

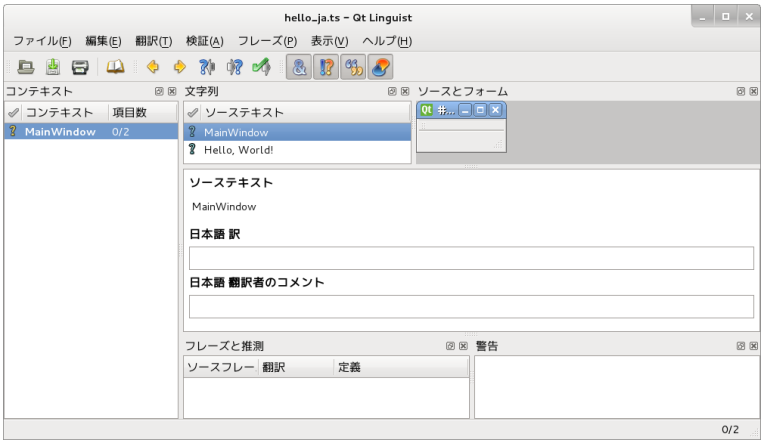


図 14.19 Qt Linguist

ソーステキストの「Hello, Wolrd!」をクリックし、日本語訳欄に「こんにちは!」を入力します。変更を確定するには、メニューの「翻訳 -> 完了」にして次へを選択します。

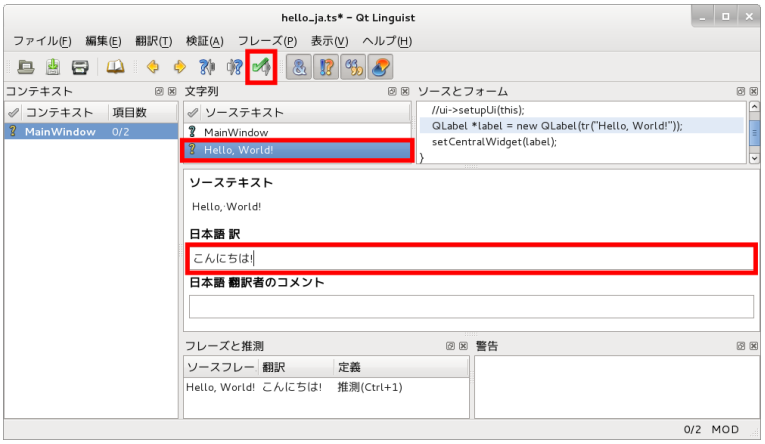


図 14.20 Qt Linguist - 翻訳

翻訳を完了する場合は、メニューの「ファイル -> 保存」を選択します。 また、翻訳ファイルをプログラムが扱える形式にするためにエクスポートします。 メニューの「ファイル -> リリース」を選択します。リリースすると QM ファイル(hello_ja.qm)が作成されます。

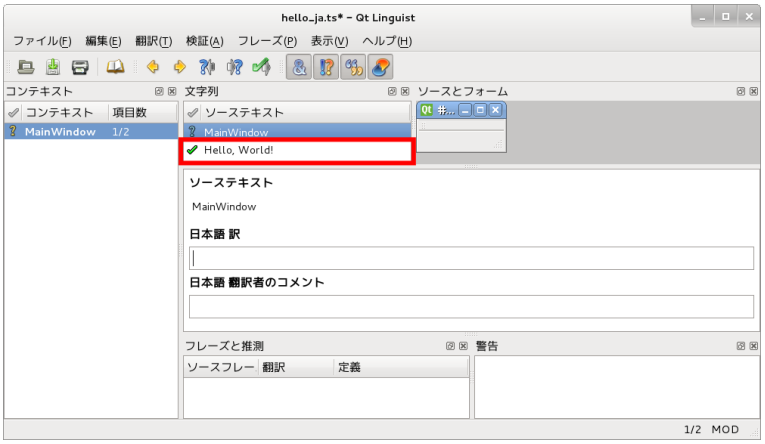


図 14.21 Qt Linguist - 翻訳確定後

これで、QM ファイルが作成できました。続いて、QM ファイルを実行ファイル(hello)に統合するためにリソース化します。まずは、プロジェクトにリソースファイル(hello.qrc)を追加します。

メニューの「ファイル -> ファイル/プロジェクトの新規作成」を選択します。

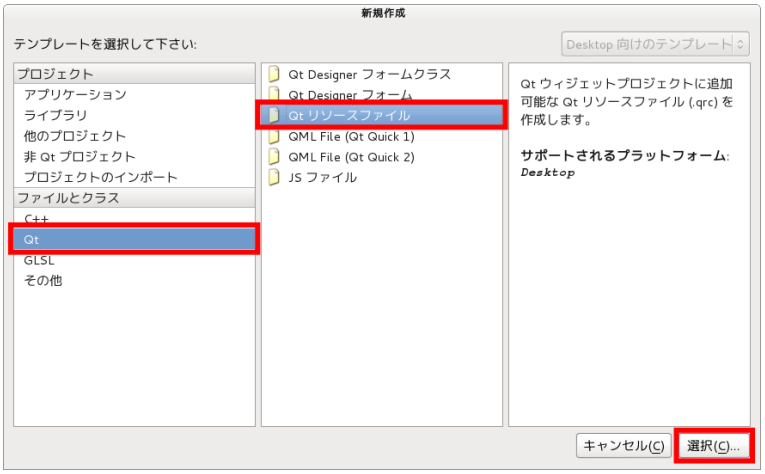


図 14.22 新規作成 - Qt リソースファイル

ファイルとクラス項目の「Qt」を選択して、「Qt リソースファイル」を選択します。

図 14.23 Qt リソースファイルの新規作成 - パス

ファイル名とパスを選択します。ここでは、名前に「hello」、パスには「/home/atmark/hello」を指定します。

図 14.24 Qt リソースファイルの新規作成 - プロジェクト管理

プロジェクト管理については、特に変更する必要はありません。

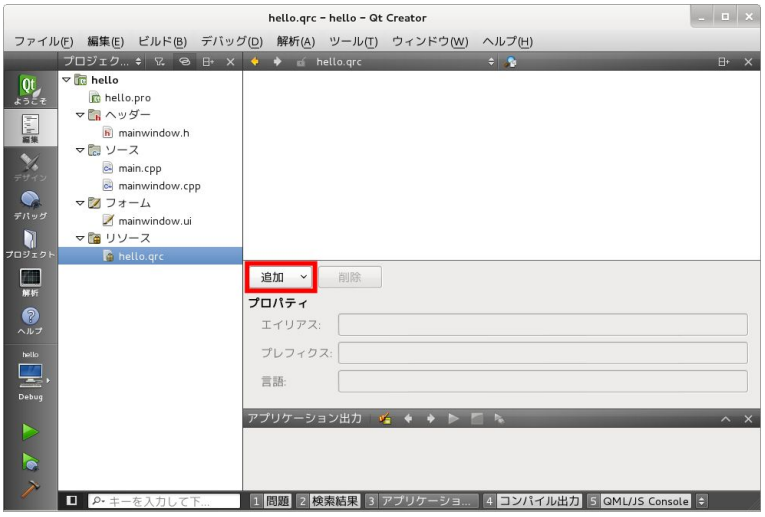


図 14.25 hello.qrc

これで、プロジェクトにリソースファイルが追加されました。続いて、リソースに翻訳ファイルを追加します。「追加」から「プレフィックスを追加」を選択し、プレフィックス欄に「/」を指定します。

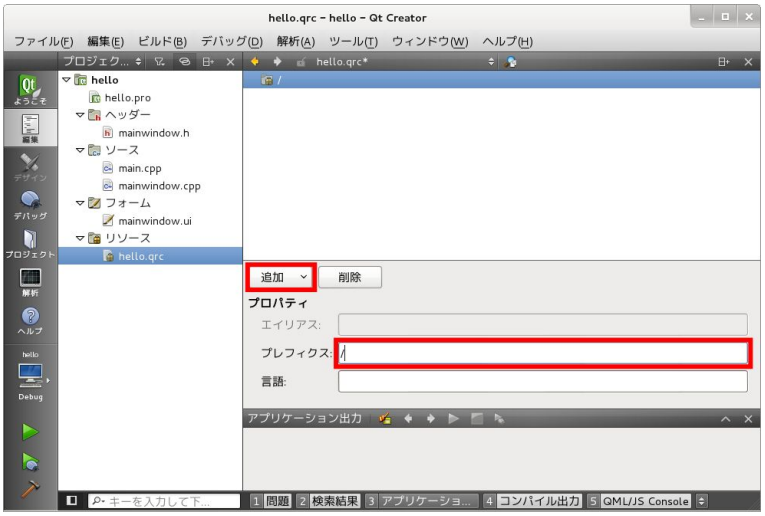


図 14.26 hello.qrc - プレフィックス

続いて、「追加」から「ファイルを追加」を選択し、ファイルダイアログから hello_ja.qm を選択します。

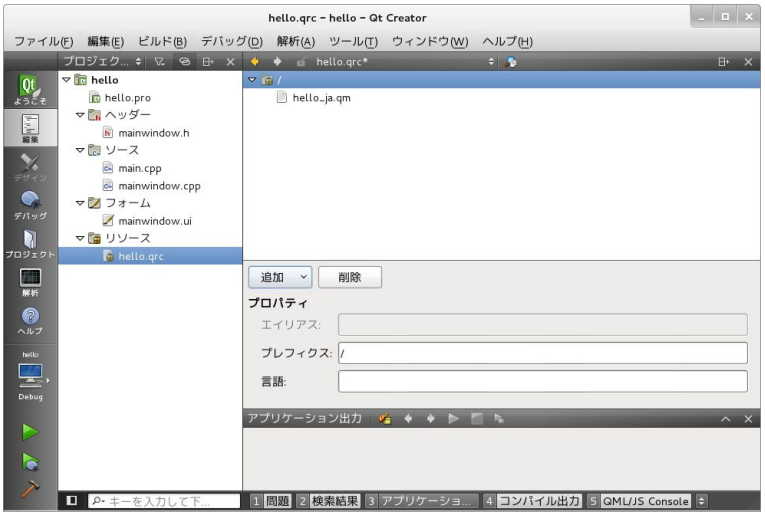


図 14.27 hello.qrc - QM ファイルを追加

これで、QM ファイルをリソースとしてプロジェクトに登録することができました。 デスクトップ上で hello を実行してみましょう。次の図のように「こんにちは!」と表示されます。

図 14.28 Hello World ウィンドウ - 日本語対応

Armadillo 上で日本語を表示するには、環境変数「LANG」を指定する必要があります。

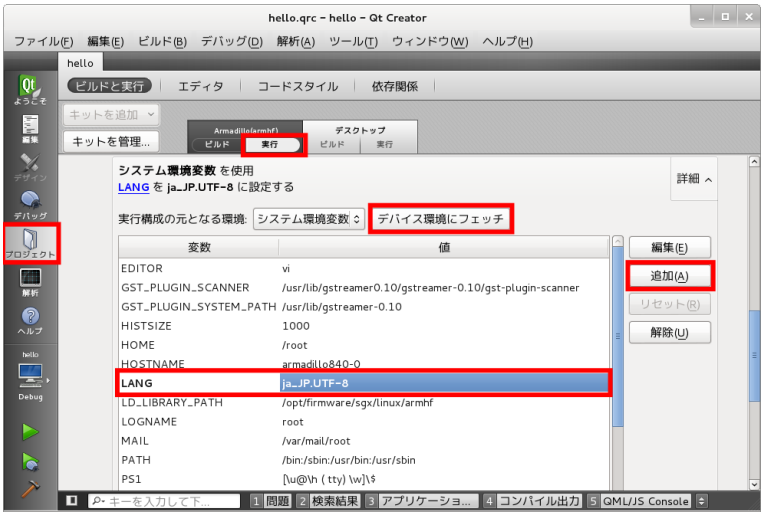


図 14.29 プロジェクト - Armadillo(armhf) - 実行 - 環境変数

左のプロジェクトタブをクリックして、Armadillo(armhf)の「実行」タブをクリックします。実行のシステム環境変数を使用の「詳細」をクリックして、詳細設定を表示させます。「デバイス環境にフェッチ」をクリックして、Armadillo の環境変数をフェッチしてください。その後、追加で変数に「LANG」、値に「ja_JP.UTF-8」を入力してください。Armadillo 用にリビルドを行い、実行すると Armadillo 上でも日本語が表示されます。

14.5. QML

QML とは、Qt Quick の一部分として実装された、ユーザーアプリケーションを開発するための言語を指します。 アニメーション制御や状態遷移などを簡単に記述でき、ボタンやスクロールなどのコンポーネントと組み合わせることで簡単にユーザーインターフェースを実現することができます。

QML を利用したアプリケーション開発では、QML アプリケーション、 QML UI の 2 つの種類があります。

QML アプリケーションは、QML ファイルや C++ファイルなどをビルドして 1 つのバイナリに統合したものを指します。

QML UI は、C++言語で記載する箇所がなく、ビルドもしません。 QML ファイルを直接 qmlscene から起動させることができます。 利点としては、コンパイラが不要のため、エディタのみでも開発できることです。

本節では、QML UI の作成・動作確認方法について説明します。

まずは、Qt Creator を利用して、QML UI のスケルトンを作成しましょう。 QML UI のスケルトンは、画面中央に「Hello World」と表示されるものとなっています。

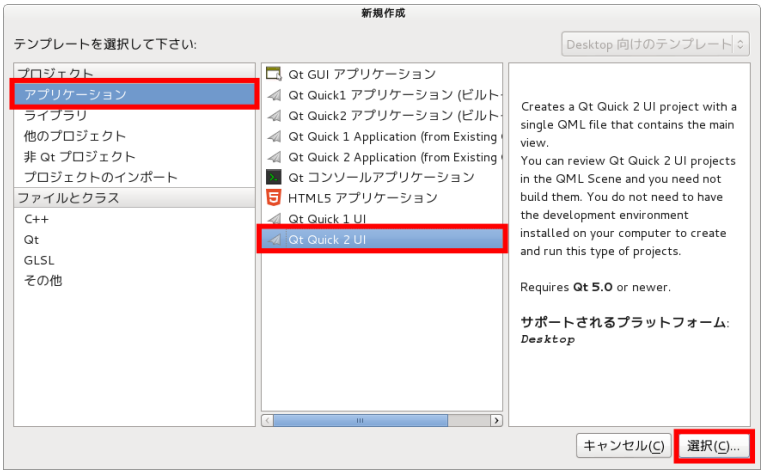


図 14.30 新規作成 - Qt Quick2 UI

メニューの「ファイル/プロジェクトの新規作成」を選択して、「Qt Quick2 UI」を選択します。

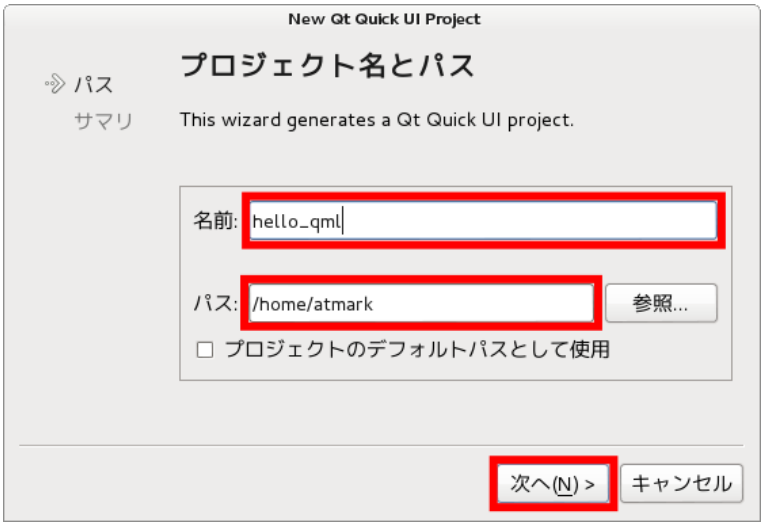


図 14.31 New Qt Quick UI Project - プロジェクト名とパス

プロジェクト名とパスを設定します。ここでは、プロジェクト名に「hello_qml」、パスに「/home/atmark/」としておきます。

図 14.32 New Qt Quick UI Project - プロジェクト管理

プロジェクト管理については、特に変更する必要はありません。

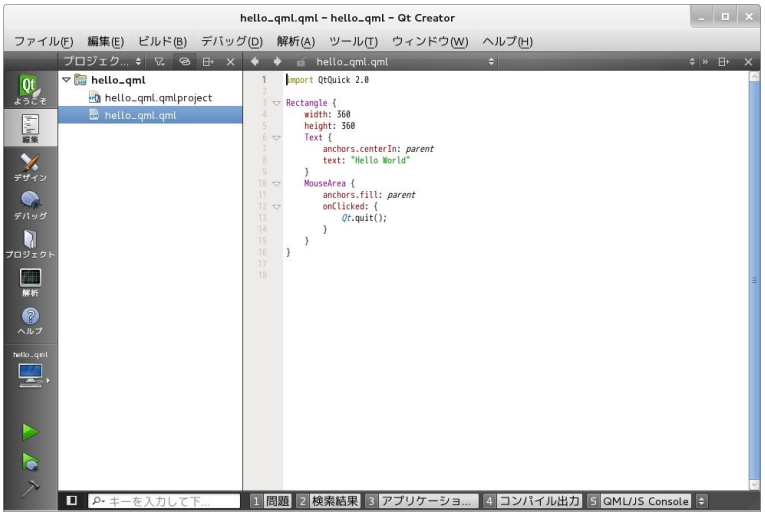


図 14.33 新規プロジェクトの作成が完了後の画面

これで、QML UI のスケルトンが作成されました。

Qt アプリケーションと違い、QML UI ではキット設定がありませんでしたね。これは、前述した通りビルドステップがないためです。

作成されたスケルトンをそのまま実行すると、次のようにウィンドウが立ち上がります。

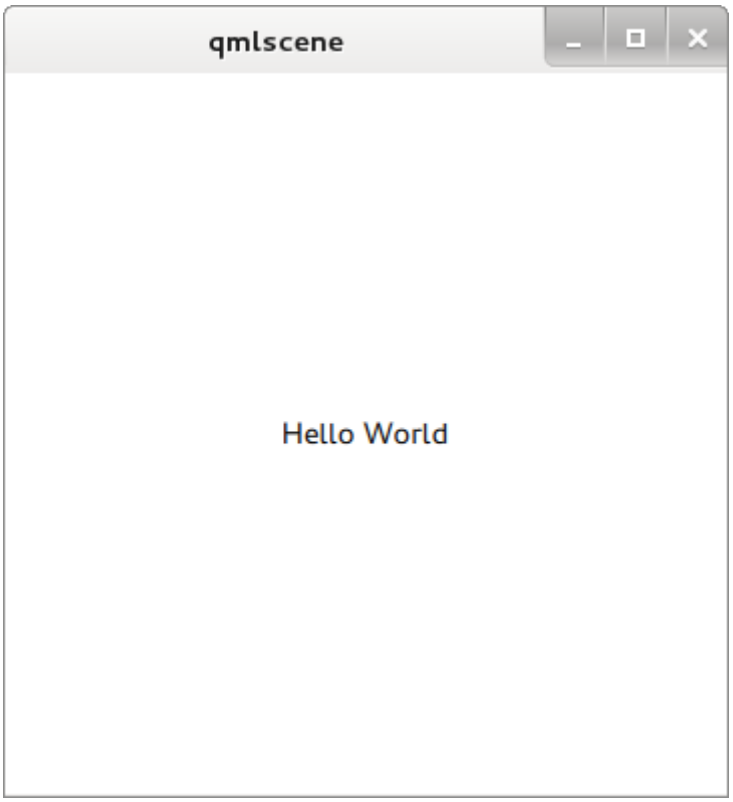


図 14.34 qmlscene - Hello World

ビルドステップが無いため、Armadillo 上で動作させるには、qml ファイルのみを転送するだけで実行準備が整います。Armadillo にファイルを転送しましょう。FTP などで転送します。

転送が終われば実際に動作させてみましょう。次の例では、FTP 経由で/home/ftp/pub/にファイルを転送した場合です。

```
[armadillo ~]# cd /home/ftp/pub
[armadillo /home/ftp/pub]# qmlscene ./hello_qml.qml
```

拡張ボード 01 の LCD に画面を表示するには、次のようにコマンドを実行します。

```
[armadillo /home/ftp/pub]# QT_QPA_EGLFS_DISPLAY=1
QT_QPA_EGLFS_WIDTH=800 QT_QPA_EGLFS_HEIGHT=480 qmlscene ./hello_qml.qml
```

14.6. オリジナル Qt アプリケーションを atmark-dist へ統合

作成したオリジナルの Qt アプリケーションを atmark-dist へ統合する方法について説明します。atmark-dist へ統合することにより、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なります。

14.6.1. Qt アプリケーションを atmark-dist に統合

まずは、「14.3.2. Hello World」で作成した hello プロジェクトを統合してみましょう。atmark-dist には、Qt アプリケーションを簡単に統合できるように実装されています。

ここでは、atmark-dist と hello は、atmark ユーザーのホームディレクトリ(~/)に存在すると仮定します。

```
[ATDE ~]$ ls -d atmark-dist/ hello/
atmark-dist/  hello/
```

まずは、hello プロジェクトをクリーンアップします。プロジェクトのディレクトリで次のように「make distclean」を行います。

```
[ATDE ~]$ cd hello
[ATDE ~/hello]$ make distclean
rm -f moc_mainwindow.cpp
rm -f ui_mainwindow.h
rm -f main.o mainwindow.o moc_mainwindow.o
rm -f *~ core *.core
rm -f hello
rm -f Makefile
```

hello ディレクトリを atmark-dist/user/qt5/以下へコピーします。

```
[ATDE ~/hello]$ cd
[ATDE ~]$ cp -a hello/ atmark-dist/user/qt5/
```

実行ファイル hello がユーザーランドの/usr/bin/ディレクトリにインストールされるようにエディタでプロジェクトファイルを修正します。

```
[ATDE ~]$ cd atmark-dist/user/qt5/hello
[ATDE ~/atmark-dist/user/qt5/hello]$ vi hello.pro
```

```
FORMS    += mainwindow.ui

INSTALLS += target
target.path = /usr/bin ❶

TRANSLATIONS = hello_ja.ts
```

❶ target.path を/usr/bin に変更

続いて、atmark-dist のビルドシステムに hello を登録します。

```
[ATDE ~/atmark-dist/user/qt5/hello]$ cd ..
[ATDE ~/atmark-dist/user/qt5]$ vi Makefile
```

```
CROSS_LIBDIR = /usr/$(CROSS_COMPILE:-=)/lib

subdir_y = qmlscene
subdir_y += hello ❶
qmldir_$(CONFIG_USER_QT5_PHOTOVIEWER) += photoviewer

BASE_LIBS = \
```

❶ subdir_y に hello を追加

これで、atmark-dist に hello を追加することができました。Armadillo-840 用に atmark-dist をビルドするとユーザーランドイメージに/usr/bin/hello が追加されます。フラッシュメモリを更新して Armadillo を起動後、次のように実行することができます。

```
[armadillo ~]# LANG=ja_JP.UTF-8 /usr/bin/hello
```

14.6.2. QML UI を atmark-dist に統合

ここでは、「14.5. QML」で作成した hello_qml を atmark-dist に統合する方法について説明します。

hello_qml は QML UI のため、前述した通りにビルドの必要がありません。そのため、インストール時の動作のみを記述します。まずは、hello_qml ディレクトリをコピーします。

```
[ATDE ~]$ ls -d atmark-dist/ hello_qml/
atmark-dist/  hello_qml/
[ATDE ~]$ cp -a hello_qml/ atmark-dist/user/qt5/
```

続いて、atmark-dist のビルドシステムに hello_qml を登録します。

```
[ATDE ~]$ cd atmark-dist/user/qt5/
[ATDE ~/atmark-dist/user/qt5]$ vi Makefile
```

```
CROSS_LIBDIR = /usr/$(CROSS_COMPILE:-=)/lib

subdir_y = qmlscene
subdir_y += hello
qmldir_y += hello_qml ❶
qmldir_$(CONFIG_USER_QT5_PHOTOVIEWER) += photoviewer

BASE_LIBS = \
```

❶ qmldir_y に hello_qml を追加

QML UI では、自動的に Makefile が生成されないため、インストール用の記述を記載した Makefile を作成します。

```
[ATDE ~/atmark-dist/user/qt5]$ cd hello_qml
[ATDE ~/atmark-dist/user/qt5/hello_qml]$ vi Makefile
```

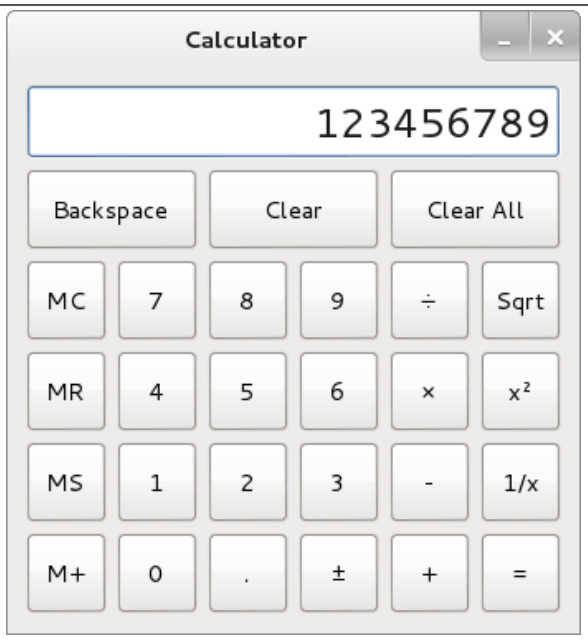
```
all:
#nothing to do here

romfs install_target:
    mkdir -p $(ROMFSDIR)/usr/share/qt5/hello_qml
    $(ROMFSINST) hello_qml.qml \
        /usr/share/qt5/hello_qml/hello_qml.qml
```

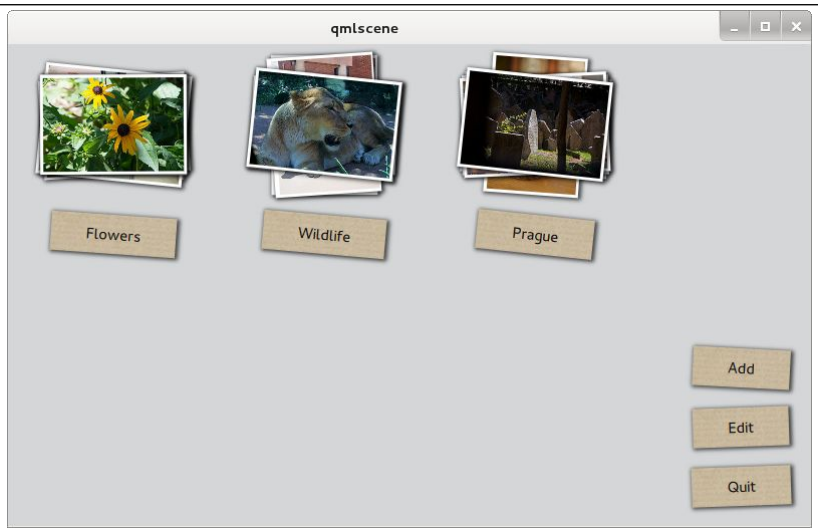
これで、atmark-dist に hello_qml を追加することができました。Armadillo-840 用に atmark-dist をビルドするとユーザーランドイメージに/usr/share/qt5/hello_qml/hello_qml.qml が追加されます。フラッシュメモリを更新して Armadillo を起動後、次のように実行することができます。

```
[armadillo ~]# qmlscene /usr/share/qt5/hello_qml/hello_qml.qml
```

14.7. サンプルソースコード

ATDE には、ユーザーインターフェースの開発に参考となるソースコードが標準的にインストールされています。Armadillo で参考となりそうなものをいくつかリストアップします。

名称	calculator
内容	Qt アプリケーションでボタンやエディットボックスを使用した基本的なアプリケーションです
パス	/usr/arm-linux-gnueabihf/lib/qt5/examples/widgets/widgets/calculator/
備考	
画像	

名称	photoviewer
内容	QML で記述されたフォトビューワーです。画像データはインターネットから取得しています
パス	/usr/arm-linux-gnueabihf/lib/qt5/examples/quick/demos/photoviewer/
備考	Armadillo-840 の工場出荷状態では、デフォルトのアプリケーションとなっています
画像	

14.8. リファレンス

Qt Creator Manual

http://qt-project.org/doc/qtcreator-2.7/index.html

Qt QML

http://qt-project.org/doc/qt-5.0/qtqml/qtqml-index.html

Qt Examples And Tutorials

http://qt-project.org/doc/qt-5.0/qtdoc/qtexamplesandtutorials.html

Qt class reference

http://qt-project.org/doc/qt-5.0/qtdoc/classes.html

15. AV コーデックミドルウェア

15.1. AV コーデックミドルウェアとは

AV コーデックミドルウェアは、Armadillo-800 シリーズでマルチメディア処理 (H.264/AVC 動画・AAC 音声・JPEG 画像の変換処理) をスムーズかつ効率的に行うためのミドルウェアです。Armadillo-800 シリーズに搭載されているアプリケーションプロセッサー「R-Mobile A1」には、メインの ARM コア以外にリアルタイム制御用の SH-4A コアとマルチメディア処理専用プロセッサが搭載されています。マルチメディア処理は、多くの場合システムに負荷をかけることが多く、メイン CPU で処理をするとシステム全体のパフォーマンスが低下します。AV コーデックミドルウェアを利用することで、メイン CPU のパフォーマンスを落すことなく、マルチメディアの処理を行うことができるようになります。

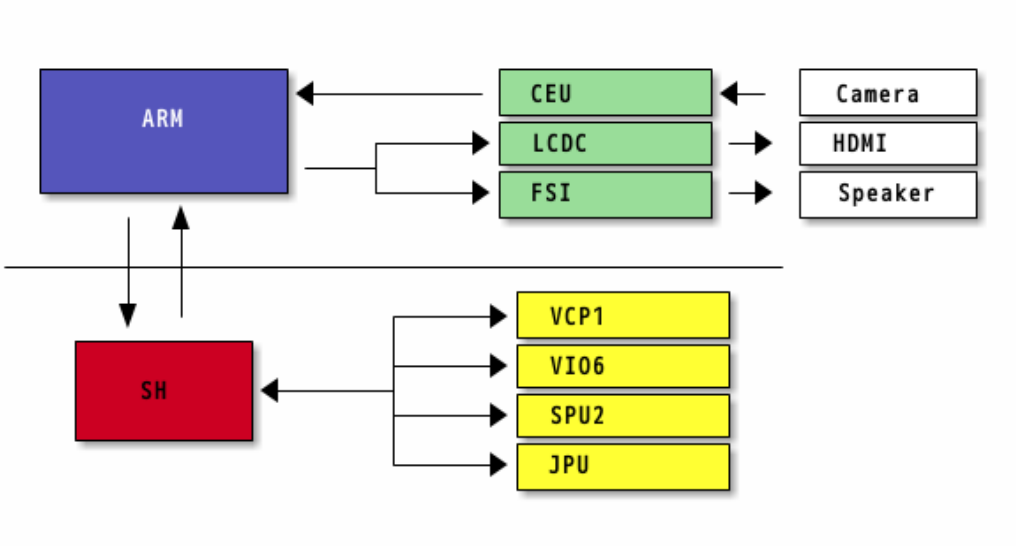


図 15.1 AV コーデックミドルウェア使用時の内蔵コアの対応

上図に示すように、AV コーデックミドルウェアを使う場合には R-Mobile A1 に搭載されている以下の専用プロセッサは SH 側で管理されます。

- Video Multi Codec (VCP1)
- Video I/O 6 (VIO6)
- Sound Processing Unit 2 (SPU2)
- JPEG Processing Unit (JPU)

AV コーデックミドルウェアが対応しているフォーマットは以下の通りです。

- デコーダーが対応しているフォーマット
  - H.264/AVC
  - AAC
- エンコーダーが対応しているフォーマット
  - H.264/AVC
  - AAC
  - JPEG

マルチメディアは、比較的大きなデータを扱います。そのためマルチメディア処理を行う専用プロセッサや管理する SH もメインメモリを利用する必要があります。AV コーデックミドルウェアを利用する時には、Armadillo-800 シリーズに搭載されている DDR3 SDRAM 512MB のうち 128MB を AV コーデックミドルウェアに割り当てる必要があります。AV コーデックミドルウェアドライバが追加された linux-3.4-at6 以降では、デフォルトで Linux カーネルが管理するメモリを 384MB に制限しています。

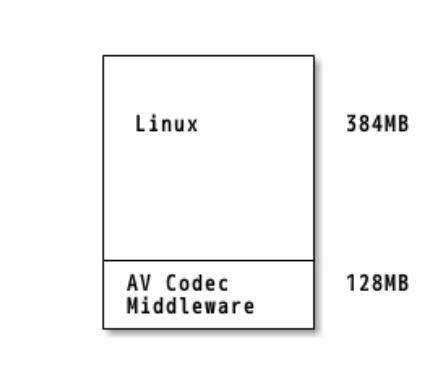


図 15.2 AV コーデックミドルウェア使用時のメモリマップ

このように、多くのプロセッサが協調動作をすることで、Armadillo-800 シリーズでは H.264/AVC 動画や AAC 音声などのマルチメディアデータをスムーズかつ効率的に扱うことができます。

さらに AV コーデックミドルウェアは、アプリケーションプログラムからマルチメディアデータを容易に扱えるよう、Linux で標準のマルチメディアフレームワーク GStreamer [http://gstreamer.freedesktop.org/]に対応しています。

図 15.3 GStreamer ロゴ

GStreamer は、エレメントと呼ばれる処理単位を繋ぎ合わせパイプラインを作成することで、複雑な要求に対応できるマルチメディアフレームワークです。MP4 ファイルを扱うエレメントや、様々な画像フォーマットを変換するエレメントなどが標準で用意されているため、このフレームワークを活用することで、様々なフォーマットのマルチメディアデータを統一的な操作で、簡単に扱うことができます。

15.2. AV コーデックミドルウェアの仕様

15.2.1. AAC デコーダー

表 15.1 AAC デコーダー仕様

符号化方式	・準拠規格 ISO/IEC 13818-7:2006、ISO/IEC 14496-3:2009 ・対応フォーマット AAC-LC、HE-AAC v1、HE-AAC v2
ビットレート	・AAC-LC 8k～576k bit/sec、VBR ・HE-AAC v1/v2 8k～144k bit/sec、VBR
入力フォーマット	・RAW 形式 / ADTS 形式

入力チャネル	・AAC-LC [a] ・1 チャネル(モノラル) ・2 チャネル (ステレオ、デュアルモノラル、パラメトリックステレオ) ・4 チャネル (3/1) [b] ・5 チャネル (3/2)[b] ・5.1 チャネル (3/2+LFE)[b] ・HE-AAC v1/v2 ・1 チャネル (モノラル) ・2 チャネル (ステレオ、デュアルモノラル、パラメトリックステレオ)
出力チャネル	1 チャネルか 2 チャネル (ダウンミックス対応)
サンプリング周波数	・AAC-LC: 8k/11.025k/12k/16k/22.05k/24k/32k/44.1k/48k/64k/88.2k/96k Hz ・HE-AAC v1/v2: 8k/11.025k/12k/16k/22.05k/24k Hz

[a] 3 チャネル(3/0、2/1)、4 チャネル(2/2)は非サポート

[b] (/)は、(前方/後方スピーカのチャネル数)を示す

15.2.2. H.264/AVC デコーダー

表 15.2 H.264/AVC デコーダー仕様

最大ビットレート	40M bit/sec [a]
入力フォーマット	Video Elementary Stream
出力フォーマット	・YUV420 ・RGB32 ・RGB24 ・RGB16
サポートプロファイル/レベル	・Baseline Profile Level 4.1 ・Constrained Baseline Profile Level 4.1 ・Main Profile Level 4.1 ・High Profile Level 4.1
プロファイル共通非サポートツール	・ASO (Arbitrary Slice Ordering) ・FMO (Flexible Macroblock Ordering) ・RS (Redundant Slices)
ピクチャ構造	・フレーム構造 (プログレッシブシーケンス / インタレースシーケンス [b]) ・フィールド構造 (インタレース) ・フレーム / フィールド混在構造 (インタレースシーケンス)
エントロピー符号化	CAVLC / CABAC
ピクチャタイプ	I / P / B ピクチャ
マルチスライス	サポート。1 フレームあたりの最大スライス数は 68 スライスまで(画像サイズ 1920x1080 の場合、1 スライス/1 マクロブロックライン相当)
マルチリファレンス	サポート
マルチシーケンス	サポート。ただし、シーケンスを通じて以下の条件満すこと ・画像サイズ、フレームレートが変化しないこと ・エンコード時のビットレート設定が同じであること
マルチストリーム	非サポート
データ・パーティショニング	非サポート
画像サイズ	・プログレッシブシーケンスの場合: 128x96〜1920x1080 [c] ・インターレースシーケンスの場合: 352x480〜1920x1080 [d]
画像の拡大・縮小	幅、高さともに拡大: 最大 16 倍まで、縮小: 最少 1/16 まで

[a] 最大ピーク時のビットレートが 40M bit/sec 以下であること

[b] インタレースシーケンスの場合、30fps の場合画像サイズ 1440x1080 まで、画像サイズ 1920x1080 であれば 20fps まで

[c] 水平 2 画素、垂直 2 ライン単位で設定可能

[d] 水平 2 画素、垂直 4 ライン単位で設定可能

15.2.3. AAC エンコーダー

表 15.3 AAC エンコーダー仕様

準拠規格	・ISO/IEC 13818-7:2006

ビットレート	16k ～ 288k bps、VBR [a]
対応チャネル	・1 チャネル(モノラル) ・2 チャネル (ステレオ、デュアルモノラル)
入力フォーマット	16bit PCM
入力サンプリング周波数	8k/11.025k/12k/16k/22.05k/24k/32k/44.1k/48k Hz
出力フォーマット	AAC-LC(RAW 形式、ADTS 形式)

[a] 1 チャネルあたり

15.2.4. H.264/AVC エンコーダー

表 15.4 H.264/AVC エンコーダー仕様

最大ビットレート	40M bit/sec [a]
最大フレームレート	水平画像サイズ 1280 以下かつ垂直画像サイズ 720 以下の場合 ・60 frame/sec 上記以外 ・30 frame/sec
最大参照フレーム	2 フレーム
入力画像サイズ	画像幅 ・80～1920 [b] 画像高さ ・80～1088 [c] [d]
入力フォーマット	YUV420
出力フォーマット	Video Elementary Stream
プロファイル/レベル	条件により自動で選択 ・Baseline Profile Level 4.1 以下条件を全て満たす場合に選択されます ・フレーム構造(プログレッシブシーケンス) ・B ピクチャなし ・画像サイズが 1920×1080 未満 ・Main Profile Level 4.1 以下条件を満たすときに選択されます ・B ピクチャあり ・High Profile Level 4.1 以下条件を満たすときに選択されます ・画像サイズが 1920×1080
プロファイル共通非サポートツール	・ASO (Arbitrary Slice Order) ・FMO (Flexible Macroblock Ordering) ・RS (Redundant Slice) ・MBAFF (Macroblock-Adaptive Frame-Field) coding ・Weighted Prediction
ピクチャ構造	フレーム構造 (プログレッシブシーケンス)
エントロピー符号化	Baseline Profile の場合 ・CAVLC Main Profile または High Profile の場合 ・CABAC

動き探索範囲	水平 ・ -64～63.75 画素 ・ -32～31.75 画素(画像幅が 144 以下のとき) 垂直 ・ -32～31.75 サブサンプリング ・ 1/4 サブサンプリング
ピクチャタイプ	I / P / B ピクチャ
イントラ MB リフレッシュ	サポート [e]
イントラ予測方式	4×4、8×8、16×16 画素単位イントラ予測
変換方式	4×4、8×8 整数変換
マルチスライス	非サポート
マルチリファレンス	非サポート
階層エンコード	非サポート
マルチシーケンス	非サポート
マルチストリーム	非サポート
リエントラント対応	なし

[a] 最大ビットレートの設定値であり、レートを保証するものではありません
[b] 2 画素単位で指定可能
[c] 2 ライン単位で指定可能
[d] 画像幅×画像高さが 32 の倍数であること
[e] 先頭のみ I ピクチャのとき適用

15.3. GStreamer - マルチメディアフレームワーク

GStreamer は、オープンソースのマルチメディアフレームワークです。小さなコアライブラリに様々な機能をプラグインとして追加できるようになっており、多彩な形式のデータを扱うことができます。GStreamer で扱うことができるデータフォーマットの一例を下記に示します。

- コンテナフォーマット: mp4, avi, mpeg-ps/ts, mkv/webm, ogg
- 動画コーデック: H.264/AVC, Vorbis
- 音声コーデック: AAC, Theora, wav
- 画像フォーマット: JPEG, PNG, BMP
- ストリーミング: http, rtp

GStreamer では、マルチメディアデータをストリームとして扱います。ストリームを流すパイプラインの中に、エレメントと呼ばれる処理単位を格納し、それらをグラフ構造で繋ぎ合わせることで、デコードやエンコードなどの処理を行います。例えば、「図 15.4. GStreamer の実行例」に示すコマンドを実行した場合のパイプラインは「図 15.5. GStreamer のパイプライン例」となります。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 \
 ! qtdemux name=demux0 \
 demux0.audio_0 ! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 ! alsasink \
 demux0.video_0 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb0
```

図 15.4 GStreamer の実行例

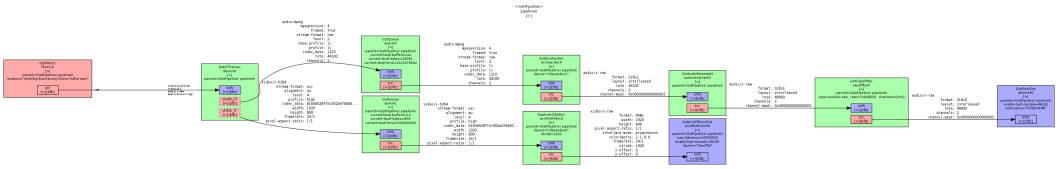


図 15.5 GStreamer のパイプライン例

拡大してご覧ください。

GStreamer のパイプラインは、シェルスクリプトのパイプ構文の構造に似ています。GStreamer の各エレメントとシェルスクリプト内のコマンドを対比することができます。構文的な違いとして、GStreamer のパイプラインは「! 」を使って各エレメントを繋ぎますが、シェルスクリプトは「| 」を使います。

シェルスクリプトで使うコマンドが引数を取るように、GStreamer のエレメントも引数を取ることができます。この引数は、「プロパティ」と呼ばれます。

各エレメントは、データの入出力の口となる「パッド(pad)」を持っています。実際にエレメント同士を繋いでいるのはパッドです。パッドにはデータを次のエレメントに渡す「ソースパッド(source pad)」とデータを受け取る「シンクパッド(sink pad)」が存在します。図中の「src 」と書かれた小さな箱と「sink 」と書かれた小さな箱がそれぞれに該当します。

パッドは自分が出力可能な、または入力可能なフォーマットを知っています。これを「ケイパビリティー (Capability)」と言います。パッドは、パイプラインが作られる時に繋がる相手パッドが持っているケイパビリティーを確認します。お互いのケイパビリティーが一致しない場合はデータの受け渡しができませんので、エラーとなり、最終的にパイプライン生成自体がエラーとなります。

ソースパッドしか持たないエレメントを「ソースエレメント」、シンクパッドしか持たないエレメントを「シンクエレメント」と呼びます。図中左端にある「filesrc0」がソースエレメントで、右端の「alsasink0」「acmfbdevsink」の 2 つがシンクエレメントになります。マルチメディアデータは、一番左側のソースエレメントから右端のシンクエレメントに流れることで形を変えていき、最終的に動画や音声として再生されることになります。

上記例は、GStreamer のデバッグ/プロトタイピング用のコマンドラインツールである gst-launch-1.0 を使って説明しましたが、GStreamer はライブラリとして提供されているため、GStreamer を使ったマルチメディア機能を自作のアプリケーションプログラムに組み込むことができます。API やアプリケーション開発マニュアルは、gstreamer.freedesktop.org の Documentation ページ [http://gstreamer.freedesktop.org/documentation/]から参照することができます。

AV コーデックミドルウェア用のエレメントには、下記のものがあります[1]。以降の章では、これらのエレメントの使い方を中心に説明します。

- H.264 デコーダー: acmh264dec
- AAC デコーダー: acmaacdec
- H.264 エンコーダー: acmh264enc
- AAC エンコーダー: acmaacenc

[1]Armadillo-800 シリーズ用の環境では、NEON 対応した libjpeg turbo が導入されています。libjpeg turbo を使うことで十分に高速な JPEG デコードを行うことができるため、AV コーデックミドルウェアには JPEG デコーダーは含まれていません。

・ JPEG エンコーダー: acmjpegenc
・ フレームバッファ用シンクエレメント: acmfbdevsink

環境にインストールされているエレメント一覧を取得したり、各エレメントの取れるケイパビリティや指定可能なプロパティは gst-inspect-1.0 コマンドを使うことで調べることができます。

```
[armadillo ~]# gst-inspect-1.0
acmaacdec:  acmaacdec: ACM AAC audio decoder
acmh264enc:  acmh264enc: ACM H264 video encoder
acmfbdevsink:  acmfbdevsink: ACM fbdev video sink
acmh264dec:  acmh264dec: ACM H264 video decoder
acmaacenc:  acmaacenc: ACM AAC audio encoder
acmjpegenc:  acmjpegenc: ACM Jpeg encoder
video4linux2:  v4l2radio: Radio (video4linux2) Tuner
video4linux2:  v4l2sink: Video (video4linux2) Sink
video4linux2:  v4l2src: Video (video4linux2) Source
fbdevsink:  fbdevsink: fbdev video sink
udp:  udpsink: UDP packet sender
udp:  multiudpsink: UDP packet sender
udp:  dynudpsink: UDP packet sender
udp:  udpsrc: UDP packet receiver
(省略)
```

図 15.6 エレメント一覧の取得

```
[armadillo ~]# gst-inspect-1.0 acmh264dec
Factory Details:
  Rank:         primary (256)
  Long-name:            ACM H264 video decoder
  Klass:                Codec/Decoder/Video
  Description:          ACM H.264/AVC decoder
  Author:               Atmark Techno, Inc.

Plugin Details:
  Name:                 acmh264dec
  Description:          ACM H264 Decoder
  Filename:             /usr/lib/gstreamer-1.0/libgstacmh264dec.so
  Version:              1.0.0
  License:              LGPL
  Source module:        gst-plugins-acm
  Binary package:       GStreamer ACM Plugins
  Origin URL:           http://armadillo.atmark-techno.com/

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstVideoDecoder
                         +----GstAcmH264Dec

Pad Templates:
  SINK template: 'sink'
    Availability: Always
    Capabilities:
      video/x-h264
```

```
          stream-format: avc
              alignment: au
                  width: [ 80, 1920 ]
                 height: [ 80, 1080 ]
              framerate: [ 0/1, 2147483647/1 ]

  SRC template: 'src'
    Availability: Always
    Capabilities:
      video/x-raw
                  format: RGB16
                   width: [ 80, 1920 ]
                  height: [ 80, 1080 ]
              framerate: [ 0/1, 2147483647/1 ]
      video/x-raw
                  format: RGB
                   width: [ 80, 1920 ]
                  height: [ 80, 1080 ]
              framerate: [ 0/1, 2147483647/1 ]
      video/x-raw
                  format: RGBx
                   width: [ 80, 1920 ]
                  height: [ 80, 1080 ]
              framerate: [ 0/1, 2147483647/1 ]
      video/x-raw
                  format: NV12
                   width: [ 80, 1920 ]
                  height: [ 80, 1080 ]
              framerate: [ 0/1, 2147483647/1 ]


Element Flags:
  no flags set

Element Implementation:
  Has change_state() function: gst_video_decoder_change_state

Element has no clocking capabilities.
Element has no indexing capabilities.
Element has no URI handling capabilities.

Pads:
  SINK: 'sink'
    Implementation:
      Has chainfunc(): gst_video_decoder_chain
      Has custom eventfunc(): gst_video_decoder_sink_event
      Has custom queryfunc(): gst_video_decoder_sink_query
      Has custom iterintlinkfunc(): gst_pad_iterate_internal_links_default
    Pad Template: 'sink'
  SRC: 'src'
    Implementation:
      Has custom eventfunc(): gst_video_decoder_src_event
      Has custom queryfunc(): gst_video_decoder_src_query
      Has custom iterintlinkfunc(): gst_pad_iterate_internal_links_default
    Pad Template: 'src'

Element Properties:
  name                : The name of the object
```

```
                        flags: readable, writable
                        String. Default: "acmh264dec0"
  parent              : The parent of the object
                        flags: readable, writable
                        Object of type "GstObject"
  device              : The video device eg: /dev/video0
                        flags: readable, writable
                        String. Default: null
  stride              : Stride of output video. (0 is unspecified)
                        flags: readable, writable
                        Unsigned Integer. Range: 0 - 65535 Default: 0
  x-offset            : X Offset of output video. (0 is unspecified)
                        flags: readable, writable
                        Unsigned Integer. Range: 0 - 65535 Default: 0
  y-offset            : Y Offset of output video. (0 is unspecified)
                        flags: readable, writable
                        Unsigned Integer. Range: 0 - 65535 Default: 0
  buf-pic-cnt         : Number of buffering picture
                        flags: readable, writable
                        Unsigned Integer. Range: 2 - 145 Default: 17
  enable-vio6         : FALSE: disable, TRUE: enable
                        flags: readable, writable
                        Boolean. Default: true
```

図 15.7 エレメント情報の取得

15.4. 有効化/無効化

AV コーデックミドルウェアは、SH で動作するファームウェアと、Linux 上で動作するデバイスドライバが協調して機能します。そのため、AV コーデックミドルウェアを有効化するには、SH にファームウェアをロードさせた後、ドライバをロードする必要があります。

SH ファームウェアのロードとドライバのロードを行うには、/sys/devices/platform/acm.0/codec に"encoder"または"decoder"という文字列を書き込みます。AV コーデックミドルウェアを無効化する場合は、"none"という文字列を書き込みます。

```
[armadillo ~]# echo encoder > /sys/devices/platform/acm.0/codec
acm_h264enc: H.264 Encoder of AV Codec Middleware
acm_aacenc: AAC Encoder of AV Coenc Middleware
acm_jpegenc: JPEG Encoder of AV Codec Middleware
```

図 15.8 AV コーデックミドルウェアの有効化(エンコーダー)

```
[armadillo ~]# echo decoder > /sys/devices/platform/acm.0/codec
acm_h264dec: H.264 Decoder of AV Codec Middleware
acm_aacdec: AAC Decoder of AV Codec Middleware
```

図 15.9 AV コーデックミドルウェアの有効化(デコーダー)

```
[armadillo ~]# echo none > /sys/devices/platform/acm.0/codec
```

図 15.10 AV コーデックミドルウェアの無効化

標準状態では、起動時に/etc/config/rc.local で AV コーデックミドルウェアを有効化しています。Armadillo-810 の場合エンコーダーが、Armadillo-840 の場合はデコーダーが有効になります。詳細は「9.1.4. /etc/config/rc.local」を参照してください。

AV コーデックミドルウェアの現在の状態は、/sys/devices/platform/acm.0/codec を読み出すことで確認できます。

```
[armadillo ~]# cat /sys/devices/platform/acm.0/codec
decoder [encoder] none
```

図 15.11 AV コーデックミドルウェアの状態確認(エンコーダーが有効化されている場合)

```
[armadillo ~]# cat /sys/devices/platform/acm.0/codec
[decoder] encoder none
```

図 15.12 AV コーデックミドルウェアの状態確認(デコーダーが有効化されている場合)

```
[armadillo ~]# cat /sys/devices/platform/acm.0/codec
decoder encoder [none]
```

図 15.13 AV コーデックミドルウェアの状態確認(無効化されている場合)

15.5. デコード

15.5.1. コンテナの扱い

ビデオ(映像)とオーディオ(音声)データを一つのファイルにまとめる(多重化する)ためのフォーマットをコンテナフォーマットと言います。H.264/AVC と AAC を格納可能なコンテナフォーマットには、MP4(MPEG-4 Part 14)、AVI(Audio Video Interface)、Matroska などがあります。また、MPEG2 TS(MPEG-2 Transport Stream)や MPEG2 PS(MPEG-2 Program Stream)は、拡張規格で H.264/AVC と AAC に対応しています。

GStreamer では、コンテナに格納されたビデオやオーディオのデータを取り出す場合、デマルチプレクサエレメントを使用します。

MP4 コンテナからビデオとオーディオを取り出し再生するパイプラインの例として、「図 15.4. GStreamer の実行例」で示したコマンドラインを再掲します。MP4 コンテナからビデオやオーディオを取り出す場合、qtdemux エレメントを使用します。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 \
 ! qtdemux name=demux0 \
```

```
demux0.audio_0 ! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 ! alsasink \
demux0.video_0 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb0
```

図 15.14 ビデオとオーディオを再生する

qtdemux エレメントは、MP4 コンテナに格納されているデータストリームごとに動的にソースパッドを作成します。ビデオ用に動的に生成されたパッドには video_*N* という名前が、オーディオ用に生成されたパッドには audio_*N* という名前がつきます。

パッド名は省略して、下記のように記述することもできます。GStreamer では、エレメント同士を接続(リンク)する際に、お互いのソースパッド(出力)とシンクパッド(入力)が受け渡しできるフォーマット(ケイパビリティ)が一致するかネゴシエーションを行い、ケイパビリティが一致した場合だけパッドがリンクされます。queue エレメントはバッファリングを行うためのエレメントで、どのようなフォーマットのデータも受け渡しでき、シンクパッドとソースパッドのケイパビリティは同じものになります。そのため、qtdemux が動的に生成したオーディオ用のパッドにはオーディオを扱う acmaacdec にリンクされている queue エレメントのシンクパッドが、ビデオ用のパッドにはビデオを扱う acmh264dec エレメントがリンクされている queue エレメントのシンクパッドがリンクされます。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 \
 ! qtdemux name=demux0 \
 demux0. ! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 ! alsasink \
 demux0. ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb0
```

図 15.15 ビデオとオーディオを再生する(パッド名の省略)

ビデオだけやオーディオだけを再生したい場合は、下記のように書くこともできます。この場合、qtdemux の name プロパティも省略されていますが、qtdemux が動的に生成したパッドのうち、後続するエレメントが受け取れるパッドだけがリンクした状態となります。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
! queue ! acmh264dec ! acmfbdevsink
```

図 15.16 ビデオのみ再生する

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
! queue ! acmaacdec ! audioresample ! audio/x-raw,rate=48000,channels=2 ! alsasink
```

図 15.17 オーディオのみ再生する

15.5.2. ビデオのデコード

15.5.2.1. 出力先を指定する

ビデオの出力先は、acmfbdevsink の device プロパティにフレームバッファのデバイスファイル名を指定することで変更できます。device プロパティを指定しなかった場合、/dev/fb0 を使用します。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb0
```

図 15.18 ビデオを再生し HDMI ディスプレイに表示する

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-800x480.mp4 ! qtdemux \
 ! queue ! acmh264dec ! acmfbdevsink device=/dev/fb1
```

図 15.19 ビデオを再生し LCD に表示する

15.5.2.2. 拡大/縮小する

acmh264dec エレメントの出力側ケイパビリティに width と height を指定することで、デコードした画像を拡大/縮小して表示することもできます。800×480 サイズのビデオを Full HD サイズに拡大して表示する場合のパイプラインは「図 15.20. ビデオを拡大する」となります。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-800x480.mp4 ! qtdemux \
 ! queue ! acmh264dec ! video/x-raw,width=1920,height=1080 ! acmfbdevsink device=/dev/fb0
```

図 15.20 ビデオを拡大する

同様に、縮小して表示することもできます。acmh264dec は、出力側のケイパビリティに合わせて R-Mobile A1 の VIO を使用して出力画像の拡大/縮小を行うため、ARM CPU 側の負荷なく拡大/縮小を行うことができます。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
 ! queue ! acmh264dec ! video/x-raw,width=800,height=480 ! acmfbdevsink device=/dev/fb1
```

図 15.21 ビデオを縮小する

15.5.2.3. 表示位置の指定

大きなディスプレイに小さな動画を表示したい場合など、出力画像と表示するディスプレイの幅が一致していない場合、acmh264dec エレメントの stride プロパティを指定する必要があります。800×480 サイズの動画をそのままのサイズで Full HD サイズの HDMI ディスプレイに表示する場合は、下記のコマンドになります。stride には出力先の幅を指定してください。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-800x480.mp4 ! qtdemux \
 ! queue ! acmh264dec stride=1920 ! acmfbdevsink device=/dev/fb0
```

図 15.22 小さな動画を大きなディスプレイに表示する

x-offset と y-offset プロパティを使うことで、任意の位置に画像を表示することもできます。下記のコマンドを実行すると、Full HD サイズのディスプレイの右下に 800×480 サイズの動画が表示されます。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-800x480.mp4 ! qtdemux \
 ! queue ! acmh264dec stride=1920 x-offset=1120 y-offset=600 ! acmfbdevsink device=/dev/fb0
```

図 15.23 オフセットを指定して任意の位置に表示する

stride、x-offset、y-offset の関係を「図 15.24. ストライドとオフセットの関係」に示します。Full HD サイズのディスプレイの右下に 800×480 サイズの動画を表示する場合、x-offset=1920-800=1120、y-offset=1080-480=600 として計算できます。

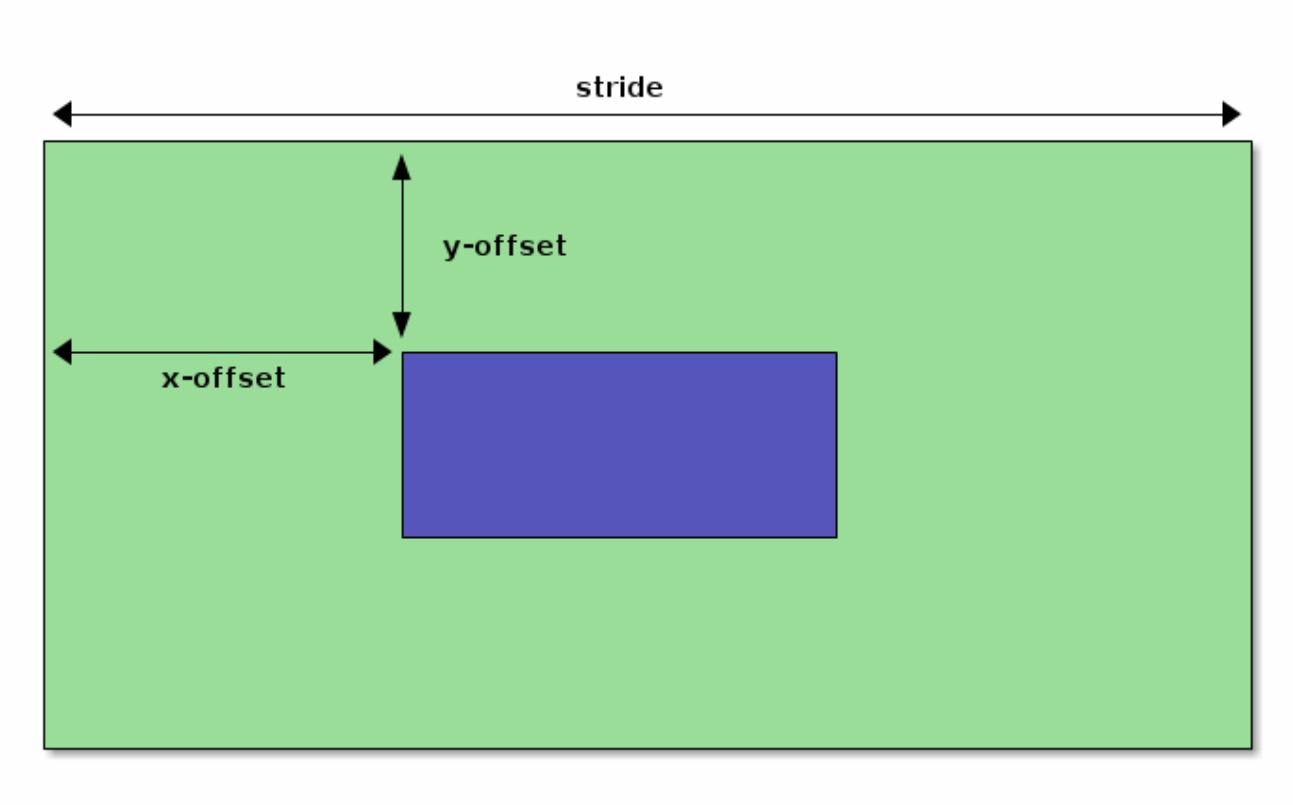


図 15.24 ストライドとオフセットの関係

15.5.3. オーディオのデコード

15.5.3.1. 出力先を指定する

オーディオの出力先は、alsasink の device プロパティに ALSA デバイス名を指定することで変更できます。device プロパティを指定しなかった場合、hw:0 を使用します。

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
 ! queue ! acmaacdec ! alsasink device=hw:0
```

図 15.25 オーディオを HDMI オーディオインターフェース(Armadillo-840: CON3)に出力する

```
[armadillo ~]# gst-launch-1.0 filesrc location=/mnt/big-buck-bunny-30sec-fullhd.mp4 ! qtdemux \
 ! queue ! acmaacdec ! alsasink device=hw:1
```

図 15.26 オーディオをステレオヘッドホン出力インターフェース(拡張ボード 01: CON6)に出力する

15.5.3.2. オーディオフォーマットの変換

Armadillo-840 の HDMI オーディオインターフェース(Armadillo-840: CON3)は、signed 16bit little endian、interleaved、48kHz、2ch PCM のみ出力可能です。入力ファイルに含まれるオーディオがそれ以外のフォーマットやサンプリング周波数の場合、変換して出力する必要があります。オーディオフォーマットの変換には audioconvert エレメントを、サンプリング周波数の変換には audioresample エレメントを使います。

```
[armadillo ~]# gst-launch-1.0 filesrc location=sample.mp4 ! qtdemux \
 ! queue ! acmaacdec ! audioconvert ! audioresample \
 ! audio/x-raw,format=S16LE,layout=interleaved,rate=48000,channels=2 \
 ! alsasink device=hw:1
```

図 15.27 オーディオフォーマットを変換する

15.6. エンコード

15.6.1. コンテナの扱い

ビデオ(映像)とオーディオ(音声)データを一つのファイルにまとめる(多重化する)ためのフォーマットをコンテナフォーマットと言います。H.264/AVC と AAC を格納可能なコンテナフォーマットには、MP4(MPEG-4 Part 14)、AVI(Audio Video Interface)、Matroska などがあります。また、MPEG2 TS(MPEG-2 Transport Stream)や MPEG2 PS(MPEG-2 Program Stream)は、拡張規格で H.264/AVC と AAC に対応しています。

GStreamer では、ビデオやオーディオのデータをコンテナに格納する場合、マルチプレクサエレメントを使用します。

ビデオのみを MP4 コンテナに格納するパイプラインの例を「図 15.28. ビデオをエンコードしてコンテナに格納する」に示します。MP4 コンテナに格納する場合、qtmux エレメントを使用します。

```
[armadillo ~]# gst-launch-1.0 -e videotestsrc \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmh264enc ! queue \
 ! qtmux ! filesink location=/mnt/output.mp4
```

図 15.28 ビデオをエンコードしてコンテナに格納する

画像の入力ソースとして、様々なパターンの画像を様々なフォーマットで生成できる videotestsrc を使用しています。acmh264enc エレメントは、入力フォーマットとして NV12 形式のみを受付ますので、acmh264enc の入力側のケイパビリティを指定しています[2]。videotestsrc は指定されたフォーマットで画像を出力します。

[2]エレメントが受け付けられるフォーマットは、gst-inspect-1.0 を実行したときの"SINK template"の"Capabilities"で確認できます。

また、パイプライン停止時に EOS イベントを発行するように、gst-launch-1.0 コマンドに-e オプションを付けています。エンコードを終了するには、Ctrl-C で gst-launch-1.0 コマンドを停止してください。

オーディオのみを MP4 コンテナに格納するパイプラインの例を「図 15.29. オーディオをエンコードしてコンテナに格納する」に示します。ここでも、様々なパターンのオーディオデータを生成できる audiotestsrc を使用しています。audiotestsrc が出力するフォーマットと、acmaacenc が受け付けられるフォーマットを合わせるため、ケイパビリティを指定しているのはビデオの場合と同じです。

```
[armadillo ~]# gst-launch-1.0 -e audiotestsrc wave=sine freq=1000 \
 ! audio/x-raw,format=S16LE,layout=interleaved,rate=48000,channels=2 \
 ! acmaacenc ! queue \
 ! qtmux ! filesink location=/mnt/output.mp4
```

図 15.29 オーディオをエンコードしてコンテナに格納する

ビデオとオーディオを同時に MP4 コンテナに格納するパイプラインの例を「図 15.30. ビデオとオーディオをエンコードしてコンテナに格納する」に示します。

```
[armadillo ~]# gst-launch-1.0 -e videotestsrc \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmh264enc ! queue ! muxer.video_0 \
 audiotestsrc wave=sine freq=1000 \
 ! audio/x-raw,format=S16LE,layout=interleaved,rate=48000,channels=2 \
 ! acmaacenc ! queue ! muxer.audio_0 \
 qtmux name="muxer" ! filesink location=/mnt/output.mp4
```

図 15.30 ビデオとオーディオをエンコードしてコンテナに格納する

qtmux エレメントは、入力されるビデオとオーディオそれぞれに対して、動的にシンクパッドを作成します。ビデオ用に動的に生成されたパッドには video_*N* という名前が、オーディオ用に生成されたパッドには audio_*N* という名前がつきます。

パッド名は省略して、下記のように記述することもできます。GStreamer では、エレメント同士を接続(リンク)する際に、お互いのソースパッド(出力)とシンクパッド(入力)が受け渡しできるフォーマット(ケイパビリティ)が一致するかネゴシエーションを行い、ケイパビリティが一致した場合だけパッドがリンクされます。queue エレメントはバッファリングを行うためのエレメントで、どのようなフォーマットのデータも受け渡しでき、シンクパッドとソースパッドのケイパビリティは同じものになります。そのため、qtmux が動的に生成したオーディオ用のパッドにはオーディオを扱う acmaacenc にリンクされている queue エレメントのソースパッドが、ビデオ用のパッドにはビデオを扱う acmh264enc エレメントがリンクされている queue エレメントのソースシンクパッドがリンクされます。

```
[armadillo ~]# gst-launch-1.0 -e videotestsrc \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmh264enc ! queue ! muxer. \
 audiotestsrc wave=sine freq=1000 \
 ! audio/x-raw,format=S16LE,layout=interleaved,rate=48000,channels=2 \
 ! acmaacenc ! queue ! muxer. \
 qtmux name="muxer" ! filesink location=/mnt/output.mp4
```

図 15.31 ビデオとオーディオをエンコードしてコンテナに格納する(パッド名の省略)

15.6.2. ビデオのエンコード

15.6.2.1. 入力ソースを指定する

v4l2src エレメントを使うことで、V4L2(Video for Linux 2)デバイスとして実装されているカメラデバイスから画像を取得できます。どのデバイスから画像を取得するかは、v4l2src エレメントの device プロパティにデバイスファイル名を指定することで変更できます。Armadillo-810 カメラモデルの場合、Armadillo-810 カメラモジュール 01 (B コネクタ用)のデバイスファイルは/dev/video1 となります。

```
[armadillo ~]# gst-launch-1.0 -e v4l2src device="/dev/video1" \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmh264enc ! queue \
 ! qtmux ! filesink location=/mnt/output.mp4
```

図 15.32 カメラモジュールからの入力画像をエンコードする

UVC 対応 USB カメラなども同様に v4l2src で扱うことができます。どのデバイスファイルが、どのカメラに対応しているかは/sys/class/video4linux/video*N*/name で確認できます。Armadillo-810 カメラモジュールなど、R-Mobile A1 の CEU に接続されたカメラの場合、"sh_mobile_ceu.*N*"と表示されます。UVC 対応 USB カメラの場合、"USB Camera"などと表示されます。

```
[armadillo ~]# cat /sys/class/video4linux/video0/name
sh_mobile_ceu.0
[armadillo ~]# cat /sys/class/video4linux/video4/name
USB Camera
```

図 15.33 どのデバイスファイルがどのカメラに対応しているか確認する

USB カメラによっては NV12 フォーマットで出力できない場合もあります。そのような場合は、videoconvert エレメントを使うことで画像フォーマットを変更できます。

```
[armadillo ~]# gst-launch-1.0 -e v4l2src device="/dev/video4" \
 ! videoconvert ! video/x-raw,format=NV12,framerate=30/1 \
 ! acmh264enc ! queue \
 ! qtmux ! filesink location=/mnt/output.mp4
```

図 15.34 USB カメラからの入力画像をエンコードする

15.6.2.2. フレームレートを指定する

Armadillo-810 カメラモジュール 01 (B コネクタ用)からの入力画像は、30fps 固定となっています。しかし、そこまでフレームレートが必要ない場合、もっと低いフレームレートでエンコードすることもできます。フレームレートの変更には、videorate エレメントを使います。「図 15.35. フレームレートを指定する」のように指定すると、15fps でエンコードを行います。

```
[armadillo ~]# gst-launch-1.0 -e v4l2src device="/dev/video1" \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! videorate ! video/x-raw,framerate=15/1 \
```

```
! acmh264enc ! queue \
! qtmux ! filesink location=/mnt/output.mp4
```

図 15.35 フレームレートを指定する

15.6.2.3. エンコード品質を指定する

acmh264enc エレメントのプロパティを指定することで、エンコード品質を調整できます。エンコード品質に影響する acmh264enc エレメントのプロパティを下記に示します。

表 15.5 エンコード品質に影響する acmh264enc エレメントのプロパティ

プロパティ	意味	最小値	最大値	デフォルト値
bitrate	目標平均ビットレート[bit/sec]	16,000	40,000,000	8,000,000
max-frame-size	最大フレームサイズ[a][byte]	0	5,000,000	0(推奨値[b]を使用)
rate-control-mode	・ 0: 固定ビットレート(ピクチャスキップあり) ・ 1: 固定ビットレート(ピクチャスキップなし) ・ 2: 可変ビットレート(ピクチャスキップなし)	0	2	2
max-gop-length	最大 GOP 長 ・ 0: ストリームの先頭のみ I ピクチャ ・ 1: 全て I ピクチャ ・ 2 以上: 指定された GOP 長で GOP を生成	0	120	30
b-pic-mode	B ピクチャモード ・ 0: B ピクチャを使用しない ・ 1〜3: B ピクチャを N フレーム挿入[c]	0	3	3

[a]0 以外を指定する場合は、目標平均ビットレート/8[byte]以上を指定すること

[b]固定ビットレート制御時: 目標平均ビットレート/3*4/8[byte]、可変ビットレート制御時: 目標平均ビットレート*2/8[byte]

[c]B ピクチャを挿入する場合、rate-control-mode=0(ピクチャスキップあり)は指定不可

15.6.2.4. 入力画像サイズの制限とオフセットの指定

acmh264enc エレメントへの入力画像サイズには、下記の制限があります。

・ 入力画像幅: 80〜1920 画素(2 の倍数であること)
・ 入力画像高さ: 80〜1088 ライン(2 の倍数であること)
・ 入力画像幅 × 入力画像高さは 32 の倍数であること

また、acmh264enc エレメントの x-offset プロパティと y-offset プロパティを使うことで、入力画像の一部だけをエンコードできます。x-offset と y-offset プロパティには下記の制限があります。

・ [入力画像幅]×[y-offset]+[x-offset]は 32 の倍数であること
・ [入力画像幅]×[y-offset]/2+[x-offset]は 32 の倍数であること

「図 15.36. オフセットを指定する」のように指定すると、640×480 サイズの入力画像のうち、中央の 320×240 サイズだけエンコードを行います。acmh264enc エレメントの出力サイズを設定するために、出力側のケイパビリティも指定している点に注意してください。

```
[armadillo ~]# gst-launch-1.0 -e videotestsrc \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmh264enc x-offset=160 y-offset=120 ! video/x-h264,width=320,height=240 ! queue \
 ! qtmux name="muxer" ! filesink location=output.mp4
```

図 15.36 オフセットを指定する

15.6.3. オーディオのエンコード

15.6.3.1. 入力ソースを指定する

alsasrc エレメントを使うことで、ALSA(Advanced Linux Sound Architecture)デバイスとして実装されているオーディオデバイスから録音できます。どのデバイスから音声を取得するかは、alsasrc エレメントの device プロパティに ALSA デバイス名を指定することで変更できます。Armadillo-840 液晶モデルの場合、Armadillo-840 拡張ボード 01 (C コネクタ用)のマイク入力インターフェース(CON5)に対応する ALSA デバイスは hw:1 となります。

```
[armadillo ~]# gst-launch-1.0 -e alsasrc device="hw:1" \
 ! audio/x-raw,format=S16LE,layout=interleaved,rate=48000,channels=2 \
 ! acmaacenc ! queue \
 ! qtmux ! filesink location=/mnt/output.mp4
```

図 15.37 マイク入力インターフェースからの入力音声をエンコードする

音声入力となることができる ALSA デバイスの一覧は、 arecord -l コマンドで調べられます。card *N* の番号を、alsasrc の device プロパティに指定してください。

```
[armadillo ~]# arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: FSI2AWM8978 [FSI2A-WM8978], device 0: wm8978 wm8978-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Camera [USB Camera], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

図 15.38 ALSA 入力デバイスの一覧表示

15.6.3.2. エンコード品質を指定する

acmaacenc エレメントのプロパティを指定することで、エンコード品質を調整できます。エンコード品質に影響する acmaacenc エレメントのプロパティを下記に示します。

表 15.6 エンコード品質に影響する acmaacenc エレメントのプロパティ

プロパティ	意味	最小値	最大値	デフォルト値
bitrate	目標平均ビットレート[bit/sec]	16,000	288,000	64,000
enable-cbr	固定ビットレート有効 ・ 0: 可変ビットレート ・ 1: 固定ビットレート	0	1	0

15.6.4. JPEG のエンコード

15.6.4.1. JPEG をファイルに保存する

Armadillo-810 カメラモジュール 01 (B コネクタ用)のカメラデバイスから取得した画像を JPEG として保存する場合、下記コマンドを実行します。10 フレームの画像を取得し、output0.jpeg 〜 output9.jpeg の 10 枚の JPEG 画像が生成されます。画像取得開始からカメラの AGC(Auto Gain Control)、AWB(Auto White Balance)の効果が十分に反映されるまでに 6 フレーム程度かかりますので、output0.jpeg 〜 output5.jpeg は色合い、明るさが激しく変化します。

最後の 1 フレームだけ取得したい場合は、 multifilesink エレメントに max-files=1 プロパティを指定してください。

```
[armadillo ~]# gst-launch-1.0 -e v4l2src device=/dev/video1 num-buffers=10 \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmjpegenc ! multifilesink location=/mnt/output%d.jpeg
```

図 15.39 カメラモジュールからの入力画像をエンコードする

下記コマンドを実行すると、JPEG エンコードした画像を Motion JPEG として mov ファイルに格納することができます。

```
[armadillo ~]# gst-launch-1.0 -e v4l2src device=/dev/video1 \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmjpegenc ! qtmux ! filesink location=/mnt/output.mov
```

図 15.40 Motion JPEG としてファイルに保存する

15.6.4.2. エンコード品質を指定する

acmjpegenc エレメントの quality プロパティを指定することで、エンコード品質を調整できます。0〜100 でエンコード品質を指定します。値が小さいほど画像が荒くなりますが、画像サイズは小さくなります。デフォルト値は 75 に設定されています。

15.6.4.3. 入出力画像サイズの制限とオフセットの指定

acmjpegenc エレメントへの入出力画像サイズには、下記の制限があります。

- 入力画像幅: 16〜1920 画素(8 の倍数であること)
- 入力画像高さ: 16〜1072 ライン(16 の倍数であること)
- 出力画像幅: 16〜1920 画素(4 の倍数であること)
- 出力画像高さ: 16〜1072 ライン(4 の倍数であること)

また、acmjpegenc エレメントの x-offset プロパティと y-offset プロパティを使うことで、入力画像の一部だけをエンコードできます。x-offset と y-offset プロパティには、下記の制限があります。

- [入力画像高さ]-[y-offset]が 16 の倍数であること

「図 15.41. オフセットを指定する」のように指定すると、640×480 サイズの入力画像のうち、中央の 320×240 サイズだけエンコードを行います。ajpegenc エレメントの出力サイズを設定するために、出力側のケイパビリティも指定している点に注意してください。

```
[armadillo ~]# gst-launch-1.0 -e videotestsrc num-buffers=10 \
 ! video/x-raw,format=NV12,width=640,height=480,framerate=30/1 \
 ! acmjpegenc x-offset=160 y-offset=112 ! image/jpeg,width=320,height=240 \
 ! multifilesink location=/mnt/output%d.jpeg
```

図 15.41 オフセットを指定する

16. SD ブートの活用

本章では、SD ブートを行うためのブートディスクの作成方法や、ブートディスクにルートファイルシステムを構築する方法など、SD ブートを活用するために必要な情報について説明します。SD ブートとは、SD カードに保存されたブートローダーイメージを起動させることを示します。

開発時に SD ブートを利用すると、以下のようなメリットがあります。

- フラッシュメモリのブートローダーを復旧することができる
- フラッシュメモリに収まらないサイズのソフトウェアを動作させることができる
- SD カードを取り替えるだけでシステムイメージを変更することができる

SD ブートを行った場合でも、ブートローダーの設定(保守モードの setenv/setboodevice コマンドで設定する項目)についてはフラッシュメモリに保存されます。

SD カードに対する作業は、ATDE で行います。そのため、ATDE に SD カードを接続する必要があります。詳しくは「4.2.2. 取り外し可能デバイスの使用」を参照してください。

ATDE に SD カードを接続すると、自動的に/media/ディレクトリにマウントされます。本章に記載されている手順を実行するためには、次のように SD カードをアンマウントしておく必要があります。

```
[ATDE ~]$ mount
(省略)
/dev/sdb1 on /media/52E6-5897 type vfat
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=cp437,iocharset=utf8,sh
ortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
[ATDE ~]$ sudo umount /dev/sdb1
[ATDE ~]$
```

図 16.1 自動マウントされた SD カードのアンマウント

本章で使用するブートローダーイメージファイルなどは、開発セット付属の DVD に収録されています。最新版のファイルは、"Armadillo サイト"で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-840 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-840/downloads

16.1. ブートディスクの作成

ATDE でブートディスクを作成します。ブートディスクの作成に使用するファイルを次に示します。

表 16.1 ブートディスクの作成に使用するファイル

ファイル	ファイル名
SD ブート用ブートローダーイメージ	loader-armadillo840-mmcsd-*[version]*.bin

SD カードにブートローダーイメージを配置する際、「表 16.2. ブートディスクの制約」に示す制約があります。本章に示す手順を実行した場合は問題になることはありませんが、独自のブートディスクを作成する場合は注意してください。

表 16.2 ブートディスクの制約

項目	制約
パーティション番号	1
パーティションのシステムタイプ	0xb(Win95 FAT32)
ファイルシステム	FAT32
ブートローダーイメージファイル名	sdboot.bin
ブートローダーイメージファイルの配置場所	ルートディレクトリ直下

「表 16.3. ブートディスクの構成例」に示すブートディスクを作成する手順を、「手順 16.1. ブートディスクの作成例」に示します。

表 16.3 ブートディスクの構成例

パーティション番号	パーティションサイズ	ファイルシステム	説明
1	128MByte	FAT32	SD ブート用のブートローダーイメージを配置します。
2	残り全て	ext3	ルートファイルシステムを構築するために ext3 ファイルシステムを構築しておきます。

手順 16.1 ブートディスクの作成例

1. SD ブート用のブートローダーイメージファイルを取得します。

```
[ATDE ~]$ ls
loader-armadillo840-mmcsd-[version].bin
```

フラッシュメモリ用のブートローダーイメージを SD カードに配置しても起動することができません。事前にファイル名を確認してください。ブートローダーイメージファイルには以下 2 種類があります。

格納場所	イメージファイル
SD カード	loader-armadillo840-mmcsd-[version].bin
フラッシュメモリ	loader-armadillo840-nor-[version].bin

2. SD カードに 2 つのプライマリパーティションを作成します。

```
[ATDE ~]$ sudo fdisk /dev/sdb ❶

Command (m for help): o ❷
Building a new DOS disklabel with disk identifier 0x8cb9edcc.
Changes will remain in memory only, until you decide to write them.
```

```
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n ❸
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): ❹
Using default response p
Partition number (1-4, default 1): ❺
Using default value 1
First sector (2048-3862527, default 2048): ❻
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-3862527, default 3862527): +128M ❼

Command (m for help): n ❽
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): ❾
Using default response p
Partition number (1-4, default 2): ❿
Using default value 2
First sector (264192-3862527, default 264192): ⓫
Using default value 264192
Last sector, +sectors or +size{K,M,G} (264192-3862527, default 3862527): ⓬
Using default value 3862527

Command (m for help): t ⓭
Partition number (1-4): 1 ⓮
Hex code (type L to list codes): b ⓯
Changed system type of partition 1 to b (W95 FAT32)

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

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
[ATDE ~]$
```

❶　SD カードのパーティションテーブル操作を開始します。USB メモリなどを接続している場合は、SD カードのデバイスファイルが sdc や sdd など本実行例と異なる場合があります。

❷　新しく空の DOS パーティションテーブルを作成します。

❸　新しくパーティションを追加します。

❹ パーティション種別にはデフォルト値(p: プライマリ)を指定するので、そのまま改行を入力してください。

❺ パーティション番号にはデフォルト値(1)を指定するので、そのまま改行を入力してください。

❻ 開始セクタにはデフォルト値(使用可能なセクタの先頭)を使用するので、そのまま改行を入力してください。

❼ 最終シリンダは、128MByte 分を指定します。

❽ 新しくパーティションを追加します。

❾ パーティション種別にはデフォルト値(p: プライマリ)を指定するので、そのまま改行を入力してください。

❿ パーティション番号にはデフォルト値(2)を指定するので、そのまま改行を入力してください。

⓫ 開始セクタにはデフォルト値(第 1 パーティションの最終セクタの次のセクタ)を使用するので、そのまま改行を入力してください。

⓬ 最終セクタにはデフォルト値(末尾セクタ)を使用するので、そのまま改行を入力してください。

⓭ パーティションのシステムタイプを変更します。

⓮ 第 1 パーティションを指定します。

⓯ パーティションのシステムタイプに 0xb(Win95 FAT32)を指定します。

⓰ 変更を SD カードに書き込みます。

3. パーティションリストを表示し、2 つのパーティションが作成されていることを確認してください。

```
[ATDE ~]$ sudo fdisk -l /dev/sdb

Disk /dev/sdb: 1977 MB, 1977614336 bytes
61 heads, 62 sectors/track, 1021 cylinders, total 3862528 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x8cb9edcc

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      264191      131072    b  W95 FAT32
/dev/sdb2          264192     3862527     1799168   83  Linux
```

4. それぞれのパーティションにファイルシステムを構築します。

```
[ATDE ~]$ sudo mkfs.vfat -F 32 /dev/sdb1  ❶
mkfs.vfat 3.0.13 (30 Jun 2012)
[ATDE ~]$ sudo mkfs.ext3 -L rootfs /dev/sdb2  ❷
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=rootfs
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
```

```
Stride=0 blocks, Stripe width=0 blocks
112448 inodes, 449792 blocks
22489 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=461373440
14 block groups
32768 blocks per group, 32768 fragments per group
8032 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

[ATDE ~]$
```

❶ 第 1 パーティションに FAT32 ファイルシステムを構築します。

❷ 第 2 パーティションに ext3 ファイルシステムを構築します。ボリュームラベルには "rootfs"を設定します。

5. SD ブート用のブートローダーイメージファイルを第 1 パーティションに配置します。

```
[ATDE ~]$ ls
loader-armadillo840-mmcsd-[version].bin
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t vfat /dev/sdb1 sd ❷
[ATDE ~]$ sudo cp loader-armadillo840-mmcsd-[version].bin sd/sdboot.bin ❸
[ATDE ~]$ sudo umount sd ❹
[ATDE ~]$ rmdir sd ❺
```

❶ SD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 1 パーティションを sd/ディレクトリにマウントします。

❸ sd/ディレクトリにブートローダーイメージをコピーします。ファイル名は"sdboot.bin" にリネームする必要があります。

❹ sd/ディレクトリにマウントした第 1 パーティションをアンマウントします。

❺ sd/ディレクトリを削除します。

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

16.2. ルートファイルシステムの構築

「16.1. ブートディスクの作成」で作成したブートディスクにルートファイルシステムを構築します。

Atmark Dist または Debian GNU/Linux のルートファイルシステムを構築することができます。ルートファイルシステムの構築に使用するファイルを次に示します。

表 16.4 ルートファイルシステムの構築に使用するファイル

Linux ディストリビューション	ファイル名	ファイルの説明
Atmark Dist	romfs-a840-*[version]*.img.gz	Atmark Dist で作成したユーザーランドイメージ
Debian GNU/Linux	debian-wheezy-armhf_a840_*[version]*.tar.gz	ARM(armhf)アーキテクチャ用 Debian GNU/Linux 7(コードネーム 「wheezy」)のルートファイルシステムアーカイブ

ブートディスクに構築した Atmark Dist ルートファイルシステムからでも、netflash を使用してフラッシュメモリを書き替えることができます。開発時にはフラッシュメモリの復旧用として準備しておくことを推奨します。

16.2.1. Atmark Dist のルートファイルシステムを構築する

Atmark Dist で作成したユーザーランドイメージから、ルートファイルシステムを構築する手順を次に示します。

手順 16.2 Atmark Dist イメージからルートファイルシステムを構築する

1. Atmark Dist で作成したユーザーランドイメージファイル(romfs-a840-*[version]*.img.gz)を準備しておきます。

```
[ATDE ~]$ ls
romfs-a840-[version].img.gz
```

2. ユーザーランドイメージファイルをマウントします。

```
[ATDE ~]$ mkdir romfs ❶
[ATDE ~]$ gzip --stdout --decompress romfs-a840-[version].img.gz > romfs-a840-
[version].img ❷
[ATDE ~]$ ls
romfs  romfs-a840-[version].img  romfs-a840-[version].img.gz
[ATDE ~]$ sudo mount -o loop romfs-a840-[version].img romfs ❸
[ATDE ~]$ ls romfs ❹
bin   dev  home  linuxrc     media  opt   root  sys  usr
boot  etc  lib   lost+found  mnt    proc  sbin  tmp  var
```

❶ ユーザーランドイメージファイルをマウントするための romfs/ディレクトリを作成します。

❷ gzip 形式で圧縮されているユーザーランドイメージファイルを伸長します。

❸ ユーザーランドイメージファイルを romfs/ディレクトリにマウントします。"-o"オプションで"loop"を指定する必要があります。

❹ マウントに成功し、ルートファイルシステムが見えるようになったことを確認します。

イメージファイルをマウントするには

Atmark Dist で作成したユーザーランドイメージファイルのマウントには「loop デバイス」を使用します。loop デバイスを使用すると、イメージファイルをブロック型デバイスとして扱うことができます。loop デバイスを使用したマウントを行うためには、mount コマンドの"-o"オプションで"loop"を指定する必要があります。

3. ルートファイルシステムをブートディスクの第 2 パーティションに構築します。

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext3 /dev/sdb2 sd ❷
[ATDE ~]$ sudo cp -a romfs/* sd ❸
[ATDE ~]$ sudo umount romfs ❹
[ATDE ~]$ rmdir romfs ❺
```

❶ SD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 2 パーティションを sd/ディレクトリにマウントします。

❸ romfs/ディレクトリから sd/ディレクトリにルートファイルシステムをコピーします。

❹ romfs/ディレクトリにマウントしたユーザーランドイメージファイルをアンマウントします。

❺ romfs/ディレクトリを削除します。

4. ユーザーランドイメージファイルの/etc/fstab はフラッシュメモリ用の設定になっているため、SD カード用の設定に変更します。

```
[ATDE ~]$ sudo vi sd/etc/fstab
/dev/mmcblk0p2          /              ext3    defaults              0 1  ❶
proc                    /proc          proc    defaults              0 0
usbfs                   /proc/bus/usb  usbfs   defaults              0 0
sysfs                   /sys           sysfs   defaults              0 0
udev                    /dev           tmpfs   mode=0755             0 0
/dev/flashblk/firmware  /opt/firmware  squashfs        defaults       0 0
/dev/flashblk/license   /opt/license   squashfs        defaults       0 0
[ATDE ~]$ sudo umount sd ❷
[ATDE ~]$ rmdir sd ❸
```

❶ "/dev/ram0"を"/dev/mmcblk0p2"に、"ext2"を"ext3"に変更します。

❷ sd/ディレクトリにマウントしたブートディスクの第 2 パーティションをアンマウントします。

❸ sd/ディレクトリを削除します。

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

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

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

手順 16.3 Debian GNU/Linux ルートファイルシステムアーカイブからルートファイルシステムを構築する

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

```
[ATDE ~]$ ls
debian-wheezy-armhf_a840_[version].tar.gz
```

2. ルートファイルシステムをブートディスクの第 2 パーティションに構築します。

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext3 /dev/sdb2 sd ❷
[ATDE ~]$ sudo tar zxf debian-wheezy-armhf_a840_[version].tar.gz -C sd ❸
[ATDE ~]$ sudo umount sd ❹
[ATDE ~]$ rmdir sd ❺
```

❶ SD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 2 パーティションを sd/ディレクトリにマウントします。

❸ ルートファイルシステムアーカイブを sd/ディレクトリに展開します。

❹ sd/ディレクトリにマウントしたブートディスクの第 2 パーティションをアンマウントします。

❺ sd/ディレクトリを削除します。

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

16.3. Linux カーネルイメージの配置

「16.2.1. Atmark Dist のルートファイルシステムを構築する」または、「16.2.2. Debian GNU/Linux のルートファイルシステムを構築する」で作成したルートファイルシステムに Linux カーネルイメージを配置します。Linux カーネルイメージの配置に使用するファイルを次に示します。

表 16.5 ブートディスクの作成に使用するファイル

ファイル	ファイル名
Linux カーネルイメージ	linux-a840-*[version]*.bin.gz

SD カードに Linux カーネルイメージを配置する際は、次の条件を満たすようにしてください。この条件から外れた場合、ブートローダーが Linux カーネルイメージを検出することができなくなる場合があります。

表 16.6 ブートローダーが Linux カーネルを検出可能な条件

項目	条件
ファイルシステム	ext2 または ext3
圧縮形式	gzip 形式 または 非圧縮
Linux カーネルイメージファイル名(gzip 形式)	Image.gz, linux.gz, Image.bin.gz, linux.bin.gz のいずれか
Linux カーネルイメージファイル名(非圧縮)	Image, linux, Image.bin, linux.bin のいずれか
Linux カーネルイメージファイルの配置場所	/boot/ディレクトリ直下

Linux カーネルイメージをルートファイルシステムに配置する手順を次に示します。

手順 16.4 Linux カーネルイメージの配置例

1. Linux カーネルイメージを準備しておきます。

```
[ATDE ~]$ ls
linux-a840-[version].bin.gz
```

2. Linux カーネルイメージをブートディスクの第 2 パーティションに配置します。

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext3 /dev/sdb2 sd ❷
[ATDE ~]$ sudo mkdir -p sd/boot ❸
[ATDE ~]$ sudo cp linux-a840-[version].bin.gz sd/boot/Image.bin.gz ❹
[ATDE ~]$ sudo umount sd ❺
[ATDE ~]$ rmdir sd ❻
```

❶ SD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 2 パーティションを sd/ディレクトリにマウントします。

❸ Linux カーネルイメージを配置するための boot/ディレクトリを作成します。

❹ Linux カーネルイメージを sd/boot/ディレクトリにコピーします。

❺ sd/ディレクトリにマウントしたブートディスクの第 2 パーティションをアンマウントします。

❻ sd/ディレクトリを削除します。

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

16.4. SD ブートの実行

「16.1. ブートディスクの作成」で作成したブートディスクから起動する方法を説明します。

Armadillo に電源を投入する前に次の準備を行います。

1. SD スロット(CON1)にブートディスクを接続します。
2. ブートディスクのブートローダーイメージを起動させ(SD ブート)、ブートローダーの起動後に保守モードとなるように、Armadillo-840 の JP1 および JP2 をショートに設定します。

準備が完了後、電源を投入すると SD ブートさせることができます。SD ブートに成功した場合は、「図 16.2. SD ブート時の起動メッセージ」のように起動メッセージが表示されます。起動デバイス(Armadillo-840/の後に表示される文字列)が"mmcsd"になっていることを確認してください。

```
Hermit-At v3.2.3 (Armadillo-840/mmcsd) compiled at 15:34:56, Jul 03 2013
hermit>
```

図 16.2 SD ブート時の起動メッセージ

「16.2. ルートファイルシステムの構築」で構築したルートファイルシステムで起動する場合は、「図 16.3. ルートファイルシステムの起動設定」のように setenv コマンドで Linux カーネル起動オプションを設定します。setenv コマンドの詳細については「10.3. ブートローダーの機能」を参照してください。

```
hermit> setenv console=ttySC2,115200 noinitrd rootwait root=/dev/mmcblk0p2 mem=384M
hermit> setenv
1: console=ttySC2,115200
2: noinitrd
3: rootwait
4: root=/dev/mmcblk0p2
5: mem=384M
```

図 16.3 ルートファイルシステムの起動設定

Linux カーネル起動オプションを出荷状態(Linux カーネル起動オプションが設定されていない状態)に戻すには、以下のようにコマンドを実行します。

```
hermit> clearenv
```

「16.3. Linux カーネルイメージの配置」で配置した Linux カーネルイメージで起動する場合は、保守モードで「図 16.4. Linux カーネルの起動設定」のように setbootdevice コマンドで Linux カーネルイ

メージを指定します。setbootdevice コマンドの詳細については「10.3.2. Linux カーネルイメージの指定方法」を参照してください。

```
hermit> setbootdevice mmcblk0p2
hermit> setbootdevice
bootdevice: mmcblk0p2
```

図 16.4 Linux カーネルの起動設定

起動デバイス設定を出荷状態(フラッシュメモリから起動)に戻すには、以下のようにコマンドを実行します。

```
hermit> setbootdevice flash
```

17. JTAG ICE を利用する

本章では ARM のデバッグを行うために、JTAG ICE を接続する方法について説明します。

17.1. 準備

JTAG ICE のケーブルを、JTAG インターフェース(Armadillo-840: CON6)に接続します。信号配列などの JTAG インターフェースについての詳細は、「18.2.6. CON6 JTAG インターフェース」を参照してください。

17.2. 接続確認

「17.1. 準備」に従って設定されている場合に、CPU は以下のように見えます。

項目	値
デバイス ID	0x4BA00477
コマンド長	4

17.3. 各種デバッガへの対応について

お使いのデバッガが Armadillo-840 に対応しているか等の情報につきましては、各メーカにお問い合わせください。

18. ハードウェア仕様

18.1. インターフェースレイアウト

Armadillo-840 のインターフェースレイアウトは次の通りです。

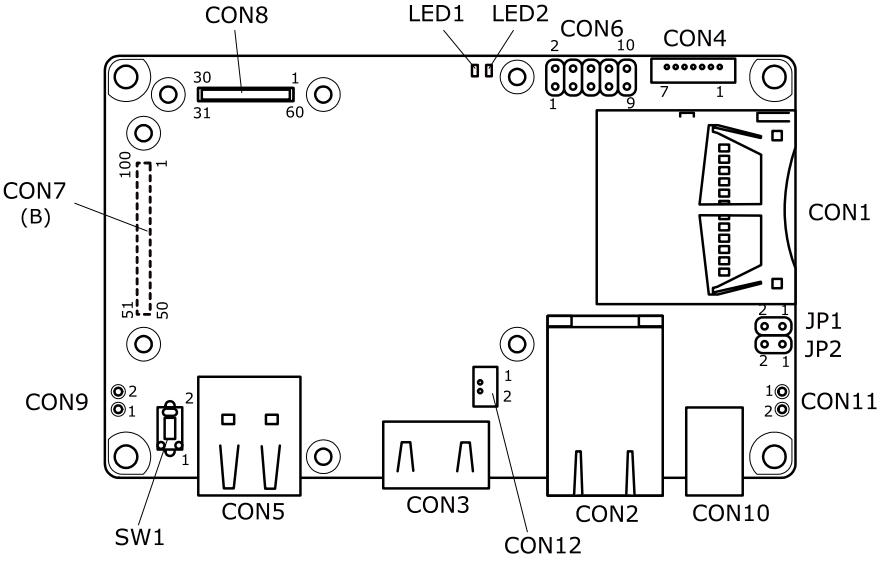


図 18.1 Armadillo-840 インターフェースレイアウト図

表 18.1 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	SD インターフェース	SCDA9A0400	ALPS ELECTRIC
CON2	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse
CON3	HDMI インターフェース	CSS5019-0711F	SMK
CON4	シリアルインターフェース	DF13C-7P-1.25V(51)	HIROSE ELECTRIC
CON5	USB インターフェース	UBA-4RS-D14T-4D(LF)(SN)	J.S.T. Mfg.
CON6	JTAG インターフェース	A1-10PA-2.54DSA(71)	HIROSE ELECTRIC
CON7	拡張インターフェース 1(C コネクタ)	DF40C-100DP-0.4V(51)	HIROSE ELECTRIC
CON8	拡張インターフェース 2(D コネクタ)	DF40C-60DP-0.4V(51)	HIROSE ELECTRIC
CON9	電源出力インターフェース	-	
CON10	電源入力インターフェース 1	HEC3600-016110	HOSIDEN
CON11	電源入力インターフェース 2	-	
CON12	RTC 外部バックアップ用電源入力インターフェース	DF13C-2P-1.25V(21)	HIROSE ELECTRIC
JP1	設定ジャンパ	A1-4PA-2.54DSA(71)	HIROSE ELECTRIC
JP2			
SW1	リセットスイッチ	SKHLACA010	ALPS ELECTRIC

18.2. インターフェース仕様

18.2.1. CON1 SD インターフェース

CON1 は SD インターフェースです。信号線は R-Mobile A1 の SD コントローラ(SDHI0)に接続されています。SD インターフェースに供給する電源は R-Mobile A1 の D21(PORT166)ピンを用いて ON/

OFF 制御が可能です。GPIO 出力モードに設定後、Low 出力で電源切断、High 出力で電源供給されます[1]。

SDHI0 信号レベル: 3.3V CMOS

搭載コネクタ SCDA9A0400/ALPS ELECTRIC

表 18.2 CON1 信号配列

ピン番号	信号名	I/O	機能
1	SDHID3_0	In/Out	データバス(bit3)、R-Mobile A1 の SDHID3_0 ピンに接続
2	SDHICMD_0	In/Out	SD コマンド/レスポンス、R-Mobile A1 の SDHICMD_0 ピンに接続
3	GND	Power	電源(GND)
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	SDHICLK_0	Out	SD クロック、R-Mobile A1 の SDHICLK_0 ピンに接続
6	GND	Power	電源(GND)
7	SDHID0_0	In/Out	データバス(bit0)、R-Mobile A1 の SDHID0_0 ピンに接続
8	SDHID1_0	In/Out	データバス(bit1)、R-Mobile A1 の SDHID1_0 ピンに接続
9	SDHID2_0	In/Out	データバス(bit2)、R-Mobile A1 の SDHID2_0 ピンに接続
10	SDHICD_0	In	カード検出、R-Mobile A1 の SDHICD_0 ピンに接続 (Low:カード挿入、High:カード未挿入)
11	GND	Power	電源(GND)
12	SDHIWP_0	In	ライトプロテクト検出、R-Mobile A1 の SDHIWP_0 ピンに接続 (Low:書き込み可能、High:書き込み不可能)
13	GND	Power	電源(GND)
14	GND	Power	電源(GND)

18.2.2. CON2 LAN インターフェース

CON2 は 10BASE-T/100BASE-TX の LAN インターフェースです。カテゴリ 5 以上のイーサネットケーブルを接続することができます。AUTO-MDIX 機能を搭載しており、ストレートまたはクロスを自動認識して送受信を切り替えます。信号線は Ethernet Phy を経由して R-Mobile A1 の Ethernet コントローラ(GETHER0)に接続されています。

搭載コネクタ 08B0-1X1T-36-F/Bel Fuse

表 18.3 CON2 信号配列

ピン番号	信号名	I/O	機能
1	TX+	In/Out	差動のツイストペア送信出力(+)
2	TX-	In/Out	差動のツイストペア送信出力(−)
3	RX+	In/Out	差動のツイストペア受信入力(+)
4	-	-	CON2 5 ピンと接続後に 75Ω 終端
5	-	-	CON2 4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	差動のツイストペア受信入力(−)
7	-	-	CON2 8 ピンと接続後に 75Ω 終端
8	-	-	CON2 7 ピンと接続後に 75Ω 終端

表 18.4 LAN コネクタ LED

名称(色)	状態	説明
LINK_ACTIVE_LED(緑色)	消灯	リンクが確立されていない。
	点灯	リンクが確立されている。
	点滅	リンクが確立されており、キャリアを検出した状態。

[1]電源回路の構成については、「図 18.4. 電源回路の構成」を参照してください。

名称(色)	状態	説明
SPEED_LED(黄色)	消灯	10BASE-T
	点灯	100BASE-TX

18.2.3. CON3 HDMI インターフェース

CON3 は HDMI インターフェースです。信号線は R-Mobile A1 の HDMI コントローラ(HDMI)に接続されています。Full HD 画面出力、リニア PCM 音声出力、CEC に対応しています。

搭載コネクタ　CSS5019-0711F/SMK

表 18.5 CON3 信号配列

ピン番号	信号名	I/O	機能
1	TX2+	Out	TMDS データ 2(+)、R-Mobile A1 の TODP2 ピンに接続
2	TX2_Shield	-	TMDS データ 2 シールド
3	TX2-	Out	TMDS データ 2(−)、R-Mobile A1 の TODN2 ピンに接続
4	TX1+	Out	TMDS データ 1(+)、R-Mobile A1 の TODP1 ピンに接続
5	TX1_Shield	-	TMDS データ 1 シールド
6	TX1-	Out	TMDS データ 1(−)、R-Mobile A1 の TODN1 ピンに接続
7	TX0+	Out	TMDS データ 0(+)、R-Mobile A1 の TODP0 ピンに接続
8	TX0_Shield	-	TMDS データ 0 シールド
9	TX0-	Out	TMDS データ 0(−)、R-Mobile A1 の TODN0 ピンに接続
10	TXC+	Out	TMDS クロック(+)、R-Mobile A1 の TOCP ピンに接続
11	TXC_Shield	-	TMDS クロックシールド
12	TXC-	Out	TMDS クロック(−)、R-Mobile A1 の TOCN ピンに接続
13	HDMI_CEC	In/Out	CEC 信号、R-Mobile A1 の HDMI_CEC ピンに接続
14	Reserved	-	未接続
15	HDMI_SCL	In/Out	DDC クロック、R-Mobile A1 の HDMI_SCL ピンに接続
16	HDMI_SDA	In/Out	DDC データ、R-Mobile A1 の HDMI_SDA ピンに接続
17	GND	Power	電源(GND)
18	HDMI_5V	Power	電源(HDMI_5V)
19	HDMI_HPD	In	ホットプラグ検出、R-Mobile A1 の HDMI_HPD ピンに接続

18.2.4. CON4 シリアルインターフェース

CON4 は非同期(調歩同期)シリアルインターフェースです。信号線は R-Mobile A1 のシリアルコントローラ(SCIFA2)に接続されています。

SCIFA2　信号レベル: 3.3V CMOS

最大 Baudrate: 1Mbps

フロー制御: RTS、CTS

搭載コネクタ　DF13C-7P-1.25V(51)/HIROSE ELECTRIC

許容電流: 1A 以下(端子 1 本あたり)

対向コネクタ例　DF13-7S-1.25C/HIROSE ELECTRIC

オプション品の「開発用 USB シリアル変換アダプタ[2]」を接続して、PC と通信可能です。開発用 USB シリアル変換アダプタには HERMIT_EN_N

[2]詳細については、「20.3. 開発用 USB シリアル変換アダプタ」を参照してください。

を制御するためのスライドスイッチが実装されており、起動モードを切り替えることが可能です。

表 18.6 CON4 信号配列

ピン番号	信号名	I/O	機能
1	SCIFA_RXD_2	In	受信データ、R-Mobile A1 の PORT200 ピンに接続
2	GND	Power	電源(GND)
3	SCIFA_TXD_2	Out	送信データ、R-Mobile A1 の PORT201 ピンに接続
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	SCIFA_CTS_2	In	送信可能、R-Mobile A1 の PORT95 ピンに接続
6	HERMIT_EN_N	In	起動モード設定、R-Mobile A1 の FCE0_N ピンに接続、JP1 の 2 ピンと共通 (Low: 保守モード、High: OS 自動起動モード)
7	SCIFA_RTS_2	Out	送信要求、R-Mobile A1 の PORT96 ピンに接続

18.2.5. CON5 USB インターフェース

CON5 は USB インターフェースです。2 段の USB コネクタを実装しており、上段、下段の信号線はそれぞれ、R-Mobile A1 の USB コントローラに接続されています。

USB0(上段) データ転送モード: USB2.0 High Speed/Full Speed

USB1(下段) データ転送モード: USB2.0 High Speed/Full Speed

搭載コネクタ UBA-4RS-D14T-4D(LF)(SN)/J.S.T. Mfg.

USB0(上段)に供給する電源は R-Mobile A1 の PPON_0/PORT85 ピン、USB1(下段)に供給する電源は R-Mobile A1 の PPON_1/PORT87 ピンを用いて ON/OFF 制御が可能です。GPIO モードに設定後、Low 出力で電源切断、High 出力で電源供給されます[3]。

USB0(上段)の 6、7 ピンは CON7 拡張インターフェース 1(C コネクタ)の 49、50 ピンと排他使用になっており、R-Mobile A1 の ET_GTX_CLK/PORT176 ピンで制御が可能です。GPIO モードに設定後、Low 出力で USB0(上段)、High 出力で拡張インターフェース 1 の USB が有効になります。

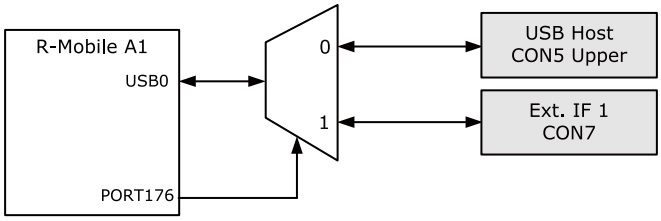


図 18.2 USB の切り替え

表 18.7 CON5 信号配列

ピン番号	信号名	I/O	機能
1	VBUS	Power	USB 電源
2	USB_DM_1	In/Out	USB マイナス側信号、R-Mobile A1 の DM_1 ピンに接続
3	USB_DP_1	In/Out	USB プラス側信号、R-Mobile A1 の DP_1 ピンに接続
4	GND	Power	電源(GND)
5	VBUS	Power	USB 電源

[3]電源回路の構成については、「図 18.4. 電源回路の構成」を参照してください。

ピン番号	信号名	I/O	機能
6	USB_DM_0	In/Out	USB マイナス側信号、R-Mobile A1 の DM_0 ピンに接続 CON7 49 ピンと排他使用
7	USB_DP_0	In/Out	USB プラス側信号、R-Mobile A1 の DP_0 ピンに接続 CON7 50 ピンと排他使用
8	GND	Power	電源(GND)

18.2.6. CON6 JTAG インターフェース

CON6 は ARM JTAG デバッガを接続することができる JTAG インターフェースです。

搭載コネクタ　　A1-10PA-2.54DSA(71)/HIROSE ELECTRIC

オプション品の「8 ピン JTAG 変換ケーブル(Armadillo-400/800 シリーズ対応)[4]」(OP-JC8P25-00)を使用して ARM 標準 20 ピンに変換することが可能です。

表 18.8 CON6 信号配列

ピン番号	信号名	I/O	機能
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	JTAG_TRST_N	In	テストリセット、R-Mobile A1 の TRST_N ピンに接続、VCC_3.3V で 10kΩ プルアップ
3	JTAG_TDI	In	テストデータ入力、R-Mobile A1 の TDI ピンに接続、VCC_3.3V で 10kΩ プルアップ
4	JTAG_TMS	In	テストモード選択、R-Mobile A1 の TMS ピンに接続、VCC_3.3V で 10kΩ プルアップ
5	JTAG_TCK	In	テストクロック、R-Mobile A1 の TCK ピンに接続、VCC_3.3V で 10kΩ プルアップ
6	JTAG_TDO	Out	テストデータ出力、R-Mobile A1 の TDO ピンに接続
7	JTAG_SRST_N	In	リセット
8	GND	Power	電源(GND)
9	JTAG_RTCK	Out	リターンテストクロック、R-Mobile A1 の RTCK ピンに接続
10	JTAG_EDBGREQ	In	デバッグリクエスト、R-Mobile A1 の EDBGREQ ピンに接続、GND に 10kΩ プルダウン

18.2.7. CON7 拡張インターフェース 1(C コネクタ)

CON7 は拡張インターフェースです。用途によって機能を選択できるように複数の機能が割り当てられたピンが多数接続されています。

搭載コネクタ　　DF40C-100DP-0.4V(51)/HIROSE ELECTRIC

許容電流: 0.3A 以下(端子 1 本あたり)

対向コネクタ例　　DF40HC(3.0)-100DS-0.4V(51)/HIROSE ELECTRIC

[4]詳細については、「20.4. 8 ピン JTAG 変換ケーブル」を参照してください。

表 18.9 CON7 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	EXT_IO0	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RXD_1 ピンに接続 CON8 40 ピンと共通
3	EXT_IO1	In/Out	拡張入出力、R-Mobile A1 の SCIFA_TXD_1 ピンに接続 CON8 41 ピンと共通
4	EXT_IO2	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RTS_1_N ピンに接続 CON8 42 ピンと共通
5	EXT_IO3	In/Out	拡張入出力、R-Mobile A1 の SCIFA_CTS_1_N ピンに接続 CON8 44 ピンと共通
6	EXT_IO4	In/Out	拡張入出力、R-Mobile A1 の D29 ピンに接続 CON8 45 ピンと共通
7	EXT_IO5	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RXD_0 ピンに接続 CON8 46 ピンと共通
8	EXT_IO6	In/Out	拡張入出力、R-Mobile A1 の SCIFA_TXD_0 ピンに接続 CON8 47 ピンと共通
9	EXT_IO7	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RTS_0_N ピンに接続 CON8 48 ピンと共通
10	EXT_IO8	In/Out	拡張入出力、R-Mobile A1 の SCIFA_CTS_0_N ピンに接続 CON8 49 ピンと共通
11	GND	Power	電源(GND)
12	EXT_IO9	In/Out	拡張入出力、R-Mobile A1 の LCDDCK_0 ピンに接続
13	EXT_IO10	In/Out	拡張入出力、R-Mobile A1 の LCDVSYN_0 ピンに接続
14	EXT_IO11	In/Out	拡張入出力、R-Mobile A1 の LCDHSYN_0 ピンに接続
15	EXT_IO12	In/Out	拡張入出力、R-Mobile A1 の LCDDISP_0 ピンに接続
16	EXT_IO13	In/Out	拡張入出力、R-Mobile A1 の LCDDON_0 ピンに接続
17	EXT_IO14	In/Out	拡張入出力、R-Mobile A1 の D24 ピンに接続
18	EXT_IO15	In/Out	拡張入出力、R-Mobile A1 の D25 ピンに接続
19	EXT_IO16	In/Out	拡張入出力、R-Mobile A1 の PORT202 ピンに接続
20	EXT_IO17	In/Out	拡張入出力、R-Mobile A1 の FRB ピンに接続
21	EXT_IO18	In/Out	拡張入出力、R-Mobile A1 の LCDVCPWC_0 ピンに接続
22	EXT_IO19	In/Out	拡張入出力、R-Mobile A1 の LCDVEPWC_0 ピンに接続
23	I2C_SCL_0	In/Out	I2C クロック、R-Mobile A1 の I2C_SCL_0 ピンに接続、VCC_3.3V で 3.3kΩ プルアップ
24	I2C_SDA_0	In/Out	I2C データ、R-Mobile A1 の I2C_SDA_0 ピンに接続、VCC_3.3V で 3.3kΩ プルアップ
25	EXT_IO20	In/Out	拡張入出力、R-Mobile A1 の D23 ピンに接続
26	EXT_IO21	In/Out	拡張入出力、R-Mobile A1 の D22 ピンに接続
27	EXT_IO22	In/Out	拡張入出力、R-Mobile A1 の DBGMD20 ピンに接続
28	EXT_IO23	In/Out	拡張入出力、R-Mobile A1 の DBGMD21 ピンに接続
29	EXT_IO24	In/Out	拡張入出力、R-Mobile A1 の DBGMDT0 ピンに接続
30	EXT_IO25	In/Out	拡張入出力、R-Mobile A1 の DBGMDT2 ピンに接続
31	EXT_IO26	In/Out	拡張入出力、R-Mobile A1 の DBGMDT1 ピンに接続
32	GND	Power	電源(GND)
33	EXT_IO27	In/Out	拡張入出力、R-Mobile A1 の PORT66 ピンに接続
34	EXT_IO28	In/Out	拡張入出力、R-Mobile A1 の PORT67 ピンに接続
35	EXT_IO29	In/Out	拡張入出力、R-Mobile A1 の PORT68 ピンに接続
36	EXT_IO30	In/Out	拡張入出力、R-Mobile A1 の PORT69 ピンに接続
37	EXT_IO31	In/Out	拡張入出力、R-Mobile A1 の PORT70 ピンに接続
38	EXT_IO32	In/Out	拡張入出力、R-Mobile A1 の PORT71 ピンに接続
39	EXT_IO33	In/Out	拡張入出力、R-Mobile A1 の PORT72 ピンに接続
40	EXT_IO34	In/Out	拡張入出力、R-Mobile A1 の PORT73 ピンに接続
41	EXT_IO35	In/Out	拡張入出力、R-Mobile A1 の PORT74 ピンに接続

ピン番号	信号名	I/O	機能
42	EXT_IO36	In/Out	拡張入出力、R-Mobile A1 の PORT75 ピンに接続
43	EXT_IO37	In/Out	拡張入出力、R-Mobile A1 の PORT97 ピンに接続
44	EXT_IO38	In/Out	拡張入出力、R-Mobile A1 の PORT98 ピンに接続
45	EXT_IO39	In/Out	拡張入出力、R-Mobile A1 の PORT99 ピンに接続
46	EXT_IO40	In/Out	拡張入出力、R-Mobile A1 の PORT100 ピンに接続
47	EXT_RESET_N	In	外部リセット、リセット IC を介して R-Mobile A1 の RESETP_N ピンに接続、 VCC_3.3V で 10kΩ プルアップ (Low: リセット状態、High: リセット解除)
48	GND	Power	電源(GND)
49	USB_DM_0	In/Out	USB マイナス側信号、R-Mobile A1 の DM_0 ピンに接続 CON5 6 ピンと排他使用
50	USB_DP_0	In/Out	USB プラス側信号、R-Mobile A1 の DP_0 ピンに接続 CON5 7 ピンと排他使用
51	VCC_5V	Power	電源(VCC_5V)
52	VCC_5V	Power	電源(VCC_5V)
53	VCC_5V	Power	電源(VCC_5V)
54	NC	-	未接続
55	YCVBSOUT	Out	コンポジットビデオ出力、R-Mobile A1 の YCVBSOUT ピンに接続
56	GND	Power	電源(GND)
57	DV_CLKI	In	27MHz ビデオクロック入力、R-Mobile A1 の DV_CLKI ピンに接続
58	GND	Power	電源(GND)
59	RESETOUTS_N	Out	リセット出力、R-Mobile A1 の RESETOUTS_N ピンに接続 R-Mobile A1 がリセット中、Low 出力
60	AUDIO_CLK	Out	オーディオクロック、R-Mobile A1 の FSIACK(12.288MHz)ピンに接続
61	EXT_IO42	In/Out	拡張入出力、R-Mobile A1 の FSIAIBT ピンに接続
62	EXT_IO43	In/Out	拡張入出力、R-Mobile A1 の FSIAILR ピンに接続
63	EXT_IO44	In/Out	拡張入出力、R-Mobile A1 の FSIAOSLD ピンに接続
64	EXT_IO45	In/Out	拡張入出力、R-Mobile A1 の DBGMD11 ピンに接続
65	EXT_IO46	In/Out	拡張入出力、R-Mobile A1 の FMSOCK ピンに接続
66	EXT_IO47	In/Out	拡張入出力、R-Mobile A1 の FSIAOMC ピンに接続
67	EXT_IO48	In/Out	拡張入出力、R-Mobile A1 の FSIAOBT ピンに接続
68	EXT_IO49	In/Out	拡張入出力、R-Mobile A1 の FSIAOLR ピンに接続
69	GND	Power	電源(GND)
70	EXT_IO50	In/Out	拡張入出力、R-Mobile A1 の LCDD18_0 ピンに接続
71	EXT_IO51	In/Out	拡張入出力、R-Mobile A1 の LCDD17_0 ピンに接続
72	EXT_IO52	In/Out	拡張入出力、R-Mobile A1 の LCDD16_0 ピンに接続
73	EXT_IO53	In/Out	拡張入出力、R-Mobile A1 の LCDD15_0 ピンに接続
74	EXT_IO54	In/Out	拡張入出力、R-Mobile A1 の LCDD14_0 ピンに接続
75	EXT_IO55	In/Out	拡張入出力、R-Mobile A1 の LCDD13_0 ピンに接続
76	EXT_IO56	In/Out	拡張入出力、R-Mobile A1 の LCDD12_0 ピンに接続
77	EXT_IO57	In/Out	拡張入出力、R-Mobile A1 の LCDD11_0 ピンに接続
78	EXT_IO58	In/Out	拡張入出力、R-Mobile A1 の LCDD10_0 ピンに接続
79	EXT_IO59	In/Out	拡張入出力、R-Mobile A1 の LCDD9_0 ピンに接続
80	EXT_IO60	In/Out	拡張入出力、R-Mobile A1 の LCDD8_0 ピンに接続
81	EXT_IO61	In/Out	拡張入出力、R-Mobile A1 の LCDD7_0 ピンに接続
82	EXT_IO62	In/Out	拡張入出力、R-Mobile A1 の LCDD6_0 ピンに接続
83	EXT_IO63	In/Out	拡張入出力、R-Mobile A1 の LCDD5_0 ピンに接続
84	EXT_IO64	In/Out	拡張入出力、R-Mobile A1 の LCDD4_0 ピンに接続
85	EXT_IO65	In/Out	拡張入出力、R-Mobile A1 の LCDD3_0 ピンに接続
86	EXT_IO66	In/Out	拡張入出力、R-Mobile A1 の LCDD2_0 ピンに接続
87	EXT_IO67	In/Out	拡張入出力、R-Mobile A1 の LCDD1_0 ピンに接続
88	EXT_IO68	In/Out	拡張入出力、R-Mobile A1 の LCDD0_0 ピンに接続
89	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
90	EXT_IO69	In/Out	拡張入出力、R-Mobile A1 の VIO_D15_0 ピンに接続 CON8 57 ピンと共通
91	EXT_IO70	In/Out	拡張入出力、R-Mobile A1 の VIO_D14_0 ピンに接続 CON8 56 ピンと共通
92	EXT_IO71	In/Out	拡張入出力、R-Mobile A1 の VIO_D13_0 ピンに接続 CON8 55 ピンと共通
93	EXT_IO72	In/Out	拡張入出力、R-Mobile A1 の VIO_D12_0 ピンに接続 CON8 54 ピンと共通
94	EXT_IO73	In/Out	拡張入出力、R-Mobile A1 の VIO_D11_0 ピンに接続 CON8 53 ピンと共通
95	EXT_IO74	In/Out	拡張入出力、R-Mobile A1 の VIO_D10_0 ピンに接続 CON8 52 ピンと共通
96	EXT_IO75	In/Out	拡張入出力、R-Mobile A1 の VIO_D9_0 ピンに接続 CON8 51 ピンと共通
97	EXT_IO76	In/Out	拡張入出力、R-Mobile A1 の VIO_D8_0 ピンに接続 CON8 50 ピンと共通
98	VCC_3.3V	Power	電源(VCC_3.3V)
99	VCC_3.3V	Power	電源(VCC_3.3V)
100	VCC_3.3V	Power	電源(VCC_3.3V)

外部からリセット信号を入力する場合、確実にリセットさせるため、200μs 以上の Low 期間を設定してください。

表 18.10 CON7 拡張入出力ピンのマルチプレクス

ピン番号	機能								
	LCD	CAMERA	I2S	UART	SSI(SPI)	SD	MMC	GPIO	その他
1									
2				SCIFA_RXD_1				PORT195	
3				SCIFA_TXD_1				PORT196	
4		VIO_CKO_1		SCIFA_RTS_1_N				PORT23	TPU0TO0
5		VIO_FIELD_1		SCIFA_CTS_1_N				PORT21	TPU0TO1
6		VIO_HD_1						PORT160	
7		VIO_CLK_1		SCIFA_RXD_0				PORT197	
8		VIO_VD_1		SCIFA_TXD_0				PORT198	
9				SCIFA_RTS_0_N				PORT194	
10				SCIFA_CTS_0_N				PORT193	
11									
12	LCDDCK_0							PORT62 (IRQ15)	
13	LCDVSYN_0							PORT63 (IRQ14)	
14	LCDHSYN_0							PORT64 (IRQ13)	
15	LCDDISP_0				MSIOF2_TSCK (SCK)			PORT65	
16	LCDDON_0				MSIOF2_TXD (MOSI)			PORT61	
17					MSIOF2_RXD (MISO)			PORT165	
18	LCDRD_0_N				MSIOF2_TSYNC (SS0)			PORT164	
19								PORT202 (IRQ21)	TPU0TO2
20	LCDLCLK_0							PORT102	
21	LCDVCPWC_0							PORT59	
22	LCDVEPWC_0							PORT60	
23									
24									
25				SCIFB_RTS_N				PORT172 (IRQ4)	

ピン番号	機能								
	LCD	CAMERA	I2S	UART	SSI(SPI)	SD	MMC	GPIO	その他
26				SCIFB_CTS_N				PORT173 (IRQ6)	
27	LCDD19_0			SCIFB_TXD				PORT4	
28	LCDD20_0			SCIFB_RXD				PORT3	
29	LCDD21_0			SCIFB_SCK				PORT2 (IRQ0)	
30	LCDD22_0			SCIFA_RXD_7				PORT0 (IRQ5)	
31	LCDD23_0			SCIFA_TXD_7				PORT1 (IRQ5)	
32									
33						SDHICLK_1	MMCCLK_0	PORT66	TPU0TO2
34					MSIOF1_SS1 (SS1)	SDHICMD_1	MMCCMD_0	PORT67 (IRQ20)	
35					MSIOF1_RSCK	SDHID0_1	MMCD0_0	PORT68 (IRQ16)	
36					MSIOF1_RSYNC	SDHID1_1	MMCD1_0	PORT69 (IRQ17)	
37					MSIOF1_MCK0	SDHID2_1	MMCD2_0	PORT70 (IRQ18)	
38					MSIOF1_MCK1	SDHID3_1	MMCD3_0	PORT71 (IRQ19)	
39					MSIOF1_TSCK (SCK)	SDHICD_1	MMCD4_0	PORT72	
40					MSIOF1_TSYNC (SS0)	SDHIWP_1	MMCD5_0	PORT73	
41					MSIOF1_TXD (MOSI)		MMCD6_0	PORT74	
42					MSIOF1_RXD (MISO)		MMCD7_0	PORT75	
43								PORT97 (IRQ12)	
44								PORT98 (IRQ13)	
45								PORT99 (IRQ14)	

ピン番号	機能								
	LCD	CAMERA	I2S	UART	SSI(SPI)	SD	MMC	GPIO	その他
46								PORT100 (IRQ15)	
47									
48									
49									
50									
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61			FSIAIBT	SCIFA_TXD_4				PORT13 (IRQ0)	
62			FSIAILR	SCIFA_RXD_4				PORT12 (IRQ2)	
63			FSIAOSLD					PORT9	
64			FSIAISLD					PORT5	
65				SCIFA_TXD_5				PORT20 (IRQ1)	
66			FSIAOMC	SCIFA_RXD_5				PORT10 (IRQ3)	
67			FSIAOBT					PORT8	
68			FSIAOLR					PORT7	
69									
70	LCDD18_0							PORT40	
71	LCDD17_0				MSIOF2_SS1 (SS1)			PORT41 (IRQ31)	
72	LCDD16_0				MSIOF2_MCK1			PORT42 (IRQ12)	
73	LCDD15_0				MSIOF2_MCK0			PORT43	KEYIN0

ピン番号	機能								
	LCD	CAMERA	I2S	UART	SSI(SPI)	SD	MMC	GPIO	その他
74	LCDD14_0				MSIOF2_RSYNC			PORT44	KEYIN1
75	LCDD13_0				MSIOF2_RSCK			PORT45	KEYIN2
76	LCDD12_0							PORT46	KEYIN3
77	LCDD11_0							PORT47	KEYIN4
78	LCDD10_0							PORT48	KEYIN5
79	LCDD9_0							PORT49 (IRQ30)	KEYIN6
80	LCDD8_0							PORT50 (IRQ29)	KEYIN7
81	LCDD7_0							PORT51	KEYOUT0
82	LCDD6_0							PORT52	KEYOUT1
83	LCDD5_0							PORT53	KEYOUT2
84	LCDD4_0							PORT54	KEYOUT3
85	LCDD3_0							PORT55	KEYOUT4
86	LCDD2_0							PORT56 (IRQ28)	KEYOUT5
87	LCDD1_0							PORT57 (IRQ27)	KEYOUT6
88	LCDD0_0							PORT58 (IRQ26)	KEYOUT7
89									
90		VIO_D7_1 VIO_D15_0		SCIFA_SCK_6				PORT24	
91		VIO_D6_1 VIO_D14_0		SCIFA_RXD_6				PORT25	
92		VIO_D5_1 VIO_D13_0		SCIFA_TXD_6				PORT26	
93		VIO_D4_1 VIO_D12_0						PORT178	
94		VIO_D3_1 VIO_D11_0						PORT179	
95		VIO_D2_1 VIO_D10_0						PORT180 (IRQ24)	TPU0TO3
96		VIO_D1_1 VIO_D9_0						PORT181	

ピン番号	機能								
	LCD	CAMERA	I2S	UART	SSI(SPI)	SD	MMC	GPIO	その他
97		VIO_D0_1 VIO_D8_0						PORT182	
98									
99									
100									

表 18.11 CON7 拡張入出力ピンの信号状態 [a]

ピン番号	信号名	R-Mobile A1 の信号名	リセット中の状態	リセット後の状態
1				
2	EXT_IO0	SCIFA_RXD_1	PD	ID
3	EXT_IO1	SCIFA_TXD_1	PD	ID
4	EXT_IO2	SCIFA_RTS_1_N	PU	IU
5	EXT_IO3	SCIFA_CTS_1_N	PD	ID
6	EXT_IO4	D29	PD	ID
7	EXT_IO5	SCIFA_RXD_0	PD	ID
8	EXT_IO6	SCIFA_TXD_0	PD	ID
9	EXT_IO7	SCIFA_RTS_0_N	PU	IU
10	EXT_IO8	SCIFA_CTS_0_N	PU	IU
11				
12	EXT_IO9	LCDDCK_0	PD	ID
13	EXT_IO10	LCDVSYN_0	PD	ID
14	EXT_IO11	LCDHSYN_0	PD	ID
15	EXT_IO12	LCDDISP_0	PD	ID
16	EXT_IO13	LCDDON_0	PD	ID
17	EXT_IO14	D24	PD	ID
18	EXT_IO15	D25	PD	ID
19	EXT_IO16	PORT202	PU	IU
20	EXT_IO17	FRB	PU	IU
21	EXT_IO18	LCDVCPWC_0	PD	ID
22	EXT_IO19	LCDVEPWC_0	PD	ID
23				
24				
25	EXT_IO20	D23	PD	ID
26	EXT_IO21	D22	PD	ID
27	EXT_IO22	DBGMD20	ID	ID
28	EXT_IO23	DBGMD21	ID	ID
29	EXT_IO24	DBGMDT0	ID	ID
30	EXT_IO25	DBGMDT2	ID	ID
31	EXT_IO26	DBGMDT1	ID	ID
32				
33	EXT_IO27	PORT66	PD	ID
34	EXT_IO28	PORT67	PU	IU
35	EXT_IO29	PORT68	PU	IU
36	EXT_IO30	PORT69	PU	IU
37	EXT_IO31	PORT70	PU	IU
38	EXT_IO32	PORT71	PU	IU
39	EXT_IO33	PORT72	PU	PU
40	EXT_IO34	PORT73	PU	PU
41	EXT_IO35	PORT74	PU	PU
42	EXT_IO36	PORT75	PU	PU
43	EXT_IO37	PORT97	PD	ID
44	EXT_IO38	PORT98	PD	ID
45	EXT_IO39	PORT99	PD	ID
46	EXT_IO40	PORT100	PD	ID
47				
48				
49				
50				
51				
52				

ピン番号	信号名	R-Mobile A1 の信号名	リセット中の状態	リセット後の状態
53				
54				
55				
56				
57				
58				
59				
60				
61	EXT_IO42	FSIAIBT	PD	ID
62	EXT_IO43	FSIAILR	PD	ID
63	EXT_IO44	FSIAOSLD	L	O
64	EXT_IO45	DBGMD11	ID	ID
65	EXT_IO46	FMSOCK	PD	ID
66	EXT_IO47	FSIAOMC	PD	ID
67	EXT_IO48	FSIAOBT	L	O
68	EXT_IO49	FSIAOLR	L	O
69				
70	EXT_IO50	LCDD18_0	PD	ID
71	EXT_IO51	LCDD17_0	PD	ID
72	EXT_IO52	LCDD16_0	PD	ID
73	EXT_IO53	LCDD15_0	PD	ID
74	EXT_IO54	LCDD14_0	PD	ID
75	EXT_IO55	LCDD13_0	PD	ID
76	EXT_IO56	LCDD12_0	PD	ID
77	EXT_IO57	LCDD11_0	PD	ID
78	EXT_IO58	LCDD10_0	PD	ID
79	EXT_IO59	LCDD9_0	PD	ID
80	EXT_IO60	LCDD8_0	PD	ID
81	EXT_IO61	LCDD7_0	PD	ID
82	EXT_IO62	LCDD6_0	PD	ID
83	EXT_IO63	LCDD5_0	PD	ID
84	EXT_IO64	LCDD4_0	PD	ID
85	EXT_IO65	LCDD3_0	PD	ID
86	EXT_IO66	LCDD2_0	PD	ID
87	EXT_IO67	LCDD1_0	PD	ID
88	EXT_IO68	LCDD0_0	PD	ID
89				
90	EXT_IO69	VIO_D15_0	PU	IU
91	EXT_IO70	VIO_D14_0	ID	ID
92	EXT_IO71	VIO_D13_0	ID	ID
93	EXT_IO72	VIO_D12_0	PU	IU
94	EXT_IO73	VIO_D11_0	PD	ID
95	EXT_IO74	VIO_D10_0	PD	ID
96	EXT_IO75	VIO_D9_0	PU	IU
97	EXT_IO76	VIO_D8_0	PU	IU
98				
99				
100				

[a]記号一覧

IU: 入力バッファ有効、プルアップ有効

ID: 入力バッファ有効、プルダウン有効

L : 出力バッファ有効、Low レベル出力

O : 出力バッファ有効

PU: 入出力バッファ無効、プルアップ有効

PD: 入出力バッファ無効、プルダウン有効

18.2.7.1. USB

R-Mobile A1 の USB コントローラに接続されています。

USB0	データ転送モード: USB2.0 High Speed/Full Speed

18.2.7.2. LCD

R-Mobile A1 の LCD コントローラ(LCDC0)に接続されています。

LCDC0	信号レベル: 3.3V CMOS
	最大ピクセル数: 1440 x 900 ピクセル/24bpp

18.2.7.3. CAMERA

R-Mobile A1 の CEU コントローラ(CEU1)に接続されています。

CEU1	信号レベル: 3.3V CMOS
	画像フォーマット: YUV422(8bit)
	最大ピクセル数: 8188 x 8188 ピクセル

18.2.7.4. I2C

R-Mobile A1 の I2C コントローラ(I2C0)に接続されています。

I2C0	信号レベル: 3.3V CMOS

18.2.7.5. I2S

R-Mobile A1 の I2S コントローラ(FSIA)に接続されています。

FSIA	信号レベル: 3.3V CMOS

18.2.7.6. UART

R-Mobile A1 のシリアル(UART)コントローラ(SCIFA0、SCIFA1、SCIFA4、SCIFA5、SCIFA6、SCIFA7、SCIFB)に接続されています。最大 7 ポート UART として使用できます。

SCIFA0	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: あり
SCIFA1	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: あり

SCIFA4 信号レベル: 3.3V CMOS
最大 Baudrate: 1Mbps
フロー制御: なし

SCIFA5 信号レベル: 3.3V CMOS
最大 Baudrate: 1Mbps
フロー制御: なし

SCIFA6 信号レベル: 3.3V CMOS
最大 Baudrate: 1Mbps
フロー制御: なし

SCIFA7 信号レベル: 3.3V CMOS
最大 Baudrate: 1Mbps
フロー制御: なし

SCIFB 信号レベル: 3.3V CMOS
最大 Baudrate: 1Mbps
フロー制御: あり

18.2.7.7. SSI(SPI)

R-Mobile A1 の SSI(SPI)コントローラ(MSIOF1、MSIOF2)に接続されています。

MSIOF1 信号レベル: 3.3V CMOS

MSIOF2 信号レベル: 3.3V CMOS

18.2.7.8. SD

R-Mobile A1 の SD コントローラ(SDHI1)に接続されています。

SDHI1 信号レベル: 3.3V CMOS

18.2.7.9. MMC

R-Mobile A1 の MMC コントローラ(MMC0)に接続されています。

MMC0 信号レベル: 3.3V CMOS

18.2.7.10. GPIO

R-Mobile A1 の GPIO コントローラに接続されています。最大 76 ポート GPIO として使用可能です。

GPIO 信号レベル: 3.3V CMOS

18.2.7.11. PWM

R-Mobile A1 の PWM コントローラ(TPU0)に接続されています。最大 4 ポート PWM として使用可能です。

TPU0 信号レベル: 3.3V CMOS

18.2.7.12. キースキャン

R-Mobile A1 のキースキャンコントローラ(KEYSC)に接続されています。

KEYSC 信号レベル: 3.3V CMOS

最大キー数: 8 x 8 キー

18.2.7.13. コンポジット

R-Mobile A1 の NTSC/PAL、ビデオエンコーダ(SDENC)に接続されています。

18.2.8. CON8 拡張インターフェース 2(D コネクタ)

CON8 は拡張インターフェースです。用途によって機能を選択できるように複数の機能が割り当てられたピンが多数接続されています。

搭載コネクタ DF40C-60DP-0.4V(51)/HIROSE ELECTRIC

許容電流: 0.3A 以下(端子 1 本あたり)

対向コネクタ例 DF40HC(4.0)-60DS-0.4V(51)/HIROSE ELECTRIC

表 18.12 CON8 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	GND	Power	電源(GND)
4	EXT_IO77	In/Out	拡張入出力、R-Mobile A1 の VIO_D0_0 ピンに接続
5	EXT_IO78	In/Out	拡張入出力、R-Mobile A1 の VIO_D1_0 ピンに接続
6	EXT_IO79	In/Out	拡張入出力、R-Mobile A1 の VIO_D2_0 ピンに接続
7	EXT_IO80	In/Out	拡張入出力、R-Mobile A1 の VIO_D3_0 ピンに接続
8	EXT_IO81	In/Out	拡張入出力、R-Mobile A1 の VIO_D4_0 ピンに接続
9	EXT_IO82	In/Out	拡張入出力、R-Mobile A1 の VIO_D5_0 ピンに接続
10	EXT_IO83	In/Out	拡張入出力、R-Mobile A1 の VIO_D6_0 ピンに接続
11	EXT_IO84	In/Out	拡張入出力、R-Mobile A1 の VIO_D7_0 ピンに接続
12	GND	Power	電源(GND)
13	EXT_IO85	In/Out	拡張入出力、R-Mobile A1 の VIO_CLK_0 ピンに接続
14	GND	Power	電源(GND)
15	EXT_IO86	In/Out	拡張入出力、R-Mobile A1 の VIO_FIELD_0 ピンに接続
16	EXT_IO87	In/Out	拡張入出力、R-Mobile A1 の VIO_HD_0 ピンに接続
17	EXT_IO88	In/Out	拡張入出力、R-Mobile A1 の VIO_VD_0 ピンに接続
18	GND	Power	電源(GND)
19	EXT_IO89	In/Out	拡張入出力、R-Mobile A1 の VIO_CKO_0 ピンに接続
20	GND	Power	電源(GND)
21	EXT_IO90	In/Out	拡張入出力、R-Mobile A1 の D31 ピンに接続
22	EXT_IO91	In/Out	拡張入出力、R-Mobile A1 の D30 ピンに接続

ピン番号	信号名	I/O	機能
23	GND	Power	電源(GND)
24	GND	Power	電源(GND)
25	GND	Power	電源(GND)
26	GND	Power	電源(GND)
27	EXT_IO92	In/Out	拡張入出力、R-Mobile A1 の PORT199 ピンに接続
28	EXT_IO93	In/Out	拡張入出力、R-Mobile A1 の PORT94 ピンに接続
29	EXT_IO94	In/Out	拡張入出力、R-Mobile A1 の PORT93 ピンに接続
30	EXT_IO95	In/Out	拡張入出力、R-Mobile A1 の SCIFA_SCK_2 ピンに接続
31	VCC_5V	Power	電源(VCC_5V)
32	VCC_5V	Power	電源(VCC_5V)
33	VCC_5V	Power	電源(VCC_5V)
34	VCC_5V	Power	電源(VCC_5V)
35	NC	-	未接続
36	GND	Power	電源(GND)
37	I2C_SCL_1	In/Out	I2C クロック、R-Mobile A1 の I2C_SCL_1 ピンに接続、VCC_3.3V で 3.3kΩ プルアップ
38	I2C_SDA_1	In/Out	I2C データ、R-Mobile A1 の I2C_SDA_1 ピンに接続、VCC_3.3V で 3.3kΩ プルアップ
39	GND	Power	電源(GND)
40	EXT_IO0	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RXD_1 ピンに接続 CON7 2 ピンと共通
41	EXT_IO1	In/Out	拡張入出力、R-Mobile A1 の SCIFA_TXD_1 ピンに接続 CON7 3 ピンと共通
42	EXT_IO2	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RTS_1_N ピンに接続 CON7 4 ピンと共通
43	GND	Power	電源(GND)
44	EXT_IO3	In/Out	拡張入出力、R-Mobile A1 の SCIFA_CTS_1_N ピンに接続 CON7 5 ピンと共通
45	EXT_IO4	In/Out	拡張入出力、R-Mobile A1 の D29 ピンに接続 CON7 6 ピンと共通
46	EXT_IO5	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RXD_0 ピンに接続 CON7 7 ピンと共通
47	EXT_IO6	In/Out	拡張入出力、R-Mobile A1 の SCIFA_TXD_0 ピンに接続 CON7 8 ピンと共通
48	EXT_IO7	In/Out	拡張入出力、R-Mobile A1 の SCIFA_RTS_0_N ピンに接続 CON7 9 ピンと共通
49	EXT_IO8	In/Out	拡張入出力、R-Mobile A1 の SCIFA_CTS_0_N ピンに接続 CON7 10 ピンと共通
50	EXT_IO76	In/Out	拡張入出力、R-Mobile A1 の VIO_D8_0 ピンに接続 CON7 97 ピンと共通
51	EXT_IO75	In/Out	拡張入出力、R-Mobile A1 の VIO_D9_0 ピンに接続 CON7 96 ピンと共通
52	EXT_IO74	In/Out	拡張入出力、R-Mobile A1 の VIO_D10_0 ピンに接続 CON7 95 ピンと共通
53	EXT_IO73	In/Out	拡張入出力、R-Mobile A1 の VIO_D11_0 ピンに接続 CON7 94 ピンと共通
54	EXT_IO72	In/Out	拡張入出力、R-Mobile A1 の VIO_D12_0 ピンに接続 CON7 93 ピンと共通
55	EXT_IO71	In/Out	拡張入出力、R-Mobile A1 の VIO_D13_0 ピンに接続 CON7 92 ピンと共通
56	EXT_IO70	In/Out	拡張入出力、R-Mobile A1 の VIO_D14_0 ピンに接続 CON7 91 ピンと共通
57	EXT_IO69	In/Out	拡張入出力、R-Mobile A1 の VIO_D15_0 ピンに接続 CON7 90 ピンと共通
58	VCC_3.3V	Power	電源(VCC_3.3V)

ピン番号	信号名	I/O	機能
59	VCC_3.3V	Power	電源(VCC_3.3V)
60	VCC_3.3V	Power	電源(VCC_3.3V)

表 18.13 CON8 拡張入出力ピンのマルチプレクス

ピン番号	機能			
	CAMERA	UART	GPIO	その他
1				
2				
3				
4	VIO_D0_0		PORT34	
5	VIO_D1_0		PORT33	
6	VIO_D2_0		PORT32	
7	VIO_D3_0		PORT31	
8	VIO_D4_0		PORT30	
9	VIO_D5_0		PORT29	
10	VIO_D6_0		PORT28	
11	VIO_D7_0		PORT27	
12				
13	VIO_CLK_0		PORT35	
14				
15	VIO_FIELD_0		PORT38 (IRQ25)	
16	VIO_HD_0		PORT37	
17	VIO_VD_0		PORT39	
18				
19	VIO_CKO_0		PORT36	
20				
21		SCIFA_SCK_3	PORT158	
22		SCIFA_RXD_3	PORT159	
23				
24				
25				
26				
27			PORT199	
28		SCIFA_RXD_4	PORT94	
29		SCIFA_TXD_4	PORT93	
30			PORT22	
31				
32				
33				
34				
35				
36				
37				
38				
39				
40		SCIFA_RXD_1	PORT195	
41		SCIFA_TXD_1	PORT196	
42	VIO_CKO_1	SCIFA_RTS_1_N	PORT23	TPU0TO0
43				
44	VIO_FIELD_1	SCIFA_CTS_1_N	PORT21	TPU0TO1
45	VIO_HD_1	SCIFA_TXD_3	PORT160	

ピン番号	機能			
	CAMERA	UART	GPIO	その他
46	VIO_CLK_1	SCIFA_RXD_0	PORT197	
47	VIO_VD_1	SCIFA_TXD_0	PORT198	
48		SCIFA_RTS_0_N	PORT194	
49		SCIFA_CTS_0_N	PORT193	
50	VIO_D0_1 VIO_D8_0		PORT182	
51	VIO_D1_1 VIO_D9_0		PORT181	
52	VIO_D2_1 VIO_D10_0		PORT180 (IRQ24)	TPU0TO3
53	VIO_D3_1 VIO_D11_0		PORT179	
54	VIO_D4_1 VIO_D12_0		PORT178	
55	VIO_D5_1 VIO_D13_0	SCIFA_TXD_6	PORT26	
56	VIO_D6_1 VIO_D14_0	SCIFA_RXD_6	PORT25	
57	VIO_D7_1 VIO_D15_0	SCIFA_SCK_6	PORT24	
58				
59				
60				

表 18.14 CON8 拡張入出力ピンの信号状態 [a]

ピン番号	信号名	R-Mobile A1 の信号名	リセット中の状態	リセット後の状態
1				
2				
3				
4	EXT_IO77	VIO_D0_0	PD	ID
5	EXT_IO78	VIO_D1_0	PD	ID
6	EXT_IO79	VIO_D2_0	PD	ID
7	EXT_IO80	VIO_D3_0	PD	ID
8	EXT_IO81	VIO_D4_0	PU	IU
9	EXT_IO82	VIO_D5_0	PU	IU
10	EXT_IO83	VIO_D6_0	PU	IU
11	EXT_IO84	VIO_D7_0	PU	IU
12				
13	EXT_IO85	VIO_CLK_0	PU	IU
14				
15	EXT_IO86	VIO_FIELD_0	PU	IU
16	EXT_IO87	VIO_HD_0	PD	ID
17	EXT_IO88	VIO_VD_0	PD	ID
18				
19	EXT_IO89	VIO_CKO_0	PU	IU
20				
21	EXT_IO90	D31	PD	ID
22	EXT_IO91	D30	PD	ID
23				
24				
25				
26				
27	EXT_IO92	PORT199	PU	IU

ピン番号	信号名	R-Mobile A1 の信号名	リセット中の状態	リセット後の状態
28	EXT_IO93	PORT94	PD	PD
29	EXT_IO94	PORT93	PD	PD
30	EXT_IO95	SCIFA_SCK_2	PU	IU
31				
32				
33				
34				
35				
36				
37				
38				
39				
40	EXT_IO0	SCIFA_RXD_1	PD	ID
41	EXT_IO1	SCIFA_TXD_1	PD	ID
42	EXT_IO2	SCIFA_RTS_1_N	PU	IU
43				
44	EXT_IO3	SCIFA_CTS_1_N	PD	ID
45	EXT_IO4	D29	PD	ID
46	EXT_IO5	SCIFA_RXD_0	PD	ID
47	EXT_IO6	SCIFA_TXD_0	PD	ID
48	EXT_IO7	SCIFA_RTS_0_N	PU	IU
49	EXT_IO8	SCIFA_CTS_0_N	PU	IU
50	EXT_IO76	VIO_D8_0	PU	IU
51	EXT_IO75	VIO_D9_0	PU	IU
52	EXT_IO74	VIO_D10_0	PD	ID
53	EXT_IO73	VIO_D11_0	PD	ID
54	EXT_IO72	VIO_D12_0	PU	IU
55	EXT_IO71	VIO_D13_0	ID	ID
56	EXT_IO70	VIO_D14_0	ID	ID
57	EXT_IO69	VIO_D15_0	PU	IU
58				
59				
60				

[a]記号一覧

IU: 入力バッファ有効、プルアップ有効

ID: 入力バッファ有効、プルダウン有効

PU: 入出力バッファ無効、プルアップ有効

PD: 入出力バッファ無効、プルダウン有効

18.2.8.1. CAMERA

R-Mobile A1 の CEU コントローラ(CEU0、CEU1)に接続されています。最大 2 ポート使用可能です。

CEU0 信号レベル: 3.3V CMOS

画像フォーマット: YUV422(8bit/16bit)

最大ピクセル数: 8188 x 8188 ピクセル

CEU1 信号レベル: 3.3V CMOS

画像フォーマット: YUV422(8bit)

最大ピクセル数: 8188 x 8188 ピクセル

18.2.8.2. I2C

R-Mobile A1 の I2C コントローラ(I2C1)に接続されています。

I2C1	信号レベル: 3.3V CMOS

18.2.8.3. UART

R-Mobile A1 のシリアル(UART)コントローラ(SCIFA0、SCIFA1、SCIFA3、SCIFA4、SCIFA6)に接続されています。最大 5 ポート UART として使用できます。

SCIFA0	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: あり
SCIFA1	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: あり
SCIFA3	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: なし
SCIFA4	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: なし
SCIFA6	信号レベル: 3.3V CMOS
	最大 Baudrate: 1Mbps
	フロー制御: なし

18.2.8.4. GPIO

R-Mobile A1 の GPIO コントローラに接続されています。最大 36 ポート GPIO として使用可能です。

GPIO	信号レベル: 3.3V CMOS

18.2.8.5. PWM

R-Mobile A1 の PWM コントローラ(TPU0)に接続されています。最大 3 ポート PWM として使用可能です。

TPU0	信号レベル: 3.3V CMOS

18.2.9. CON9 電源出力インターフェース

CON9 は+5V 電源出力インターフェースです。コネクタは実装されていません。

搭載コネクタ例　　B2B-EH(LF)(SN)/J.S.T. Mfg.

表 18.15 CON9 信号配列

ピン番号	信号名	I/O	機能
1	VOUT	Power	+5V 電源出力(VOUT)
2	GND	Power	電源(GND)

18.2.10. CON10 電源入力インターフェース 1

CON10 は+5V 電源入力インターフェースです。DC ジャックが実装されています。AC アダプターのジャック形状は EIAJ RC-5320A 準拠(電圧区分 2)です。「図 18.3. AC アダプターの極性マーク」と同じ極性マークのある AC アダプターが使用できます。

図 18.3 AC アダプターの極性マーク

搭載コネクタ　　HEC3600-016110/HOSIDEN

CON10 を使用する場合、同時に CON11 から電源供給しないでください。故障の原因となる可能性があります。

18.2.11. CON11 電源入力インターフェース 2

CON11 は+5V 電源入力インターフェースです。

搭載コネクタ例　　B2B-EH(LF)(SN)/J.S.T. Mfg.

CON11 を使用する場合、同時に CON10 から電源供給しないでください。故障の原因となる可能性があります。

表 18.16 CON11 信号配列

ピン番号	信号名	I/O	機能
1	VIN	Power	+5V 電源入力(VIN)
2	GND	Power	電源(GND)

18.2.12. CON12 RTC 外部バックアップ用電源入力インターフェース

CON12 はリアルタイムクロックの外部バックアップ用電源入力インターフェースです。電源が切断されても長時間時刻を保持させたい場合は、別途外付けバッテリー(対応バッテリー例: CR2032 WK11)[5]を接続することができます。

搭載コネクタ　　DF13C-2P-1.25V(21)/HIROSE ELECTRIC

許容電流: 1A 以下(端子 1 本あたり)

対向コネクタ例　　DF13-2S-1.25C/HIROSE ELECTRIC

表 18.17 CON12 信号配列

ピン番号	信号名	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)

CON12 の入力電圧範囲は 1.5V～3.8V です。3.8V 以上加えると内部デバイスが正常に動作しなくなる可能性があります。

リアルタイムクロックの平均月差は周囲温度 25℃で±90 秒程度(参考値)です。時間精度は、周囲温度に大きく影響を受けますので、ご使用の際は十分に特性の確認をお願いします。

リアルタイムクロックのバックアップ時間は、周囲温度、電圧印加時間等に大きく影響を受けますので、ご使用の際は十分に特性の確認をお願いします。

18.2.13. JP1、JP2 設定ジャンパ

JP1、JP2 は設定ジャンパです。JP1 は起動モードの設定、JP2 は起動デバイスの設定に使用します。

搭載コネクタ　　A1-4PA-2.54DSA(71)/HIROSE ELECTRIC

表 18.18 JP1 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	HERMIT_EN_N	In	起動モード設定、R-Mobile A1 の FCE0_N ピンに接続、 CON4 の 6 ピンと共通、VCC_3.3V で 10kΩ プルアップ (Low: 保守モード、High: OS 自動起動モード)

[5]詳しくは、各 Armadillo 販売代理店にお問い合わせください。

表 18.19 JP2 信号配列

ピン番号	信号名	I/O	機能
1	SDBOOT_EN	In	起動デバイス設定、R-Mobile A1 の MD3 ピンに接続、GND に 10kΩ プルダウン (Low: オンボードフラッシュメモリから起動、High: SD から起動)
2	VCC_3.3V	Power	電源(VCC_3.3V)

表 18.20 ジャンパの機能

ジャンパ	機能	動作
JP1	起動モード設定	オープン: OS を自動起動します。 ショート: ブートローダーを保守モードにします。
JP2	起動デバイス設定	オープン: オンボードフラッシュメモリのブートローダーを起動します。 ショート: SD カードのブートローダーを起動します。

18.2.14. LED1、LED2 ユーザー LED

LED1、LED2 はユーザー側で自由に利用できる面実装の黄色 LED です。LED に接続された R-Mobile A1 の信号が GPIO の出力モードに設定されている場合に制御できます。

表 18.21 ユーザー LED の機能

LED	名称(色)	機能
LED1	ユーザー LED(黄色)	R-Mobile A1 の WE3_N ピンに接続(Low:消灯、High:点灯)
LED2		R-Mobile A1 の WE2_N ピンに接続(Low:消灯、High:点灯)

18.2.15. SW1 リセットスイッチ

SW1 はリセットスイッチです。押下でリセットされます。

搭載スイッチ　　SKHLACA010/ALPS ELECTRIC

表 18.22 SW1 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	SW1	In	リセット、CON7 の 47 ピンと接続 (押されていない状態:オープン、押された状態:GND とショート)

18.3. 電気的仕様

18.3.1. 絶対最大定格

表 18.23 絶対最大定格

項目	記号	Min	Max	単位	備考
電源電圧	VIN	-0.3	5.3	V	
入力電圧	VIO	-0.5	VCC_3.3V+0.5	V	
動作温度範囲	Topr	-20	70	℃	ただし結露なきこと

絶対最大定格はあらゆる使用条件、又は試験条件であっても瞬時たりとも越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。

18.3.2. 推奨動作条件

表 18.24 推奨動作条件

項目	記号	Min	Typ	Max	単位	備考
電源電圧	VIN	4.75	5	5.25	V	
使用周囲温度	Ta	-20	25	70	℃	ただし結露なきこと

18.3.3. 入出力インターフェースの電気的仕様

表 18.25 入出力インターフェースの電気的仕様

項目	記号	Min	Typ	Max	単位	備考
入出力インターフェース電源電圧	VCC_3.3V	3.135	3.3	3.465	V	
入力電圧	VIH	VCC_3.3Vx0.8	-	VCC_3.3V +0.3	V	
	VIL	-0.3	-	VCC_3.3V x 0.2	V	
出力電圧	VOH	VCC_3.3V-0.5	-	-	V	IOH=-2mA の時
	VOL	-	-	0.5	V	IOL=2mA の時
出力電流(per pin)	IOL	-	-	2.0	mA	
	IOH	-	-	-2.0	mA	
出力電流(total)	ΣIOL	-	-	120	mA	
	ΣIOH	-	-	-40	mA	
出力電流(I2C)	I2C_IOL	-	-	5	mA	
プルアップ/ダウン抵抗値	PULL	25	50	100	kΩ	

18.4. 電源回路の構成

電源回路の構成は次の通りです。

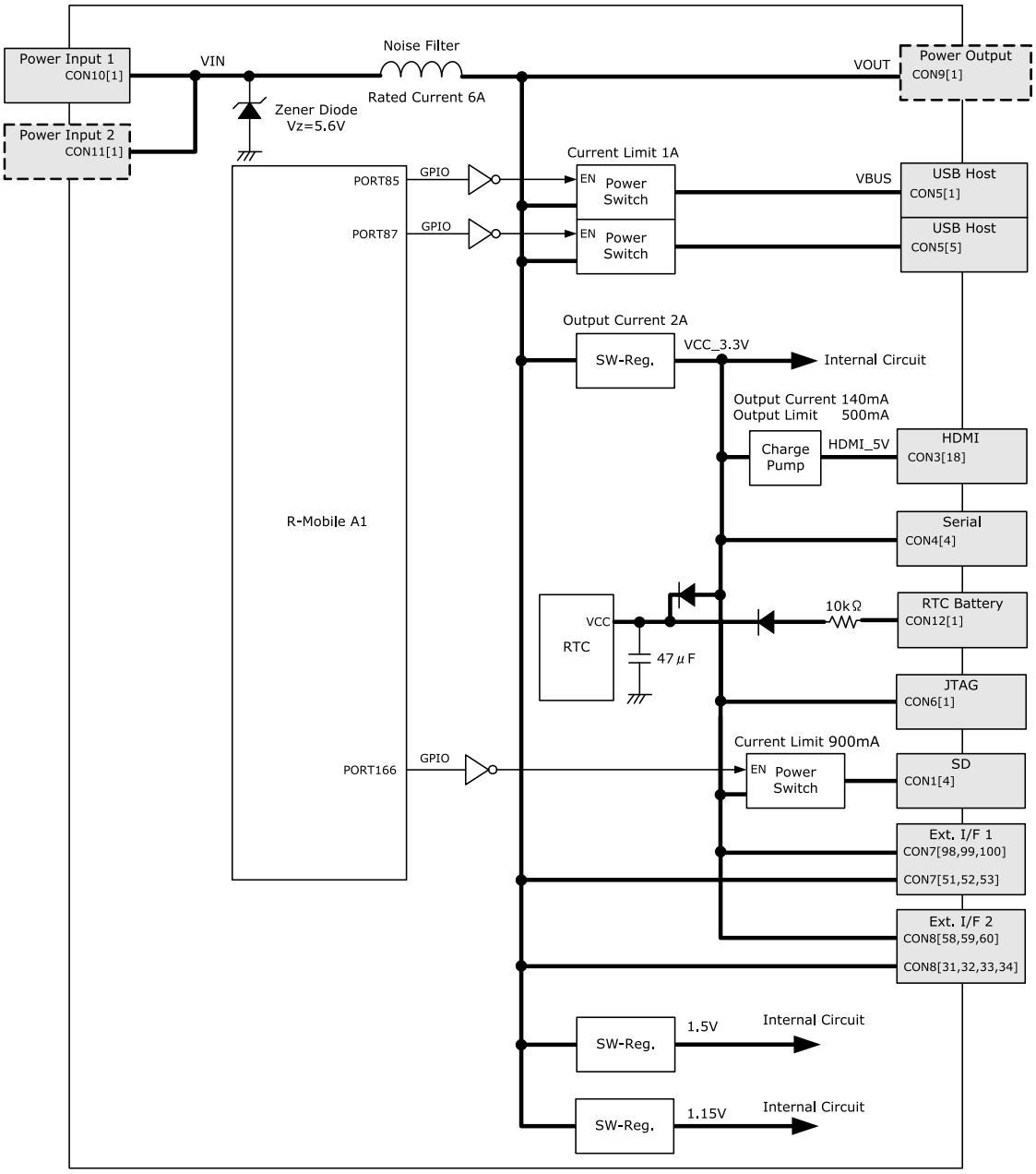


図 18.4 電源回路の構成

CON1 SD インターフェースと CON7 拡張インターフェース 1(C コネクタ)と CON8 拡張インターフェース 2(D コネクタ)の VCC_3.3V 系統から供給可能な電流は合計で最大 1.4A となります。電流容量の制限を超えないように、外部機器の接続や供給電源の設定を行ってください。

CON10 と CON11 から同時に電源を供給しないでください。故障の原因となります。

18.5. リセット回路の構成

リセット回路の構成は次の通りです。

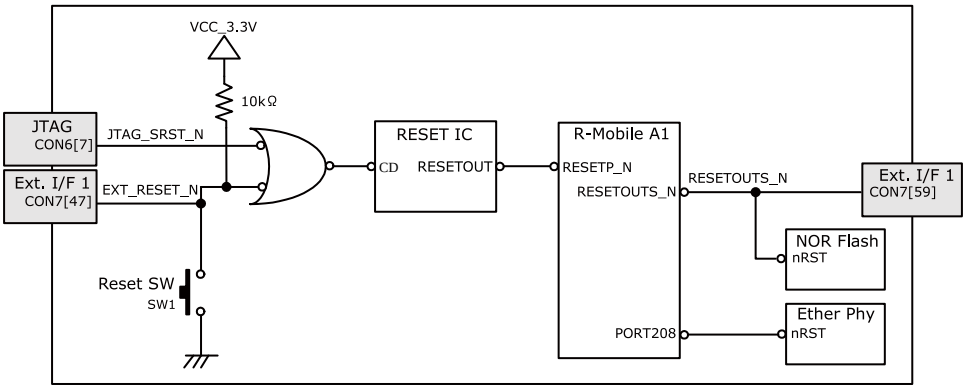


図 18.5 リセット回路の構成

拡張インターフェース 1(CON7)からリセットする場合、オープンドレインで接続してください。プッシュプルで接続した場合、リセットスイッチからの信号とショートし、故障の原因となる可能性があります。

外部からリセットする場合、確実にリセットさせるため、200μs 以上の Low 期間を設定してください。

19. 基板形状図

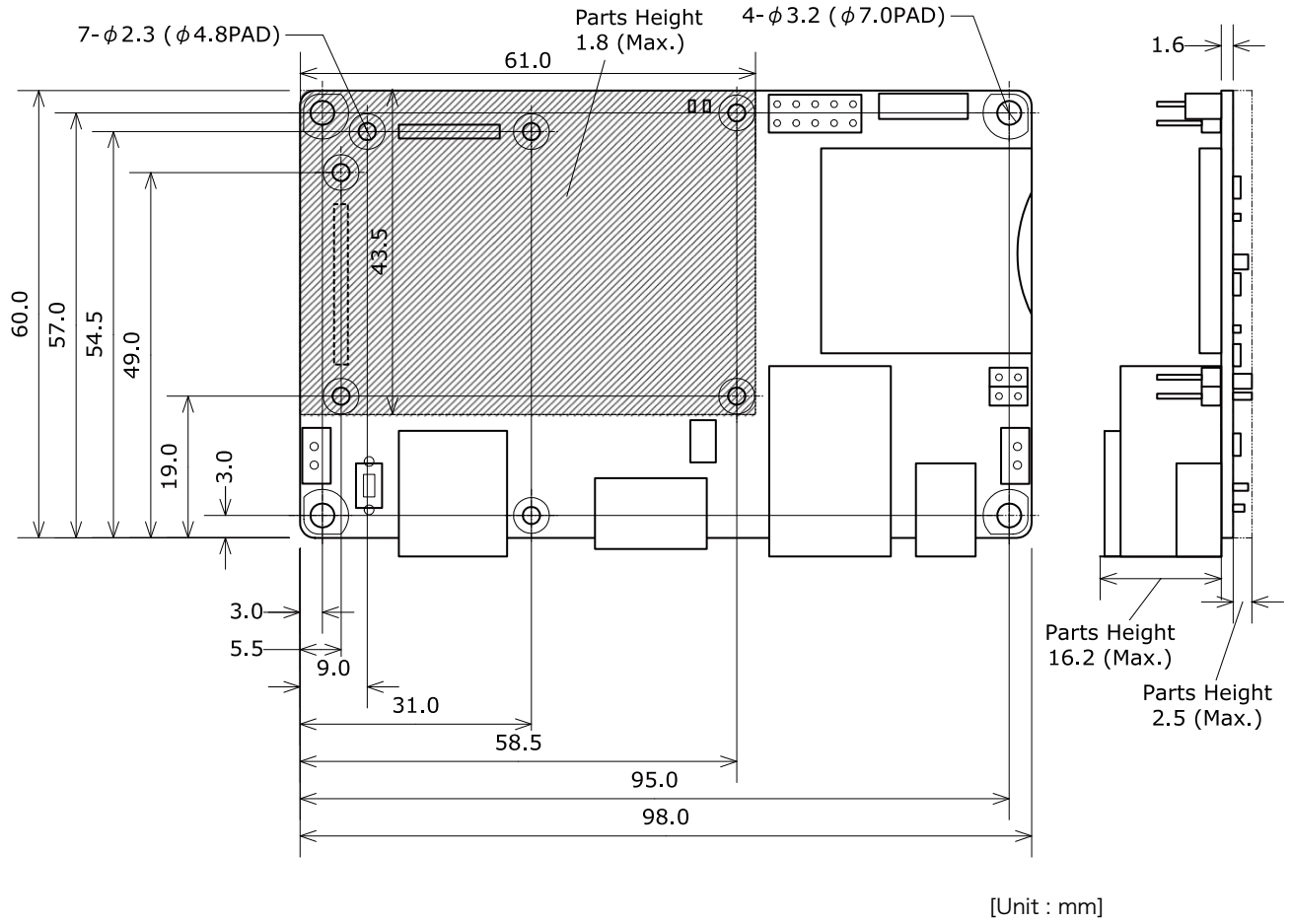
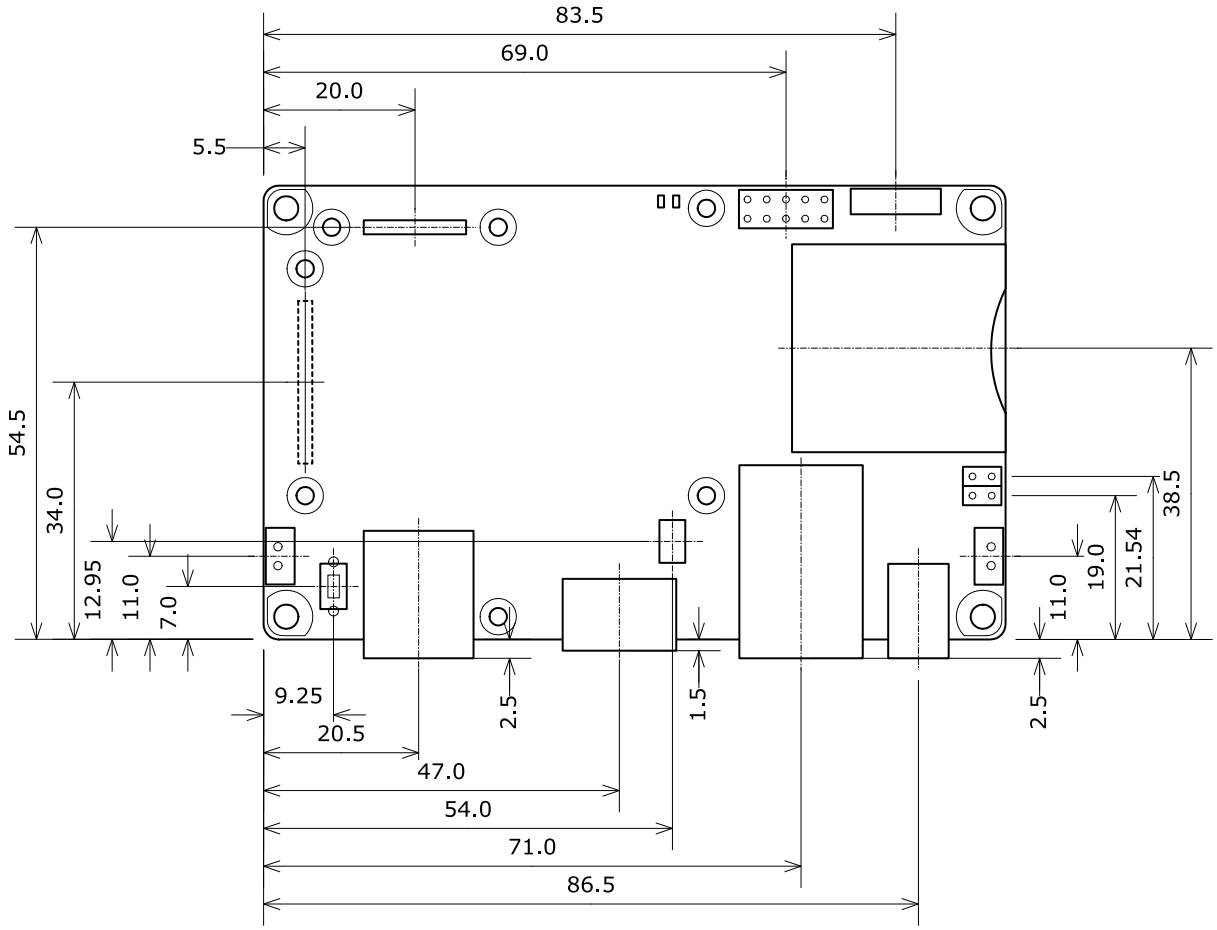


図 19.1 基板形状および固定穴寸法



[Unit : mm]

図 19.2 コネクタ中心寸法

20. オプション品

本章では、Armadillo-840 関連のオプション品について説明します。

20.1. Armadillo-840 拡張ボード 01(C コネクタ用)

20.1.1. ボード概要

Armadillo-840 拡張ボード 01(C コネクタ用)は Armadillo-840 の CON7 拡張インターフェース 1(C コネクタ)に接続可能な拡張ボード 01 です。タッチパネル LCD、オーディオコーデック、ビデオアンプ、SD スロット等を搭載しています。

Armadillo-840 拡張ボード 01(C コネクタ用)の主な仕様は次の通りです。

表 20.1 Armadillo-840 拡張ボード 01(C コネクタ用)の仕様

LCD	DataImage 製 タッチパネル LCD「SCF0500133GFR03」用コネクタ
オーディオ	Wolfson 製オーディオコーデック「WM8978CGEFL/V」搭載 モノラルマイク入力ミニジャック ステレオヘッドホン出力ミニジャック オーディオライン/コンポジットビデオ出力インターフェース
SD/MMC	SD スロット
無線 LAN	アットマークテクノ製無線 LAN モジュール「AWL13-U00Z」用コネクタ
カメラ	アットマークテクノ製カメラモジュール「OP-A810-CAM01-00」用コネクタ
USB	USB mini B コネクタ、USB2.0 Device(High Speed 対応)
拡張インターフェース	LCD、カメラ、UART、GPIO、I2C、SPI、I2S、SD、MMC、PWM 等
スイッチ	ユーザー用タクトスイッチ x 4 リセット用タクトスイッチ
LED	ユーザー用 LED(黄色、面実装) x 6 リセット用 LED(黄色、面実装)
電源電圧	DC 5V±5%、3,3V±5%
使用周囲温度	10〜40℃(ただし結露なきこと)
基板サイズ	96 x 150mm(突起部を除く)

表 20.2 タッチパネル LCD の仕様

型番	SCF0500133GFR03
メーカー名	DataImage
タイプ	TFT
サイズ	5 インチ
解像度	WVGA(800×480 ピクセル)
バックライト	LED
色数	True Color(1677 万色、24bit)
タッチパネル	静電容量方式、マルチタッチ対応(最大 2 フィンガー)

20.1.2. ブロック図

Armadillo-840 拡張ボード 01(C コネクタ用)のブロック図は次の通りです。

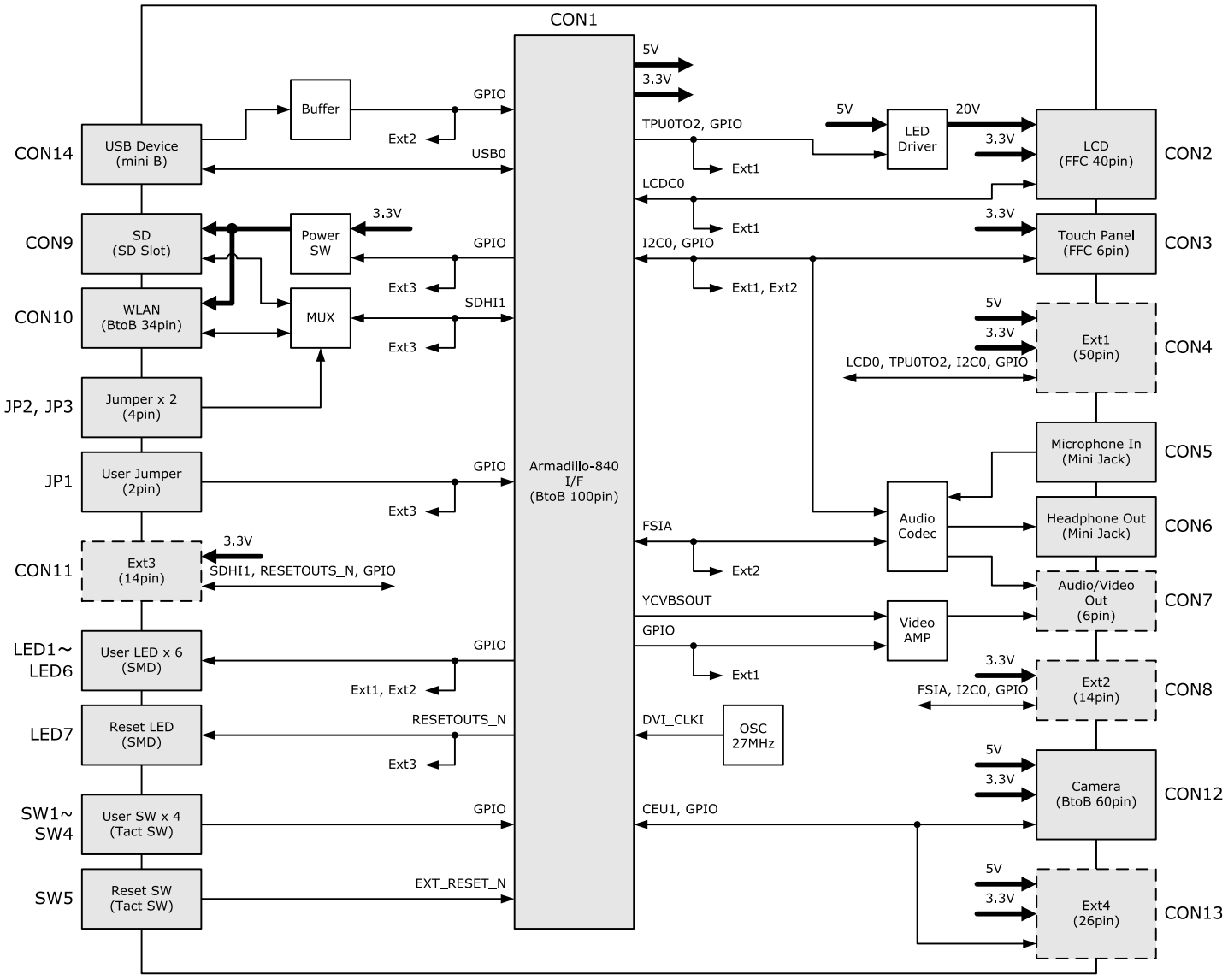


図 20.1 Armadillo-840 拡張ボード 01(C コネクタ用)のブロック図

20.1.3. インターフェースレイアウト

Armadillo-840 拡張ボード 01(C コネクタ用)のインターフェースレイアウトは次の通りです。

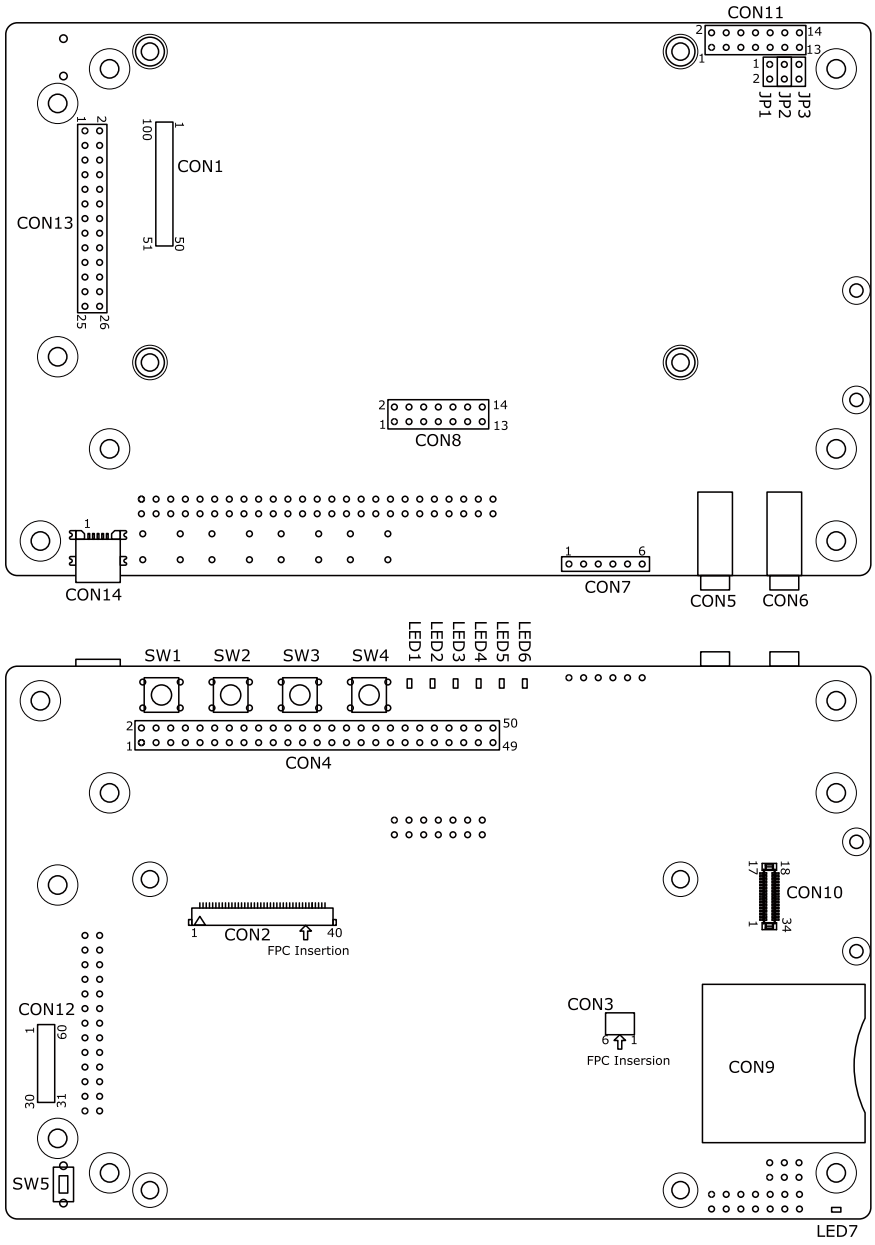


図 20.2 Armadillo-840 拡張ボード 01(C コネクタ用) インターフェースレイアウト図

表 20.3 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	Armadillo-840 インターフェース	DF40HC(3.0)-100DS-0.4V(51)	HIROSE ELECTRIC
CON2	LCD インターフェース	FH28-40S-0.5SH(05)	HIROSE ELECTRIC
CON3	タッチパネルインターフェース	FH34S-6S-0.5SH(50)	HIROSE ELECTRIC
CON4	拡張インターフェース 1	-	
CON5	マイク入力インターフェース	SJ-3524-SMT	CUI
CON6	ヘッドホン出力インターフェース	SJ-3524-SMT	CUI
CON7	オーディオライン/コンポジットビデオ出力インターフェース	-	

部品番号	インターフェース名	型番	メーカー
CON8	拡張インターフェース 2	-	
CON9	SD インターフェース	SCDA9A0400	ALPS ELECTRIC
CON10	WLAN インターフェース	AXK6F34347YG または AXK6F34347YG-E	Panasonic
CON11	拡張インターフェース 3	-	
CON12	カメラインターフェース	DF40C-60DP-0.4V(51)	HIROSE ELECTRIC
CON13	拡張インターフェース 4	-	
CON14	USB インターフェース	54819-0572	Molex
JP1	ユーザージャンパ	A2-2PA-2.54DSA(71)	HIROSE ELECTRIC
JP2	設定ジャンパ		
JP3			
SW1	ユーザースイッチ	SKHHAMA010	ALPS ELECTRIC
SW2			
SW3			
SW4			
SW5	リセットスイッチ	SKHLACA010	ALPS ELECTRIC

20.1.4. インターフェース仕様

20.1.4.1. CON1 Armadillo-840 インターフェース

CON1 は Armadillo-840 の拡張インターフェース 1(C コネクタ)との接続用インターフェースです。

搭載コネクタ　　DF40HC(3.0)-100DS-0.4V(51)/HIROSE ELECTRIC

許容電流: 0.3A 以下(端子 1 本あたり)

表 20.4 CON1 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	CAM_RSTB	In/Out	CON12 の 40 ピン、CON13 の 24 ピンに接続
3	CAM_PWDN	In/Out	CON12 の 41 ピン、CON13 の 23 ピンに接続
4	A1_VIO_CKO_1	In/Out	CON12 の 19 ピン、CON13 の 21 ピンに接続
5	A1_VIO_FIELD_1	In/Out	CON12 の 15 ピン、CON13 の 15 ピンに接続
6	A1_VIO_HD_1	In/Out	CON12 の 16 ピン、CON13 の 16 ピンに接続
7	A1_VIO_CLK_1	In/Out	CON12 の 13 ピン、CON13 の 19 ピンに接続
8	A1_VIO_VD_1	In/Out	CON12 の 17 ピン、CON13 の 17 ピンに接続
9	CAM_SCL	In/Out	CON12 の 37 ピン、CON13 の 26 ピンに接続
10	CAM_SDA	In/Out	CON12 の 38 ピン、CON13 の 25 ピンに接続
11	GND	Power	電源(GND)
12	A1_LCDDCK_0	In/Out	CON2 の 30 ピン、CON4 の 32 ピンに接続
13	A1_LCDVSYN_0	In/Out	CON2 の 33 ピン、CON4 の 34 ピンに接続
14	A1_LCDHSYN_0	In/Out	CON2 の 32 ピン、CON4 の 33 ピンに接続
15	A1_LCDDISP_0	In/Out	CON2 の 34 ピン、CON4 の 35 ピンに接続
16	A1_LCDDON_0	In/Out	CON2 の 31 ピン、CON4 の 36 ピンに接続
17	VAMP_PSAVE_N	In/Out	ビデオアンプのパワーセーブピン、CON4 の 42 ピンに接続
18	BL_SHDN_N	In/Out	LED バックライトドライバのイネーブルピン、CON4 の 41 ピンに接続
19	A1_TPU0TO2	In/Out	LED バックライトドライバの電流調整ピン、CON4 の 40 ピンに接続
20	EXT_LED1	In/Out	LED1、CON4 の 39 ピンに接続
21	EXT_LED2	In/Out	LED2、CON4 の 38 ピンに接続
22	EXT_LED3	In/Out	LED3、CON4 の 37 ピンに接続

ピン番号	信号名	I/O	機能
23	A1_I2C_SCL_0	In/Out	CON3 の 2 ピン、CON4 の 46 ピン、CON8 の 13 ピン、オーディオコーデックの I2C クロックピンに接続
24	A1_I2C_SDA_0	In/Out	CON3 の 3 ピン、CON4 の 45 ピン、CON8 の 14 ピン、オーディオコーデックの I2C データピンに接続
25	TP_INT_N	In/Out	CON3 の 4 ピン、CON4 の 47 ピンに接続
26	TP_RST_N	In/Out	CON3 の 5 ピン、CON4 の 48 ピンに接続
27	A1_LCDD19_0	In/Out	CON2 の 8 ピン、CON4 の 10 ピンに接続
28	A1_LCDD20_0	In/Out	CON2 の 9 ピン、CON4 の 11 ピンに接続
29	A1_LCDD21_0	In/Out	CON2 の 10 ピン、CON4 の 12 ピンに接続
30	A1_LCDD22_0	In/Out	CON2 の 11 ピン、CON4 の 13 ピンに接続
31	A1_LCDD23_0	In/Out	CON2 の 12 ピン、CON4 の 14 ピンに接続
32	GND	Power	電源(GND)
33	A1_SDHICLK_1	In/Out	CON11 の 3 ピン、マルチプレクサを経由して CON9 の 5 ピン、CON10 の 7 ピンに接続
34	A1_SDHICMD_1	In/Out	CON11 の 4 ピン、マルチプレクサを経由して CON9 の 2 ピン、CON10 の 10 ピンに接続
35	A1_SDHID0_1	In/Out	CON11 の 5 ピン、マルチプレクサを経由して CON9 の 7 ピン、CON10 の 2 ピンに接続
36	A1_SDHID1_1	In/Out	CON11 の 6 ピン、マルチプレクサを経由して CON9 の 8 ピン、CON10 の 1 ピンに接続
37	A1_SDHID2_1	In/Out	CON11 の 7 ピン、マルチプレクサを経由して CON9 の 9 ピン、CON10 の 12 ピンに接続
38	A1_SDHID3_1	In/Out	CON11 の 8 ピン、マルチプレクサを経由して CON9 の 9 ピン、CON10 の 11 ピンに接続
39	A1_SDHICD_1	In/Out	CON11 の 9 ピン、マルチプレクサを経由して CON9 の 10 ピンに接続
40	A1_SDHIWP_1	In/Out	CON11 の 10 ピン、マルチプレクサを経由して CON9 の 12 ピンに接続
41	EXT_SD_PWR_EN	In/Out	CON11 の 11 ピン、CON9、CON12 の電源ピンのパワースイッチに接続
42	EXT_JP1	In/Out	CON11 の 12 ピン、JP1 の 2 ピンに接続
43	SW1	In/Out	SW1 の 1 ピンに接続
44	SW2	In/Out	SW2 の 1 ピンに接続
45	SW3	In/Out	SW3 の 1 ピンに接続
46	SW4	In/Out	SW4 の 1 ピンに接続
47	EXT_RESET_N	Out	SW5 の 1 ピンに接続
48	GND	Power	電源(GND)
49	USB0_DM	In/Out	CON14 の 2 ピンに接続
50	USB0_DP	In/Out	CON14 の 3 ピンに接続
51	VCC_5V	Power	電源(VCC_5V)
52	VCC_5V	Power	電源(VCC_5V)
53	VCC_5V	Power	電源(VCC_5V)
54	NC	-	未接続
55	A1_YCVBSOUT	In	コンポジットビデオ入力、ビデオアンプの入力ピンに接続
56	GND	Power	電源(GND)
57	A1_DV_CLKI	Out	27MHz ビデオクロック出力、27MHz クロックに接続
58	GND	Power	電源(GND)
59	A1_RESETOUTS_N	In	R-Mobile A1 リセット入力、CON11 の 14 ピン、LED7 に接続
60	AUDIO_CLK	In	オーディオコーデックの MCLK ピン、CON8 の 3 ピンに接続
61	A1_FSIAIBT	In/Out	オーディオコーデックの BCLK ピン、CON8 の 4 ピンに接続
62	A1_FSIAILR	In/Out	オーディオコーデックの LRC ピン、CON8 の 5 ピンに接続
63	A1_FSIAOSLD	In/Out	オーディオコーデックの DACDAT ピン、CON8 の 6 ピンに接続
64	A1_FSIAISLD	In/Out	オーディオコーデックの ADCDAT ピン、CON8 の 7 ピンに接続
65	USB0_VBUS	In/Out	CON8 の 8 ピン、CON14 の VBUS ピンに接続
66	EXT_LED4	In/Out	CON8 の 9 ピン、LED4 に接続
67	EXT_LED5	In/Out	CON8 の 10 ピン、LED5 に接続
68	EXT_LED6	In/Out	CON8 の 11 ピン、LED6 に接続
69	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
70	A1_LCDD18_0	In/Out	CON2 の 7 ピン、CON4 の 9 ピンに接続
71	A1_LCDD17_0	In/Out	CON2 の 6 ピン、CON4 の 8 ピンに接続
72	A1_LCDD16_0	In/Out	CON2 の 5 ピン、CON4 の 7 ピンに接続
73	A1_LCDD15_0	In/Out	CON2 の 20 ピン、CON4 の 22 ピンに接続
74	A1_LCDD14_0	In/Out	CON2 の 19 ピン、CON4 の 21 ピンに接続
75	A1_LCDD13_0	In/Out	CON2 の 18 ピン、CON4 の 20 ピンに接続
76	A1_LCDD12_0	In/Out	CON2 の 17 ピン、CON4 の 19 ピンに接続
77	A1_LCDD11_0	In/Out	CON2 の 16 ピン、CON4 の 18 ピンに接続
78	A1_LCDD10_0	In/Out	CON2 の 15 ピン、CON4 の 17 ピンに接続
79	A1_LCDD9_0	In/Out	CON2 の 14 ピン、CON4 の 16 ピンに接続
80	A1_LCDD8_0	In/Out	CON2 の 13 ピン、CON4 の 15 ピンに接続
81	A1_LCDD7_0	In/Out	CON2 の 28 ピン、CON4 の 30 ピンに接続
82	A1_LCDD6_0	In/Out	CON2 の 27 ピン、CON4 の 29 ピンに接続
83	A1_LCDD5_0	In/Out	CON2 の 26 ピン、CON4 の 28 ピンに接続
84	A1_LCDD4_0	In/Out	CON2 の 25 ピン、CON4 の 27 ピンに接続
85	A1_LCDD3_0	In/Out	CON2 の 24 ピン、CON4 の 26 ピンに接続
86	A1_LCDD2_0	In/Out	CON2 の 23 ピン、CON4 の 25 ピンに接続
87	A1_LCDD1_0	In/Out	CON2 の 22 ピン、CON4 の 24 ピンに接続
88	A1_LCDD0_0	In/Out	CON2 の 21 ピン、CON4 の 23 ピンに接続
89	GND	Power	電源(GND)
90	A1_VIO_D7_1	In/Out	CON12 の 57 ピン、CON13 の 14 ピンに接続
91	A1_VIO_D6_1	In/Out	CON12 の 56 ピン、CON13 の 13 ピンに接続
92	A1_VIO_D5_1	In/Out	CON12 の 55 ピン、CON13 の 12 ピンに接続
93	A1_VIO_D4_1	In/Out	CON12 の 54 ピン、CON13 の 11 ピンに接続
94	A1_VIO_D3_1	In/Out	CON12 の 53 ピン、CON13 の 10 ピンに接続
95	A1_VIO_D2_1	In/Out	CON12 の 52 ピン、CON13 の 9 ピンに接続
96	A1_VIO_D1_1	In/Out	CON12 の 51 ピン、CON13 の 8 ピンに接続
97	A1_VIO_D0_1	In/Out	CON12 の 50 ピン、CON13 の 7 ピンに接続
98	VCC_3.3V	Power	電源(VCC_3.3V)
99	VCC_3.3V	Power	電源(VCC_3.3V)
100	VCC_3.3V	Power	電源(VCC_3.3V)

20.1.4.2. CON2 LCD インターフェース

CON2 は DataImage 製タッチパネル LCD「SCF0500133GFR03」の接続用インターフェースです。R-Mobile A1 の LCD コントローラ(LCD0)に接続されます。

搭載コネクタ　　FH28-40S-0.5SH(05)/HIROSE ELECTRIC

許容電流: 0.5A 以下(端子 1 本あたり)

CON4 と共通の信号が接続されています。同時利用には対応していませんので、CON4 を使用する場合は CON2、CON3 からタッチパネル LCD を取り外してください。

表 20.5 CON2 信号配列

ピン番号	信号名	I/O	機能
1	LCD_LED−	Out	LED バックライト用電源(カソード)、LED バックライトドライバに接続
2	LCD_LED+	Out	LED バックライト用電源(アノード)、LED バックライトドライバに接続
3	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	A1_LCDD16_0	Out	LCD データ(bit16)、CON1 の 72 ピンに接続 CON4 の 7 ピンと共通
6	A1_LCDD17_0	Out	LCD データ(bit17)、CON1 の 71 ピンに接続 CON4 の 8 ピンと共通
7	A1_LCDD18_0	Out	LCD データ(bit18)、CON1 の 70 ピンに接続 CON4 の 9 ピンと共通
8	A1_LCDD19_0	Out	LCD データ(bit19)、CON1 の 27 ピンに接続 CON4 の 10 ピンと共通
9	A1_LCDD20_0	Out	LCD データ(bit20)、CON1 の 28 ピンに接続 CON4 の 11 ピンと共通
10	A1_LCDD21_0	Out	LCD データ(bit21)、CON1 の 29 ピンに接続 CON4 の 12 ピンと共通
11	A1_LCDD22_0	Out	LCD データ(bit22)、CON1 の 30 ピンに接続 CON4 の 13 ピンと共通
12	A1_LCDD23_0	Out	LCD データ(bit23)、CON1 の 31 ピンに接続 CON4 の 14 ピンと共通
13	A1_LCDD8_0	Out	LCD データ(bit8)、CON1 の 80 ピンに接続 CON4 の 15 ピンと共通
14	A1_LCDD9_0	Out	LCD データ(bit9)、CON1 の 79 ピンに接続 CON4 の 16 ピンと共通
15	A1_LCDD10_0	Out	LCD データ(bit10)、CON1 の 78 ピンに接続 CON4 の 17 ピンと共通
16	A1_LCDD11_0	Out	LCD データ(bit11)、CON1 の 77 ピンに接続 CON4 の 18 ピンと共通
17	A1_LCDD12_0	Out	LCD データ(bit12)、CON1 の 76 ピンに接続 CON4 の 19 ピンと共通
18	A1_LCDD13_0	Out	LCD データ(bit13)、CON1 の 75 ピンに接続 CON4 の 20 ピンと共通
19	A1_LCDD14_0	Out	LCD データ(bit14)、CON1 の 74 ピンに接続 CON4 の 21 ピンと共通
20	A1_LCDD15_0	Out	LCD データ(bit15)、CON1 の 73 ピンに接続 CON4 の 22 ピンと共通
21	A1_LCDD0_0	Out	LCD データ(bit0)、CON1 の 88 ピンに接続 CON4 の 23 ピンと共通
22	A1_LCDD1_0	Out	LCD データ(bit1)、CON1 の 87 ピンに接続 CON4 の 24 ピンと共通
23	A1_LCDD2_0	Out	LCD データ(bit2)、CON1 の 86 ピンに接続 CON4 の 25 ピンと共通
24	A1_LCDD3_0	Out	LCD データ(bit3)、CON1 の 85 ピンに接続 CON4 の 26 ピンと共通
25	A1_LCDD4_0	Out	LCD データ(bit4)、CON1 の 84 ピンに接続 CON4 の 27 ピンと共通
26	A1_LCDD5_0	Out	LCD データ(bit5)、CON1 の 83 ピンに接続 CON4 の 28 ピンと共通
27	A1_LCDD6_0	Out	LCD データ(bit6)、CON1 の 82 ピンに接続 CON4 の 29 ピンと共通
28	A1_LCDD7_0	Out	LCD データ(bit7)、CON1 の 81 ピンに接続 CON4 の 30 ピンと共通
29	GND	Power	電源(GND)
30	A1_LCDDCK_0	Out	DCK 信号、CON1 の 12 ピンに接続 CON4 の 32 ピンと共通
31	A1_LCDDON_0	Out	DON 信号、CON1 の 16 ピンに接続 CON4 の 36 ピンと共通
32	A1_LCDHSYN_0	Out	HSYNC 信号、CON1 の 14 ピンに接続 CON4 の 33 ピンと共通

ピン番号	信号名	I/O	機能
33	A1_LCDVSYN_0	Out	VSYNC 信号、CON1 の 13 ピンに接続 CON4 の 34 ピンと共通
34	A1_LCDDISP_0	Out	DISP 信号、CON1 の 15 ピンに接続 CON4 の 35 ピンと共通
35	NC	-	未接続
36	GND	Power	電源(GND)
37	NC	-	未接続
38	NC	-	未接続
39	NC	-	未接続
40	NC	-	未接続

20.1.4.3. CON3 タッチパネルインターフェース

CON3 は DataImage 製タッチパネル LCD「SCF0500133GFR03」のタッチパネル接続用インターフェースです。R-Mobile A1 の I2C コントローラ(I2C0)、GPIO コントローラに接続されます。

搭載コネクタ　　FH34S-6S-0.5SH(50)/HIROSE ELECTRIC

許容電流: 0.5A 以下(端子 1 本あたり)

CON4 と共通の信号が接続されています。同時利用には対応していませんので、CON4 を使用する場合は、CON2、CON3 からタッチパネル LCD を取り外してください。

表 20.6 CON3 信号配列

ピン番号	信号名	I/O	機能
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	A1_I2C_SCL_0	In/Out	I2C クロック、CON1 の 23 ピンに接続 CON4 の 46 ピン、CON8 の 13 ピン、オーディオコーデックの I2C クロックピンと共通
3	A1_I2C_SDA_0	In/Out	I2C データ、CON1 の 24 ピンに接続 CON4 の 45 ピン、CON8 の 14 ピン、オーディオコーデックの I2C データピンと共通
4	TP_INT_N	In	タッチパネル割り込み信号(Active-Low)、CON1 の 25 ピンに接続 CON4 の 47 ピンと共通
5	TP_RST_N	Out	タッチパネルリセット信号、CON1 の 26 ピンに接続 CON4 の 48 ピンと共通
6	GND	Power	電源(GND)

20.1.4.4. CON4 拡張インターフェース 1

CON4 は機能拡張用インターフェースです。主に、LCD 接続用の信号が配線されています。

CON2、CON3 と共通の信号が接続されています。同時利用には対応していませんので、CON4 を使用する場合は、CON2、CON3 からタッチパネル LCD を取り外してください。

搭載コネクタ例　　A1-50PA-2.54DSA(71)/HIROSE ELECTRIC

表 20.7 CON4 信号配列

ピン番号	信号名	I/O	機能
1	VCC_5V	Power	電源(VCC_5V)
2	VCC_5V	Power	電源(VCC_5V)
3	VCC_3.3V	Power	電源(VCC_3.3V)
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	GND	Power	電源(GND)
6	GND	Power	電源(GND)
7	A1_LCDD16_0	In/Out	拡張入出力、CON1 の 72 ピンに接続 CON2 の 5 ピンと共通
8	A1_LCDD17_0	In/Out	拡張入出力、CON1 の 71 ピンに接続 CON2 の 6 ピンと共通
9	A1_LCDD18_0	In/Out	拡張入出力、CON1 の 70 ピンに接続 CON2 の 7 ピンと共通
10	A1_LCDD19_0	In/Out	拡張入出力、CON1 の 27 ピンに接続 CON2 の 8 ピンと共通
11	A1_LCDD20_0	In/Out	拡張入出力、CON1 の 28 ピンに接続 CON2 の 9 ピンと共通
12	A1_LCDD21_0	In/Out	拡張入出力、CON1 の 29 ピンに接続 CON2 の 10 ピンと共通
13	A1_LCDD22_0	In/Out	拡張入出力、CON1 の 30 ピンに接続 CON2 の 11 ピンと共通
14	A1_LCDD23_0	In/Out	拡張入出力、CON1 の 31 ピンに接続 CON2 の 12 ピンと共通
15	A1_LCDD8_0	In/Out	拡張入出力、CON1 の 80 ピンに接続 CON2 の 13 ピンと共通
16	A1_LCDD9_0	In/Out	拡張入出力、CON1 の 79 ピンに接続 CON2 の 14 ピンと共通
17	A1_LCDD10_0	In/Out	拡張入出力、CON1 の 78 ピンに接続 CON2 の 15 ピンと共通
18	A1_LCDD11_0	In/Out	拡張入出力、CON1 の 77 ピンに接続 CON2 の 16 ピンと共通
19	A1_LCDD12_0	In/Out	拡張入出力、CON1 の 76 ピンに接続 CON2 の 17 ピンと共通
20	A1_LCDD13_0	In/Out	拡張入出力、CON1 の 75 ピンに接続 CON2 の 18 ピンと共通
21	A1_LCDD14_0	In/Out	拡張入出力、CON1 の 74 ピンに接続 CON2 の 19 ピンと共通
22	A1_LCDD15_0	In/Out	拡張入出力、CON1 の 73 ピンに接続 CON2 の 20 ピンと共通
23	A1_LCDD0_0	In/Out	拡張入出力、CON1 の 88 ピンに接続 CON2 の 21 ピンと共通
24	A1_LCDD1_0	In/Out	拡張入出力、CON1 の 87 ピンに接続 CON2 の 22 ピンと共通
25	A1_LCDD2_0	In/Out	拡張入出力、CON1 の 86 ピンに接続 CON2 の 23 ピンと共通
26	A1_LCDD3_0	In/Out	拡張入出力、CON1 の 85 ピンに接続 CON2 の 24 ピンと共通
27	A1_LCDD4_0	In/Out	拡張入出力、CON1 の 84 ピンに接続 CON2 の 25 ピンと共通
28	A1_LCDD5_0	In/Out	拡張入出力、CON1 の 83 ピンに接続 CON2 の 26 ピンと共通
29	A1_LCDD6_0	In/Out	拡張入出力、CON1 の 82 ピンに接続 CON2 の 27 ピンと共通

ピン番号	信号名	I/O	機能
30	A1_LCDD7_0	In/Out	拡張入出力、CON1 の 81 ピンに接続 CON2 の 28 ピンと共通
31	GND	Power	電源(GND)
32	A1_LCDDCK_0	In/Out	拡張入出力、CON1 の 12 ピンに接続 CON2 の 30 ピンと共通
33	A1_LCDHSYN_0	In/Out	拡張入出力、CON1 の 14 ピンに接続 CON2 の 32 ピンと共通
34	A1_LCDVSYN_0	In/Out	拡張入出力、CON1 の 13 ピンに接続 CON2 の 33 ピンと共通
35	A1_LCDDISP_0	In/Out	拡張入出力、CON1 の 15 ピンに接続 CON2 の 34 ピンと共通
36	A1_LCDDON_0	In/Out	拡張入出力、CON1 の 16 ピンに接続 CON2 の 31 ピンと共通
37	EXT_LED3	In/Out	拡張入出力、CON1 の 22 ピン、LED3 に接続
38	EXT_LED2	In/Out	拡張入出力、CON1 の 21 ピン、LED2 に接続
39	EXT_LED1	In/Out	拡張入出力、CON1 の 20 ピン、LED1 に接続
40	A1_TPU0TO2	In/Out	拡張入出力、CON1 の 19 ピンに接続
41	BL_SHDN_N	In/Out	拡張入出力、CON1 の 18 ピンに接続
42	VAMP_PSAVE_N	In/Out	拡張入出力、CON1 の 17 ピンに接続
43	NC	-	未接続
44	NC	-	未接続
45	A1_I2C_SDA_0	In/Out	I2C データ、CON1 の 24 ピンに接続 CON3 の 3 ピン、CON8 の 14 ピン、オーディオコーデックの I2C データピンと共通
46	A1_I2C_SCL_0	In/Out	I2C クロック、CON1 の 23 ピンに接続 CON3 の 2 ピン、CON8 の 13 ピン、オーディオコーデックの I2C クロックピンと共通
47	TP_INT_N	In/Out	拡張入出力、CON1 の 25 ピンに接続 CON3 の 4 ピンと共通
48	TP_RST_N	In/Out	拡張入出力、CON1 の 26 ピンに接続 CON3 の 5 ピンと共通
49	VCC_3.3V	Power	電源(VCC_3.3V)
50	GND	Power	電源(GND)

表 20.8 CON4 マルチプレクス

ピン番号	機能				
	LCD	UART	SSI	GPIO	その他
1					
2					
3					
4					
5					
6					
7	LCDD16_0		MSIOF2_MCK1	PORT42	
8	LCDD17_0		MSIOF2_SS1	PORT41	
9	LCDD18_0			PORT40	
10	LCDD19_0	SCIFB_TXD		PORT4	
11	LCDD20_0	SCIFB_RXD		PORT3	
12	LCDD21_0	SCIFB_SCK		PORT2	
13	LCDD22_0	SCIFA_RXD_7		PORT0	
14	LCDD23_0	SCIFA_TXD_7		PORT1	
15	LCDD8_0			PORT50	KEYIN7
16	LCDD9_0			PORT49	KEYIN6
17	LCDD10_0			PORT48	KEYIN5

ピン番号	機能				
	LCD	UART	SSI	GPIO	その他
18	LCDD11_0			PORT47	KEYIN4
19	LCDD12_0			PORT46	KEYIN3
20	LCDD13_0		MSIOF2_RSCK	PORT45	KEYIN2
21	LCDD14_0		MSIOF2_RSYNC	PORT44	KEYIN1
22	LCDD15_0		MSIOF2_MCK0	PORT43	KEYIN0
23	LCDD0_0			PORT58	KEYOUT7
24	LCDD1_0			PORT57	KEYOUT6
25	LCDD2_0			PORT56	KEYOUT5
26	LCDD3_0			PORT55	KEYOUT4
27	LCDD4_0			PORT54	KEYOUT3
28	LCDD5_0			PORT53	KEYOUT2
29	LCDD6_0			PORT52	KEYOUT1
30	LCDD7_0			PORT51	KEYOUT0
31					
32	LCDDCK_0			PORT62	
33	LCDHSYN_0			PORT64	
34	LCDVSYN_0			PORT63	
35	LCDDISP_0		MSIOF2_TSCK	PORT65	
36	LCDDON_0		MSIOF2_TXD	PORT61	
37	LCDVEPWC_0			PORT60	
38	LCDVCPWC_0			PORT59	
39	LCDLCLK_0			PORT102	
40				PORT202	TPU0TO2
41	LCDRD_0_N		MSIOF2_TSYNC	PORT164	
42			MSIOF2_RXD	PORT165	
43					
44					
45					I2C_SDA_0
46					I2C_SCL_0
47		SCIFB_RTS_N		PORT172	
48		SCIFB_CTS_N		PORT173	
49					
50					

20.1.4.5. CON5 マイク入力インターフェース

CON5 はモノラルマイク入力インターフェースです。オーディオコーデックを経由して R-Mobile A1 の I2S コントローラ(FSIA)に接続されます。

搭載コネクタ　　SJ-3524-SMT/CUI

表 20.9 CON5 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	MIC_IN	In	マイク入力
3	NC	-	未接続
10	NC	-	未接続

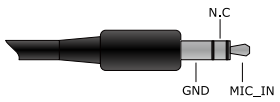


図 20.3 ミニプラグ(マイク)のピンアサイン

20.1.4.6. CON6 ヘッドホン出力インターフェース

CON6 はステレオヘッドホン出力インターフェースです。オーディオコーデックを経由して R-Mobile A1 の I2S コントローラ(FSIA)に接続されます。

搭載コネクタ　　SJ-3524-SMT/CUI

表 20.10 CON6 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	HP_OUT_L	Out	ヘッドホン出力(左チャンネル)
3	HP_OUT_R	Out	ヘッドホン出力(右チャンネル)
10	NC	-	未接続

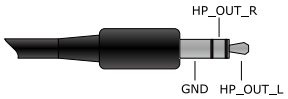


図 20.4 ミニプラグ(ヘッドホン)のピンアサイン

20.1.4.7. CON7 オーディオライン/コンポジットビデオ出力インターフェース

CON7 はオーディオライン出力、コンポジットビデオ出力インターフェースです。オーディオライン信号は Wolfson 製オーディオコーデック「WM8978CGEFL/V」を経由して R-Mobile A1 の I2S コントローラ(FSIA)に接続されます。コンポジット信号はビデオアンプを経由して R-Mobile A1 の NTSC/PAL、ビデオエンコーダ(SDENC)に接続されます。

搭載コネクタ例　　A2-06PA-2.54DSA(71)/HIROSE ELECTRIC

表 20.11 CON7 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	CVBS_OUT	Out	コンポジットビデオ出力、ビデオアンプに接続
3	GND	Power	電源(GND)
4	LINE_OUT_R	Out	ライン出力(右チャンネル)
5	GND	Power	電源(GND)
6	LINE_OUT_L	Out	ライン出力(左チャンネル)

20.1.4.8. CON8 拡張インターフェース 2

CON8 は機能拡張用インターフェースです。主にオーディオ用の信号が配線されています。

搭載コネクタ例　　A1-14PA-2.54DSA(71)/HIROSE ELECTRIC

オーディオコーデックと共通の信号が接続されていますので、CON8 を使用する場合はハードウェアの改造が必要となります。詳細につきましては「Armadillo-840 拡張ボード 01(C コネクタ用)の回路図」でご確認ください。

表 20.12 CON8 信号配列

ピン番号	信号名	I/O	機能
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	GND	Power	電源(GND)
3	AUDIO_CLK	Out	オーディオクロック(12.288MHz)、オーディオコーデックの MCLK ピン、CON1 の 60 ピンに接続
4	A1_FSIAIBT	In/Out	拡張入出力、オーディオコーデックの BCLK ピン、CON1 の 61 ピンに接続
5	A1_FSIAILR	In/Out	拡張入出力、オーディオコーデックの LRC ピン、CON1 の 62 ピンに接続
6	A1_FSIAOSLD	In/Out	拡張入出力、オーディオコーデックの DACDAT ピン、CON1 の 63 ピンに接続
7	A1_FSIAISLD	In/Out	拡張入出力、オーディオコーデックの ADCDAT ピン、CON1 の 64 ピンに接続
8	USB0_VBUS	In/Out	拡張入出力、CON1 の 65 ピン、CON14 の VBUS ピンに接続
9	EXT_LED4	In/Out	拡張入出力、CON1 の 66 ピン、RTC の INT1 ピン、LED4 に接続
10	EXT_LED5	In/Out	拡張入出力、CON1 の 67 ピン、RTC の SCL ピン、LED5 に接続
11	EXT_LED6	In/Out	拡張入出力、CON1 の 68 ピン、RTC の SDA ピン、LED6 に接続
12	GND	Power	電源(GND)
13	A1_I2C_SCL_0	In/Out	I2C クロック、CON3 の 2 ピン、CON4 の 46 ピン、CON1 の 23 ピン、オーディオコーデックの I2C クロックピンに接続
14	A1_I2C_SDA_0	In/Out	I2C データ、CON3 の 3 ピン、CON4 の 45 ピン、CON1 の 24 ピン、オーディオコーデックの I2C データピンに接続

表 20.13 CON8 マルチプレクス

ピン番号	マルチプレクス			
	I2S	UART	GPIO	I2C
1				
2				
3				
4	FSIAIBT	SCIFA_TXD_4	PORT13	
5	FSIAILR	SCIFA_RXD_4	PORT12	
6	FSIAOSLD		PORT9	
7	FSIAISLD		PORT5	
8		SCIFA_TXD_5	PORT20	
9	FSIAOMC	SCIFA_RXD_5	PORT10	
10	FSIAOBT		PORT8	
11	FSIAOLR		PORT7	
12				
13				I2C_SCL_0
14				I2C_SDA_0

20.1.4.9. CON9 SD インターフェース

CON9 は SD インターフェースです。マルチプレクサを経由して R-Mobile A1 の SD コントローラ(SDHI1)に接続されます。JP2、JP3 により CON9、CON10、CON11 のどのインターフェースを有効にするかを設定して使用します。

搭載コネクタ　　SCDA9A0400/ALPS ELECTRIC

表 20.14 ジャンパの設定(CON9 を有効)

ジャンパ	状態
JP2	オープン
JP3	オープン

表 20.15 CON9 信号配列

ピン番号	信号名	I/O	機能
1	EXT_SD_DAT3	In/Out	SD データ(bit3)
2	EXT_SD_CMD	In/Out	SD コマンド/レスポンス
3	GND	Power	電源(GND)
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	EXT_SD_CLK	Out	SD クロック
6	GND	Power	電源(GND)
7	EXT_SD_DAT0	In/Out	SD データ(bit0)
8	EXT_SD_DAT1	In/Out	SD データ(bit1)
9	EXT_SD_DAT2	In/Out	SD データ(bit2)
10	EXT_SD_CD	In	カード検出 (Low:カード挿入、High:カード未挿入)
11	GND	Power	電源(GND)
12	EXT_SD_WP	In	ライトプロテクト検出 (Low:書き込み可能、High:書き込み不可能)
13	GND	Power	電源(GND)
14	GND	Power	電源(GND)

20.1.4.10. CON10 WLAN インターフェース

CON10 はアットマークテクノ製無線 LAN モジュール「AWL13-U00Z」の接続用インターフェースです。SDIO 起動モードで動作するよう設定されています。マルチプレクサを経由して R-Mobile A1 の SD コントローラ(SDHI1)に接続されます。JP2、JP3 により CON9、CON10、CON11 のどのインターフェースを有効にするかを設定して使用します。

搭載コネクタ　　AXK6F34347YG または AXK6F34347YG-E/Panasonic

対向コネクタ例　　AXK5F34347YG/Panasonic

表 20.16 ジャンパの設定(CON10 有効)

ジャンパ	状態
JP2	ショート
JP3	オープン

表 20.17 CON10 信号配列

ピン番号	信号名	I/O	機能
1	AWLAN_DAT1	In/Out	SDIO データ(bit1)
2	AWLAN_DAT0	In/Out	SDIO データ(bit0)
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	AWLAN_CLK	Out	SDIO クロック
8	VCC_3.3V	Power	電源(VCC_3.3V)
9	NC	-	未接続
10	AWLAN_CMD	In/Out	SDIO コマンド
11	AWLAN_DAT3	In/Out	SDIO データ(bit3)
12	AWLAN_DAT2	In/Out	SDIO データ(bit2)
13	UART_RXD	-	未接続
14	UART_TXD	-	未接続

ピン番号	信号名	I/O	機能
15	BOOT_SEL1	Out	起動モード設定
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 10kΩ プルダウン
22	NC	-	未接続
23	NC	-	未接続
24	NC	-	未接続
25	NC	-	未接続
26	NC	-	未接続
27	NC	-	未接続
28	HRST	Out	VCC_3.3V に接続
29	NC	-	未接続
30	NC	-	未接続
31	NC	-	未接続
32	NC	-	未接続
33	NC	-	未接続
34	NC	-	未接続

20.1.4.11. CON11 拡張インターフェース 3

CON11 は機能拡張用インターフェースです。主に、SD 用の信号が配線されています。JP2、JP3 により CON9、CON10、CON11 のどのインターフェースを有効にするかを設定して使用します。

搭載コネクタ例　　A1-14PA-2.54DSA(71)/HIROSE ELECTRIC

表 20.18 ジャンパの設定(CON11 有効)

ジャンパ	状態
JP2	-
JP3	ショート

表 20.19 CON11 信号配列

ピン番号	信号名	I/O	機能
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	GND	Power	電源(GND)
3	A1_SDHICLK_1	In/Out	拡張入出力、CON1 の 33 ピンに接続
4	A1_SDHICMD_1	In/Out	拡張入出力、CON1 の 34 ピンに接続
5	A1_SDHID0_1	In/Out	拡張入出力、CON1 の 35 ピンに接続
6	A1_SDHID1_1	In/Out	拡張入出力、CON1 の 36 ピンに接続
7	A1_SDHID2_1	In/Out	拡張入出力、CON1 の 37 ピンに接続
8	A1_SDHID3_1	In/Out	拡張入出力、CON1 の 38 ピンに接続
9	A1_SDHICD_1	In/Out	拡張入出力、CON1 の 39 ピンに接続
10	A1_SDHIWP_1	In/Out	拡張入出力、CON1 の 40 ピンに接続
11	EXT_SD_PWR_EN	In/Out	拡張入出力、CON1 の 41 ピンに接続
12	EXT_JP1	In/Out	拡張入出力、CON1 の 42 ピン、JP1 の 2 ピンに接続
13	GND	Power	電源(GND)
14	A1_RESETOUTS_N	Out	R-Mobile A1 リセット入力、CON1 の 59 ピン、LED7 に接続

表 20.20 CON11 マルチプレクス

ピン番号	マルチプレクス				
	SSI	SD	MMC	GPIO	その他
1					
2					
3		SDHICLK_1	MMCCLK_0	PORT66	TPU0TO2
4	MSIOF1_SS1	SDHICMD_1	MMCCMD_0	PORT67	
5	MSIOF1_RSCK	SDHID0_1	MMCD0_0	PORT68	
6	MSIOF1_RSYNC	SDHID1_1	MMCD1_0	PORT69	
7	MSIOF1_MCK0	SDHID2_1	MMCD2_0	PORT70	
8	MSIOF1_MCK1	SDHID3_1	MMCD3_0	PORT71	
9	MSIOF1_TSCK	SDHICD_1	MMCD4_0	PORT72	
10	MSIOF1_TSYNC	SDHIWP_1	MMCD5_0	PORT73	
11	MSIOF1_TXD		MMCD6_0	PORT74	
12	MSIOF1_RXD		MMCD7_0	PORT75	
13					
14					

20.1.4.12. CON12 カメラインターフェース

CON12 はアットマークテクノ製カメラモジュール「OP-A810-CAM01-00」の接続用インターフェースです。R-Mobile A1 の CEU コントローラ(CEU1)に接続されます。

搭載コネクタ　　DF40C-60DP-0.4V(51)/HIROSE ELECTRIC

許容電流: 0.3A 以下(端子 1 本あたり)

対向コネクタ例　　DF40HC(4.0)-60DS-0.4V(51)/HIROSE ELECTRIC

CON13 と共通の信号が接続されていますが、同時利用には対応していませんので、どちらか一方のコネクタのみでご使用ください。

表 20.21 CON12 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)
3	GND	Power	電源(GND)
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	GND	Power	電源(GND)
13	A1_VIO_CLK_1	In	カメラクロック入力、CON1 の 7 ピンに接続 CON13 の 19 ピンと共通
14	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
15	A1_VIO_FIELD_1	In	FIELD 信号、CON1 の 5 ピンに接続 CON13 の 15 ピンと共通
16	A1_VIO_HD_1	In	HSYNC 信号、CON1 の 6 ピンに接続 CON13 の 16 ピンと共通
17	A1_VIO_VD_1	In	VSYNC 信号、CON1 の 8 ピンに接続 CON13 の 17 ピンと共通
18	GND	Power	電源(GND)
19	A1_VIO_CKO_1	Out	カメラクロック出力、CON1 の 4 ピンに接続 CON13 の 21 ピンと共通
20	GND	Power	電源(GND)
21	NC	-	未接続
22	NC	-	未接続
23	GND	Power	電源(GND)
24	GND	Power	電源(GND)
25	GND	Power	電源(GND)
26	GND	Power	電源(GND)
27	NC	-	未接続
28	NC	-	未接続
29	NC	-	未接続
30	NC	-	未接続
31	VCC_5V	Power	電源(VCC_5V)
32	VCC_5V	Power	電源(VCC_5V)
33	VCC_5V	Power	電源(VCC_5V)
34	VCC_5V	Power	電源(VCC_5V)
35	NC	-	未接続
36	GND	Power	電源(GND)
37	CAM_SCL	In/Out	I2C クロック、CON1 の 9 ピンに接続 CON13 の 26 ピンと共通
38	CAM_SDA	In/Out	I2C データ、CON1 の 10 ピンに接続 CON13 の 25 ピンと共通
39	GND	Power	電源(GND)
40	CAM_RSTB	Out	カメラリセット信号出力、CON1 の 2 ピンに接続 CON13 の 24 ピンと共通
41	CAM_PWDN	Out	カメラスタンバイモード出力、CON1 の 3 ピンに接続 CON13 の 23 ピンと共通
42	NC	-	未接続
43	GND	Power	電源(GND)
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	A1_VIO_D0_1	In/Out	カメラデータ(bit0)、CON1 の 97 ピンに接続 CON13 の 7 ピンと共通
51	A1_VIO_D1_1	In/Out	カメラデータ(bit1)、CON1 の 96 ピンに接続 CON13 の 8 ピンと共通
52	A1_VIO_D2_1	In/Out	カメラデータ(bit2)、CON1 の 95 ピンに接続 CON13 の 9 ピンと共通
53	A1_VIO_D3_1	In/Out	カメラデータ(bit3)、CON1 の 94 ピンに接続 CON13 の 10 ピンと共通
54	A1_VIO_D4_1	In/Out	カメラデータ(bit4)、CON1 の 93 ピンに接続 CON13 の 11 ピンと共通
55	A1_VIO_D5_1	In/Out	カメラデータ(bit5)、CON1 の 92 ピンに接続 CON13 の 12 ピンと共通

ピン番号	信号名	I/O	機能
56	A1_VIO_D6_1	In/Out	カメラデータ(bit6)、CON1 の 91 ピンに接続 CON13 の 13 ピンと共通
57	A1_VIO_D7_1	In/Out	カメラデータ(bit7)、CON1 の 90 ピンに接続 CON13 の 14 ピンと共通
58	VCC_3.3V	Power	電源(VCC_3.3V)
59	VCC_3.3V	Power	電源(VCC_3.3V)
60	VCC_3.3V	Power	電源(VCC_3.3V)

20.1.4.13. CON13 拡張インターフェース 4

CON13 は機能拡張用のインターフェースです。主に、カメラ用の信号が配線されています。

搭載コネクタ例　　A1-26PA-2.54DSA(71)/HIROSE ELECTRIC

CON12 と共通の信号が接続されていますが、同時利用には対応していませんので、どちらか一方のコネクタのみでご使用ください。

表 20.22 CON13 信号配列

ピン番号	信号名	I/O	機能
1	VCC_5V	Power	電源(VCC_5V)
2	VCC_5V	Power	電源(VCC_5V)
3	VCC_3.3V	Power	電源(VCC_3.3V)
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	GND	Power	電源(GND)
6	GND	Power	電源(GND)
7	A1_VIO_D0_1	In/Out	カメラデータ(bit0)、CON1 の 97 ピンに接続 CON12 の 50 ピンと共通
8	A1_VIO_D1_1	In/Out	CON1 の 96 ピンに接続 CON12 の 51 ピンと共通
9	A1_VIO_D2_1	In/Out	拡張入出力、CON1 の 95 ピンに接続 CON12 の 52 ピンと共通
10	A1_VIO_D3_1	In/Out	拡張入出力、CON1 の 94 ピンに接続 CON12 の 53 ピンと共通
11	A1_VIO_D4_1	In/Out	拡張入出力、CON1 の 93 ピンに接続 CON12 の 54 ピンと共通
12	A1_VIO_D5_1	In/Out	拡張入出力、CON1 の 92 ピンに接続 CON12 の 55 ピンと共通
13	A1_VIO_D6_1	In/Out	拡張入出力、CON1 の 91 ピンに接続 CON12 の 56 ピンと共通
14	A1_VIO_D7_1	In/Out	拡張入出力、CON1 の 90 ピンに接続 CON12 の 57 ピンと共通
15	A1_VIO_FIELD_1	In/Out	拡張入出力、CON1 の 5 ピンに接続 CON12 の 15 ピンと共通
16	A1_VIO_HD_1	In/Out	拡張入出力、CON1 の 6 ピンに接続 CON12 の 16 ピンと共通
17	A1_VIO_VD_1	In/Out	拡張入出力、CON1 の 8 ピンに接続 CON12 の 17 ピンと共通
18	GND	Power	電源(GND)
19	A1_VIO_CLK_1	In/Out	拡張入出力、CON1 の 7 ピンに接続 CON12 の 13 ピンと共通
20	GND	Power	電源(GND)

ピン番号	信号名	I/O	機能
21	A1_VIO_CKO_1	In/Out	拡張入出力、CON1 の 4 ピンに接続 CON12 の 19 ピンと共通
22	GND	Power	電源(GND)
23	CAM_PWDN	In/Out	拡張入出力、CON1 の 3 ピンに接続 CON12 の 41 ピンと共通
24	CAM_RSTB	In/Out	拡張入出力、CON1 の 2 ピンに接続 CON12 の 40 ピンと共通
25	CAM_SDA	In/Out	I2C データ、CON1 の 10 ピンに接続 CON12 の 38 ピンと共通
26	CAM_SCL	In/Out	I2C クロック、CON1 の 9 ピンに接続 CON12 の 37 ピンと共通

表 20.23 CON13 マルチプレクス

ピン番号	マルチプレクス			
	CAMERA	UART	GPIO	その他
1				
2				
3				
4				
5				
6				
7	VIO_D0_1		PORT182	
8	VIO_D1_1		PORT181	
9	VIO_D2_1		PORT180	TPU0TO3
10	VIO_D3_1		PORT179	
11	VIO_D4_1		PORT178	
12	VIO_D5_1	SCIFA_TXD_6	PORT26	
13	VIO_D6_1	SCIFA_RXD_6	PORT25	
14	VIO_D7_1	SCIFA_SCK_6	PORT24	
15	VIO_FIELD_1	SCIFA_CTS_1_N	PORT21	TPU0TO1
16	VIO_HD_1		PORT160	
17	VIO_VD_1	SCIFA_TXD_0	PORT198	
18				
19	VIO_CLK_1	SCIFA_RXD_0	PORT197	
20				
21	VIO_CKO_1	SCIFA_RTS_1_N	PORT23	TPU0TO0
22				
23		SCIFA_TXD_1	PORT196	
24		SCIFA_RXD_1	PORT195	
25		SCIFA_CTS_0_N	PORT193	
26		SCIFA_RTS_0_N	PORT194	

20.1.4.14. CON14 USB インターフェース

CON14 は USB デバイスインターフェースです。R-Mobile A1 の USB コントローラ(USB0)に接続されます。

USB0　　データ転送モード: USB2.0 High Speed/Full Speed

搭載コネクタ　　54819-0572/Molex

表 20.24 CON14 信号配列

ピン番号	信号名	I/O	機能
1	USB0_VBUS	In	VBUS 検知
2	USB0_DM	In/Out	USB マイナス側信号、CON1 の 49 ピンに接続
3	USB0_DP	In/Out	USB プラス側信号、CON1 の 50 ピンに接続
4	NC	-	未接続
5	GND	Power	電源(GND)

20.1.4.15. JP1 ユーザージャンパ

JP1 はユーザー側で自由に使用できるジャンパです。R-Mobile A1 の GPIO コントローラ(PORT75)に接続されます。

搭載コネクタ　　A2-2PA-2.54DSA(71)/HIROSE ELECTRIC

表 20.25 JP1 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	EXT_JP1	In	ユーザージャンパ、CON1 の 42 ピンに接続 CON11 の 12 ピンと共通

20.1.4.16. JP2、JP3 設定ジャンパ

JP2、JP3 は SD 設定ジャンパです。

搭載コネクタ　　A2-2PA-2.54DSA(71)/HIROSE ELECTRIC

表 20.26 JP2 信号配列

ピン番号	信号名	I/O	機能
1	GND	Power	電源(GND)
2	EXT_JP2	In	SD インターフェース選択、マルチプレクサのセレクトピンに接続

表 20.27 JP3 信号配列

ピン番号	信号名	I/O	機能
1	EXT_JP3	In	SD インターフェース選択、マルチプレクサのイネーブルピンに接続
2	VCC_3.3V	Power	電源(VCC_3.3V)

表 20.28 ジャンパの設定

有効インターフェース	JP2	JP3
SD インターフェース(CON9)	オープン	オープン
AWL13 接続インターフェース(CON10)	ショート	オープン
拡張インターフェース 3(CON11)	-	ショート

20.1.4.17. SW1～SW4 ユーザースイッチ

SW1～SW4 はユーザー側で自由に使用できるスイッチです。R-Mobile A1 の GPIO コントローラに接続されます。

搭載スイッチ　　SKHHAMA010/ALPS ELECTRIC

表 20.29 ユーザースイッチの機能

スイッチ	機能	R-Mobile A1 の設定
SW1	ユーザースイッチ、CON1 の 43 ピンと接続 (押されていない状態: High、押された状態: Low)	PORT97
SW2	ユーザースイッチ、CON1 の 44 ピンと接続 (押されていない状態: High、押された状態: Low)	PORT98
SW3	ユーザースイッチ、CON1 の 45 ピンと接続 (押されていない状態: High、押された状態: Low)	PORT99
SW4	ユーザースイッチ、CON1 の 46 ピンと接続 (押されていない状態: High、押された状態: Low)	PORT100

20.1.4.18. SW5 リセットスイッチ

SW5 はリセットスイッチです。

搭載スイッチ　　SKHLACA010/ALPS ELECTRIC

表 20.30 リセットスイッチの機能

スイッチ	機能
SW5	外部リセット、CON1 の 47 ピンと接続 (押されていない状態: Open、押された状態: Low)

20.1.4.19. LED1～LED6 ユーザー LED

LED1～LED6 はユーザー側で自由に使用できる面実装の黄色 LED です。LED に接続された R-Mobile A1 の信号が GPIO の出力モードに設定されている場合に制御できます。

表 20.31 ユーザー LED の機能

LED	機能	R-Mobile A1 の設定
LED1	CON1 の 20 ピンに接続 CON4 の 39 ピンと共通 (Low:消灯、High:点灯)	PORT102
LED2	CON1 の 21 ピンに接続 CON4 の 38 ピンと共通 (Low:消灯、High:点灯)	PORT59
LED3	CON1 の 22 ピンに接続 CON4 の 37 ピンと共通 (Low:消灯、High:点灯)	PORT60
LED4	CON1 の 66 ピンに接続 CON8 の 9 ピンと共通 (Low:消灯、High:点灯)	PORT10
LED5	CON1 の 67 ピンに接続 CON8 の 10 ピンと共通 (Low:消灯、High:点灯)	PORT8
LED6	CON1 の 68 ピンに接続 CON8 の 11 ピンと共通 (Low:消灯、High:点灯)	PORT7

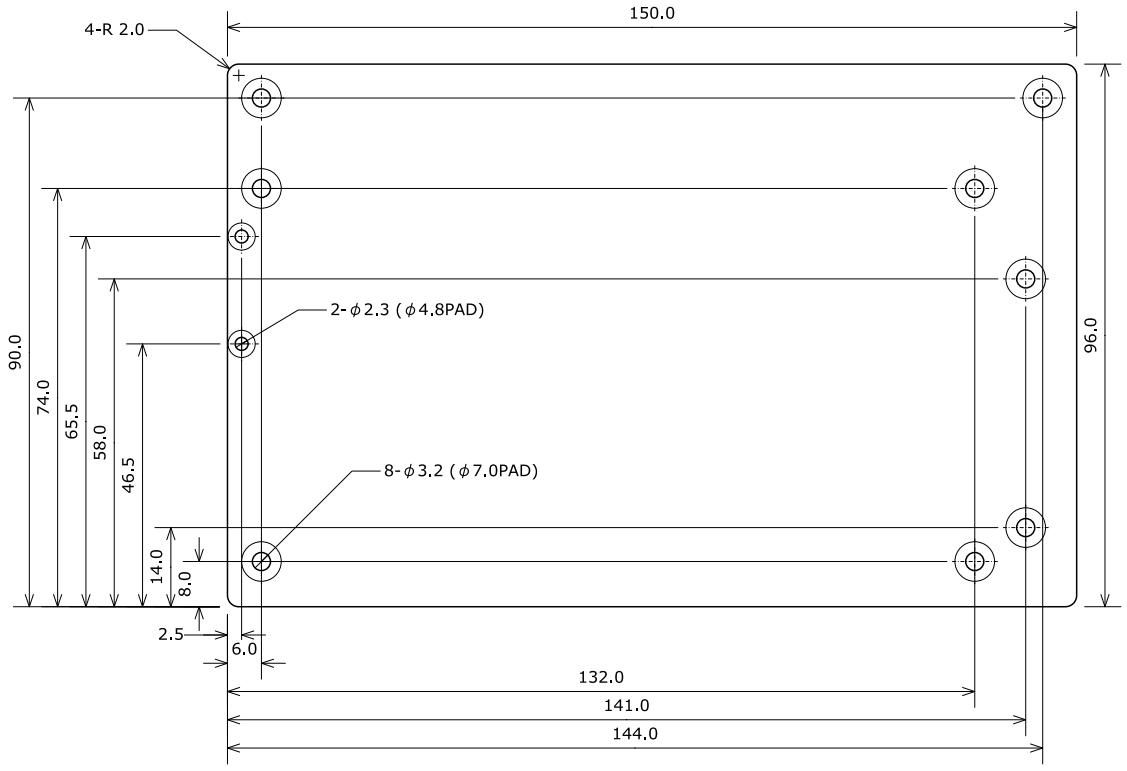
20.1.4.20. LED7 リセット LED

LED7 はリセット LED です。R-Mobile A1 の RESETOUTS_N ピンに接続されるため、R-Mobile A1 のリセット状態を確認することが可能です。

表 20.32 リセット LED の機能

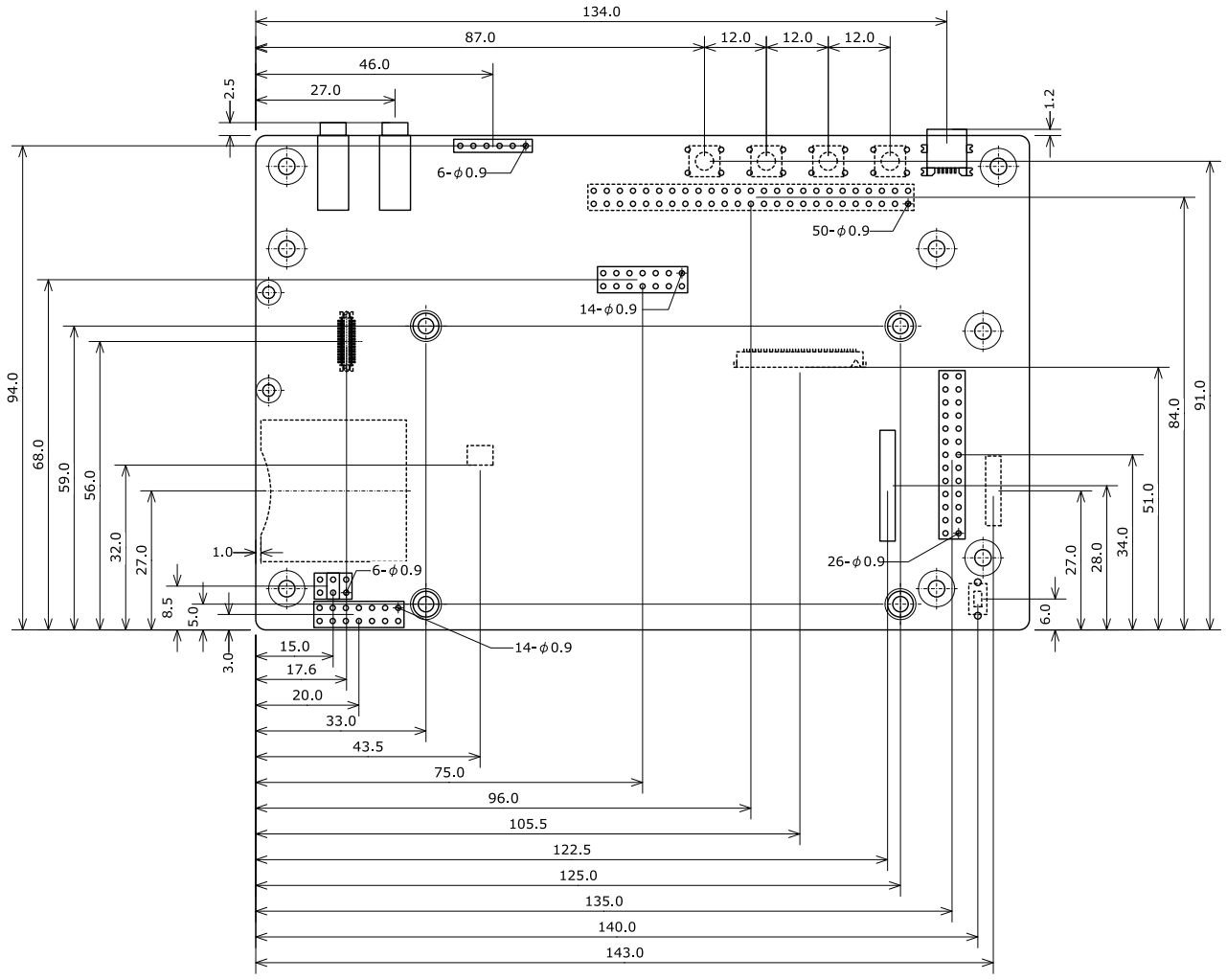
LED	名称(色)	機能
LED7	リセット LED(黄色)	CON1 の 59 ピン、CON11 の 14 ピンに接続 R-Mobile A1 がリセット中に消灯

20.1.5. 基板形状図



[Unit : mm]

図 20.5 基板形状および固定穴寸法

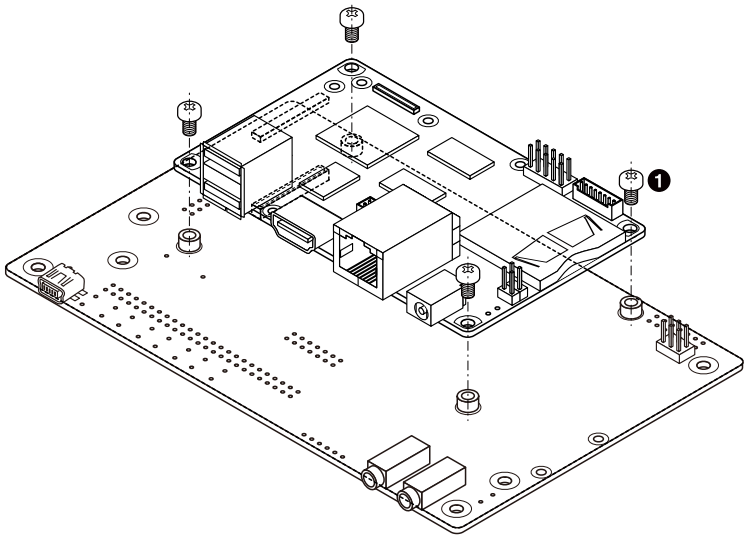


[Unit : mm]

図 20.6 コネクタ中心寸法およびコネクタ穴寸法

20.1.6. 組み立て

20.1.6.1. Armadillo-840 と拡張ボード 01 の組み立て

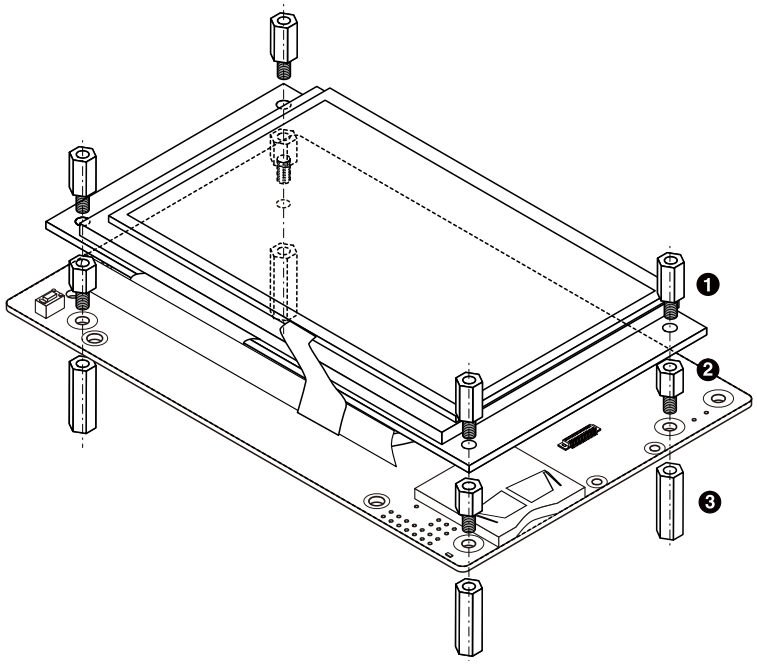


❶ なべ小ねじ(M3、L=5mm)

図 20.7 Armadillo-840 と拡張ボード 01 の組み立て

ねじの締め付けトルクは 5kgf・cm(49N・cm)以下とし、締め付け過ぎにご注意ください。

20.1.6.2. タッチパネル LCD と拡張ボード 01 の組み立て



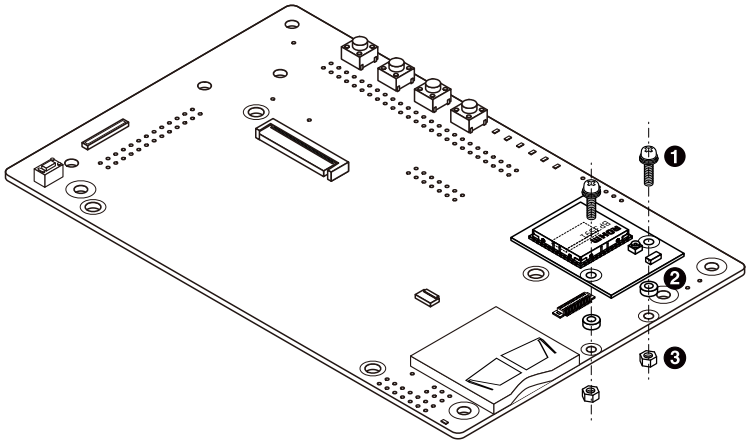
❶ スペーサ(M3、L=10mm)

❷ スペーサ(M3、L=6.5mm)

❸ スペーサ(M3、L=24mm)

図 20.8 タッチパネル LCD と拡張ボード 01 の組み立て

20.1.6.3. WLAN と拡張ボード 01 の組み立て



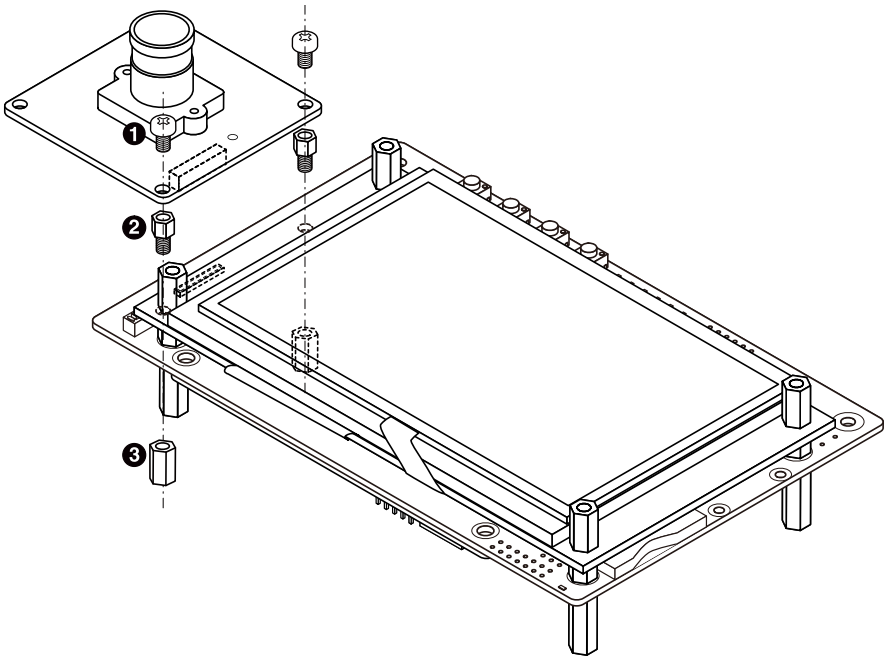
❶ なべ小ねじ スプリングワッシャー、小径平ワッシャー付(M2、L=8mm)

❷ 金属スペーサ(M2、L=1.5mm、直径=4mm)

❸ ナット(M2、L=1.6mm、平径=4mm)

図 20.9 WLAN と拡張ボード 01 の組み立て

20.1.6.4. カメラと拡張ボード 01 の組み立て



❶　なべ小ねじ(M3、L=4mm)

❷　金属スペーサ(M3、L=4mm)

❸　ナット(M2、L=1.6mm、平径=4mm)

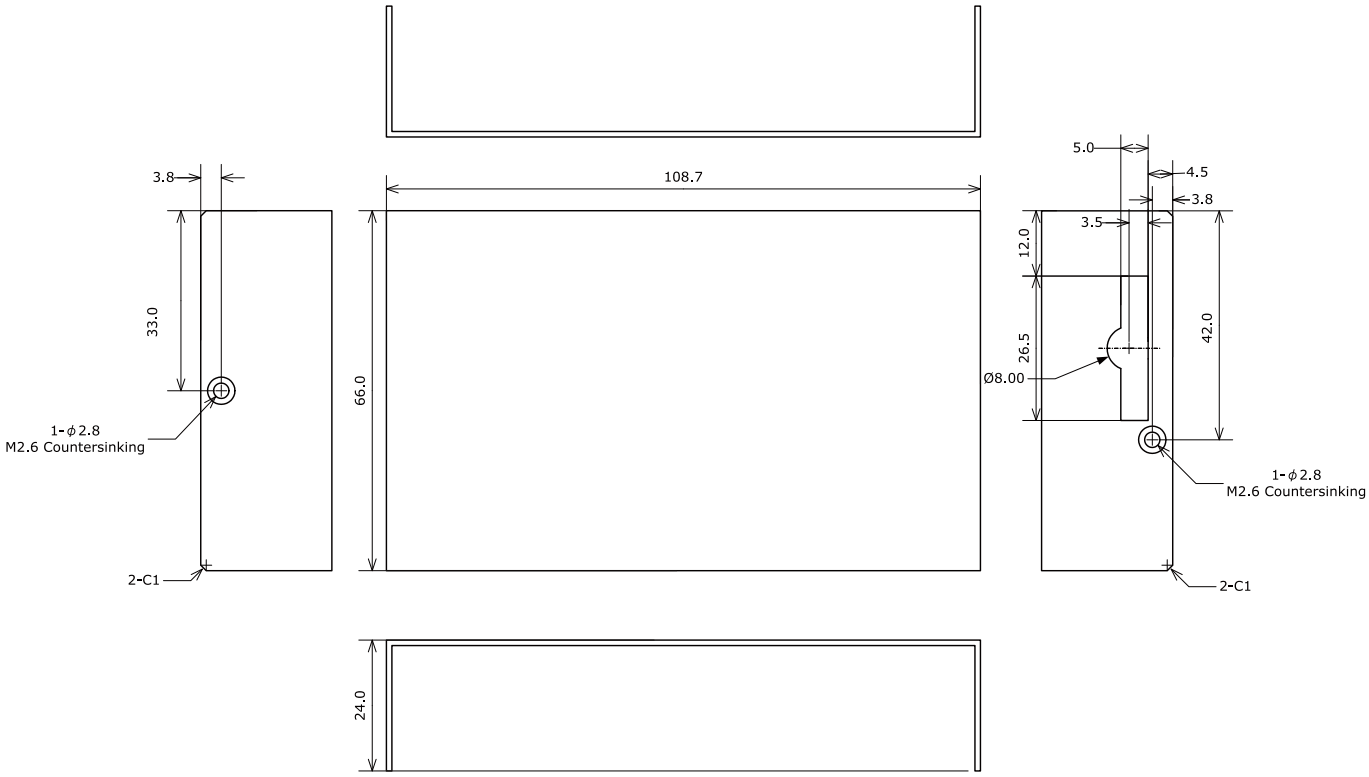
図 20.10 カメラと拡張ボード 01 の組み立て

20.2. Armadillo-840 オプションケース(金属製)

Armadillo-840 オプションケース(金属製)は、アルミ製の Armadillo-840 専用ケースです。Armadillo-840 を収めた状態で、DC ジャック、LAN インターフェース、HDMI インターフェース、USB インターフェース×2 にアクセス可能となっています。

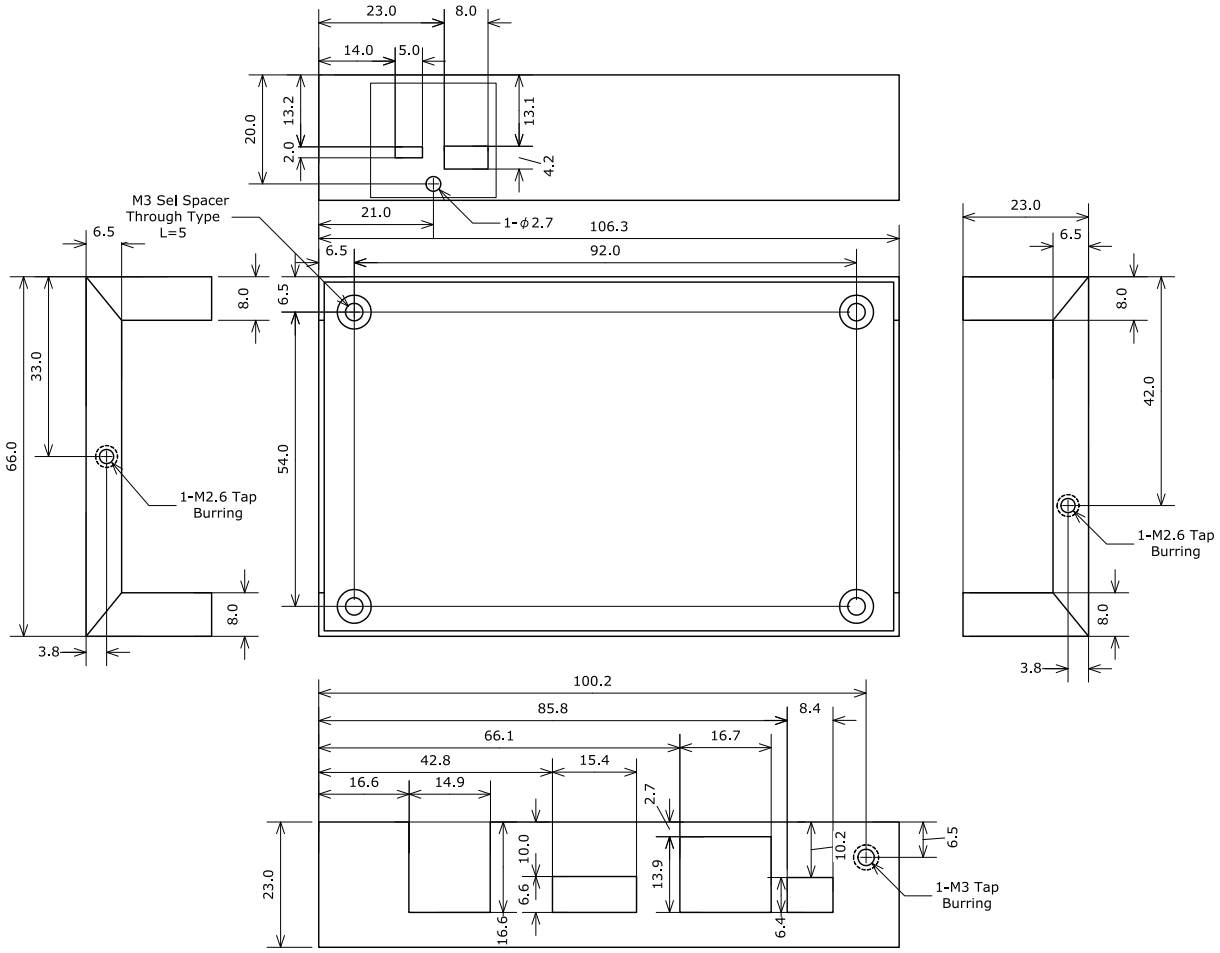
表 20.33 Armadillo-840 オプションケース(金属製)仕様

項目	説明
板厚	1mm
材質	アルミ
表面処理	アルマイト白、梨地



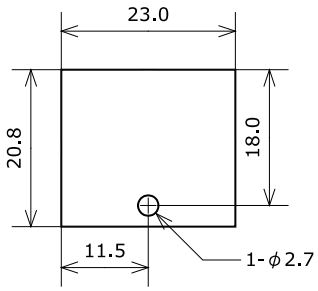
[Unit : mm]

図 20.11 Armadillo-840 オプションケース(金属製)上板寸法図



[Unit : mm]

図 20.12 Armadillo-840 オプションケース(金属製)下板寸法図

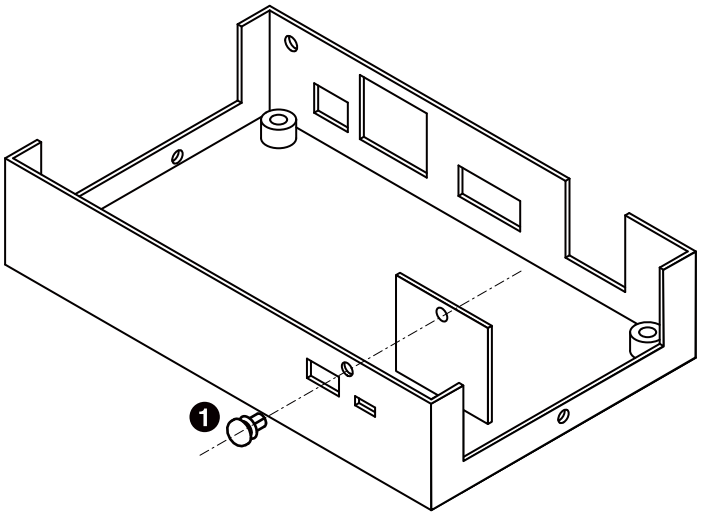


[Unit : mm]

図 20.13 Armadillo-840 オプションケース(金属製)目隠しプレート寸法図

20.2.1. 組み立て

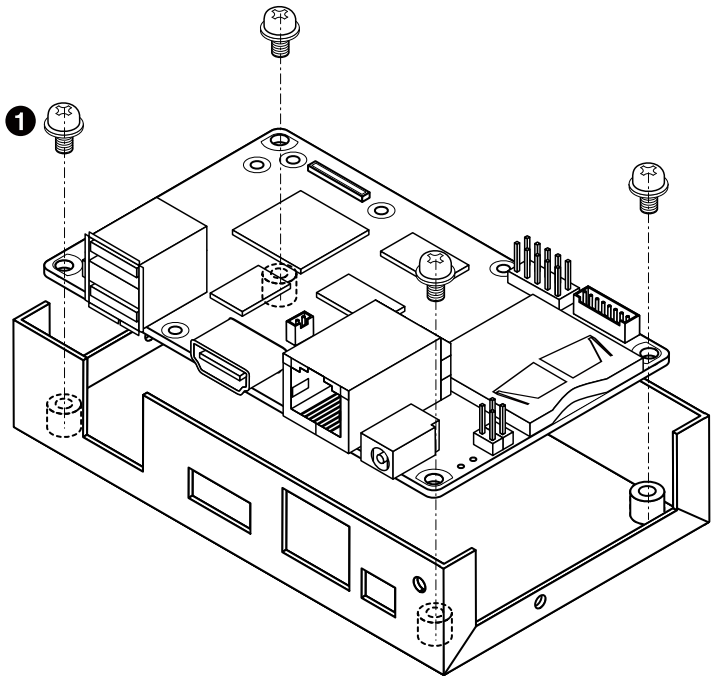
20.2.1.1. 目隠しプレートの取り付け



❶ プッシュリベット(NP-2632W/廣杉計器)

図 20.14 目隠しプレートの取り付け

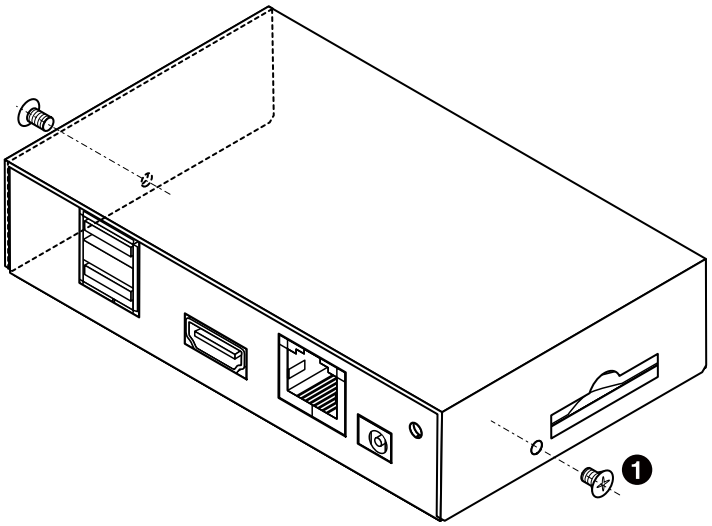
20.2.1.2. Armadillo-840 の組み込み



❶ なべ小ねじ 小径平ワッシャー付(M3、L=4mm)

図 20.15 Armadillo-840 の組み込み

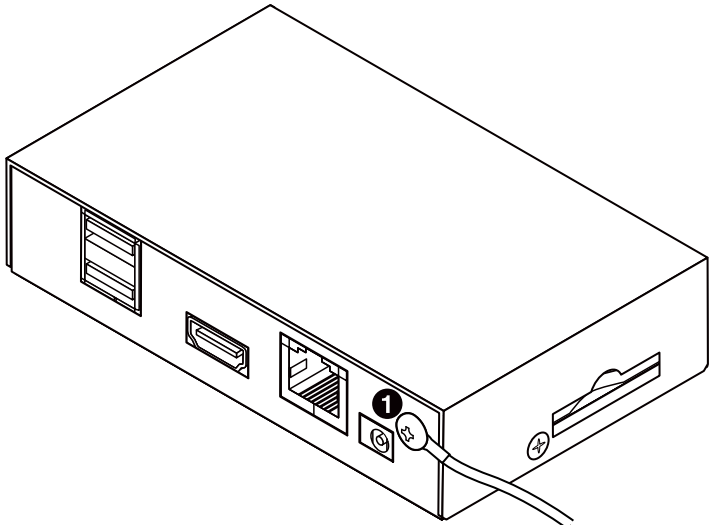
20.2.1.3. オプションケース上板の取り付け



❶ 皿小ねじ(M2.6、L=4mm)

図 20.16 オプションケース上板の取り付け

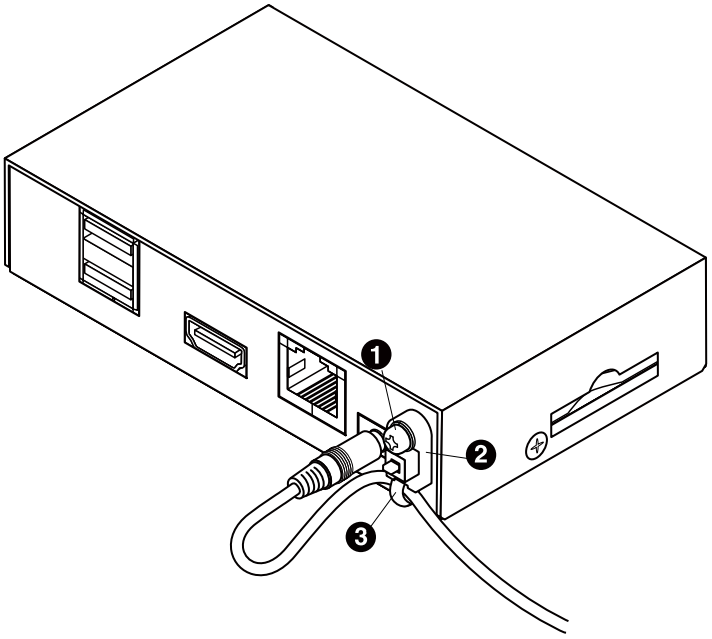
20.2.1.4. アース端子の取り付け例



❶ トラス小ねじ(M3、L=5mm)

図 20.17 アース端子の取り付け例

20.2.1.5. AC アダプタケーブル抜け防止パーツの取り付け例



❶　なべ小ねじ スプリングワッシャー付(M3、L=10mm)

❷　マウントヘッド

❸　インシュロックタイ

図 20.18 AC アダプタケーブル抜け防止パーツの取り付け例

20.3. 開発用 USB シリアル変換アダプタ

開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)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。Armadillo-840 のシリアルインターフェース(CON4)に接続して使用することが可能です。

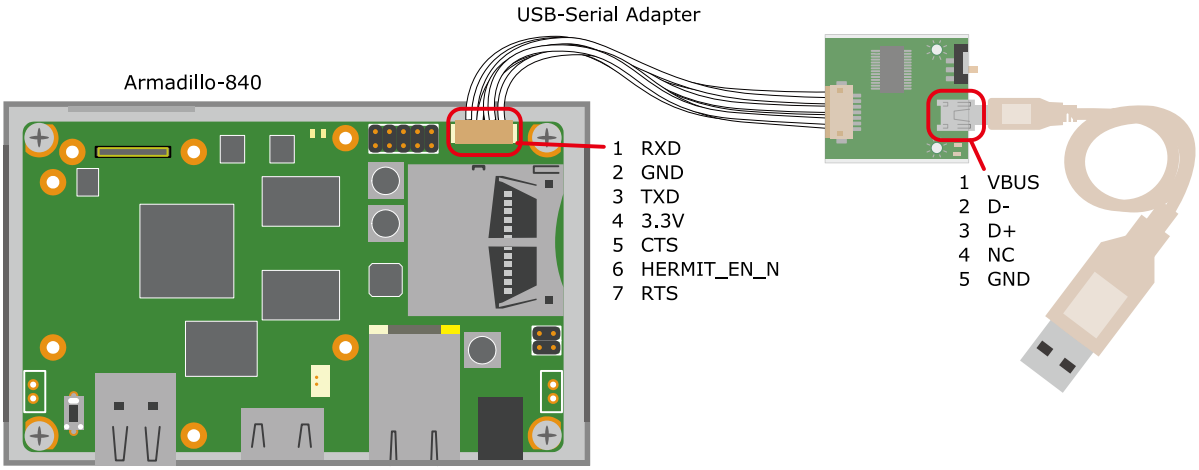
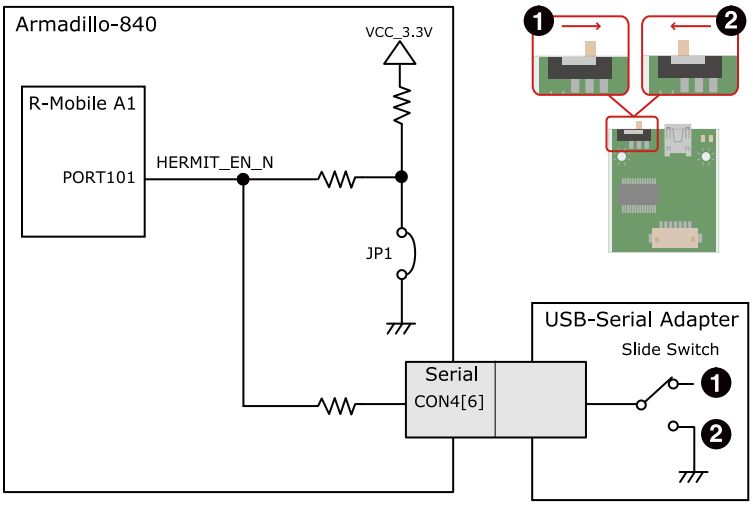


図 20.19 開発用 USB シリアル変換アダプタの配線

スライドスイッチが実装されており、JP1 がオープンの場合に Armadillo-840 の起動モードを設定することができます。



❶　OS 自動起動モード

❷　保守モード

図 20.20 スライドスイッチについて

開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)の取扱い上の注意

USB シリアル変換アダプタには電源投入順序があります。Armadillo-840 に接続する際は、以下の手順に従ってご使用ください。接続手順に従わない場合は、USB シリアル変換アダプタが故障する可能性がありますのでご注意ください。

1. 起動中の作業用 PC と USB シリアル変換アダプタを USB2.0 ケーブルで接続します。
2. Armadillo-840 のシリアルインターフェース(CON4)に USB シリアル変換アダプタを接続します。
3. 上記接続を確認後、Armadillo-840 に電源を投入します。

また、Armadillo-840 に USB シリアル変換アダプタを接続した状態のまま、作業用 PC または USB シリアル変換アダプタから USB2.0 ケーブルを抜く場合や作業用 PC をシャットダウンする場合は、Armadillo-840 の電源が切断されていることを確認してから行ってください。

20.4. 8 ピン JTAG 変換ケーブル

8 ピン JTAG 変換ケーブルは JTAG インターフェースを ARM 標準コネクタ(20 ピン、2.54mm ピッチ)に変換するケーブルです。

8 ピン JTAG 変換ケーブルの接続図、参考回路を以下に示します。

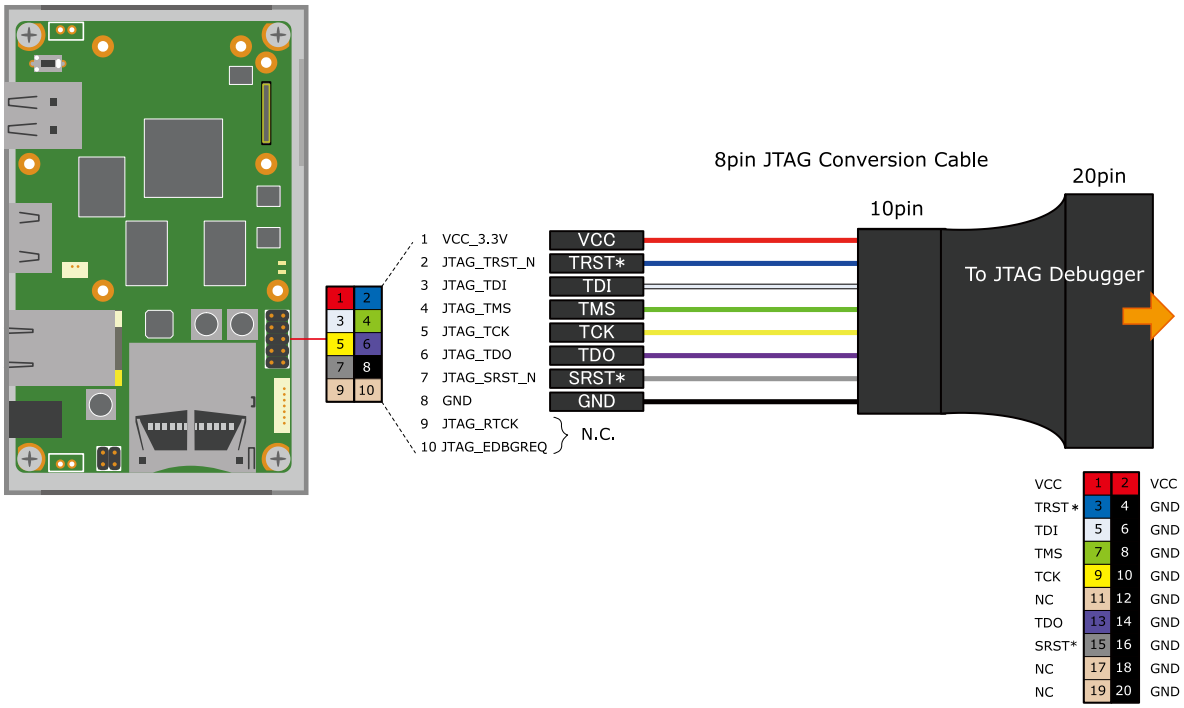


図 20.21 8 ピン JTAG 変換ケーブルの接続図

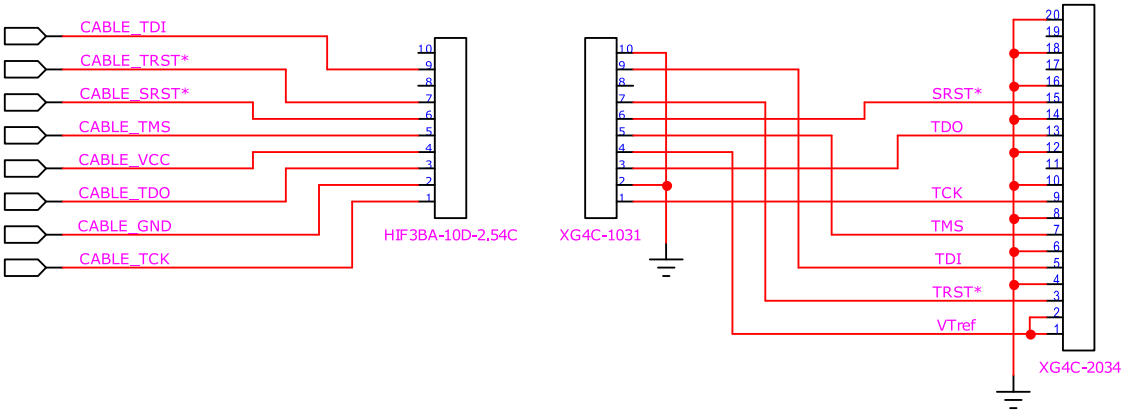


図 20.22 8 ピン JTAG 変換ケーブルの参考回路

21. Howto

本章では、Armadillo-840 のソフトウェアをカスタマイズをする方法や、様々な機能を使用する方法などについて説明します。

21.1. 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 21.1 イメージをカスタマイズ

1. **ソースコードの準備**

   ソースコードを準備します。Atmark Dist と Linux カーネルのソースコードアーカイブを準備し展開します。展開後、Atmark Dist に Linux カーネルのソースコードを登録するために、シンボリックリンクを作成します。

```
[ATDE ~]$ ls
atmark-dist.tar.gz  linux-3.4-at.tar.gz
[ATDE ~]$ tar zxf atmark-dist.tar.gz
[ATDE ~]$ tar zxf linux-3.4-at.tar.gz
[ATDE ~]$ ls
atmark-dist  atmark-dist.tar.gz  linux-3.4-at  linux-3.4-at.tar.gz
[ATDE ~]$ ln -s ../linux-3.4-at atmark-dist/linux-3.x ❶
```

   ❶ シンボリックリンク名は常に linux-3.x である必要があります

2. **コンフィギュレーションの開始**

   make menuconfig を行い「Vendor/Product Selection --->」を選択します。

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

```
 atmark-dist v1.32.0 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 >
```

3. **ベンダー/プロダクト名の選択**

Vendor には "AtmarkTechno" を選択し、AtmarkTechno Products には "Armadillo-840" を選択します。その後、前のメニューに戻るために"Exit"を選択し、「Kernel/Library/Defaults Selection --->」を選択します。

```
 atmark-dist v1.32.0 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-840) AtmarkTechno Products ❷


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

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

❶ "AtmarkTechno"を選択します

❷ "Armadillo-840"を選択します

4. **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 v1.32.0 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

5. **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/arm 3.4-at4 Kernel Configuration
------------------------------------------------------------------------------
                Linux/arm 3.4-at4 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search. Legend: [*] built-in  [ ] excluded  <M> module  < >
  ------------------------------------------------------------------------

     [*] Patch physical to virtual translations at runtime
         General setup  --->
     [*] Enable loadable module support  --->
     [*] Enable the block layer  --->
         System Type  --->
         Bus support  --->
         Kernel Features  --->
         Boot options  --->
         CPU Power Management  --->
         Floating point emulation  --->
  ------------------------------------------------------------------------

                    <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。

6. 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 v1.32.0 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 >
```

7. **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
```

8. **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-840 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
```

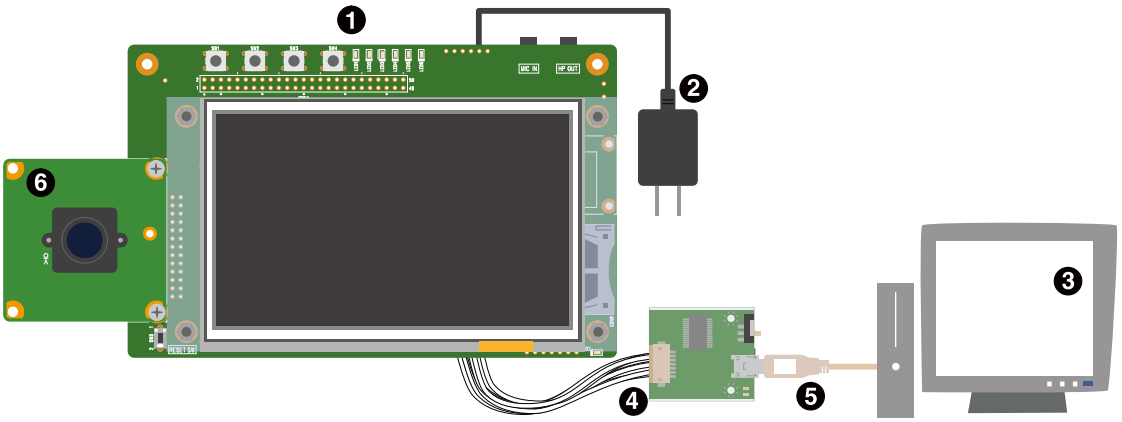
21.2. Armadillo-810 カメラモジュール 01 (B コネクタ用)を使用する 液晶モデル

「Armadillo-810 カメラモジュール 01 (B コネクタ用)」を使用する方法について説明します。以降の説明では、Armadillo-810 カメラモジュール 01 (B コネクタ用)をカメラモジュールと表記します。

カメラモジュールを使用するためには、コンフィギュレーションを変更した Linux カーネルイメージファイルを作成する必要があります。

21.2.1. 接続方法

Armadillo-840 と周辺装置の接続例を次に示します。



1. Armadillo-840 液晶モデル
2. AC アダプタ(5V/2.0A EIAJ#2)[1]
3. 作業用 PC
4. 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[1]
5. USB2.0 ケーブル(A-miniB タイプ)[1]
6. カメラモジュール

21.2.2. ビルド手順

カーネルコンフィギュレーションを変更して、カメラモジュールに対応した Linux カーネルイメージファイルを作成します。ユーザーランドイメージはデフォルトのものが使用できます。

21.2.2.1. カメラモジュール対応イメージのビルド

カーネルコンフィギュレーションを変更して、カメラモジュールを使用可能にします。「21.1. イメージをカスタマイズする」を参照して次のコンフィギュレーションを有効化します。

```
System Type  --->
  Armadillo-840 System Configuration  --->
    [*]   use CEU1 [CLKs, SYNCs and D7-0]
    [*]     CAMERA: KBCR-iC01VG
    [*] use I2C-GPIO3 [SCL:PORT194, SDA:PORT193]
```

カーネルコンフィギュレーションの確定後、make コマンドを実行してイメージファイルを作成します。

21.2.3. フラッシュメモリの書き換え

フラッシュメモリの kernel パーティションを「21.2.2. ビルド手順」で作成した Linux カーネルイメージファイルで書き換えます。フラッシュメモリの書き換え方法については「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。

[1] Armadillo-840 液晶モデル開発セット付属品

21.2.4. 動作確認方法

カメラモジュールから取得した映像を、拡張ボード 01 の LCD に表示させます。GStreamer を利用して映像の取得および表示を行う例を、次に示します。

```
[armadillo ~]# gst-launch-1.0 v4l2src ! videoconvert ! fbdevsink device=/dev/fb1
```

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 v4l2src ! ffmpegcolorspace ! fbdevsink
device=/dev/fb1
```

コマンドの違いは、インストールされている Gstreamer のバージョンによるものです。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では Gstreamer0.10 がインストールされていましたが、ユーザーランドイメージ romfs-a840-v1.02.img 以降(Atmark Dist v20140131 以降)では Gstreamer1.0 がインストールされています。

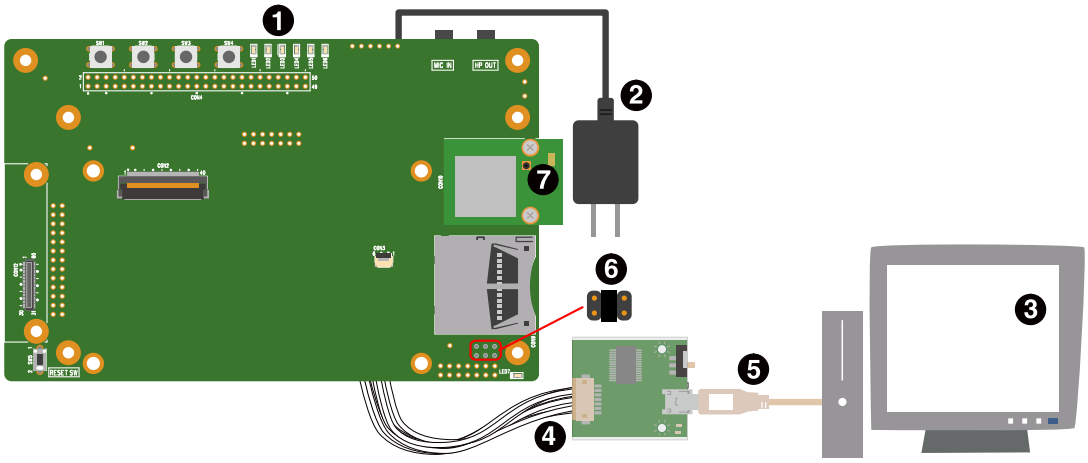
21.3. Armadillo-WLAN モジュール(AWL13)を使用する 液晶モデル

無線通信を行うために、「Armadillo-WLAN モジュール(AWL13)」をインフラストラクチャモード(STA)で使用する方法を説明します。以降の説明では、Armadillo-WLAN モジュール(AWL13)を AWL13 と表記します。

AWL13 を使用するためには、コンフィギュレーションを変更したユーザーランドイメージファイルを作成する必要があります。

21.3.1. 接続方法

Armadillo-840 と周辺装置の接続例を次に示します。



❶ Armadillo-840 液晶モデル[2]

❷ AC アダプタ(5V/2.0A EIAJ#2)[3]

❸ 作業用 PC

❹ 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[3]

❺ USB2.0 ケーブル(A-miniB タイプ)[3]

❻ ジャンパソケット[3][4]

❼ AWL13

21.3.2. ビルド手順

ユーザーランドコンフィギュレーションを変更して、AWL13 に対応したユーザーランドイメージファイルを作成します。カーネルコンフィギュレーションは変更しません。

21.3.2.1. ソースコードの準備

ソースコードを準備します。AWL13 に対応したユーザーランドイメージファイルを作成するためには、Atmark Dist と Linux カーネルの他に、AWL13 ドライバーのソースコードが必要です。Atmark Dist に、Linux カーネルと AWL13 ドライバーのソースコードを登録するために、シンボリックリンクを作成します。

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

[2]AWL13 を接続するために、タッチパネル LCD を取り外します。AWL13 接続後にタッチパネル LCD を接続しても問題ありません。
[3]Armadillo-840 液晶モデル開発セット付属品
[4]AWL13 を使用するために、拡張ボード 01 の JP2 をショート、JP3 をオープンに設定します。

```
[ATDE ~]$ ln -s ../linux-3.4-at atmark-dist/linux-3.x ❶
[ATDE ~]$ ln -s ../awl13 atmark-dist/awl13 ❷
```

❶ シンボリックリンク名は常に linux-3.x である必要があります

❷ シンボリックリンク名は常に awl13 である必要があります

21.3.2.2. AWL13 対応イメージのビルド

ユーザーランドコンフィギュレーションを変更して、AWL13 サポートを有効化します。「21.1. イメージをカスタマイズする」を参照して次のようにコンフィギュレーションを変更します。

```
Vendor specific  --->
  [*] Armadillo-WLAN
  (AWL13) Armadillo-WLAN Products
  (SDIO) AWL13 Support interface
  (STA) AWL13 Support mode
```

ユーザーランドコンフィギュレーションの確定後、make コマンドを実行してイメージファイルを作成します。

21.3.3. フラッシュメモリの書き換え

フラッシュメモリの userland および kernel パーティションを「21.3.2. ビルド手順」で作成したイメージファイルで書き換えます。フラッシュメモリの書き換え方法については「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。

21.3.4. 無線設定

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。これで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ます。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の詳細は、「6.1. ネットワーク」を参照してください。

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。

「21.3.2. ビルド手順」で作成したユーザーランドイメージは、システム起動時に AWL13 が適切に初期化されるように作られています。通常、AWL13 を動作させる場合は、

1. カーネルモジュール「awl13_sdio.ko」の組み込み

2. AWL13 へファームウェアをロード
3. AWL13 の無線設定

の 3 つの手順が必要となりますが、1. 及び 2. を起動スクリプトで実行するため、無線設定のみを行うことで無線通信が可能になります。

21.3.4.1. 無線設定の保存

無線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無線設定を行う必要が無くなります。コンフィグ領域を保存する方法については「7.2. コンフィグ領域の保存」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/etc/config/interfaces の編集例を次に示します。

```
[armadillo ~]# vi /etc/config/interfaces
iface usb0 inet manual
      up ifconfig usb0 up
      post-up zcip usb0 /etc/zcip.script > /dev/null
      down ifconfig usb0 down
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

/etc/config/interfaces の編集後、次のようにコマンドを実行すると、無線設定、IP 設定およびネットワークの有効化が行われます。

```
[armadillo ~]# ifup awlan0
```

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 ~]# /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
```

21.3.5. 動作確認方法

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。次の例では、通信機器が「192.168.10.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping 192.168.10.20
```

事前に「無線設定」、「IP 設定」、「ネットワークの有効化」が行われている必要があります。「21.3.4. 無線設定」および「6.1. ネットワーク」を参照し、行ってください。

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、ネットワーク通信時に 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、「6.1.2. ネットワークの有効化、無効化」を参照して 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。

21.4. USB ガジェットを使用する 液晶モデル

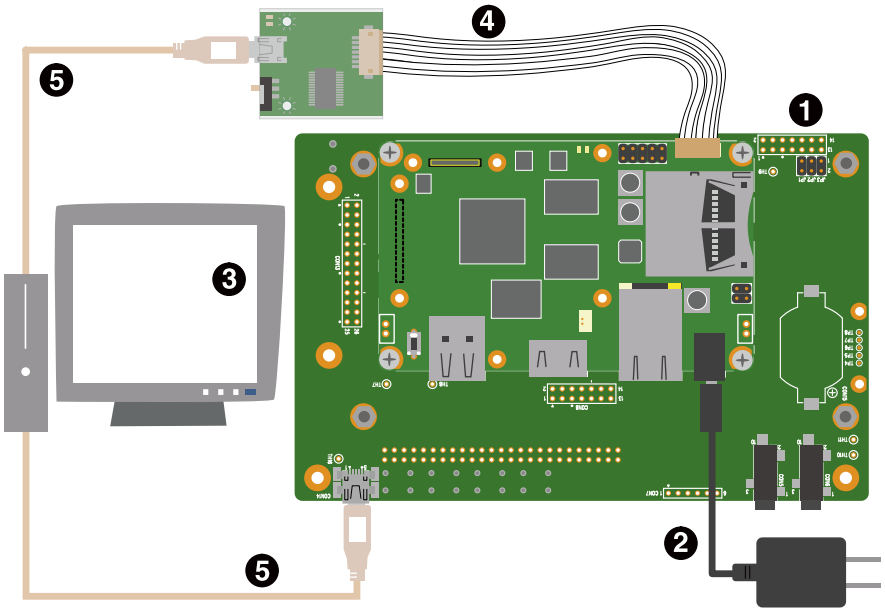
USB ガジェットを使用する方法について説明します。USB デバイス機能を提供する USB ガジェットドライバーには様々な種類のものが用意されていますが、ここでは例としてイーサネットガジェットドライバー[5]を有効化した Armadillo-840 を USB デバイスとして使用する方法を紹介します。

イーサネットガジェットを使用するためには、コンフィギュレーションを変更した Linux カーネルイメージファイルを作成する必要があります。

21.4.1. 接続方法

Armadillo-840 と周辺装置の接続例を次に示します。

[5]イーサネット機能を提供する USB ガジェットドライバー



❶ Armadillo-840 液晶モデル

❷ AC アダプタ(5V/2.0A EIAJ#2)[6]

❸ 作業用 PC

❹ 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[6]

❺ USB2.0 ケーブル(A-miniB タイプ)[6]

21.4.2. ビルド手順

カーネルコンフィギュレーションを変更して、イーサネットガジェットに対応した Linux カーネルイメージファイルを作成します。ユーザーランドイメージはデフォルトのものが使用できます。

21.4.2.1. イーサネットガジェット対応イメージのビルド

カーネルコンフィギュレーションを変更して、イーサネットガジェットを使用可能にします。「21.1. イメージをカスタマイズする」を参照して次のようにコンフィギュレーションを変更します。

```
System Type  --->
  Armadillo-840 System Configuration  --->
    USB1 selection (CON7 - Device)  --->
      ( ) CON5 - Host
      (X) CON7 - Device
Device Drivers  --->
  [*] USB support  --->
    <*>   USB Gadget Support  --->
      <*>   USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support))  --->
        ( ) Audio Gadget (EXPERIMENTAL)
        (X) Ethernet Gadget (with CDC Ethernet support)
        ( ) Network Control Model (NCM) support
      [*]       RNDIS support
      [ ]       Ethernet Emulation Model (EEM) support
```

[6]Armadillo-840 液晶モデル開発セット付属品

カーネルコンフィギュレーションの確定後、make コマンドを実行してイメージファイルを作成します。

21.4.3. フラッシュメモリの書き換え

フラッシュメモリの kernel パーティションを「21.4.2. ビルド手順」で作成した Linux カーネルイメージファイルで書き換えます。フラッシュメモリの書き換え方法については「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。

21.4.4. 動作確認方法

ATDE5 と Armadillo-840 で、イーサネットガジェットを使用した PING 通信を行います。

事前に「4.2.2. 取り外し可能デバイスの使用」を参照して、ATDE5 に Armadillo-840 を接続する必要があります。Armadillo-840 のデバイス名は「Netchip RNDIS/Ethernet Gadget」と表示されます。

接続が完了すると、ATDE5 では自動的にネットワークインターフェース"usb0"が有効化されます。有効化が完了し、リンクローカルアドレス[7]が割り当てられると、"ifconfig"コマンドの出力結果は次のように表示されます。

```
[ATDE ~]$ LANG=C sudo ifconfig
(省略)
usb0      Link encap:Ethernet  HWaddr a2:e4:92:6f:c6:54  ❶
          inet6 addr: fe80::a0e4:92ff:fe6f:c654/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

usb0:avahi Link encap:Ethernet  HWaddr a2:e4:92:6f:c6:54  ❷
          inet addr:169.254.9.18  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

❶　イーサネットガジェットを認識すると"usb0"が表示されます

❷　リンクローカルアドレスが割り当てられると"usb0:avahi"が表示されます

Armadillo-840 では、手動でネットワークインターフェース"usb0"を有効化する必要があります。有効化が完了し、リンクローカルアドレスが割り当てられると、"ifconfig"コマンドの出力結果は次のように表示されます。

```
[armadillo ~]# ifup usb0
[armadillo ~]# ifconfig
(省略)
usb0      Link encap:Ethernet  HWaddr F2:13:6A:0F:7D:24  ❶
          inet addr:169.254.134.176  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1827 (1.7 KiB)
```

[7]IPv4LL という機構によって割り当てられる特定のアドレス範囲(169.254.0.1～169.254.255.254)の IP アドレス。

❶ ネットワークインターフェースが有効化されると"usb0"が表示されます

Armadillo の起動時に自動的に usb0 が有効化されるようにするには、/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 manual
      up ifconfig usb0 up
      post-up zcip usb0 /etc/zcip.script > /dev/null
      down ifconfig usb0 down
```

❶ "usb0"を追加します

編集後、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-840 から ATDE5 に、イーサネットガジェットを使用した PING 通信を行う例を次に示します。

```
[armadillo ~]# ping 169.254.9.18
```

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、ネットワーク通信時に 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、「6.1.2. ネットワークの有効化、無効化」を参照して 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。

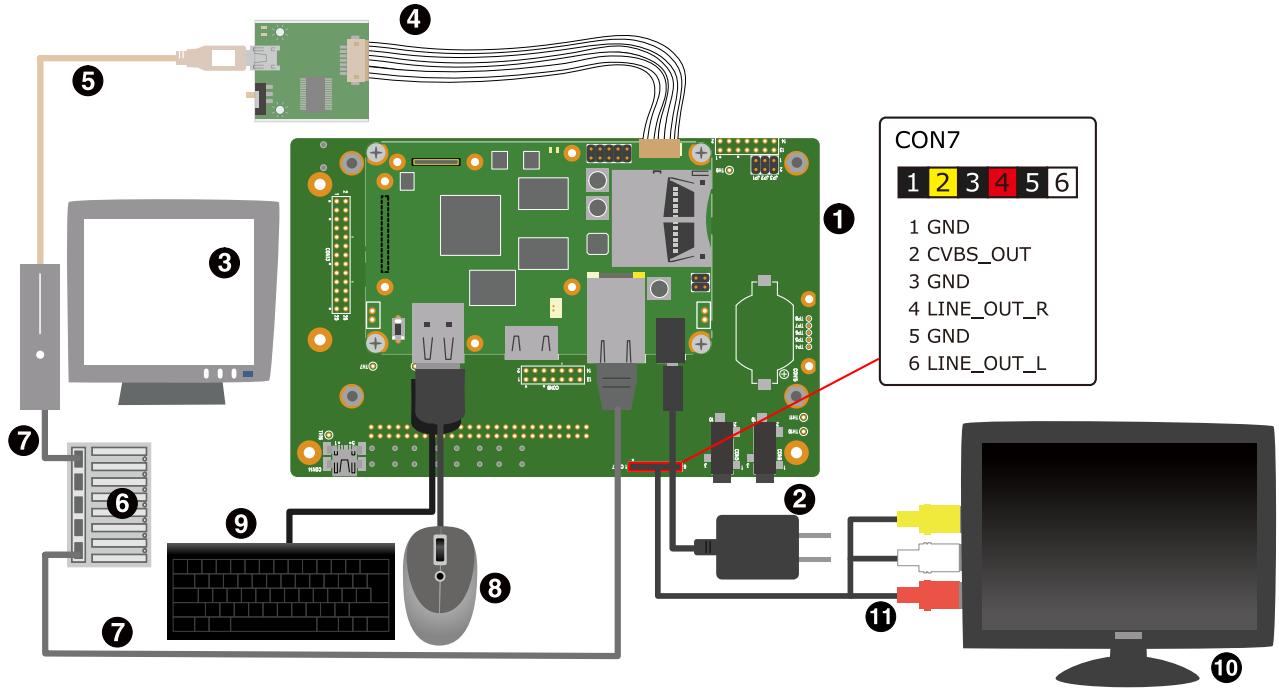
21.5. コンポジットビデオ/ライン出力を使用する 液晶モデル

コンポジットビデオ出力およびライン出力を使用する方法について説明します。

コンポジットビデオ出力を使用するためには、コンフィギュレーションを変更した Linux カーネルイメージファイルを作成する必要があります。ライン出力は、デフォルトの Linux カーネルイメージでも使用することができます。

21.5.1. 接続方法

Armadillo-840 と周辺装置の接続例を次に示します。



❶ Armadillo-840 液晶モデル
❷ AC アダプタ(5V/2.0A EIAJ#2)[8]
❸ 作業用 PC
❹ 開発用 USB シリアル変換アダプタ(Armadillo-800 シリーズ対応)[8]
❺ USB2.0 ケーブル(A-miniB タイプ)[8]
❻ LAN HUB
❼ LAN ケーブル
❽ USB マウス
❾ USB キーボード
❿ RCA 対応ディスプレイ
⓫ RCA ケーブル

21.5.2. ビルド手順

カーネルコンフィギュレーションを変更して、コンポジットビデオ出力に対応した Linux カーネルイメージファイルを作成します。ユーザーランドイメージはデフォルトのものが使用できます。

21.5.2.1. コンポジットビデオ出力対応イメージのビルド

カーネルコンフィギュレーションを変更して、コンポジットビデオ出力を使用可能にします。「21.1. イメージをカスタマイズする」を参照して次のようにコンフィギュレーションを変更します。

```
System Type  --->
  Armadillo-840 System Configuration  --->
```

[8]Armadillo-840 液晶モデル開発セット付属品

```
    [*] use SDENC
    [*]   have Power-Save [PSAVE_B:PORT165]
Device Drivers  --->
  Graphics support  --->
    <*> Support for frame buffer devices  --->
      <*>     SuperH Mobile SDENC controller support
```

カーネルコンフィギュレーションの確定後、make コマンドを実行してイメージファイルを作成します。

21.5.3. フラッシュメモリの書き換え

フラッシュメモリの kernel パーティションを「21.5.2. ビルド手順」で作成した Linux カーネルイメージファイルで書き換えます。フラッシュメモリの書き換え方法については「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。

21.5.4. 動作確認方法

21.5.4.1. コンポジットビデオ出力

「6.2.2. HDMI - フレームバッファデバイス /dev/fb0」で紹介されている Photo Viewer というデモアプリケーションを、RCA 対応ディスプレイに表示します。

「21.5.1. 接続方法」のように Armadillo-840 に HDMI 対応ディスプレイを接続せずに電源を入れると、Photo Viewer が RCA 対応ディスプレイに表示されます。

21.5.4.2. ライン出力

RCA 対応ディスプレイへのサウンドの再生を行います。GStreamer を利用してテストサウンドの出力を行う例を、次に示します。

```
[armadillo ~]# gst-launch-1.0 audiotestsrc ! 'audio/x-raw,channels=2,width=16' ! alsasink
device=hw:1
```

ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では、次のようにコマンドを実行する必要があります。

```
[armadillo ~]# gst-launch-0.10 audiotestsrc ! 'audio/x-raw-
int,channels=2,width=16' ! alsasink device=hw:1
```

コマンドの違いは、インストールされている Gstreamer のバージョンによるものです。ユーザーランドイメージ romfs-a840-v1.01.img 以前(Atmark Dist v20131018 以前)では Gstreamer0.10 がインストールされていましたが、ユーザーランドイメージ romfs-a840-v1.02.img 以降(Atmark Dist v20140131 以降)では Gstreamer1.0 がインストールされています。

22. 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/

22.1. 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-840 購入製品登録

https://users.atmark-techno.com/armadillo-840/register

Armadillo-840 の購入製品登録を行うには、Armadillo-840 から取り出した「正規認証ファイル」をアットマークテクノ ユーザーズサイトからアップロードする必要があります。Armadillo-840 から正規認証ファイル(board-info.txt)を取り出す手順は以下の通りです。

22.1.1. 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-840 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
```

[1] 拡張ボード 01 の回路図データや、ミドルウェアパッケージなど

```
armadillo840-0 login: root
Password:
[root@armadillo840-0 (ttySC2) ~]#
```

2. "get-board-info-a840"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@armadillo840-0 (ttySC2) ~]# get-board-info-a840
[root@armadillo840-0 (ttySC2) ~]# ls
board-info.txt
[root@armadillo840-0 (ttySC2) ~]#
```

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@armadillo840-0 (ttySC2) ~]# mv board-info.txt /home/www-data/
[root@armadillo840-0 (ttySC2) ~]# 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://armadillo840-0.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-840 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 192.168.10.10 の場合、http://192.168.10.10/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	2013/07/12	・初版発行
1.1.0	2013/08/09	・Armadillo-840 液晶モデル開発セットに関する情報追記 ・「1. はじめに」の構成変更 ・「2.1. 製品本体開封についてのご注意」を追記 ・「2.2. 評価ボードについてのご注意」を追記 ・「4.2.1.3. ATDE5 アーカイブの展開」を追記 ・「21. Howto」を追記 ・「8. Linux カーネル仕様」の内容拡充 ・「6. 動作確認方法」の内容拡充 ・横浜営業所の住所追記 ・誤記修正 ・表記ゆれ修正
1.2.0	2014/01/29	・AV コーデックミドルウェアに対応 ・「表 3.1. 仕様」に RAM とフラッシュメモリの型番を追加 ・「表 6.14. direction の設定」の内容を修正 ・「18.2. インターフェース仕様」に搭載コネクタの許容電流を追加 ・「20.1.4. インターフェース仕様」に搭載コネクタの許容電流を追加 ・「14.1. ライセンス」の内容明確化 ・誤記、表記ゆれ修正
1.3.0	2014/07/31	・「表 20.3. 搭載コネクタ、スイッチ型番一覧」、「20.1.4.10. CON10 WLAN インターフェース」に AWL13 接続コネクタの型番を追加 ・誤記修正

Armadillo-840 製品マニュアル
Version 1.3.0
2014/07/31

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