Armadillo-610 製品マニュアル

A6100-U00Z
A6100-D00Z

Version 1.5.2
2022/04/26
Debian GNU/Linux 9 (stretch) 対応

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

Armadillo サイト [https://armadillo.atmark-techno.com]

Armadillo-610 製品マニュアル

株式会社アットマークテクノ

Version 1.5.2
2022/04/26

目次

図目次

表目次

1. はじめに

Armadillo-610 をご利用いただき、ありがとうございます。

Armadillo-610 は Armadillo-640 の性能と品質をそのままユーザーオリジナルの拡張ボードとの接続を可能にした、50mm×50mm の小型コンピューターモジュールです。

Ethernet PHY や microSD カードコネクタは既にボード上に搭載しているため、簡単な外付け部品で自由に拡張ボードを設計していただけます。

有線 LAN 通信時でも約 1W(モジュール部分のみ)で動作し、スリープ時には 0.1W 以下の低消費電力で設計しているためバッテリー駆動を想定した製品開発にも最適です。

以降、本書では他の Armadillo ブランド製品にも共通する記述については、製品名を Armadillo と表記します。

1.1. 本書で扱うこと扱わないこと

1.1.1. 扱うこと

本書では、Armadillo-610 の使い方、製品仕様(ソフトウェアおよびハードウェア)、オリジナルの製品を開発するために必要となる情報、その他注意事項について記載しています。Linux あるいは組み込み機器に不慣れな方でも読み進められるよう、コマンドの実行例なども記載しています。

また、本書では、アットマークテクノが運営する Armadillo サイトをはじめ、開発に有用な情報を得る方法についても、随時説明しています。

1.1.2. 扱わないこと

本書では、一般的な Linux のプログラミング、デバッグ方法やツールの扱い方、各種モジュールの詳細仕様など、一般的な情報や、他に詳しい情報があるものは扱いません。また、(Armadillo-610 を使用した)最終製品あるいはサービスに固有な情報や知識も含まれていません。

1.2. 本書で必要となる知識と想定する読者

本書は、読者として Armadillo-610 を使ってオリジナルの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-610 を使うと、どのようなことが実現可能なのか」を知りたいと考えている設計者・企画者も対象としています。Armadillo-610 は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

ソフトウェアエンジニア	端末からのコマンドの実行方法など、基本的な Linux の扱い方を知っているエンジニアを対象読者として想定しています。プログラミング言語として C/C++を扱えることは必ずしも必要ではありませんが、基礎的な知識がある方が理解しやすい部分もあります。
ハードウェアエンジニア	電子工学の基礎知識を有したエンジニアを対象読者として想定しています。回路図や部品表を読み、理解できる必要があります。

1.3. ユーザー限定コンテンツ

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

- 各種信頼性試験データ・納入仕様書等製造関連情報
- Armadillo-610 拡張ボード、一部のオプション品の回路図

限定コンテンツを取得するには、「21. 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-610 ドキュメントダウンロード

https://armadillo.atmark-techno.com/armadillo-610/resources/documents

Armadillo サイト - Armadillo-610 ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-610/resources/software

1.5. 本書の構成

本書には、Armadillo-610 をベースに、オリジナルの製品を開発するために必要となる情報を記載しています。また、取扱いに注意が必要な事柄についても説明しています。

- **はじめにお読みください。**

  「1. はじめに」、「2. 注意事項」

- **Armadillo-610 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」

- **工場出荷状態のソフトウェア仕様について紹介します。**

  「7. Linux カーネル仕様」、「8. Debian ユーザーランド仕様」、「9. ブートローダー (U-Boot) 仕様」

- **システム開発に必要な情報を紹介します。**

  「10. ビルド手順」、「11. イメージファイルの書き換え方法」、「12. 開発の基本的な流れ」、「13. i.MX6ULL の電源制御方法」

- **拡張ボードの開発や、ハードウェアをカスタマイズする場合に必要な情報を紹介します。**

  「14. SD ブートの活用」、「15. 電気的仕様」、「16. インターフェース仕様」、「17. 基板形状図」、「18. オプション品」、「19. 設計情報」

- **ソフトウェアのカスタマイズ方法を紹介します。**

  「20. 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。**

  「21. 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 の root ユーザで実行
[PC /]$	作業用 PC の一般ユーザで実行
[ATDE ~/]#	ATDE 上の root ユーザで実行
[ATDE ~/]$	ATDE 上の一般ユーザで実行
[armadillo /]#	Armadillo 上 Linux の root ユーザで実行
[armadillo /]$	Armadillo 上 Linux の一般ユーザで実行
⇒	Armadillo 上 U-Boot 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。適宜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。

用語の説明や補足的な説明を記載します。

1.7. 謝辞

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

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

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

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

2.2. 取扱い上の注意事項

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

破損しやすい箇所　基板間コネクタ、microSD スロットおよびそのカバーやフラットケーブルコネクタは、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(SD [3]、LAN、USB)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。

使用場所の制限　無線機能を搭載した製品は、テレビ・ラジオに近接する場所で使用すると、受信障害を招く恐れがあります。

振動　振動が発生する環境では、Armadillo が動かないよう固定して使用してください。

電波に関する注意事項　2.4GHz 帯の電波を使用する機能(無線 LAN 等)は、自動ドアなどの自動制御電子機器に影響が出る場合、すぐに使用を中止してください。

この無線機(Armadillo-WLAN モジュール(AWL13))は 2.4GHz 帯を使用します。全帯域を使用し、かつ移動体識別装置の帯域が回避可能です。変調方式として DS-SS および OFDM 方式を採用し、想定される与干渉距離は 40m 以下です。

[1]本書を含めた関連マニュアルで改造方法を記載している箇所および、コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2]改造やコネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。
[3]Armadillo-610 の microSD スロットは活線挿抜非対応です

2.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて

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

パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェアに付属する 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。

以下のソフトウェアは、オープンソースソフトウェアではありません。

ボード情報取得ツール(get-board-info)

2.4. 電波障害について

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

Armadillo-610 と Armadillo-610 拡張ボードの組み合わせ(型番: A6100-D00Z)において、VCCI の技術基準に適合することを確認しています。新たに設計開発した拡張ボードを組み合わせる場合は、再び妨害波を測定し、VCCI 協会に届出をする必要があります。

2.5. 無線モジュールの安全規制について

WLAN インターフェース(Armadillo-610 拡張ボード: CON18)に接続可能な無線モジュール「Armadillo-WLAN モジュール(AWL13)」は、電波法に基づく工事設計認証を受けています。

これらの無線モジュールを国内で使用するときに無線局の免許は必要ありません。

以下の事項を行うと法律により罰せられることがあります。

- 無線モジュールやアンテナを分解/改造すること。
- 無線モジュールや筐体、基板等に直接印刷されている証明マーク・証明番号、または貼られている証明ラベルをはがす、消す、上からラベルを貼るなどし、見えない状態にすること。

認証番号は次のとおりです。

表 2.1 Armadillo-WLAN モジュール(AWL13) 適合証明情報

項目	内容
型式又は名称	BP3591
電波法に基づく工事設計認証における認証番号	003WWA100913

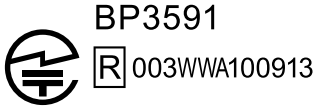


図 2.1 Armadillo-WLAN モジュール(AWL13) 認証マーク

2.6. 保証について

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

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

2.7. 輸出について

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

2.8. 商標について

- Armadillo は株式会社アットマークテクノの登録商標です。その他の記載の商品名および会社名は、各社・各団体の商標または登録商標です。™、®マークは省略しています。

・SD、SDHC、SDXC、microSD、microSDHC、microSDXC、SDIO ロゴは SD-3C, LLC の商標です。

3. 製品概要

3.1. 製品の特長

3.1.1. Armadillo とは

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

- ARM プロセッサ搭載・省電力設計

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

- 小型・手のひらサイズ

  CPU ボードは名刺サイズ程度の手のひらサイズが主流です。名刺の 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-610 とは

- 50mm×50mm の小型モジュール型組み込みプラットフォーム

  Armadillo-610 は、Arm Cortex-A7 コアの SoC「i.MX6ULL」(NXP セミコンダクターズ製)を搭載した、小型 CPU モジュール型の組み込みプラットフォームです。ユーザー自身が新たに拡張ボードを開発して機器に組み込むことを想定しており、設計が煩雑になりがちな CPU 周りは Armadillo-610 をそのまま使い、開発セット用拡張ボードの回路図(購入者向けに無償で提供)を参考にして設計開発が比較的簡単なインターフェース部分だけを拡張設計することにより、設計時間を節約しつつさまざまな形状の基板を実現することができます。

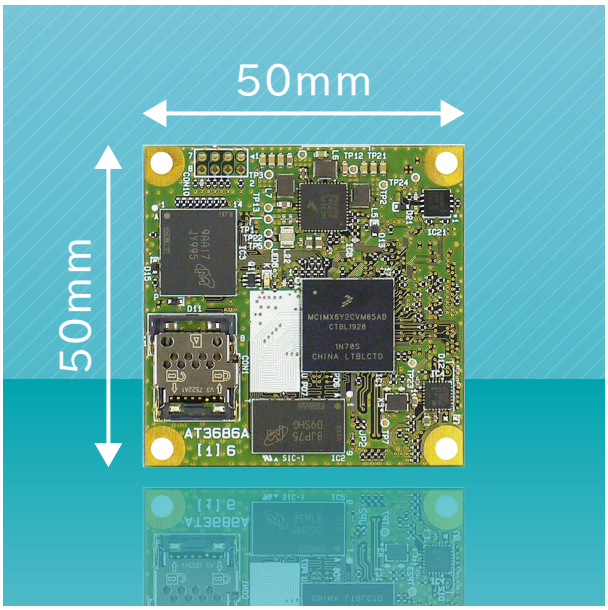


図 3.1 Armadillo-610 とは

- Armadillo-640 と同等の機能を搭載

  Armadillo-610 は、シングルボード型の組み込みプラットフォーム「Armadillo-640」の姉妹製品で、Armadillo-640 と同等のインターフェースを拡張することができます。

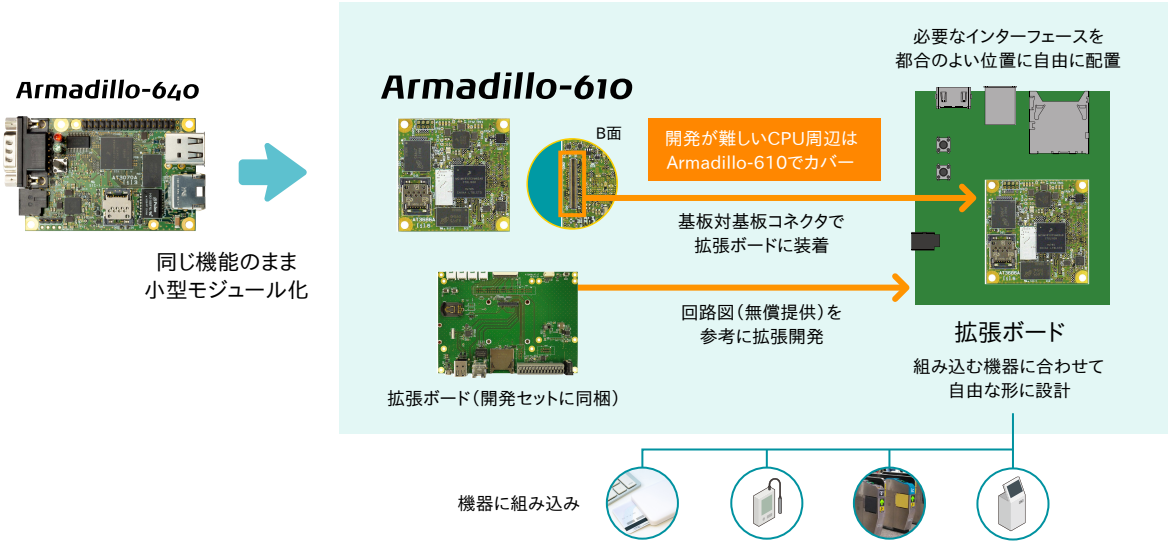


図 3.2 拡張ボードの例

- 耐環境性能に配慮・省電力、産業用途向け

  Armadillo-610 は動作温度範囲-20℃～＋ 70℃までをカバーする産業用途向けの仕様です。また、わずか数 W 程度で動作するので、低消費電力が必須条件となるバッテリ駆動の機器にも適しています。

- Debian GNU/Linux プリインストール

Armadillo-610 は Debian GNU/Linux をプリインストールしており、ユーザーが開発したアプリケーションを書き込んで、さまざまな機能を実現します。

・ Device Tree Blob（DTB）作成ツール「at-dtweb」

使いたいインターフェースをブラウザ上の GUI で選択指定するだけで Device Tree Blob（DTB）を作成できるツールです。Linux カーネルのソースコードを手動で変更することなく GUI で選択したインターフェースを利用できます。

3.2. 製品ラインアップ

Armadillo-610 の製品ラインアップは次のとおりです。

表 3.1 Armadillo-610 ラインアップ

名称	型番
Armadillo-610 量産ボード	A6100-U00Z
Armadillo-610 開発セット	A6100-D00Z

3.2.1. Armadillo-610 開発セット

Armadillo-610 開発セット(型番: A6100-D00Z)は、Armadillo-610 を使った開発がすぐに開始できるように、USB シリアル変換アダプタや AC アダプタなど、開発に必要なものを一式にしています。また、拡張ボードの回路図は製品購入者向けに無償で公開しています。

表 3.2 Armadillo-610 開発セットのセット内容

Armadillo-610
Armadillo-610 拡張ボード
USB(A オス-miniB)ケーブル
USB シリアル変換アダプタ
スピーカー
AC アダプタ(12V/2.0A)
ジャンパソケット
ねじ
スペーサ

3.2.2. Armadillo-610 量産ボード

Armadillo-610 量産ボード(型番: A6100-U00Z)は、量産向けのラインアップです。

3.3. 仕様

Armadillo-610 の主な仕様は次のとおりです。

表 3.3 仕様

プロセッサ	NXP Semiconductors i.MX6ULL
	ARM Cortex-A7 x 1 ・命令/データキャッシュ 32KByte/32KByte ・L2 キャッシュ 128KByte ・内部 SRAM 128KByte ・メディアプロセッシングエンジン(NEON)搭載 ・Thumb code(16bit 命令セット)サポート

システムクロック	CPU コアクロック(ARM Cortex-A7): 528MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz, 24MHz
RAM	DDR3L: 512MByte バス幅: 16bit
ROM	eMMC: 約 3.8GB(約 3.6GiB)
SD	microSD スロット x 1 [a]
拡張インターフェース	LANx1、USBx2、UARTx8、SDx1 [a]、LCDx1、I2Sx2、S/PDIFx1、MQSx1、I2Cx2、SPIx4、CANx2、A/Dx7、PWMx8、GPIOx66 等 [b]
カレンダー時計	SoC 内蔵リアルタイムクロック [c]
LED	黄色(面実装) x 1
電源電圧	DC 3.6〜4.5V
消費電力(参考値)	約 0.8W(待機時)、約 1W(LAN 通信時) [d]
使用温度範囲	-20〜+70℃(結露なきこと)
外形サイズ	50×50mm

[a]microSD スロットと拡張インターフェース側の SD は排他利用となります。

[b]i.MX6ULL のピンマルチプクレスの設定で、優先的に機能を割り当てた場合に拡張可能な最大数を記載しています。

[c]電池は付属していません。

[d]Armadillo-610 拡張ボードとの組み合わせで LAN、シリアルコネクタにケーブルを接続した状態での消費電力です。外部接続機器の消費分は含みません。

3.4. ブロック図

Armadillo-610 のブロック図は次のとおりです。

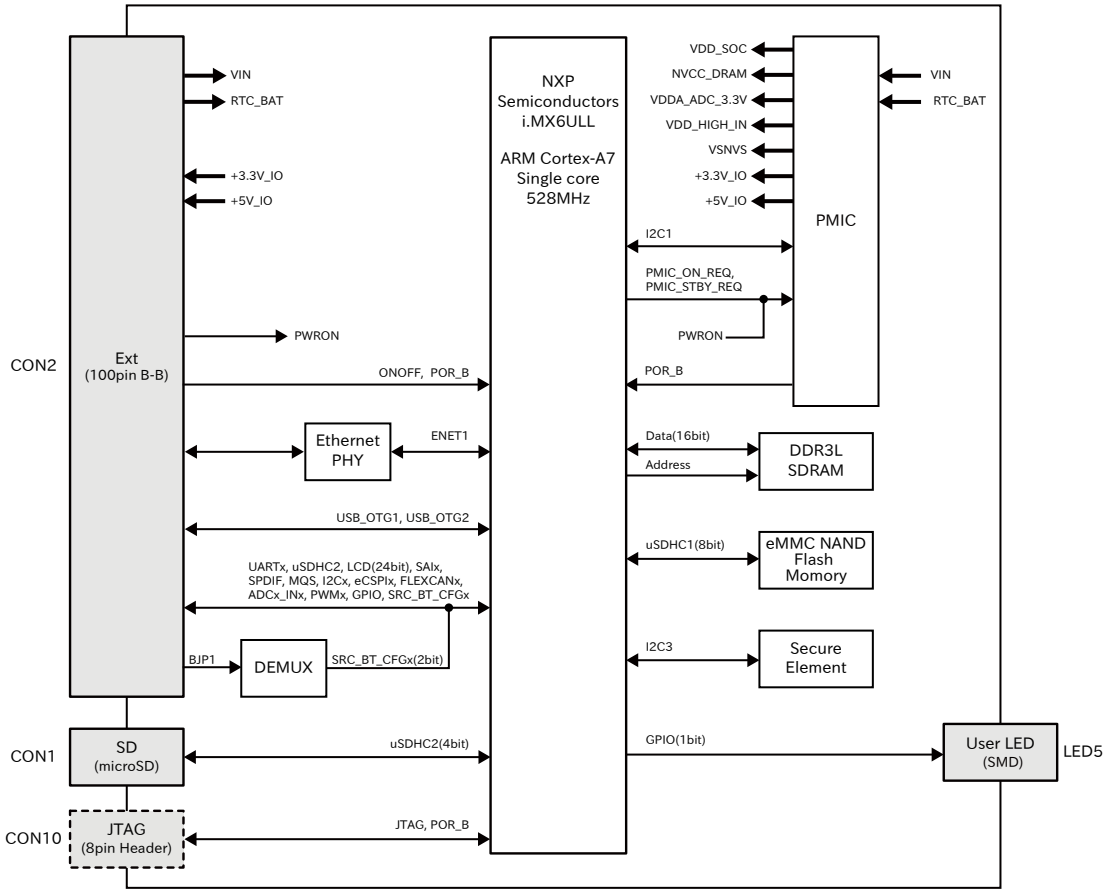


図 3.3 Armadillo-610 ブロック図

3.5. ソフトウェア構成

Armadillo-610 で動作するソフトウェアの構成について説明します。Armadillo-610 で利用可能なソフトウェアを「表 3.4. Armadillo-610 で利用可能なソフトウェア」に示します。

表 3.4 Armadillo-610 で利用可能なソフトウェア

ソフトウェア	説明
U-Boot	ブートローダーです。工場出荷状態ではブートローダーは eMMC に配置されています[a]。microSD カードに配置することもできます。ブートローダーが使う環境変数は常に eMMC に保存されます。
Linux カーネル	uImage 形式の Linux カーネルイメージが利用可能です。 工場出荷状態では Linux カーネルイメージは eMMC に配置されています[a]。ブートローダーの機能により microSD カード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。工場出荷状態では Debian GNU/Linux のルートファイルシステムは eMMC に配置されていますが[a]、Linux カーネルがサポートしている microSD カードなどのストレージデバイスに配置することもできます。

[a]工場出荷状態でソフトウェアが配置されているのは、Armadillo-610 開発セット(A6100-D00Z)のみです。

Armadillo-610 の eMMC のメモリマップを「表 3.5. eMMC メモリマップ」に示します。

表 3.5 eMMC メモリマップ

ディスクデバイス	サイズ	説明
/dev/mmcblk0p1	30.6MByte	予約領域

ディスクデバイス	サイズ	説明
/dev/mmcblk0p2	3.4GByte	Linux カーネルイメージ, Device Tree Blob, Debian GNU/Linux
/dev/mmcblk0p3	122.1MByte	予約領域

Armadillo-610 の eMMC(GPP)のメモリマップを「表 3.6. eMMC(GPP)メモリマップ」に示します。

表 3.6 eMMC(GPP)メモリマップ

ディスクデバイス	サイズ	説明
/dev/mmcblk0gp0	8.389 MByte	ライセンス情報等の保存
/dev/mmcblk0gp1	8.389 MByte	予約領域
/dev/mmcblk0gp2	8.389 MByte	ユーザー領域
/dev/mmcblk0gp3	8.389 MByte	ユーザー領域

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。「開発/動作確認環境の構築」を参照して、作業用 PC 上に開発/動作確認環境を構築してください。
ネットワーク環境	Armadillo と作業用 PC をネットワーク通信ができるようにしてください。
SD カード	SD スロットの動作を確認する場合などに利用します。
microSD カード	microSD スロットの動作を確認する場合などに利用します。
USB メモリ	USB の動作を確認する場合などに利用します。
tar.xz 形式のファイルを展開するソフトウェア	開発/動作確認環境を構築するために利用します。Linux では、tar で展開できます。Windows では、7-Zip や Lhaz などが対応しています。

4.2. 開発/動作確認環境の構築

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

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

- GPL v2(General Public License version 2)で提供されている [1]
- VMware 形式の仮想ディスク(.vmdk)ファイルに対応している

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

ATDE は、バージョンにより対応するアットマークテクノ製品が異なります。本製品に対応している ATDE は、ATDE7 の v20200226 以降です。

ATDE7 は Debian GNU/Linux 9 (コードネーム Stretch) をベースに、Armadillo-610 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-610 の動作確認を行うために必要なツールが事前にインストールされています。

[1]バージョン 3.x までは PUEL(VirtualBox Personal Use and Evaluation License)が適用されている場合があります。

4.2.1. ATDE のセットアップ

4.2.1.1. VMware のインストール

ATDE 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、ATDE のアーカイブは 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 ATDE 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. ATDE のアーカイブを取得

ATDE のアーカイブは Armadillo サイト(http://armadillo.atmark-techno.com)から取得可能です。

本製品に対応している ATDE のバージョンは ATDE7 v20200226 以降です。

作業用 PC の動作環境(ハードウェア、VMware、ATDE の対応アーキテクチャなど)により、ATDE 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. ATDE のアーカイブを展開

ATDE のアーカイブを展開します。ATDE 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「4.2.1.4. Windows で ATDE のアーカイブ展開する」に、Linux での展開方法を手順「4.2.1.5. Linux で tar.xz 形式のファイルを展開する」に示します。

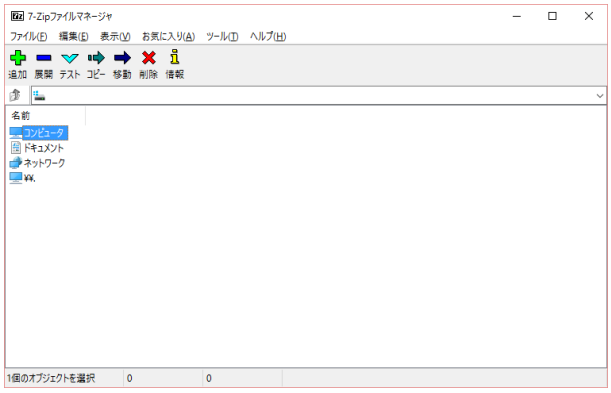
4.2.1.4. Windows で ATDE のアーカイブ展開する

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 のサイト (http://sevenzip.sourceforge.jp)からダウンロード取得可能です。

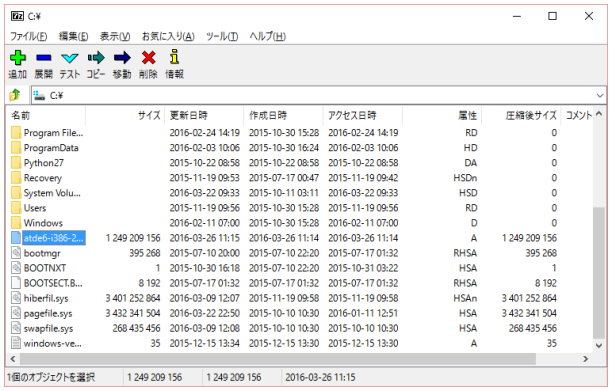
2. 7-Zip の起動

   7-Zip を起動します。



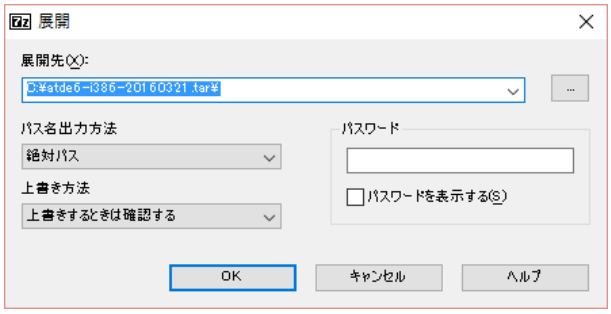
3. xz 圧縮ファイルの選択

   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。

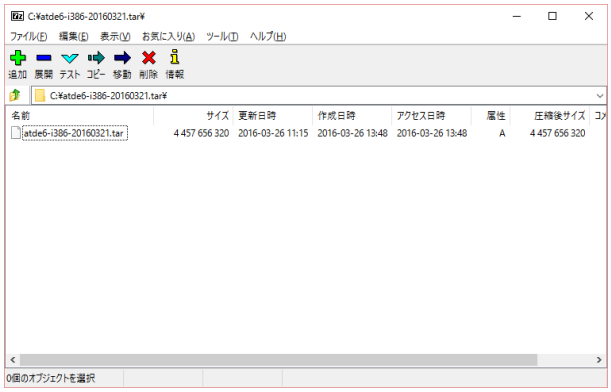


5. xz 圧縮ファイルの展開

   展開が始まります。

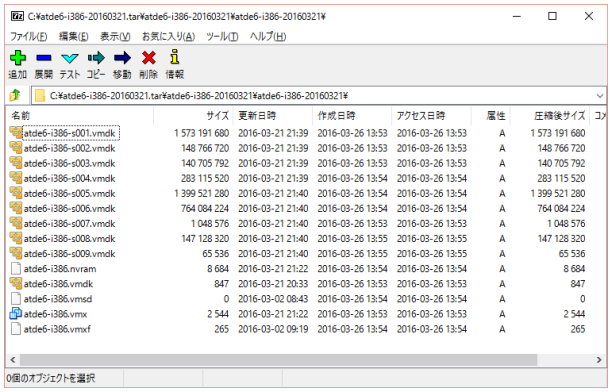
6. tar アーカイブファイルの選択

   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 ATDE 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、ATDE 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 ATDE のデータイメージが出力されています。



4.2.1.5. 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 xf オプションを使用して tar.xz 圧縮ファイルを展開します。

```
[PC ~]$ tar xf atde7-i386-[version].tar.xz
```

2. 展開の完了確認

tar.xz 圧縮ファイルの展開が終了すると、ATDE アーカイブの展開は完了です。 **atde7-i386-[version]** ディレクトリに ATDE のデータイメージが出力されています。

```
[PC ~]$ ls atde7-i386-[version]/
atde7-i386-s001.vmdk  atde7-i386-s009.vmdk  atde7-i386-s017.vmdk
atde7-i386-s002.vmdk  atde7-i386-s010.vmdk  atde7-i386.nvram
atde7-i386-s003.vmdk  atde7-i386-s011.vmdk  atde7-i386.vmdk
atde7-i386-s004.vmdk  atde7-i386-s012.vmdk  atde7-i386.vmsd
atde7-i386-s005.vmdk  atde7-i386-s013.vmdk  atde7-i386.vmx
atde7-i386-s006.vmdk  atde7-i386-s014.vmdk  atde7-i386.vmxf
atde7-i386-s007.vmdk  atde7-i386-s015.vmdk
atde7-i386-s008.vmdk  atde7-i386-s016.vmdk
```

4.2.1.6. ATDE の起動

ATDE 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、ATDE を起動することができます。ATDE7 にログイン可能なユーザーを、「表 4.1. ユーザー名とパスワード」に示します [2]。

表 4.1 ユーザー名とパスワード

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

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

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。

[2]特権ユーザーで GUI ログインを行うことはできません

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-610 の動作確認を行うためには、「表 4.2. 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 4.2 動作確認に使用する取り外し可能デバイス

デバイス	デバイス名
USB シリアル変換アダプタ	Future Devices FT232R USB UART
作業用 PC の物理シリアルポート	シリアルポート

4.2.3. コマンドライン端末(GNOME 端末)の起動

ATDE 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。ATDE 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アクティビティから「terminal」と入力し「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 4.3. 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 4.3 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

1. 「図 4.3. minicom の設定の起動」に示すコマンドを実行し、minicom の設定画面を起動してください。

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

図 4.3 minicom の設定の起動

2. 「図 4.4. minicom の設定」が表示されますので、「Serial port setup」を選択してください。

```
            +-----[configuration]------+
            | Filenames and paths      |
```

```
            | File transfer protocols  |
            | Serial port setup        |
            | Modem and dialing        |
            | Screen and keyboard      |
            | Save setup as dfl        |
            | Save setup as..          |
            | Exit                     |
            | Exit from Minicom        |
            +--------------------------+
```

図 4.4 minicom の設定

3. 「図 4.5. minicom のシリアルポートの設定」が表示されますので、A キーを押して Serial Device を選択してください。

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

図 4.5 minicom のシリアルポートの設定

4. Serial Device に使用するシリアルポートを入力して Enter キーを押してください。

USB to シリアル変換ケーブル使用時のデバイスファイル確認方法

Linux で USB to シリアル変換ケーブルを接続した場合、コンソールに以下のようなログが表示されます。ログが表示されなくても、dmesg コマンドを実行することで、ログを確認することができます。

```
usb 2-1.2: new full-speed USB device number 5 using ehci-pci
usb 2-1.2: New USB device found, idVendor=0403, idProduct=6001
usb 2-1.2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
usb 2-1.2: Product: FT232R USB UART
usb 2-1.2: Manufacturer: FTDI
usb 2-1.2: SerialNumber: A702ZLZ7
usbcore: registered new interface driver usbserial
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial support registered for generic
usbcore: registered new interface driver ftdi_sio
usbserial: USB Serial support registered for FTDI USB Serial
Device
```

```
ftdi_sio 2-1.2:1.0: FTDI USB Serial Device converter detected
usb 2-1.2: Detected FT232RL
usb 2-1.2: Number of endpoints 2
usb 2-1.2: Endpoint 1 MaxPacketSize 64
usb 2-1.2: Endpoint 2 MaxPacketSize 64
usb 2-1.2: Setting MaxPacketSize 64
usb 2-1.2: FTDI USB Serial Device converter now attached to
ttyUSB0
```

図 4.6 例. USB to シリアル変換ケーブル接続時のログ

上記のログから USB to シリアル変換ケーブルが ttyUSB0 に割り当てられたことが分かります。

5. F キーを押して Hardware Flow Control を No に設定してください。
6. G キーを押して Software Flow Control を No に設定してください。
7. キーボードの E キーを押してください。「図 4.7. minicom のシリアルポートのパラメータの設定」が表示されます。

```
+---------[Comm Parameters]----------+
|                                    |
|     Current: 115200 8N1            |
| Speed            Parity      Data  |
| A: <next>        L: None     S: 5  |
| B: <prev>        M: Even     T: 6  |
| C:   9600        N: Odd      U: 7  |
| D:  38400        O: Mark     V: 8  |
| E: 115200        P: Space          |
|                                    |
| Stopbits                           |
| W: 1             Q: 8-N-1          |
| X: 2             R: 7-E-1          |
|                                    |
|                                    |
| Choice, or <Enter> to exit?        |
+------------------------------------+
```

図 4.7 minicom のシリアルポートのパラメータの設定

8. 「図 4.7. minicom のシリアルポートのパラメータの設定」では、転送レート、データ長、ストップビット、パリティの設定を行います。
9. 現在の設定値は「Current」に表示されています。それぞれの値の内容は「図 4.8. minicom シリアルポートの設定値」を参照してください。

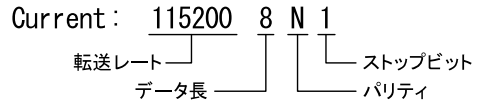


図 4.8 minicom シリアルポートの設定値

10. E キーを押して、転送レートを 115200 に設定してください。
11. Q キーを押して、データ長を 8、パリティを None、ストップビットを 1 に設定してください。
12. Enter キーを 2 回押して、「図 4.4. minicom の設定」に戻ってください。
13.「図 4.4. minicom の設定」から、「Save setup as dfl」を選択し、設定を保存してください。
14.「Exit from Minicom」を選択し、minicom の設定を終了してください。

minicom を起動させるには、「図 4.9. minicom 起動方法」のようにしてください。

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

図 4.9 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 がオープンする /dev/ttyS0 や /dev/ttyUSB0 といったデバイスファイルは、 root または dialout グループに属しているユーザーしかアクセスできません。

ユーザーを dialout グループに入れることで、以降、sudo を使わずに minicom で /dev/ttyUSB0 をオープンすることができます。

```
[ATDE ~]$ sudo usermod -aG dialout atmark
[ATDE ~]$ LANG=C minicom --wrap --device /dev/ttyUSB0
```

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 4.10 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

Armadillo-610 および開発セット付属の拡張ボードインターフェースレイアウトは次のとおりです。各インターフェースの配置場所等を確認してください。

コネクタの挿抜寿命を記載していますが、製品出荷時における目安であり、実際に挿抜可能な回数を保証するものではありません。また、無理な挿抜は接触不良や破損の原因となりますので、取り扱いには十分ご注意ください。

4.3.1. Armadillo-610 インターフェースレイアウト

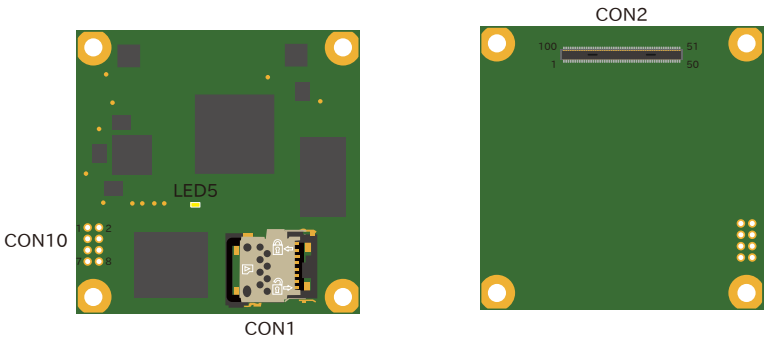


図 4.11 Armadillo-610 インターフェースレイアウト

表 4.4 Armadillo-610 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	microSD スロット(ヒンジタイプ)	
CON2	拡張インターフェース	基板間コネクタ 100 ピン(0.4mm ピッチ)	挿抜寿命: 20 回
CON10	JTAG インターフェース	ピンヘッダ 8 ピン(2mm ピッチ)	コネクタ非搭載
LED5	ユーザー LED	LED(黄色,面実装)	

4.3.2. Armadillo-610 拡張ボード インターフェースレイアウト

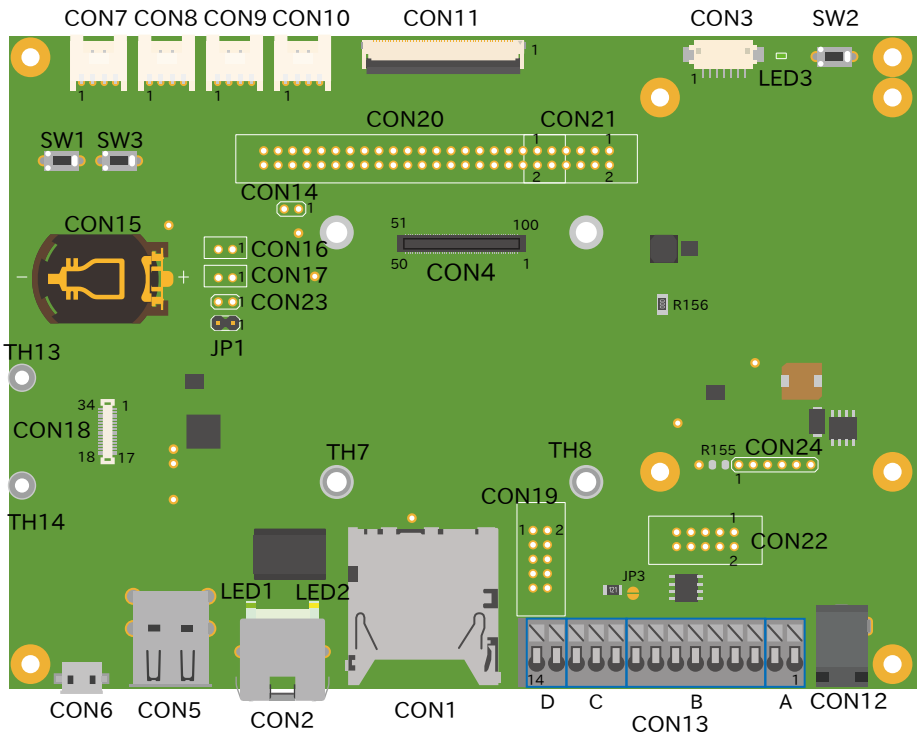


図 4.12 Armadillo-610 拡張ボード インターフェースレイアウト

表 4.5 Armadillo-610 拡張ボード 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	シリアルインターフェース	ピンヘッダ 7 ピン(1.25mm ピッチ)	挿抜寿命: 40 回
CON4	Armadillo-610 インターフェース	基板間コネクタ 100 ピン(0.4mm ピッチ)	挿抜寿命: 20 回
CON5	USB ホストインターフェース	Type-A コネクタ(2 ポートスタック)	
CON6	USB OTG インターフェース	Micro-AB コネクタ	
CON7	Grove インターフェース	ピンヘッダ 4 ピン(2.54mm ピッチ)	
CON8		ピンヘッダ 4 ピン(2.54mm ピッチ)	
CON9		ピンヘッダ 4 ピン(2.54mm ピッチ)	
CON10		ピンヘッダ 4 ピン(2.54mm ピッチ)	
CON11	LCD インターフェース	FFC コネクタ 50 ピン(0.5mm ピッチ)	挿抜寿命: 20 回
CON12	電源入力インターフェース	DC ジャック	対応プラグ: 内径 2.1 外形 5.5mm
CON13 A	電源出力インターフェース	端子台 2 ピン	
CON13 B	DIDO インターフェース	端子台 7 ピン	
CON13 C	RS485 インターフェース	端子台 3 ピン	
CON13 D	オーディオインターフェース	端子台 2 ピン	
CON14	電源出力インターフェース	ピンヘッダ 2 ピン(2.54mm ピッチ)	コネクタ非搭載
CON15	RTC バックアップインターフェース	電池ボックス	対応電池: CR2032
CON16		ピンヘッダ 2 ピン(2.54mm ピッチ)	コネクタ非搭載

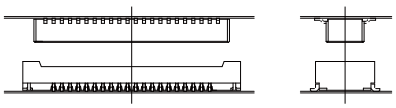
部品番号	インターフェース名	形状	備考
CON17	内蔵 RTC バックアップインターフェース	ピンヘッダ 2 ピン(2.54mm ピッチ)	コネクタ非搭載
CON18	WLAN インターフェース	基板間コネクタ 34 ピン(0.5mm ピッチ)	
CON19	拡張インターフェース	ピンヘッダ 10 ピン(2.54mm ピッチ)	コネクタ非搭載
CON20		ピンヘッダ 40 ピン(2.54mm ピッチ)	コネクタ非搭載
CON21		ピンヘッダ 10 ピン(2.54mm ピッチ)	コネクタ非搭載
CON22		ピンヘッダ 10 ピン(2.54mm ピッチ)	コネクタ非搭載
CON23	リセットインターフェース	ピンヘッダ 2 ピン(2.54mm ピッチ)	コネクタ非搭載
CON24	電源入力インターフェース	ピンヘッダ 6 ピン(2.54mm ピッチ)	コネクタ非搭載
JP1	起動デバイス設定ジャンパ	ピンヘッダ 2 ピン(2.54mm ピッチ)	
SW1	ユーザースイッチ	タクトスイッチ	
SW2	リセットスイッチ	タクトスイッチ	
SW3	ONOFF スイッチ	タクトスイッチ	
LED1	LAN スピード LED	LED(緑色,面実装)	
LED2	LAN リンクアクティビティ LED	LED(黄色,面実装)	
LED3	ユーザー LED	LED(緑色,面実装)	
TH7	Armadillo-610 用スタッド	スペーサー M3 (L=3mm)	
TH8		スペーサー M3 (L=3mm)	
TH13	WLAN 用スタッド	スペーサー M2 (L=1.5mm)	
TH14		スペーサー M2 (L=1.5mm)	

4.4. 組み立て

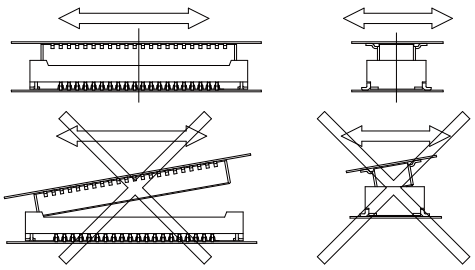
Armadillo-610 拡張ボードの 4 隅のねじ穴は、スペーサー固定用です。Armadillo-610 開発セットに同梱されている M3 のねじとスペーサーを取り付けてください。

コネクタ嵌合時の注意

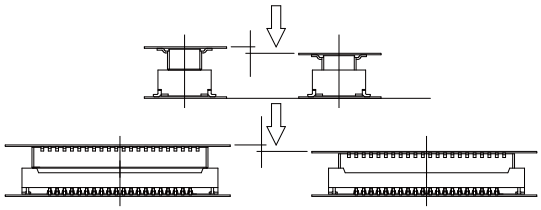
コネクタの中心を合わせて嵌合してください。



位置合わせをする際は、無理な力を加えることなく誘い込み口を探してください。無理な力を加えると、モールドの破損、削れが発生し、接触抵抗の不具合等に繋る場合があります。



コネクタが誘い込まれると、コネクタ間の距離が近くなり、平行になって前後左右に動かなくなります。この状態からまっすぐに嵌合してください。



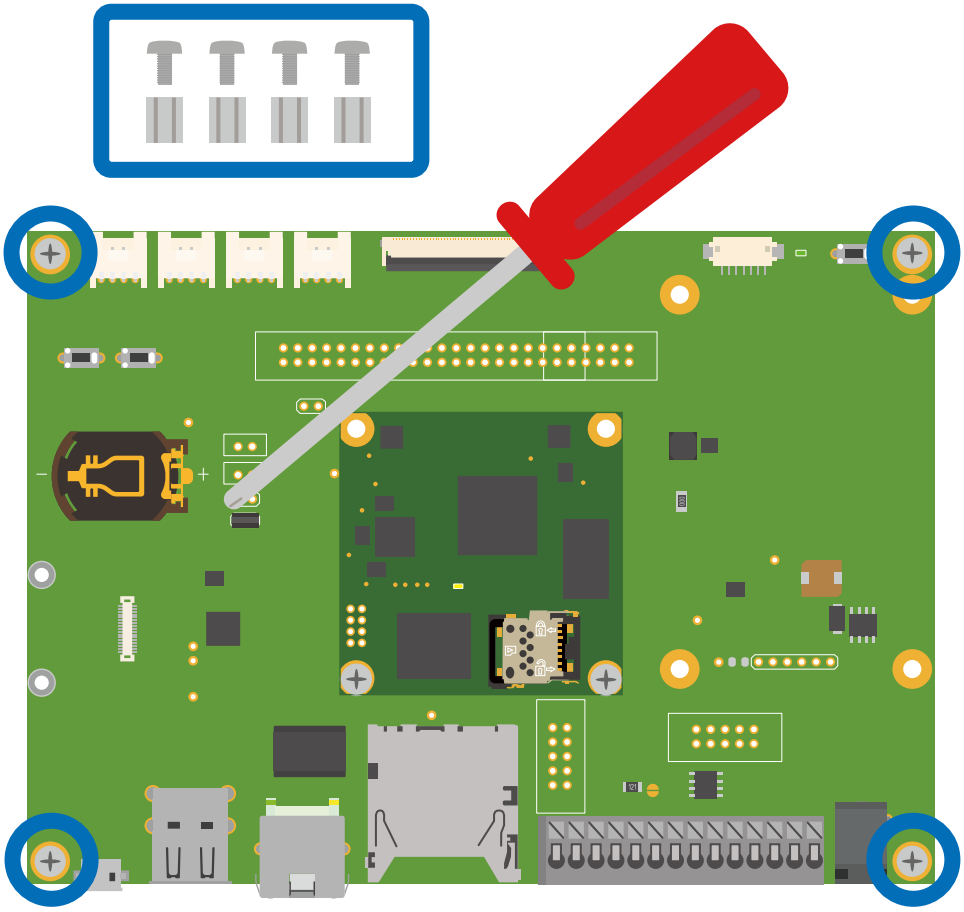
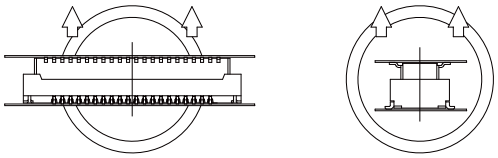


図 4.13 Armadillo-610 開発セットの組み立て

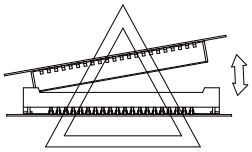
Armadillo-610 開発セットでは、Armadillo-610 は Armadillo-610 拡張ボードに搭載した状態で出荷されます。M2 のねじが同梱されていますが、こちらは Armadillo-WLAN(AWL13)固定用としてご使用ください。

コネクタ抜去時の注意

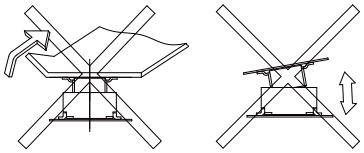
コネクタは平行に抜去してください。



平行に抜去することが困難な場合、コネクタ幅の狭い方向から斜めに抜去してください。



コネクタが損傷する可能性が高いため、コネクタのコーナー方向や幅の広い方向から斜めに抜去しないでください。



4.5. 接続方法

Armadillo-610 開発セットと周辺装置の接続例を次に示します。

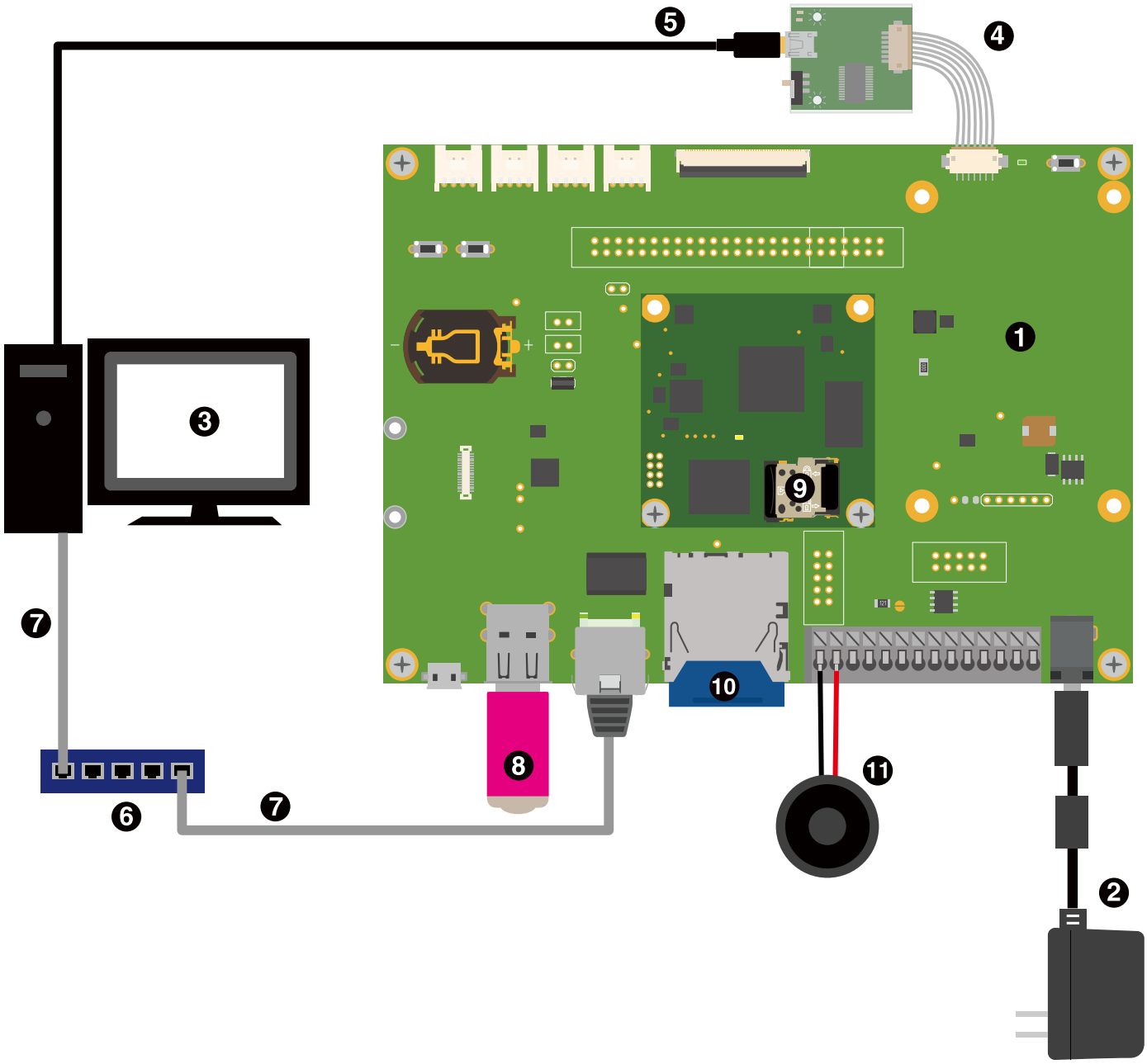


図 4.14 Armadillo-610 開発セットの接続例

1. Armadillo-610 開発セット
2. AC アダプタ(12V/2A)
3. 作業用 PC
4. USB シリアル変換アダプタ
5. USB2.0 ケーブル(A-miniB タイプ)
6. LAN HUB

❼ Ethernet ケーブル

❽ USB メモリ

❾ microSD カード

❿ SD カード

⓫ スピーカー

AC アダプタから電源を供給する際、DC プラグを Armadillo-610 拡張ボードの DC ジャックに接続してから、AC プラグをコンセントに接続してください。突入電流により、故障する可能性があります。

シリアルインターフェース(Armadillo-610 拡張ボード: CON3)に USB シリアル変換アダプタを接続する際は、ケーブルの根本を軽く握り、指先でコネクタを押すようにして挿入してください。取り外しの際は、全ケーブルが均等に引きぬかれるようにケーブルをつかみ、引き抜いてください。また、両コネクタを水平にして挿入・抜去してください。30°以上傾けた状態での斜め挿入・抜去は、端子変形、ケース破損の原因となります。

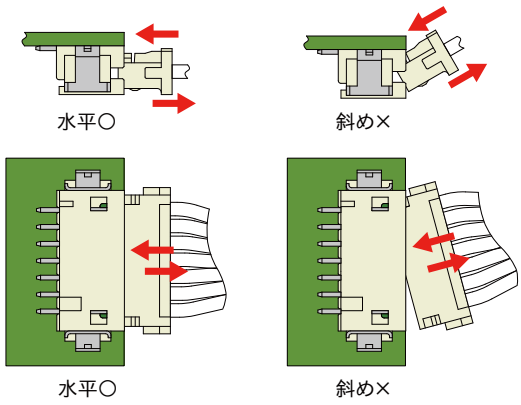


図 4.15 USB シリアル変換アダプタの挿抜角度

Armadillo-610 開発セットに同梱されているスピーカーを直接端子台に取り付ける場合、8～9mm ワイヤーストリッパー等で剥いてください。剥かずに取り付けた場合、音がでない等のトラブルの原因となる可能性があります。

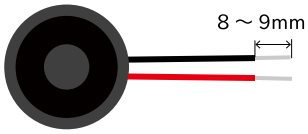


図 4.16 スピーカーのリード線

4.6. 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-610 の動作を変更することができます。ジャンパの機能については「18.2.4.24. JP1(起動デバイス設定ジャンパ)」を参照してください。

ジャンパピンの位置は「図 4.17. JP1 の位置」で確認してください。

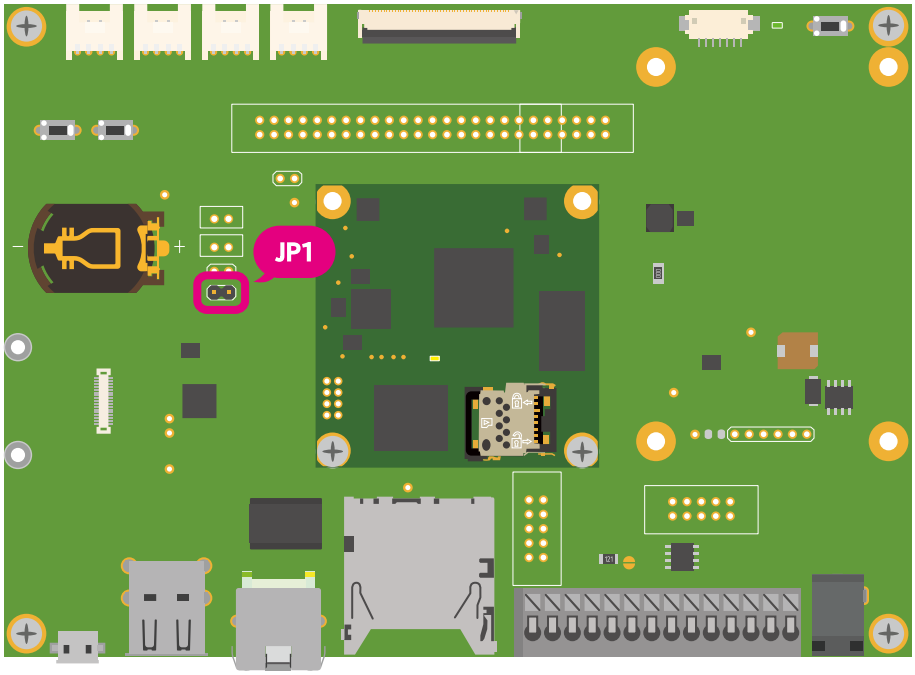


図 4.17 JP1 の位置

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、JP1 をオープンに設定しておきます。

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。

4.7. スライドスイッチの設定について

USB シリアル変換アダプタのスライドスイッチを操作することで、ブートローダーの起動モードを変更することができます。

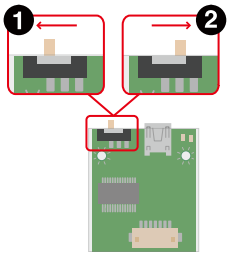


図 4.18 スライドスイッチの設定

❶ ブートローダーは保守モードになります。保守モードでは、ブートローダーのコマンドプロンプトが起動します。

❷ ブートローダーはオートブートモードになります。オートブートモードでは、ブートローダーのコマンドプロンプトが表示されず、OS を自動起動します。

4.8. 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。

4.8.1. 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 4.19 vi の起動

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

4.8.2. 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 4.6. 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 4.6 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 4.20. 入力モードに移行するコマンドの説明」に示します。

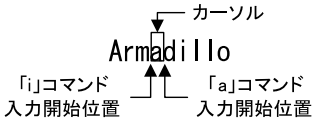


図 4.20 入力モードに移行するコマンドの説明

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、「4.8.4. 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。

4.8.3. 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 4.7. 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 4.7 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

4.8.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.8. 文字の削除コマンド」に示すコマンドを入力します。

表 4.8 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 4.21. 文字を削除するコマンドの説明」に示します。

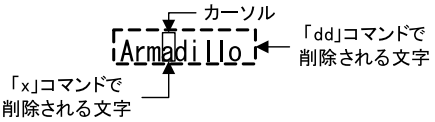


図 4.21 文字を削除するコマンドの説明

4.8.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.9. 保存・終了コマンド」に示します。

表 4.9 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

電源入力インターフェース (Armadillo-610 拡張ボード: CON12) に電源を接続すると、Armadillo-610 が起動します。

USB シリアル変換アダプタのスライドスイッチやユーザースイッチ (Armadillo-610 拡張ボード: SW1)によって起動モードが変わります。 詳しくは 「4.5. 接続方法」 、 「4.7. スライドスイッチの設定について」 、 「9.1. U-Boot の起動モード」 を参照してください。

本章では、保守モードに設定しているときの例を示します。 オートブートモードを選択した場合は、途中でコマンドを入力することなく起動が完了します。

初めて起動した時は、 U-Boot の環境変数が eMMC に書き込まれていないために、次のように Warning が表示されます。

```
U-Boot 2018.03-at8 (Feb 14 2020 - 10:21:57 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

Failed (-5)
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 5.1 電源投入直後のログ (U-Boot の環境変数が eMMC に無い場合)

env save すると、次の起動から表示されなくなります。 詳しくは 「9. ブートローダー (U-Boot) 仕様」 を参照してください。

```
U-Boot 2018.03-at8 (Feb 14 2020 - 10:21:57 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
```

```
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 5.2 電源投入直後のログ (U-Boot の環境変数が eMMC にある場合)

Linux システム (Debian 9 "Stretch") を起動するには、次のように boot コマンドを実行してください。コマンドを実行するとブートローダーが Linux システムを起動させます。シリアル通信ソフトウェアには Linux の起動ログが表示されます。

```
=> boot
6601520 bytes read in 205 ms (30.7 MiB/s)
27838 bytes read in 53 ms (512.7 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at19
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    6601456 Bytes = 6.3 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02cbd ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.14-at19 (atmark@atde7) (gcc version 6.3.0 20170516 (Debian 6.3.0-18)) ⏎
#1 Wed Feb 19 11:40:11 JST 2020
[    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c53c7d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[    0.000000] OF: fdt: Machine model: Atmark Techno Armadillo-610
[    0.000000] Memory policy: Data cache writeback
[    0.000000] cma: Reserved 16 MiB at 0x9f000000
[    0.000000] CPU: All CPU(s) started in SVC mode.
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 129920
[    0.000000] Kernel command line: root=/dev/mmcblk0p2 rootwait
[    0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)
[    0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[    0.000000] Memory: 490908K/524288K available (6144K kernel code, 259K rwdata, 1288K rodata, ⏎
3072K init, 242K bss, 16996K reserved, 16384K cma-reserved)
[    0.000000] Virtual kernel memory layout:
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
[    0.000000]     vmalloc : 0xe0800000 - 0xff800000   ( 496 MB)
[    0.000000]     lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
[    0.000000]       .text : 0xc0008000 - 0xc0700000   (7136 kB)
[    0.000000]       .init : 0xc0900000 - 0xc0c00000   (3072 kB)
[    0.000000]       .data : 0xc0c00000 - 0xc0c40d00   ( 260 kB)
[    0.000000]        .bss : 0xc0c45d64 - 0xc0c82734   ( 243 kB)
```

```
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS: 16, nr_irqs: 16, preallocated irqs: 16
[    0.000000] Switching to timer-based delay loop, resolution 41ns
[    0.000016] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 89478484971ns
[    0.000054] clocksource: mxc_timer1: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
79635851949 ns
[    0.001816] Console: colour dummy device 80x30
[    0.002456] console [tty0] enabled
[    0.002521] Calibrating delay loop (skipped), value calculated using timer frequency.. 48.00
BogoMIPS (lpj=240000)
[    0.002582] pid_max: default: 32768 minimum: 301
[    0.002987] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.003036] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.004234] CPU: Testing write buffer coherency: ok
[    0.005295] Setting up static identity map for 0x80100000 - 0x80100060
[    0.007298] devtmpfs: initialized
[    0.017708] random: get_random_u32 called from bucket_table_alloc+0x84/0x1bc with crng_init=0
[    0.018208] VFP support v0.3: implementor 41 architecture 2 part 30 variant 7 rev 5
[    0.018641] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
19112604462750000 ns
[    0.018715] futex hash table entries: 256 (order: -1, 3072 bytes)
[    0.020062] pinctrl core: initialized pinctrl subsystem
[    0.021375] NET: Registered protocol family 16
[    0.023244] DMA: preallocated 256 KiB pool for atomic coherent allocations
[    0.032210] vdd3p0: supplied by regulator-dummy
[    0.033021] cpu: supplied by regulator-dummy
[    0.033796] vddsoc: supplied by regulator-dummy
[    0.048804] imx6ul-pinctrl 20e0000.iomuxc: initialized IMX pinctrl driver
[    0.049515] imx6ul-pinctrl 2290000.iomuxc-snvs: initialized IMX pinctrl driver
[    0.064034] GPIO line 45 (SWITCH_LCD_DATA18) hogged as output/high
[    0.064118] GPIO line 47 (SE_RST_N) hogged as output/high
[    0.069175] GPIO line 104 (ISO3086TDWR_RE) hogged as output/low
[    0.077905] SCSI subsystem initialized
[    0.078264] usbcore: registered new interface driver usbfs
[    0.078373] usbcore: registered new interface driver hub
[    0.078510] usbcore: registered new device driver usb
[    0.080308] i2c i2c-0: IMX I2C adapter registered
[    0.080374] i2c i2c-0: can't use DMA, using PIO instead.
[    0.081590] i2c i2c-1: IMX I2C adapter registered
[    0.081655] i2c i2c-1: can't use DMA, using PIO instead.
[    0.082423] i2c i2c-2: IMX I2C adapter registered
[    0.082487] i2c i2c-2: can't use DMA, using PIO instead.
[    0.083170] Advanced Linux Sound Architecture Driver Initialized.
[    0.085470] clocksource: Switched to clocksource mxc_timer1
[    0.100973] NET: Registered protocol family 2
[    0.102187] TCP established hash table entries: 4096 (order: 2, 16384 bytes)
[    0.102339] TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
[    0.102465] TCP: Hash tables configured (established 4096 bind 4096)
[    0.102661] UDP hash table entries: 256 (order: 0, 4096 bytes)
[    0.102723] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[    0.103077] NET: Registered protocol family 1
[    0.103769] RPC: Registered named UNIX socket transport module.
[    0.103827] RPC: Registered udp transport module.
[    0.103859] RPC: Registered tcp transport module.
[    0.103888] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.415196] workingset: timestamp_bits=14 max_order=17 bucket_order=3
[    0.431121] squashfs: version 4.0 (2009/01/31) Phillip Lougher
[    0.433052] NFS: Registering the id_resolver key type
```

```
[    0.433156] Key type id_resolver registered
[    0.433191] Key type id_legacy registered
[    0.433295] fuse init (API version 7.26)
[    0.441764] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 250)
[    0.441836] io scheduler noop registered
[    0.441867] io scheduler deadline registered
[    0.442305] io scheduler cfq registered (default)
[    0.452886] imx-sdma 20ec000.sdma: loaded firmware 3.3
[    0.454029] pfuze100-regulator 0-0008: Full layer: 1, Metal layer: 1
[    0.454732] pfuze100-regulator 0-0008: FAB: 0, FIN: 0
[    0.454777] pfuze100-regulator 0-0008: pfuze3000 found.
[    0.467415] 2020000.serial: ttymxc0 at MMIO 0x2020000 (irq = 18, base_baud = 5000000) is a IMX
[    0.467500] Console IMX rounded baud rate from 114943 to 114900
[    1.011614] console [ttymxc0] enabled
[    1.017160] 21e8000.serial: ttymxc1 at MMIO 0x21e8000 (irq = 55, base_baud = 5000000) is a IMX
[    1.027110] 21f4000.serial: ttymxc4 at MMIO 0x21f4000 (irq = 56, base_baud = 5000000) is a IMX
[    1.038310] libphy: Fixed MDIO Bus: probed
[    1.042790] tun: Universal TUN/TAP device driver, 1.6
[    1.048340] CAN device driver interface
[    1.054177] fec 2188000.ethernet: 2188000.ethernet supply phy not found, using dummy regulator
[    1.064249] libphy: fec_enet_mii_bus: probed
[    1.103797] usbcore: registered new interface driver awl13
[    1.109425] pegasus: v0.9.3 (2013/04/25), Pegasus/Pegasus II USB Ethernet driver
[    1.116979] usbcore: registered new interface driver pegasus
[    1.122761] usbcore: registered new interface driver rtl8150
[    1.128573] usbcore: registered new interface driver r8152
[    1.134162] usbcore: registered new interface driver lan78xx
[    1.139954] usbcore: registered new interface driver asix
[    1.145483] usbcore: registered new interface driver ax88179_178a
[    1.151684] usbcore: registered new interface driver cdc_ether
[    1.157645] usbcore: registered new interface driver net1080
[    1.163430] usbcore: registered new interface driver cdc_subset
[    1.169486] usbcore: registered new interface driver zaurus
[    1.175173] usbcore: registered new interface driver cdc_ncm
[    1.180912] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[    1.187507] ehci-mxc: Freescale On-Chip EHCI Host driver
[    1.193107] usbcore: registered new interface driver cdc_acm
[    1.198861] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
[    1.207030] usbcore: registered new interface driver usb-storage
[    1.213235] usbcore: registered new interface driver usbserial
[    1.219237] usbcore: registered new interface driver usbserial_generic
[    1.225904] usbserial: USB Serial support registered for generic
[    1.232007] usbcore: registered new interface driver cp210x
[    1.237704] usbserial: USB Serial support registered for cp210x
[    1.243717] usbcore: registered new interface driver ftdi_sio
[    1.249582] usbserial: USB Serial support registered for FTDI USB Serial Device
[    1.257023] usbcore: registered new interface driver mxuport
[    1.262765] usbserial: USB Serial support registered for MOXA UPort
[    1.269160] usbcore: registered new interface driver pl2303
[    1.274814] usbserial: USB Serial support registered for pl2303
[    1.293583] ci_hdrc ci_hdrc.1: EHCI Host Controller
[    1.298700] ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus number 1
[    1.335508] ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
[    1.342432] hub 1-0:1.0: USB hub found
[    1.346398] hub 1-0:1.0: 1 port detected
[    1.352521] using random self ethernet address
[    1.357114] using random host ethernet address
[    1.362786] usb0: HOST MAC 5a:03:14:fd:6e:79
```

```
[    1.367277] usb0: MAC 56:6f:92:11:2e:bd
[    1.371247] g_cdc gadget: CDC Composite Gadget, version: King Kamehameha Day 2008
[    1.378835] g_cdc gadget: g_cdc ready
[    1.387553] rtc-nr3225sa 1-0032: Voltage low, temperature compensation stopped.
[    1.394942] rtc-nr3225sa 1-0032: Voltage low, data loss detected.
[    1.402201] rtc-nr3225sa 1-0032: Voltage low, data is invalid.
[    1.408531] rtc-nr3225sa 1-0032: registered as rtc0
[    1.414718] snvs_rtc 20cc000.snvs:snvs-rtc-lp: registered as rtc1
[    1.421232] i2c /dev entries driver
[    1.425739] IR NEC protocol handler initialized
[    1.430327] IR RC5(x/sz) protocol handler initialized
[    1.435407] IR RC6 protocol handler initialized
[    1.440010] IR JVC protocol handler initialized
[    1.444574] IR Sony protocol handler initialized
[    1.449248] IR SANYO protocol handler initialized
[    1.453981] IR Sharp protocol handler initialized
[    1.458757] IR MCE Keyboard/mouse protocol handler initialized
[    1.464623] IR XMP protocol handler initialized
[    1.470997] imx2-wdt 20bc000.wdog: timeout 60 sec (nowayout=0)
[    1.477533] sdhci: Secure Digital Host Controller Interface driver
[    1.483768] sdhci: Copyright(c) Pierre Ossman
[    1.488214] sdhci-pltfm: SDHCI platform and OF driver helper
[    1.535478] random: fast init done
[    1.555512] mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
[    1.605096] mmc0: new DDR MMC card at address 0001
[    1.610757] mmcblk0: mmc0:0001 S0J35A 3.53 GiB
[    1.615769] mmcblk0boot0: mmc0:0001 S0J35A partition 1 31.5 MiB
[    1.622080] mmcblk0boot1: mmc0:0001 S0J35A partition 2 31.5 MiB
[    1.628333] mmc1: SDHCI controller on 2194000.usdhc [2194000.usdhc] using ADMA
[    1.637540] usbcore: registered new interface driver usbhid
[    1.643172] usbhid: USB HID core driver
[    1.648435] mmcblk0gp0: mmc0:0001 S0J35A partition 4 8.00 MiB
[    1.654808] mmcblk0gp1: mmc0:0001 S0J35A partition 5 8.00 MiB
[    1.670662] mmcblk0gp2: mmc0:0001 S0J35A partition 6 8.00 MiB
[    1.683476] mmcblk0gp3: mmc0:0001 S0J35A partition 7 8.00 MiB
[    1.689892] mmcblk0rpmb: mmc0:0001 S0J35A partition 3 4.00 MiB, chardev (248:0)
[    1.701847] imx-mqs sound-mqs: fsl-mqs-dai <-> 2028000.sai mapping ok
[    1.709000]  mmcblk0: p1 p2 p3
[    1.717824] imx-mqs sound-mqs: snd-soc-dummy-dai <-> 2034000.asrc mapping ok
[    1.725213] imx-mqs sound-mqs: fsl-mqs-dai <-> 2028000.sai mapping ok
[    1.731917] usb 1-1: new high-speed USB device number 2 using ci_hdrc
[    1.747838] ip_tables: (C) 2000-2006 Netfilter Core Team
[    1.754642] NET: Registered protocol family 10
[    1.764089] Segment Routing with IPv6
[    1.768048] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[    1.775222] NET: Registered protocol family 17
[    1.779829] can: controller area network core (rev 20170425 abi 9)
[    1.786432] NET: Registered protocol family 29
[    1.790928] can: raw protocol (rev 20170425)
[    1.795230] can: broadcast manager protocol (rev 20170425 t)
[    1.801047] can: netlink gateway (rev 20170425) max_hops=1
[    1.806967] Key type dns_resolver registered
[    1.812134] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 900000000, volt:
1275000, enabled: 1. New: freq: 900000000, volt: 1275000, enabled: 1
[    1.826493] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 792000000, volt:
1225000, enabled: 1. New: freq: 792000000, volt: 1225000, enabled: 1
[    1.840781] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 528000000, volt:
1175000, enabled: 1. New: freq: 528000000, volt: 1175000, enabled: 1
```

```
[    1.855050] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 396000000, volt:
1025000, enabled: 1. New: freq: 396000000, volt: 1025000, enabled: 1
[    1.869319] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 198000000, volt: 950000,
enabled: 1. New: freq: 198000000, volt: 950000, enabled: 1
[    1.884334] ThumbEE CPU extension supported.
[    2.396913] input: gpio-keys as /devices/soc0/gpio-keys/input/input0
[    2.404789] rtc-nr3225sa 1-0032: Voltage low, data is invalid.
[    2.410782] rtc-nr3225sa 1-0032: hctosys: unable to read the hardware clock
[    2.418465] USB_OTG1_VBUS: disabling
[    2.422095] ALSA device list:
[    2.425086]   #0: mqs-audio
[    2.428151] Warning: unable to open an initial console.
[    2.439030] Freeing unused kernel memory: 3072K
[    2.457439] hub 1-1:1.0: USB hub found
[    2.463257] hub 1-1:1.0: 3 ports detected
[    2.587787] systemd-udevd[77]: starting version 215
[    2.595444] random: systemd-udevd: uninitialized urandom read (16 bytes read)
[    3.749332] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[    4.103435] systemd[1]: System time before build time, advancing clock.
[    4.124545] random: systemd: uninitialized urandom read (16 bytes read)
[    4.134560] random: systemd: uninitialized urandom read (16 bytes read)
[    4.149035] systemd[1]: systemd 232 running in system mode. (+PAM +AUDIT +SELINUX +IMA +APPARMOR
+SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD
+IDN)
[    4.168001] systemd[1]: Detected architecture arm.

Welcome to Debian GNU/Linux 9 (stretch)!

[    4.209170] systemd[1]: Set hostname to <armadillo>.
[    4.431504] random: systemd-cryptse: uninitialized urandom read (16 bytes read)
[    4.812626] systemd[1]: Listening on Journal Socket.
[  OK  ] Listening on Journal Socket.
[    4.846247] systemd[1]: Listening on Syslog Socket.
[  OK  ] Listening on Syslog Socket.
[    4.877322] systemd[1]: Created slice User and Session Slice.
[  OK  ] Created slice User and Session Slice.
[    4.916866] systemd[1]: Listening on udev Control Socket.
[  OK  ] Listening on udev Control Socket.
[    4.956126] systemd[1]: Listening on udev Kernel Socket.
[  OK  ] Listening on udev Kernel Socket.
[    4.985922] systemd[1]: Reached target Remote File Systems.
[  OK  ] Reached target Remote File Systems.
[    5.015909] systemd[1]: Reached target Swap.
[  OK  ] Reached target Swap.
[  OK  ] Listening on /dev/initctl Compatibility Named Pipe.
[UNSUPP] Starting of Arbitrary Executable Fi…tem Automount Point not supported.
[  OK  ] Started Forward Password Requests to Wall Directory Watch.
[  OK  ] Started Dispatch Password Requests to Console Directory Watch.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Encrypted Volumes.
[  OK  ] Listening on fsck to fsckd communication Socket.
[  OK  ] Listening on Journal Socket (/dev/log).
[  OK  ] Created slice System Slice.
[  OK  ] Created slice system-getty.slice.
         Starting Journal Service...
[  OK  ] Reached target Slices.
[  OK  ] Created slice system-serial¥x2dgetty.slice.
         Starting Remount Root and Kernel File Systems...
```

```
         Starting Create Static Device Nodes in /dev...
         Starting Load Kernel Modules...
[    5.631604] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[  OK  ] Started Remount Root and Kernel File Systems.
[  OK  ] Started Load Kernel Modules.
[  OK  ] Started Create Static Device Nodes in /dev.
         Starting udev Kernel Device Manager...
         Mounting FUSE Control File System...
         Mounting Configuration File System...
         Starting Apply Kernel Variables...
         Starting udev Coldplug all Devices...
         Starting Load/Save Random Seed...
[  OK  ] Reached target Local File Systems (Pre).
[  OK  ] Reached target Local File Systems.
         Starting netfilter persistent configuration...
[  OK  ] Mounted Configuration File System.
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Started Journal Service.
[  OK  ] Started Apply Kernel Variables.
[  OK  ] Started Load/Save Random Seed.
[  OK  ] Started netfilter persistent configuration.
[  OK  ] Started udev Kernel Device Manager.
[  OK  ] Reached target Network (Pre).
         Starting Raise network interfaces...
         Starting Flush Journal to Persistent Storage...
[    6.939306] systemd-journald[144]: Received request to flush runtime journal from PID 1
[  OK  ] Started Flush Journal to Persistent Storage.
         Starting Create Volatile Files and Directories...
[  OK  ] Started Create Volatile Files and Directories.
         Starting Update UTMP about System Boot/Shutdown...
         Starting Network Time Synchronization...
[  OK  ] Started Update UTMP about System Boot/Shutdown.
[  OK  ] Started Raise network interfaces.
[  OK  ] Started Network Time Synchronization.
[  OK  ] Reached target System Time Synchronized.
[  OK  ] Reached target Network.
[  OK  ] Reached target Network is Online.
[  OK  ] Started udev Coldplug all Devices.
[  OK  ] Reached target System Initialization.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Started Daily apt download activities.
[  OK  ] Started Daily apt upgrade and clean activities.
[  OK  ] Reached target Timers.
[  OK  ] Reached target Basic System.
         Starting Login Service...
         Starting LSB: exim Mail Transport Agent...
         Starting System Logging Service...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Regular background program processing daemon.
         Starting Permit User Sessions...
         Starting /etc/rc.local Compatibility...
[  OK  ] Started System Logging Service.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
[  OK  ] Found device /dev/ttymxc0.
[  OK  ] Found device /dev/mmcblk0gp0.
```

```
[  OK  ] Reached target Sound Card.
[  OK  ] Started ifup for eth0.
         Mounting /opt/license...
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttymxc0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Mounted /opt/license.
[  OK  ] Started Login Service.
[   11.777015] SMSC LAN8710/LAN8720 2188000.ethernet-1:00: attached PHY driver [SMSC LAN8710/
LAN8720] (mii_bus:phy_addr=2188000.ethernet-1:00, irq=POLL)
[   11.825773] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[  OK  ] Started LSB: exim Mail Transport Agent.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 9 armadillo ttymxc0

[   16.086034] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[   16.093944] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
armadillo login:
```

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
atmark	atmark	一般ユーザ

初めて機器を接続したときは、必ず以下の手順に従って、初期パスワードを変更してください。

1. root でログイン

   初期パスワードを変更します。

   ```
   [armadillo ~]# passwd
   Enter new UNIX password: # 新しいパスワードを入力
   Retype new UNIX password: # 再入力
   ```

2. atmark でログイン

   初期パスワードを変更します。

   ```
   [armadillo ~]$ passwd
   Enter new UNIX password: # 新しいパスワードを入力
   Retype new UNIX password: # 再入力
   ```

Armadillo-610 はネットワークに接続されることを前提としている機器です。初期パスワードのままご利用になると、セキュリティリスクが非常に高まります。最終製品として作り上げる場合は、外部からのログインは極力できないようにすることをお勧めします。どうしてもログインが必要な場合は、セキュリティ強度の高いパスワードに変更し、その後も適切にパスワードを運用されることを強くお勧めします。

5.3. Debian のユーザを管理する

例として guest というユーザを作成、パスワードの変更、sudo を許可する方法を紹介します。

```
[armadillo ~]# adduser guest
Adding user ` [user_name] ' ...
Adding new group `guest' (1001) ...
Adding new user `guest' (1001) with group `guest' ...
Creating home directory `/home/guest' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: # パスワードを入力
Retype new UNIX password: # 再入力
passwd: password updated successfully
Changing the user information for guest
Enter the new value, or press ENTER for the default
        Full Name []: # Enter を入力
        Room Number []: # Enter を入力
        Work Phone []: # Enter を入力
        Home Phone []: # Enter を入力
        Other []: # Enter を入力
Is the information correct? [Y/n] # Enter を入力
```

図 5.3 ユーザの作成

```
[armadillo ~]# passwd guest
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 再入力
```

図 5.4 パスワードの変更

```
[armadillo ~]# sudo usermod -a -G sudo guest
```

図 5.5 sudo を許可

```
[armadillo ~]# deluser guest
  or
[armadillo ~]# deluser --remove-home guest
```

図 5.6 ユーザの削除

ホームディレクトリも合わせて消したいときは、 --remove-home オプションをつけます。

5.4. 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、「reboot: 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
         Stopping Session c1 of user root.
[  OK  ] Stopped target Graphical Interface.
[  OK  ] Stopped target Multi-User System.
         Stopping D-Bus System Message Bus...
         Stopping System Logging Service...
         Stopping LSB: exim Mail Transport Agent...
[  OK  ] Stopped target Timers.
[  OK  ] Stopped Daily Cleanup of Temporary Directories.
[  OK  ] Stopped Daily apt upgrade and clean activities.
[  OK  ] Stopped target Login Prompts.
         Stopping Serial Getty on ttymxc0...
         Stopping User Manager for UID 0...
         Stopping Getty on tty1...
         Unmounting /opt/license...
         Stopping Regular background program processing daemon...
[  OK  ] Stopped target Sound Card.
[  OK  ] Stopped Daily apt download activities.
[  OK  ] Stopped Regular background program processing daemon.
[  OK  ] Stopped System Logging Service.
[  OK  ] Stopped D-Bus System Message Bus.
[  OK  ] Stopped Serial Getty on ttymxc0.
[  OK  ] Stopped Getty on tty1.
[  OK  ] Stopped User Manager for UID 0.
[  OK  ] Stopped Session c1 of user root.
[  OK  ] Unmounted /opt/license.
[  OK  ] Removed slice User Slice of root.
         Stopping Login Service...
[  OK  ] Removed slice system-getty.slice.
[  OK  ] Stopped /etc/rc.local Compatibility.
[  OK  ] Removed slice system-serial¥x2dgetty.slice.
         Stopping Permit User Sessions...
[  OK  ] Stopped Login Service.
[  OK  ] Stopped Permit User Sessions.
[  OK  ] Stopped LSB: exim Mail Transport Agent.
[  OK  ] Stopped target System Time Synchronized.
[  OK  ] Stopped target Network is Online.
[  OK  ] Stopped target Remote File Systems.
[  OK  ] Stopped target Network.
         Stopping ifup for eth0...
         Stopping Raise network interfaces...
[  OK  ] Stopped target Basic System.
[  OK  ] Stopped target Paths.
[  OK  ] Stopped target Sockets.
[  OK  ] Closed Syslog Socket.
[  OK  ] Closed D-Bus System Message Bus Socket.
[  OK  ] Stopped target Slices.
[  OK  ] Removed slice User and Session Slice.
[  OK  ] Stopped target System Initialization.
         Stopping Update UTMP about System Boot/Shutdown...
```

```
         Stopping Load/Save Random Seed...
         Stopping Network Time Synchronization...
[  OK  ] Stopped target Encrypted Volumes.
[  OK  ] Stopped Forward Password Requests to Wall Directory Watch.
[  OK  ] Stopped Dispatch Password Requests to Console Directory Watch.
[  OK  ] Stopped target Swap.
[  OK  ] Stopped Network Time Synchronization.
[  OK  ] Stopped Load/Save Random Seed.
[  OK  ] Stopped Update UTMP about System Boot/Shutdown.
[  OK  ] Stopped Create Volatile Files and Directories.
[  OK  ] Stopped Raise network interfaces.
[  OK  ] Stopped ifup for eth0.
[  OK  ] Stopped target Network (Pre).
         Stopping netfilter persistent configuration...
[  OK  ] Stopped Apply Kernel Variables.
[  OK  ] Stopped netfilter persistent configuration.
[  OK  ] Stopped Load Kernel Modules.
[  OK  ] Stopped target Local File Systems.
         Unmounting /run/user/0...
[  OK  ] Unmounted /run/user/0.
[  OK  ] Reached target Unmount All Filesystems.
[  OK  ] Stopped target Local File Systems (Pre).
[  OK  ] Stopped Create Static Device Nodes in /dev.
[  OK  ] Stopped Remount Root and Kernel File Systems.
[  OK  ] Reached target Shutdown.
[   29.070874] systemd-shutdow: 32 output lines suppressed due to ratelimiting
[   29.118128] systemd-shutdown[1]: Sending SIGTERM to remaining processes...
[   29.137325] systemd-journald[151]: Received SIGTERM from PID 1 (systemd-shutdow).
[   29.153917] systemd-shutdown[1]: Sending SIGKILL to remaining processes...
[   29.171503] systemd-shutdown[1]: Unmounting file systems.
[   29.178450] systemd-shutdown[1]: Remounting '/' read-only with options 'data=ordered'.
[   29.199284] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[   29.211365] systemd-shutdown[1]: Remounting '/' read-only with options 'data=ordered'.
[   29.219946] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[   29.226001] systemd-shutdown[1]: All filesystems unmounted.
[   29.231656] systemd-shutdown[1]: Deactivating swaps.
[   29.236939] systemd-shutdown[1]: All swaps deactivated.
[   29.242277] systemd-shutdown[1]: Detaching loop devices.
[   29.250115] systemd-shutdown[1]: All loop devices detached.
[   29.269592] ci_hdrc ci_hdrc.1: remove, state 1
[   29.274188] usb usb1: USB disconnect, device number 1
[   29.279272] usb 1-1: USB disconnect, device number 2
[   29.288775] ci_hdrc ci_hdrc.1: USB bus 1 deregistered
[   29.299013] reboot: System halted
```

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。

poweroff コマンドでも Armadillo-610 を終了させることができます。

```
[armadillo ~]# poweroff
:（省略）
[   30.356097] reboot: Power down
```

poweroff と halt では、コマンド実行後の Armadillo-610 の状態が異なります。

poweroff の場合、Armadillo-610 は、ONOFF ピンを GND とショートすることで電源をオフした場合と同じ状態になります。そのため、RTC_BAT ピンからバックアップ電源が供給されている限り、i.MX6ULL への電源を再投入しても Armadillo-610 が起動しません。詳しくは「15.6.1. ONOFF ピンからの電源制御」を参照してください。

6. 動作確認方法

6.1. 動作確認を行う前に

Armadillo には、 OS として Debian がインストールされています。基本的には PC Linux と同じように動作します。ここではネットワークの設定やストレージのように一般的な動作に加え、GPIO や LED などについて説明します。

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、最新版でない可能性があります。最新版のイメージファイルは Armadillo サイトからダウンロード可能です。最新版のイメージファイルに書き換えてからのご使用を推奨します。

イメージファイルの書き換えについては、「11. イメージファイルの書き換え方法」を参照してください。

6.2. ネットワーク

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

6.2.1. 接続可能なネットワーク

Armadillo-610 は、 Ethernet に対応しています。Linux からは、`eth0` に見えます。

表 6.1 ネットワークとネットワークデバイス

ネットワーク	ネットワークデバイス	出荷時の設定
Ethernet	`eth0`	DHCP

6.2.2. ネットワークの設定方法

ここでは有線 LAN の使用方法について説明します。Armadillo-610 では、通常の Linux システムと同様にネットワーク設定を行います。出荷状態では `eth0` が DHCP [1] でネットワークの設定を行います。 DHCP が無い環境の場合は、 「6.2.3.3. 固定 IP アドレスに設定する」 を参照し設定してください。

6.2.3. 基本的な使い方

最近の GNU/Linux OS では、古くから使われてきた `ifconfig` (net-tools) に代り iproute2 を使用します。 `ifconfig` は Deprecated されています。本書でも `ifconfig` ではなく、 iproute2 に含まれている `ip` コマンドなどを使用します。

6.2.3.1. IP アドレスの一覧

IP アドレスの一覧を確認するには、次のようにコマンドを実行します。

[1] Dynamic Host Configuration Protocol

```
[armadillo ~]# ip address
ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
oup default qlen 1000
    link/ether 00:11:0c:00:07:a4 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.107/16 brd 172.16.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

図 6.1 インターフェースの一覧確認

ネットワークデバイスの一覧を確認するには link を使います。

```
[armadillo ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether 00:11:0c:00:07:a4 brd ff:ff:ff:ff:ff:ff
```

図 6.2 ネットワークデバイスの一覧確認

他にも ip コマンドではルーティングテーブルの表示やトンネルの設定などもできます。詳しくは ip コマンドの man ページを参照してください。

6.2.3.2. インターフェースの有効化・無効化

インターフェースを有効化するには、次のようにコマンドを実行します。 ifup コマンドは Debian 特有のコマンドで ifupdown パッケージに含まれています。 ifconfig とは関係なく抽象的にネットワークを操作するためのコマンドです。設定は /etc/network/interfaces で行います。

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

図 6.3 インターフェースの有効化

インターフェースを無効化するには、次のようにコマンドを実行します。

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

図 6.4 インターフェースの無効化

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。

/etc/network/interfaces の変更は、インターフェースを無効化した状態で行ってください。 DHCP が動作している場合など、設定が反映されない場合があります。

6.2.3.3. 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」の内容に設定する例を、以下に示します。

表 6.2 固定 IP アドレス設定例

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

```
[armadillo ~]# vi /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

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

図 6.5 固定 IP アドレス設定

6.2.3.4. DHCP に設定する

DHCP に設定する例を以下にに示します。

```
[armadillo ~]# vi /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

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

図 6.6 DHCP 設定

6.2.3.5. DNS サーバーを指定する

固定 IP 時、または DHCP で DNS 情報が取得できない場合は、DNS サーバーを指定する必要があります。DNS サーバーを指定する例を、以下に示します。

```
[armadillo ~]# vi /etc/resolv.conf
domain local-network
```

```
search local-network
nameserver 192.0.2.1
```

図 6.7 DNS サーバーの指定

6.2.3.6. インターフェースの修正を反映する

有効化されているインターフェースは、修正しないでください。必ず無効化してから設定を変更してください。

```
[armadillo ~]# ifdown eth0
[armadillo ~]# vi /etc/network/interfaces
  :
[armadillo ~]# ifup eth0
```

6.2.3.7. 有線 LAN の接続を確認する

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

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

```
[armadillo ~]# ping 192.0.2.20
ping -c 3 192.0.2.20
PING 192.0.2.20 (192.0.2.20) 56(84) bytes of data.
64 bytes from 192.0.2.20: icmp_seq=1 ttl=63 time=1.39 ms
64 bytes from 192.0.2.20: icmp_seq=2 ttl=63 time=1.35 ms
64 bytes from 192.0.2.20: icmp_seq=3 ttl=63 time=1.34 ms

--- 192.0.2.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.343/1.365/1.395/0.021 ms
```

図 6.8 有線 LAN の PING 確認

有線 LAN 以外のインターフェースが有効化されている場合、ルーティングの設定などにより、ネットワーク通信に有線 LAN が使用されない場合があります。設定を必ず確認してください。確実に有線 LAN の接続確認をする場合は、有線 LAN 以外のインターフェースを無効化してください。

6.2.4. ファイアーウォール

Armadillo-610 では、ファイアーウォールの実現に iptables を使用しています。工場出荷状態の Armadillo-610 では、開発時の利便性のために、すべての通信(送信・受信・転送)を許可する設定になっています。

Armadillo を製品として運用する際には、最低限、踏み台として利用されない程度のファイアーウォールを設定しておかなければいけません。

ここでは、iptables のポリシーの設定と、Armadillo がネットワークに接続される前に自動的に設定を適用する方法を紹介します。

6.2.4.1. iptables のポリシーの設定

送信はすべて許可、受信・転送はすべて破棄するように設定します。

```
[armadillo ~]# iptables --policy INPUT DROP
[armadillo ~]# iptables --policy FORWARD DROP
[armadillo ~]# iptables --policy OUTPUT ACCEPT
```

iptables のポリシーの設定で受信と転送を許可する

iptables のポリシーの設定をもとに戻す(受信・転送を許可する)には次のコマンドを実行します。

```
[armadillo ~]# iptables --policy INPUT ACCEPT
[armadillo ~]# iptables --policy FORWARD ACCEPT
```

6.2.4.2. lo (ローカルループバックインターフェース)の許可

```
[armadillo ~]# iptables --append INPUT --in-interface lo --jump ACCEPT
```

6.2.4.3. iptables の設定確認

設定されている内容を参照するには、次のコマンドを実行します。

```
[armadillo ~]# iptables --list --verbose
Chain INPUT (policy DROP 1 packets, 72 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere


Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

図 6.9 iptables 設定確認

「図 6.9. iptables 設定確認」の設定では受信パケットが全て破棄されます。これが最も安全で最小の設定です。

この設定をベースに、SSH や HTTPS などの通信プロトコルから利用するものだけを許可していくことをおすすめします。

6.2.4.4. iptables の設定を保存し自動的に適用する

ここまでの手順で行った iptables の設定は、Armadillo を再起動すると失われてしまいます。そこで、Armadillo-610 では iptables-persistent パッケージを利用して、あらかじめ保存しておいた設定を自動的に適用します。

iptables の設定を保存するには、次のコマンドを実行します。

```
[armadillo ~]# iptables-save > /etc/iptables/rules.v4
```

図 6.10 iptables 設定保存

iptables-persistent がインストールされていなかった場合は、/etc/iptables/ ディレクトリが存在しないため、「図 6.10. iptables 設定保存」が失敗します。次のコマンドを実行して iptables-persistent をインストールし、再度 「図 6.10. iptables 設定保存」 を行ってください。

```
[armadillo ~]# iptables --policy INPUT ACCEPT
[armadillo ~]# apt-get update && apt-get install iptables-persistent
```

図 6.11 iptables のポリシー設定(受信許可)と iptables-persistent のインストール

次回起動時から、Armadillo がネットワークに接続される前のタイミングで、自動的に iptables の設定が適用されます。

6.3. ストレージ

Armadillo-610 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 6.3 ストレージデバイス

デバイス種類	ディスクデバイス	先頭パーティション	インターフェース
オンボード eMMC	/dev/mmcblk0	/dev/mmcblk0p1	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp2	なし	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp3	なし	オンボード
SD/SDHC/SDXC カード	/dev/mmcblk1	/dev/mmcblk1p1	SD インターフェース (Armadillo-610: CON1) [a] SD インターフェース (Armadillo-610 拡張ボード: CON1) [a]
USB メモリ	/dev/sd* [b]	/dev/sd*1	USB ホストインターフェース (Armadillo-610 拡張ボード: CON5)

[a]Armadillo-610: CON1 と Armadillo-610 拡張ボード: CON1 は排他利用となります。詳しくは「20.6. Armadillo-610 拡張ボードの SD インターフェースを利用する」を参照してください。

[b]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。

GPP(General Purpose Partition)について

GPP は、eMMC の通常の記憶領域を割譲して eMMC 内部に作られた記憶領域です。 eMMC の通常の記憶領域とはアドレス空間が異なるため、/dev/mmcblk0 および /dev/mmcblk0p* に対してどのような書き込みを行っても /dev/mmcblk0gp* のデータが書き換わることはありません。

Armadillo-610 では、8 MiB の GPP を 4 つ作成しています。各領域の用途を「表 6.4. eMMC の GPP の用途」に示します。

表 6.4 eMMC の GPP の用途

ディスクデバイス	用途
/dev/mmcblk0gp0	ライセンス情報等の保存
/dev/mmcblk0gp1	予約領域
/dev/mmcblk0gp2	ユーザー領域
/dev/mmcblk0gp3	ユーザー領域

GPP のユーザー領域を使用する例を「20.5. eMMC の GPP(General Purpose Partition) を利用する」に記載しています。

6.3.1. ストレージの使用方法

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

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

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

mount コマンドの典型的なフォーマットは、次の通りです。

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

図 6.12 mount コマンド書式

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

通常、購入したばかりの SDHC カードは FAT32 または exFAT ファイルシステムでフォーマットされています。

device には、ストレージデバイスのデバイスファイル名を指定します。microSD カードのパーティション 1 の場合は /dev/mmcblk1p1 、パーティション 2 の場合は /dev/mmcblk1p2 となります。

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

microSD スロット (Armadillo-610: CON1) に SDHC カードを挿入し[2]、以下に示すコマンドを実行すると、 /media 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することができます。microSD カード内のファイルは、/media ディレクトリ以下に見えるようになります。

```
[armadillo ~]# mount -t vfat /dev/mmcblk1p1 /media
[armadillo ~]# ls /media
  :
  :
```

図 6.13 ストレージのマウント

ストレージを安全に取り外すには、アンマウントという作業が必要です。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 /media
```

図 6.14 ストレージのアンマウント

6.3.2. ストレージのパーティション変更とフォーマット

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

パーティション構成を変更したい場合、 fdisk コマンドを使用します。 fdisk コマンドの使用例として、一つのパーティションで構成されている microSD カードのパーティションを、2 つに分割する例を「図 6.15. 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/mmcblk1p1 、二つめは /dev/mmcblk1p2 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。

[2]Armadillo-600 シリーズの microSD スロットは活線挿抜非対応のため、Armadillo の電源を OFF してから microSD カードを挿抜してください。

```
[armadillo ~]# fdisk /dev/mmcblk1

Welcome to fdisk (util-linux 2.29.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-7744511, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-7744511, default 7744511): +100M

Created a new partition 1 of type 'Linux' and of size 100 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (206848-7744511, default 206848):
Last sector, +sectors or +size{K,M,G,T,P} (206848-7744511, default 7744511):

Created a new partition 2 of type 'Linux' and of size 3.6 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[  447.905671]  mmcblk1: p1 p2
Syncing disks.
```

図 6.15 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、 EXT4 ファイルシステムでフォーマットするには、mkfs.ext2 や mkfs.ext3 、 mkfs.ext4 コマンドを使用します。microSD カードのパーティション 1 を EXT4 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 ~]# mkfs.ext4 /dev/mmcblk1p1
```

図 6.16 EXT4 ファイルシステムの構築

6.4. LED

Armadillo-610 の LED は GPIO で接続されているため、ソフトウェアで制御することができます。

利用しているデバイスドライバは LED クラスとして実装されているため、LED クラスディレクトリ以下のファイルによって LED の制御を行うことができます。LED クラスディレクトリと各 LED の対応を次に示します。

表 6.5 LED クラスディレクトリと LED の対応

LED クラスディレクトリ	インターフェース	デフォルトトリガ
/sys/class/leds/green/	ユーザー LED 緑	default-on
/sys/class/leds/yellow/	ユーザー LED 黄	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。[LED] の部分を適宜読みかえてください。

6.4.1. LED を点灯/消灯する

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

brightness に 0 以外の値を書き込むと LED が点灯します。

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

図 6.17 LED を点灯させる

Armadillo-610 の LED には輝度制御の機能がないため、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 6.18 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 6.19 LED の状態を表示する

6.4.2. トリガを使用する

Linux では、LED をある特定のタイミングで光らせることができます。これを「トリガ」と呼びます。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 でサポートされている値は以下の通りです。

表 6.6 LED トリガの種類

設定	説明
none	トリガを設定しません
mmc0	eMMC のアクセスランプにします
mmc1	microSD スロットのアクセスランプにします
timer	任意のタイミングで点灯/消灯を行います。この設定にすることにより、LED クラスディレクトリ以下に delay_on, delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します
heartbeat	心拍のように点灯/消灯を行います
default-on	主に Linux カーネルから使用します。LED が点灯します

trigger ファイルを読み出すとサポートしているトリガと、現在有効のトリガが表示されます。[] 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] rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shif
tlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock kbd-c
trlllock kbd-ctrlrlock mmc0 mmc1 timer oneshot heartbeat default-on
```

図 6.20 対応している LED トリガを表示

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 6.21 LED のトリガに timer を指定する

6.5. ユーザースイッチ

Armadillo-610 のユーザースイッチのデバイスドライバは、インプットデバイスとして実装されています。インプットデバイスのデバイスファイルからボタンプッシュ/リリースイベントを取得することができます。

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

表 6.7 インプットデバイスファイルとイベントコード

ユーザースイッチ	インプットデバイスファイル	イベントコード
SW1	/dev/input/event0	28 (KEY_ENTER)

インプットデバイスは検出された順番にインデックスが割り振られます。USB デバイスなどを接続してインプットデバイスを追加している場合は、デバイスファイルのインデックスが異なる可能性があります。

6.5.1. イベントを確認する

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

```
[armadillo ~]# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)
Event: time 1523249446.289965, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0  ❶
Event: time 1523249446.289965, ---------- SYN_REPORT ------------
Event: time 1523249446.349969, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1  ❷
Event: time 1523249446.349969, ---------- SYN_REPORT ------------
```

図 6.22 ユーザースイッチ: イベントの確認

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

6.6. RTC

Armadillo-610 は、i.MX6ULL の RTC 機能を利用しています。また、Armadillo-610 拡張ボードには、日本電波工業(NDK)製 NR3225SA が搭載されており、こちらも利用することができます。

6.6.1. 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 6.8 時刻フォーマットのフィールド

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

2018 年 3 月 2 日 12 時 34 分 56 秒に設定する例を次に示します。

```
[armadillo ~]# date
Sat Jan  1 09:00:00 JST 2000
[armadillo ~]# systemctl stop systemd-timesyncd.service
[armadillo ~]# date 030212342018.56
Fri Mar  2 12:34:56 JST 2018
[armadillo ~]# date
Fri Mar  2 12:34:57 JST 2018
```

図 6.23 システムクロックを設定

Armadillo-610 では、標準で systemd-timesyncd.service が動作しています。systemd-timesyncd.service は、自身が正しいと考えている時刻となるように、自動でシステムクロックおよびハードウェアクロックを設定します。

そのため、date コマンドで過去の時刻を設定しても、すぐに systemd-timesyncd.service によって変更前の正しい時刻に再設定されてしまいます。これを避けるため、システムクロックを設定する前に systemd-timesyncd.service を停止する必要があります。

```
[armadillo ~]# systemctl stop systemd-timesyncd.service
```

systemd-timesyncd.service は、次の NTP サーバーを利用します。利用する NTP サーバーは/etc/systemd/timesyncd.conf で設定することができます。詳しくは man timesyncd.conf を参照してください。

- 0.debian.pool.ntp.org
- 1.debian.pool.ntp.org
- 2.debian.pool.ntp.org
- 3.debian.pool.ntp.org

systemd-timesyncd.service を自動で起動しないようにするには、次のようにしてサービスを無効化します。

```
[armadillo ~]# systemctl disable systemd-timesyncd.service
```

Armadillo-610 のタイムゾーンはデフォルトで JST に設定されています。timedatectl コマンドで、これを変更することができます。

タイムゾーンを UTC に変更するには次のようにコマンドを実行します。

```
root@armadillo:~# date
Tue Feb 12 10:32:07 JST 2019
```

```
root@armadillo:~# timedatectl set-timezone Etc/UTC
root@armadillo:~# date
Tue Feb 12 01:32:10 UTC 2019
```

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
2000-01-01 00:00:00.000000+0900
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
2018-03-02 12:35:08.213911+0900
```

図 6.24 ハードウェアクロックを設定

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。

i.MX6ULL の RTC に時刻を設定する場合は、--rtc オプションで /dev/rtc1 を指定します。

```
[armadillo ~]# hwclock --rtc /dev/rtc1
[armadillo ~]# hwclock --rtc /dev/rtc1 --utc --systohc
[armadillo ~]# hwclock --rtc /dev/rtc1 --utc
```

6.6.2. RTC から時刻を取得する

hwclock コマンドで RTC から時刻を取得することができます。

```
[armadillo ~]# hwclock
2018-11-19 09:51:53.743739+0900
```

NR3225SA に接続した電池残量が少ない状態 (厳密には電源電圧が 1.3V 以上 1.5V 未満の状態) で Armadillo を起動すると、NR3225SA から時刻を取得できなくなります。

```
[armadillo ~]# hwclock
[   44.986529] rtc-nr3225sa 5-0032: Voltage low, data is invalid.
hwclock: ioctl(RTC_RD_TIME) to /dev/rtc to read the time failed: Invalid
argument
```

この状態になった場合は、以下のどちらかの方法を実施すると取得可能になります。

- Armadillo の電源を OFF にして古い電池を取り外し、10 秒以上経過した後に新しい電池を取り付ける。
- RTC に時刻を再設定する。

6.7. GPIO

Armadillo-610 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。

Armadillo-610 開発セットの工場出荷状態では、DIDO インターフェース(Armadillo-610 拡張ボード: CON13B)の各ピンを GPIO として利用することができます。ピン番号と GPIO 番号の対応を次に示します。

表 6.9 Armadillo-610 拡張ボード: CON13B ピン番号と GPIO 番号の対応

ピン番号	ピン名	GPIO 番号
4	UART2_TX_DATA	20
5	UART2_RX_DATA	21
6/7	UART5_TX_DATA	30
8/9	UART5_RX_DATA	31

at-dtweb を利用すると、「表 6.9. Armadillo-610 拡張ボード: CON13B ピン番号と GPIO 番号の対応」以外のピンも GPIO として利用することができるようになります。at-dtweb の利用方法については「20.3. Device Tree をカスタマイズする」を参照してください。

拡張インターフェース(Armadillo-610: CON2)のピン番号と GPIO 番号の対応を次に示します。

表 6.10 Armadillo-610: CON2 ピン番号と GPIO 番号の対応

ピン番号	ピン名	GPIO 番号
11	UART1_RTS_B	19
12	CSI_MCLK	113
14	GPIO1_IO04	4
15	GPIO1_IO03	3
16	GPIO1_IO02	2
17	GPIO1_IO01	1
18	LCD_DATA00	69
19	LCD_DATA01	70
20	LCD_DATA02	71
21	LCD_DATA03	72
22	LCD_DATA04	73
23	LCD_DATA05	74
24	LCD_DATA06	75
25	LCD_DATA07	76
26	LCD_DATA08	77
27	LCD_DATA09	78

ピン番号	ピン名	GPIO 番号
28	LCD_DATA10	79
29	LCD_DATA11	80
30	LCD_DATA12	81
31	LCD_DATA13	82
32	LCD_DATA14	83
33	LCD_DATA15	84
34	LCD_DATA16	85
35	LCD_DATA17	86
37	LCD_CLK	64
38	LCD_HSYNC	66
39	LCD_VSYNC	67
40	LCD_ENABLE	65
41	NAND_DQS	112
43	JTAG_MOD	10
55	CSI_VSYNC	115
56	CSI_HSYNC	116
57	CSI_DATA04	121
58	CSI_DATA05	122
59	CSI_DATA06	123
60	CSI_DATA07	124
61	CSI_DATA02	119
62	CSI_DATA01	118
63	CSI_DATA03	120
64	CSI_DATA00	117
65	CSI_PIXCLK	114
66	NAND_DATA07	105
67	NAND_DATA06	104
68	NAND_DATA05	103
69	NAND_DATA04	102
70	LCD_DATA23	92
71	LCD_DATA22	91
72	LCD_DATA21	90
73	LCD_DATA20	89
74	LCD_DATA19	88
76	LCD_DATA18	87
80	GPIO1_IO08	8
81	GPIO1_IO05	5
82	UART5_TX_DATA	30
83	UART1_TX_DATA	16
84	UART5_RX_DATA	31
85	UART1_RX_DATA	17
86	UART2_RTS_B	23
87	UART2_CTS_B	22
88	UART2_RX_DATA	21
89	UART2_TX_DATA	20
90	GPIO1_IO00	0

ピン番号	ピン名	GPIO 番号
91	UART3_CTS_B	26
92	UART3_RTS_B	27
93	UART3_RX_DATA	25
94	UART3_TX_DATA	24

GPIO 番号は次の式より導くことができます。

```
GPIOx_IOy -> (x - 1) * 32 + y
```

例えば、GPIO4_IO8 の場合は、以下のようになります。

```
(4 - 1) * 32 + 8 = 104
```

6.7.1. GPIO クラスディレクトリを作成する

GPIO を利用するには、まず GPIO ディレクトリを作成する必要があります。

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

```
[armadillo ~]# echo 20 > /sys/class/gpio/export
[armadillo ~]# ls /sys/class/gpio/gpio20/
active_low device direction edge subsystem uevent value
```

図 6.25 GPIO クラスディレクトリを作成する

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。

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

```
[armadillo ~]# echo 20 > /sys/class/gpio/unexport
[armadillo ~]# ls /sys/class/gpio/gpio20/
ls: cannot access '/sys/class/gpio/gpio20/': No such file or directory
```

6.7.2. 入出力方向を変更する

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

表 6.11 direction の設定

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

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

図 6.26 GPIO の入出力方向を設定する(INPUT に設定)

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

図 6.27 GPIO の入出力方向を設定する(OUTPUT に設定)

6.7.3. 入力レベルを取得する

GPIO ディレクトリ以下の value ファイルから値を読み出すことによって、入力レベルを取得することができます。"0"は LOW レベル、"1"は HIGH レベルを表わします。入力レベルの取得は入出力方向が INPUT, OUTPUT のどちらでも行うことができます。

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

図 6.28 GPIO の入力レベルを取得する

6.7.4. 出力レベルを設定する

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

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

図 6.29 GPIO の出力レベルを設定する

6.8. RS485

Armadillo-610 拡張ボードは、電気的に絶縁された RS485 のシリアルポートが 1 ポート登載されています。シリアルポートのデバイスドライバは、TTY デバイスとして実装されているため TTY デバイスファイルから制御を行うことができます。

シリアルポートインターフェースと、TTY デバイスファイルの対応を次に示します。

表 6.12 シリアルポートインターフェースと TTY デバイスファイルの対応

シリアルポートインターフェース	TTY デバイスファイル
Armadillo-610 拡張ボード: CON13C	/dev/ttymxc1

6.8.1. RS485 の通信設定を変更する

RS485 設定は、アプリケーションプログラムまたは、Device Tree から変更することができます。

アプリケーションプログラムから変更が可能な RS485 設定とその初期値を「表 6.13. RS485 設定と初期値(アプリケーションプログラム)」に示します。flags は各ビットごとの論理和を示します。

表 6.13 RS485 設定と初期値(アプリケーションプログラム)

設定		説明	初期値
flags	ENABLED(bit0)	0: RS485 無効 1: RS485 有効	1
	RTS_ON_SEND(bit1)	0: データ送信時の RTS(Driver Enable)が Low 1: データ送信時の RTS(Driver Enable)が High	1
	RTS_AFTER_SEND(bit2)	0: データ非送信時の RTS(Driver Enable)が Low 1: データ非送信時の RTS(Driver Enable)が High	0
	RX_DURING_TX(bit4)	0: 半二重通信 1: 全二重通信	0
delay_rts_before_send		送信前遅延時間(ミリ秒)	0
delay_rts_after_send		送信後遅延時間(ミリ秒)	0

flags は初期値を変更しないでください。変更した場合はデータ送信を行うことができなくなります。

RS485 をコンソールとして利用することはできません。

アプリケーションプログラムの作成方法については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/serial/serial-rs485.txt)を参照してください。

Device Tree から変更が可能な RS485 設定とその初期値を「表 6.14. RS485 設定と初期値(Device Tree)」に示します。flags は各ビットごとの論理和を示します。

表 6.14 RS485 設定と初期値(Device Tree)

プロパティ	説明	初期値
rs485-enabled-at-boot-time	未指定: RS485 無効 指定: RS485 有効	指定

プロパティ	説明	初期値
rs485-rts-on-send	未指定: データ送信時の RTS(Driver Enable)が Low 指定: データ送信時の RTS(Driver Enable)が High	指定
rs485-rts-after-send	未指定: データ非送信時の RTS(Driver Enable)が Low 指定: データ非送信時の RTS(Driver Enable)が High	未指定
rs485-rx-during-tx	未指定: 半二重通信 指定: 全二重通信	未指定
rs485-delay-rts-before-send	送信前遅延時間(ミリ秒)	0
rs485-delay-rts-after-send	送信後遅延時間(ミリ秒)	0

rs485-delay-rts-*-send 以外のプロパティの指定を変更しないでください。変更した場合はデータ送信を行うことができなくなります。

RS485 をコンソールとして利用することはできません。

6.9. オーディオ

Armadillo-610 では、オーディオ機能を ALSA デバイスとして利用できます。オーディオインターフェースと ALSA デバイスの対応を次に示します。

表 6.15 オーディオインターフェースと ALSA デバイスの対応

オーディオインターフェース	ALSA デバイス
Armadillo-610 拡張ボード: CON13D	card 0: mqsaudio [mqs-audio]

サンプリング周波数などの対応機能については、「7.3.13. アナログオーディオ」を参照してください。

6.9.1. サウンドを再生する

例として aplay コマンドでサウンドを再生する方法を記載します。まず、次のコマンドを実行して aplay コマンドが含まれる alsa-utils をインストールします。

```
[armadillo ~]# apt-get update && apt-get install alsa-utils
```

図 6.30 alsa-utils のインストール

aplay コマンドでサウンドを再生します。例として sample.wav という名前の WAV ファイルを再生します。

```
[armadillo ~]# aplay sample.wav
```

図 6.31 サウンドの再生

音声ファイルのサンプリング周波数などが非対応の場合でも、自動的にソフトウェアで変換されます。

7. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-610 開発セットの Linux カーネル仕様について説明します。

7.1. デフォルトコンフィギュレーション

工場出荷時の Armadillo-610 開発セットに書き込まれている Linux カーネルは、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-610 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-v4.14-at[VERSION].tar.gz)に含まれる arch/arm/configs/armadillo-640_defconfig です。

Linux カーネルのデフォルトコンフィギュレーションは、Armadillo-640 と Armadillo-610 で共通です。そのため、ファイル名が armadillo-640_defconfig となっています。

armadillo-640_defconfig で有効になっている主要な設定を「表 7.1. Linux カーネル主要設定」に示します。

表 7.1 Linux カーネル主要設定

コンフィグ	説明
VMSPLIT_3G	3G/1G user/kernel split
AEABI	Use the ARM EABI to compile the kernel
COMPACTION	Allow for memory compaction
MIGRATION	Page migration

7.2. デフォルト起動オプション

工場出荷状態の Armadillo-610 開発セットの 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 7.2 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=ttymxc0	起動ログなどが出力されるイニシャルコンソールに ttymxc0 を指定します。
root=/dev/mmcblk0p2	ルートファイルシステムに eMMC を指定します。
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。

7.3. Linux ドライバー覧

Armadillo-610 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うち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。

7.3.1. Armadillo-610

Armadillo-610 のハードウェアの構成情報やピンのマルチプレクス情報、i.MX6ULL の初期化手順などが定義されています。

関連するソースコード
- arch/arm/mach-imx/
- arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dts
- arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dts
- arch/arm/boot/dts/armadillo-610-extboard-eva-common.dtsi
- arch/arm/boot/dts/armadillo-610.dtsi
- arch/arm/boot/dts/imx6ull.dtsi
- arch/arm/boot/dts/imx6ul.dtsi

カーネルコンフィギュレーション

```
System Type --->
  [*] Freescale i.MX family  --->                          <ARCH_MXC>
    [*]   i.MX6 UltraLite support                        <SOC_IMX6UL>
```

7.3.2. UART

Armadillo-610 のシリアルは、i.MX6ULL の UART (Universal Asynchronous Receiver/Transmitter) を利用しています。Armadillo-610 開発セットの標準状態では、シリアルインターフェース(Armadillo-610 拡張ボード: CON3)が UART1 をコンソールとして利用しています。

また、Grove インターフェース(Armadillo-610 拡張ボード: CON7)には UART5 を利用しています。

フォーマット
- データビット長: 7 or 8 ビット
- ストップビット長: 1 or 2 ビット
- パリティ: 偶数 or 奇数 or なし
- フロー制御: CTS/RTS or XON/XOFF or なし
- 最大ボーレート:230.4kbps

関連するソースコード
- drivers/tty/n_null.c
- drivers/tty/n_tty.c
- drivers/tty/pty.c
- drivers/tty/tty_baudrate.c
- drivers/tty/tty_buffer.c
- drivers/tty/tty_io.c
- drivers/tty/tty_ioctl.c
- drivers/tty/tty_jobctrl.c

- drivers/tty/tty_ldisc.c
- drivers/tty/tty_ldsem.c
- drivers/tty/tty_mutex.c
- drivers/tty/tty_port.c
- drivers/tty/serial/earlycon.c
- drivers/tty/serial/serial_core.c
- drivers/tty/serial/serial_mctrl_gpio.c
- drivers/tty/serial/imx.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/serial/fsl-imx-uart.txt
- Documentation/devicetree/bindings/serial/serial.txt

デバイスファイル

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

カーネルコンフィギュレーション

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

7.3.3. Ethernet

Armadillo-610 の Ethernet (LAN) は、i.MX6ULL の ENET(10/100-Mbps Ethernet MAC)を利用しています。

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

関連するソースコード
- drivers/net/Space.c
- drivers/net/loopback.c
- drivers/net/ethernet/freescale/fec_main.c
- drivers/net/ethernet/freescale/fec_ptp.c
- drivers/net/phy/fixed_phy.c

- drivers/net/phy/mdio-boardinfo.c
- drivers/net/phy/mdio_bus.c
- drivers/net/phy/mdio_device.c
- drivers/net/phy/phy-core.c
- drivers/net/phy/phy.c
- drivers/net/phy/phy_device.c
- drivers/net/phy/smsc.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/net/fsl-fec.txt
- Documentation/devicetree/bindings/net/phy.txt

ネットワークデバイス
- eth0

カーネルコンフィギュレーション

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

7.3.4. WLAN

アットマークテクノ製 Armadillo-WLAN モジュール(AWL13)を WLAN インターフェース(Armadillo-610 拡張ボード: CON18)に接続することができます。AWL13 は、「7.3.8. USB ハブ」に示す USB ハブに接続されています。

機能
- チャンネル(2.4GHz): 1-13
- 通信速度(規格上の理論値)
  - IEEE 802.11b: 最大 11 Mbps
  - IEEE 802.11g: 最大 54 Mbps
  - IEEE 802.11n: 最大 72.2 Mbps
- チャンネル帯域: 20MHz
- MIMO(Multi Input Multi Output) 1x1、シングルストリーム
- セキュリティ機能: WEP(64bit, 128bit), WPA-PSK(TKIP,AES), WPA2-PSK(TKIP,AES)

・アクセス方式: インフラストラクチャモード(STA [1], AP [2]対応), アドホックモード

ネットワークデバイス
・awlan0

関連するソースコード
・drivers/net/wireless/awl13/

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                          <NETDEVICES>
    -*- Wireless LAN  --->                                        <WLAN>
      [*]   Armadillo-WLAN(AWL13)                <ARMADILLO_WLAN_AWL13>
        Armadillo-WLAN(AWL13) Driver Options  --->
          Selected AWL13 interface (USB)  --->
            (X) USB                            <ARMADILLO_WLAN_AWL13_USB>
            ( ) SDIO (NOT TESTED)             <ARMADILLO_WLAN_AWL13_SDIO>
```

7.3.5. SD ホスト

Armadillo-610 の SD ホストは、i.MX6ULL の uSDHC (Ultra Secured Digital Host Controller) を利用しています。

Armadillo-610 開発セットでは、SD インターフェース(Armadillo-610: CON1) と SD インターフェース(Armadillo-610 拡張ボード: CON1)が uSDHC2 を共用しています。そのため、どちらか一方しか利用することができません。Armadillo-610 開発セットの標準状態では、SD インターフェース(Armadillo-610: CON1)が有効になっています。

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: Default Speed (24.75MHz), High Speed (49.5MHz)
・カードディテクトサポート

デバイスファイル
・/dev/mmcblk1

関連するソースコード
・drivers/mmc/core/
・drivers/mmc/host/sdhci-esdhc-imx.c
・drivers/mmc/host/sdhci-pltfm.c
・drivers/mmc/host/sdhci.c

Device Tree ドキュメント
・Documentation/devicetree/bindings/mmc/fsl-imx-esdhc.txt
・Documentation/devicetree/bindings/mmc/mmc.txt
・Documentation/devicetree/bindings/regulator/fixed-regulator.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] MMC/SD/SDIO card support  --->                              <MMC>
```

[1]STA=ステーション
[2]AP=アクセスポイント

```
    [*]   MMC block device driver                          <MMC_BLOCK>
    (8)     Number of minors per block device    <MMC_BLOCK_MINORS>
          *** MMC/SD/SDIO Host Controller Drivers ***
    [*]   Secure Digital Host Controller Interface support
                                                           <MMC_SDHCI>
    [*]     SDHCI platform and OF driver helper   <MMC_SDHCI_PLTFM>
    [*]       SDHCI support for the Freescale eSDHC/uSDHC i.MX
controller support
                                                 <MMC_SDHCI_ESDHC_IMX>
```

7.3.6. USB ホスト

Armadillo-610 の USB ホストは、i.MX6ULL の USB-PHY (Universal Serial Bus 2.0 Integrated PHY) および USB (Universal Serial Bus Controller) を利用しています。

Armadillo-610 開発セットでは、「7.3.8. USB ハブ」に示す USB ハブが USB_OTG1 を利用しています。USB インターフェース(Armadillo-610 拡張ボード: CON5)には USB ハブが接続されています。

機能

- Universal Serial Bus Specification Revision 2.0 準拠
- Enhanced Host Controller Interface (EHCI)準拠
- 転送レート: 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/chipidea/
- drivers/usb/host/ehci-hcd.c
- drivers/usb/phy/of.c
- drivers/usb/phy/phy-generic.c
- drivers/usb/phy/phy.c

Device Tree ドキュメント

- Documentation/devicetree/bindings/usb/ci-hdrc-usb2.txt
- Documentation/devicetree/bindings/usb/usbmisc-imx.txt
- Documentation/devicetree/bindings/regulator/fixed-regulator.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] USB support  --->                                  <USB_SUPPORT>
    [*]   Support for Host-side USB                             <USB>
          *** USB Host Controller Drivers ***
    [*]   EHCI HCD (USB 2.0) support                  <USB_EHCI_HCD>
    [*]     Support for Freescale i.MX on-chip EHCI USB controller
                                                      <USB_EHCI_MXC>
    [*]   ChipIdea Highspeed Dual Role Controller     <USB_CHIPIDEA>
    [*]     ChipIdea host controller             <USB_CHIPIDEA_HOST>
```

```
            USB Physical Layer drivers  --->
        [*] NOP USB Transceiver Driver                <NOP_USB_XCEIV>
```

7.3.7. USB OTG

Armadillo-610 の USB OTG は、i.MX6ULL の USB-PHY (Universal Serial Bus 2.0 Integrated PHY) および USB (Universal Serial Bus Controller) を利用しています。

Armadillo-610 開発セットでは、USB インターフェース(Armadillo-610 拡張ボード: CON6)が USB OTG2 を利用しています。

機能

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

関連するソースコード

- drivers/usb/chipidea/
- drivers/usb/gadget/
- drivers/usb/host/ehci-hcd.c
- drivers/usb/phy/of.c
- drivers/usb/phy/phy-generic.c
- drivers/usb/phy/phy.c

Device Tree ドキュメント

- Documentation/devicetree/bindings/usb/ci-hdrc-usb2.txt
- Documentation/devicetree/bindings/usb/usbmisc-imx.txt
- Documentation/devicetree/bindings/regulator/fixed-regulator.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] USB support  --->                                   <USB_SUPPORT>
    [*]   Support for Host-side USB                              <USB>
          *** USB Host Controller Drivers ***
    [*]   EHCI HCD (USB 2.0) support                    <USB_EHCI_HCD>
    [*]     Support for Freescale i.MX on-chip EHCI USB controller
                                                        <USB_EHCI_MXC>
    [*]   ChipIdea Highspeed Dual Role Controller       <USB_CHIPIDEA>
    [*]     ChipIdea host controller                <USB_CHIPIDEA_HOST>
          USB Physical Layer drivers  --->
      [*] NOP USB Transceiver Driver                    <NOP_USB_XCEIV>
    [*]   USB Gadget Support  --->                        <USB_GADGET>
      [*]   USB Gadget precomposed configurations (CDC Composite
Device (Ethernet and ACM))
                                                    <USB_CDC_COMPOSITE>
```

7.3.8. USB ハブ

Armadillo-610 拡張ボードには、Microchip 製 USB2513B が搭載されています。USB2513B には、WLAN インターフェース(Armadillo-610 拡張ボード: CON18)および USB インターフェース(Armadillo-610 拡張ボード: CON5)が接続されています。

機能
- Universal Serial Bus Specification Revision 2.0 準拠

関連するソースコード
- drivers/usb/core/

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] USB support  --->                                   <USB_SUPPORT>
    [*]   Support for Host-side USB                              <USB>
```

7.3.9. リアルタイムクロック

Armadillo-610 のリアルタイムクロックは、i.MX6ULL の RTC 機能を利用しています。

また、Armadillo-610 拡張ボードには、日本電波工業(NDK)製 NR3225SA が搭載されています。NR3225SA は、「7.3.12. I2C」に示す I2C2 (I2C ノード: 2-0032) に接続されています。

LCD インターフェース(Armadillo-610 拡張ボード: CON11)および拡張インターフェース(Armadillo-610 拡張ボード: CON20)には、NR3225SA に接続された I2C と共通の信号線が接続されています。そのため、同時に利用できない場合があります。

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
- /dev/rtc1

関連するソースコード
- drivers/rtc/rtc-lib.c
- drivers/rtc/rtc-core.c
- drivers/rtc/hctosys.c
- drivers/rtc/systohc.c
- drivers/rtc/nvmem.c
- drivers/rtc/rtc-sysfs.c
- drivers/rtc/rtc-proc.c
- drivers/rtc/rtc-dev.c
- drivers/rtc/rtc-nr3225sa.c

・ drivers/rtc/rtc-snvs.c

カーネルコンフィギュレーション

```
Device Drivers --->
  [*] Real Time Clock                                                 <RTC_CLASS>
    [*]   Set system time from RTC on startup and resume                          ⏎
<RTC_HCTOSYS>
    (rtc0)  RTC used to set the system time                                       ⏎
<RTC_HCTOSYS_DEVICE>
    [*]   Set the RTC time based on NTP synchronization                           ⏎
<RTC_SYSTOHC>
    (rtc0)  RTC used to synchronize NTP adjustment                                ⏎
<RTC_SYSTOHC_DEVICE>
    [*]   RTC non volatile storage support                                        ⏎
<RTC_NVMEM>
          *** RTC interfaces ***
    [*]   /sys/class/rtc/rtcN (sysfs)                                             ⏎
<RTC_INTF_SYSFS>
    [*]   /proc/driver/rtc (procfs for rtcN)                                      ⏎
<RTC_INTF_PROC>
    [*]   /dev/rtcN (character devices)                                           ⏎
<RTC_INTF_DEV>
          *** I2C RTC drivers ***
   [*]  NDK NR3225SA                                    <RTC_DRV_NR3225SA>
          *** on-CPU RTC drivers ***
    [*]   Freescale SNVS RTC support                                              ⏎
<RTC_DRV_SNVS>
```

アラーム割り込みは、デバイスファイル経由で利用することができます。

詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/rtc.txt)やサンプルプログラム(tools/testing/selftests/timers/rtctest.c)を参照してください。

7.3.10. LED

Armadillo-610 および Armadillo-610 拡張ボードに搭載されているソフトウェア制御可能な LED には、GPIO が接続されています。Linux では、GPIO 接続用 LED ドライバ (leds-gpio) で制御することができます。

sysfs LED クラスディレクトリ
- /sys/class/leds/green
- /sys/class/leds/yellow

関連するソースコード
- drivers/leds/led-class.c
- drivers/leds/led-core.c
- drivers/leds/led-triggers.c
- drivers/leds/leds-gpio.c
- drivers/leds/trigger/

Device Tree ドキュメント
- Documentation/devicetree/bindings/leds/leds-gpio.txt

カーネルコンフィギュレーション

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

7.3.11. ユーザースイッチ

Armadillo-610 拡張ボードに搭載されているユーザースイッチには、GPIO が接続されています。GPIO が接続されユーザー空間でイベント (Press/Release) を検出することができます。Linux では、GPIO 接続用キーボードドライバ (gpio-keys) で制御することができます。

ユーザースイッチには、次に示すキーコードが割り当てられています。

表 7.3 キーコード

ユーザースイッチ	キーコード	イベントコード
SW1	KEY_ENTER	28

デバイスファイル
- /dev/input/event1 [3]

関連するソースコード
- drivers/input/evdev.c
- drivers/input/input-compat.c
- drivers/input/input.c
- drivers/input/keyboard/gpio_keys.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/input/gpio-keys.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  Input device support  --->
    -*- Generic input layer (needed for keyboard, mouse, ...)
                                                           <INPUT>
    [*]   Event interface                               <INPUT_EVDEV>
          *** Input Device Drivers ***
    [*]   Keyboards  --->                             <INPUT_KEYBOARD>
      [*]   GPIO Buttons                               <KEYBOARD_GPIO>
```

7.3.12. I2C

Armadillo-610 の I2C インターフェースは、i.MX6ULL の I2C(I2C Controller) を利用します。また、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。

[3]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-610 開発セットで利用している I2C バスと、接続される I2C デバイスを次に示します。

表 7.4 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
0(I2C1)	0x08	PF3000
1(I2C2)	0x32	NR3225SA

Armadillo-610 開発セットの標準状態では、CONFIG_I2C_CHARDEV が有効となっているためユーザードライバで I2C デバイスを制御することができます。ユーザードライバを利用する場合は、Linux カーネルで I2C デバイスに対応するデバイスドライバを無効にする必要があります。

機能
- 最大転送レート: 400kbps

デバイスファイル
- /dev/i2c-0 (I2C1)
- /dev/i2c-1 (I2C2)

関連するソースコード
- drivers/i2c/i2c-core.c
- drivers/i2c/i2c-boardinfo.c
- drivers/i2c/i2c-dev.c
- drivers/i2c/algos/i2c-algo-bit.c
- drivers/i2c/busses/i2c-gpio.c
- drivers/i2c/busses/i2c-imx.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/i2c/i2c-imx.txt
- Documentation/devicetree/bindings/i2c/i2c-gpio.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  I2C support  --->
    [*] I2C support                                           <I2C>
    [*]   Enable compatibility bits for old user-space  <I2C_COMPAT>
    [*]   I2C device interface                        <I2C_CHARDEV>
      I2C Algorithms  --->
        -*- I2C bit-banging interfaces                <I2C_ALGOBIT>
      I2C Hardware Bus support  --->
        [*] GPIO-based bitbanging I2C                    <I2C_GPIO>
        [*] IMX I2C interface                             <I2C_IMX>
```

7.3.13. アナログオーディオ

Armadillo-610 のアナログオーディオは、i.MX6ULL の MQS (Medium Quality Sound) を利用しています。

機能
- 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 bit, Little-endian

	・再生(Playback)のみサポート
オーディオデバイス	・hw:0
関連するソースコード	・sound/soc/fsl/imx-mqs.c ・sound/soc/fsl/fsl_asrc.c ・sound/soc/fsl/fsl_asrc_dma.c ・sound/soc/fsl/fsl_sai.c ・sound/soc/fsl/imx-pcm-dma-v2.c

7.3.14. AD コンバーター

Armadillo-610 の AD コンバーターは、i.MX6ULL の ADC (Analog-to-Digital Converter) を利用しています。

Armadillo-610 開発セットでは、Grove インターフェース(Armadillo-610 拡張ボード: CON9)と Grove インターフェース(Armadillo-610 拡張ボード: CON10)が ADC1 を利用しています。

LCD インターフェース(Armadillo-610 拡張ボード: CON11)および拡張インターフェース(Armadillo-610 拡張ボード: CON20)には、AD コンバーターと共通の信号線が接続されています。そのため、同時に利用できない場合があります。

機能	・分解能: 最大 12bit ・サンプリングレート: 最大 1MS/s ・測定範囲: 0V ~ 3.3V
sysfs ファイル	・デバイスを認識した順番で /sys/bus/iio/devices/iio:deviceN (N は'0'からの連番)となります。
デバイスファイル	・デバイスを認識した順番で /dev/iio:deviceN (N は'0'からの連番)となります。
関連するソースコード	・drivers/iio/industrialio-buffer.c ・drivers/iio/industrialio-core.c ・drivers/iio/industrialio-event.c ・drivers/iio/industrialio-trigger.c ・drivers/iio/inkern.c ・drivers/iio/adc/vf610_adc.c
Device Tree ドキュメント	・Documentation/devicetree/bindings/iio/adc/vf610-adc.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Industrial I/O support  --->                                  <IIO>
          Analog to digital converters  --->
      [*] Freescale vf610 ADC driver                          <VF610_ADC>
```

7.3.15. パワーマネジメント

Armadillo-610 のパワーマネジメント機能は、Linux の SPM(System Power Management)および DPM(Device Power Management)を利用しています。パワーマネジメント状態を省電力モードに遷移させることにより、Armadillo-610 の消費電力を抑えることができます。

パワーマネジメント状態を省電力モードに遷移させると、アプリケーションの実行は一時停止し、Linux カーネルはサスペンド状態となります。起床要因が発生すると、Linux カーネルのリジューム処理が行われた後、アプリケーションの実行を再開します。

sysfs ファイル ・ /sys/power/state

関連するソースコード ・ kernel/power/

カーネルコンフィギュレーション

```
Power management options  --->
  [*] Suspend to RAM and standby                              <SUSPEND>
  -*- Device power management core functionality                 <PM>
```

Armadillo-610 が対応するパワーマネジメント状態と、/sys/power/state に書き込む文字列の対応を次に示します。

表 7.5 対応するパワーマネジメント状態

パワーマネジメント状態	文字列	説明
Suspend-to-RAM	mem	最も消費電力を抑えることができる
Power-On Suspend	standby	Suspend-to-RAM よりも短時間で復帰することができ、Suspend-to-Idle よりも消費電力を抑えることができる
Suspend-to-Idle	freeze	最も短時間で復帰することができる

起床要因として利用可能なデバイスは次の通りです。

UART1 (Armadillo-610 拡張ボード: CON3) 起床要因 データ受信

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/                ⏎
drivers/imx-uart/2020000.serial/tty/ttymxc0/power/              ⏎
wakeup
```

USB OTG2 (Armadillo-610 拡張ボード: CON6) 起床要因 USB デバイスの挿抜

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/                ⏎
devices/2184200.usb/power/wakeup
[armadillo ~]# echo enabled > /sys/bus/platform/                ⏎
drivers/ci_hdrc/ci_hdrc.1/power/wakeup
```

```
[armadillo ~]# echo enabled > /sys/bus/platform/
drivers/ci_hdrc/ci_hdrc.1/usb2/power/wakeup
```

RTC(i.MX6ULL)	起床要因	アラーム割り込み
	有効化	`[armadillo ~]# echo enabled > /sys/bus/platform/devices/20cc000.snvs¥:snvs-rtc-lp/power/wakeup`
RTC(NR3225SA)	起床要因	アラーム割り込み
	有効化	`[armadillo ~]# echo enabled > /sys/bus/i2c/devices/1-0032/power/wakeup`

8. Debian ユーザーランド仕様

本章では、工場出荷状態の Armadillo-600 シリーズの Debian ユーザーランドの基本的な仕様について説明します。

8.1. Debian ユーザーランド

Armadillo-600 シリーズの標準ルートファイルシステムは、 32-bit hard-float ARMv7(「armhf」)アーキテクチャ用の Debian GNU/Linux9(コードネーム「stretch」)です。出荷状態、または標準イメージを展開した直後のユーザーランド内には、Armadillo の動作に必要な最小限のパッケージや設定が含まれています。

Armadillo-600 シリーズにインストールされた Debian GNU/Linux 9 は、eMMC または microSD カード上で動作します。Linux カーネルが動作している状態で Armadillo の電源を切断する場合は、必ず「halt」コマンドによる終了を行い、RAM 上にキャッシュされている eMMC または microSD カードへの書き込み処理を完了するようにしてください。再起動を行う場合も同様に、reboot コマンドによる再起動を行なってください。

8.2. パッケージ管理

パッケージ管理システム APT(Advanced Packaging Tool)を使用して、パッケージを管理する方法について記載します。工場出荷状態の Debian には動作に必要な最低限のパッケージしかインストールされていませんが、APT を使用することで、簡単にパッケージを追加することができます。

工場出荷状態では、APT はインターネット上の Debian サイト(HTTP サーバー)から利用可能なパッケージのインデックスを取得します[1]そのため、APT を使用するためにはネットワークを有効化し、インターネットに接続できる状態にしておく必要があります。

ネットワークを有効化する方法については、「6.2. ネットワーク」を参照してください。

システムクロックが大幅にずれた状態で、APT を利用すると警告メッセージが出力される場合があります。事前に「6.6. RTC」を参照してシステムクロックを合わせてください。

apt-get update

パッケージインデックファイルを最新の状態にアップデートします。

引数　なし

使用例

```
[armadillo ~]# apt-get update
```

apt-get upgrade

現在インストールされている全てのパッケージを最新バージョンにアップグレードします。

引数　なし

[1]/etc/apt/sources.list で設定しています。記述ルールなどについては、sources.list のマニュアルページを参照してください。

使用例

```
[armadillo ~]# apt-get upgrade
```

apt-get install [パッケージ名]

引数に指定したパッケージをインストールします。すでにインストール済みの場合はアップグレードします。

引数　パッケージ名(複数指定可能)

使用例

```
[armadillo ~]# apt-get install gcc
```

apt-get remove [パッケージ名]

引数に指定したパッケージをアンインストールします。インストールされていない場合は何もしません。

引数　パッケージ名(複数指定可能)

使用例

```
[armadillo ~]# apt-get remove apache2
```

apt-cache search [キーワード]

引数に指定したキーワードをパッケージ名または説明文に含むパッケージを検索します。

引数　キーワード(正規表現が使用可能)

使用例

```
[armadillo ~]# apt-cache search "Bourne Again SHell"
bash-doc - Documentation and examples for the The ⏎
GNU Bourne Again SHell
bash-static - The GNU Bourne Again SHell (static ⏎
version)
bash - The GNU Bourne Again SHell
```

9. ブートローダー (U-Boot) 仕様

本章では、Armadillo-600 シリーズのブートローダーである U-Boot 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。

Armadillo-200 シリーズ、400 シリーズでは、ブートローダーに Hermit を使用していました。 Armadillo-600 シリーズ では、他の最近の Armadillo シリーズ (Armadillo-IoT など) に合せ、U-Boot を採用しています。

U-Boot は Open Source で開発されているブートローダーで、特に組み込み機器に良く使われています。 U-Boot のマニュアルは、 Denx Software Engineering の U-Boot のページ (https://www.denx.de/wiki/U-Boot/WebHome) からアクセスできます。

9.1. U-Boot の起動モード

U-Boot はブートローダーなので、OS を起動するのが仕事です。しかし OS を起動する以外にも、いろいろと便利な機能が U-Boot には備わっています。

Armadillo-600 シリーズ の U-Boot には 2 つの起動モードがあります。「保守モード」と「オートブートモード」です。 Armadillo-600 シリーズ に接続している USB シリアル変換アダプターのスライドスイッチによって、モードを切り替えることができます。 Armadillo-400 シリーズの Hermit にもあった機能です。このモード切り換えは、 GPIO によって実現しています。 U-Boot 本家にはまだマージされておらず、 Armadillo-600 シリーズ 用の U-Boot に独自実装されている機能です。

また、U-Boot バージョン at5 以降では、Armadillo-600 シリーズ のユーザースイッチ(SW1) [1]を押しながら [2]電源を投入した場合、スライドスイッチの状態に関係なく保守モードでブートローダーが起動します。

ブートローダーが起動すると、USB シリアル変換アダプタのスライドスイッチの状態により、2 つのモードのどちらかに遷移します。USB シリアル変換アダプタのスライドスイッチの詳細については、「スライドスイッチの設定について」を参照してください。

表 9.1 ブートローダー起動モード

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

[1]Armadiilo-610 では開発セット同梱の拡張ボード上に搭載されています。
[2]Armadillo-610 においては、拡張インターフェース(Armadillo-610: CON2)の 43 ピンに High を印加している時、ユーザースイッチを押している状態と同様の挙動をします。

表 9.2 各種スイッチの状態とブートローダー起動モード

起動モードの種別	スライドスイッチ	ユーザースイッチ [a]
保守モード	外側	押す
		押さない
	内側	押す
オートブートモード		押さない

[a]U-Boot バージョン at5 以降のみ対応

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

U-Boot が起動すると、U-Boot のバージョンや、ビルド時間、CPU の情報、DRAM のサイズなどボード情報が表示されます。

```
U-Boot 2018.03-at8 (Feb 17 2020 - 19:19:11 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 9.1 U-Boot の起動

⇒ が U-Boot のプロンプトです。プロンプトが出るのは保守モードの時だけです。Armadillo-600 シリーズ では U-Boot のプロンプトが表示され、コマンド入力を受け付ける状態を「保守モード」と呼んでいます。

9.2. U-Boot の機能

U-Boot の 機能を使うには U-Boot のコマンドプロンプトからコマンドを入力します。コマンドプロンプトは保守モードにすることで表示されます。

U-Boot の保守モードでは、U-Boot のバージョン番号を表示したり、あるメモリアドレスの値を表示したり Linux カーネルの起動オプションの設定などを行うことができます。保守モードで利用できる有用なコマンドは、プロンプトで help と入力すると表示されます。

```
=> help
?       - alias for 'help'
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
```

```
bootd   - boot default, i.e., run 'bootcmd'
bootefi - Boots an EFI payload from memory
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
clocks  - display clocks
cmp     - memory compare
config  - print .config
cp      - memory copy
crc32   - checksum calculation
dcache  - enable or disable data cache
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
editenv - edit environment variable
env     - environment handling commands
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
ext4load- load binary file from a Ext4 filesystem
ext4ls  - list files in a directory (default /)
ext4size- determine a file's size
ext4write- create a file in the root directory
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
fatsize - determine a file's size
fdt     - flattened device tree utility commands
fstype  - Look up a filesystem type
fsuuid  - Look up a filesystem UUID
fuse    - Fuse sub-system
grepenv - search environment variables
help    - print command description/usage
icache  - enable or disable instruction cache
load    - load binary file from a filesystem
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loadx   - load binary file over serial line (xmodem mode)
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
loopw   - infinite write loop on address range
ls      - list files in a directory (default /)
md      - memory display
md5sum  - compute MD5 message digest
meminfo - display memory information
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - display MMC info
mw      - memory write (fill)
nm      - memory modify (constant address)
part    - disk partition related commands
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
save    - save file to a filesystem
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sha1sum - compute SHA1 message digest
size    - determine a file's size
strings - display strings
```

```
tftpboot- boot image via network using TFTP protocol
usb     - USB sub-system
version - print monitor, compiler and linker version
=>
```

図 9.2 U-Boot コマンドのヘルプを表示

各コマンドのヘルプを表示するには U-Boot コマンドのヘルプを表示のようにします。

```
=> help [コマンド]
```

図 9.3 U-Boot コマンドのヘルプを表示

良く使うと思われるコマンドを以下で説明します。

boot	環境変数 bootcmd に指定されているコマンドを実行。デフォルトでは Linux を起動。オートブートモード時はこのコマンドが呼ばれている
env	U-Boot の環境変数に関連したコマンド (下記で詳しく説明)
ext4load	Ext4 ファイルシステムからファイルをメモリにロード
ext4ls	Ext4 ファイルシステムにあるファイルをリスト
fuse	CPU の内部 Fuse の値の読み書き
help	コマンド一覧、または指定されたコマンドのヘルプを表示
mmc	MMC/SD 関連のコマンド群 「9.2.2. mmc コマンド」 で詳しく説明
ping	ICMP ECHO_REQUEST を送信
run	環境変数に登録されているコマンドの実行
tftpboot	TFTP による起動
usb	USB 関連のコマンド群
version	U-Boot のバージョン番号表示

help で表示されるコマンドには、Git のようにサブコマンドを持つものがあります。 env や usb などがそうです。 help env とすることで、 指定したコマンドのサブコマンドが表示されます。

9.2.1. env コマンド

```
=> help env
env - environment handling commands

Usage:
env default [-f] -a - [forcibly] reset default environment
env default [-f] var [...] - [forcibly] reset variable(s) to their default values
env delete [-f] var [...] - [forcibly] delete variable(s)
env edit name - edit environment variable
env exists name - tests for existence of variable
```

```
env export [-t | -b | -c] [-s size] addr [var ...] - export environment
env grep [-e] [-n | -v | -b] string [...] - search environment
env import [-d] [-t [-r] | -b | -c] addr [size] - import environment
env print [-a | name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env set [-f] name [arg ...]

=>
```

図 9.4 env コマンドのヘルプを表示

env default	環境変数をリセット
env delete	指定した環境変数を削除
env grep	指定した文字列を環境変数から検索
env print	指定した環境変数を表示します。指定が無ければ、すべて表示。 printenv コマンドと同じ
env save	環境変数を eMMC に保存。 saveenv コマンドと同じ。「9.3. U-Boot の環境変数」で詳しく説明
env set	環境変数を設定。 setenv コマンドと同じ

9.2.2. mmc コマンド

```
=> mmc
mmc - MMC sub system

Usage:
mmc info - display info of the current MMC device
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
mmc hwpartition [args...] - does hardware partitioning
  arguments (sizes in 512-byte blocks):
    [user [enh start cnt] [wrrel {on|off}]] - sets user data area attributes
    [gp1|gp2|gp3|gp4 cnt [enh] [wrrel {on|off}]] - general purpose partition
    [check|set|complete] - mode, complete set partitioning completed
  WARNING: Partitioning is a write-once setting once it is set to complete.
  Power cycling is required to initialize partitions after set to complete.
mmc setdsr <value> - set DSR register value

=>
```

図 9.5 mmc コマンドのヘルプを表示

mmc info	現在指定されている MMC デバイスの情報を表示

`mmc list`　　ボード上の MMC デバイスのリストを表示

`mmc dev`　　現在指定されている MMC デバイスを表示。または指定された番号で示される MMC デバイスを選択

9.3. U-Boot の環境変数

U-Boot は、環境変数を持つことができます。デフォルトの環境変数は、U-Boot をビルドした時に値が決定します。実行に環境変数を変更したり、変更した環境変数を保存したりすることも可能です。

`env print` コマンドで、現在設定されているすべての環境変数を表示できます。

```
=> env print
baudrate=115200
bootcmd=run setup_mmcargs; ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load m
mc 0:2 0x83000000 /boot/${fdt_file}; bootm ${loadaddr} - 0x83000000;
bootdelay=0
enable_pf3000_lpm=no
ethact=FEC
fdt_file=a640.dtb ❶
loadaddr=0x82000000
setup_bootcmd_usb=setenv bootcmd run setup_usbargs¥¥; usb start¥¥; ext4load usb
0:2 ¥¥${loadaddr} /boot/uImage¥¥; ext4load usb 0:2 0x83000000 /boot/¥¥${fdt_file
}¥¥; usb stop¥¥; bootm ¥¥${loadaddr} - 0x83000000¥¥;
setup_mmcargs=setenv bootargs root=/dev/mmcblk0p2 rootwait ${optargs};
setup_usbargs=setenv bootargs root=/dev/sda2 rootwait rw ${optargs};
stderr=serial
stdin=serial
stdout=serial
stop_nr3225sa_alarm=no;
tftpboot=tftpboot uImage; tftpboot 0x83000000 ${fdt_file}; bootm ${loadaddr} - 0
x83000000;
usbboot=run setup_bootcmd_usb; boot;

Environment size: 826/524284 bytes
=>
```

❶　fdt_file 変数のデフォルト値は、Armadillo-640 では "a640.dtb"、Armadillo-610 では "a610.dtb" となります。

または `env print` コマンドで変数を指定すると、その変数の値だけを表示することも可能です。

```
=> env print loadaddr
loadaddr=0x82000000
=>
```

新しく変数を追加したり、すでに設定されている値を変更するには `env set` を使います。

```
=> env set hello world
=> env print hello
hello=world
=> env set hello armadillo
=> env print hello
```

```
hello=armadillo
=>
```

不要な変数を削除するには env delete を使います。

```
=> env delete hello
=> env print hello
## Error: "hello" not defined
=>
```

環境変数を保存するには env save コマンドを使います。

```
=> env set hello armadillo
=> env save
Saving Environment to MMC... Writing to MMC(0)... OK
=>

....電源入れなおし....

U-Boot 2018.03-at8 (Feb 14 2020 - 10:21:57 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC

=> env print hello
hello=armadillo
=>
```

表示された文字からも分るように、Armadillo-600 シリーズでは環境変数を MMC の 0 番、つまりオンボード eMMC に保存します。U-Boot の環境変数が保存される場所は、オンボード eMMC の先頭から 512 KByte オフセットです。eMMC の 1 MByte オフセットから第 1 パーティションが始まっているので、環境変数を保存できるのは 512 KByte になります。 eMMC のアドレスマップについては「表 3.5. eMMC メモリマップ」 を参照してください。

環境変数をデフォルト値に戻したい場合は env default コマンドを使います。

```
=> env print bootdelay
bootdelay=0
=> env set bootdelay 1
=> env print bootdelay
bootdelay=1
=> env default bootdelay
=> env print bootdelay
```

```
bootdelay=0
=>
```

もし、すべての環境変数をデフォルト値に戻したい場合は、 オプション -a を付けてください。

```
=> env default -a
## Resetting to default environment
=>
```

図 9.6 全ての環境変数をデフォルト値に戻す

特定の変数は、値を変更するタイミングで U-Boot の関数が実行されます。環境変数 baudrate もそのうちの 1 つです。つまり値が変更されるだけでなく、実際にボーレートが変更されます。

他にも値の変更できなくなっていたり、変更回数が決まっているものもあります。環境変数 ethaddr もその 1 つです。それらの変数は、変更する必要はあまり無いと思いますが、もし変更する場合には U-Boot のマニュアルを参照してください。

9.4. U-Boot が Linux を起動する仕組み

U-Boot は、 boot コマンドによって、OS を起動します。 boot コマンドは、 環境変数 bootcmd に登録されているコマンドを実行するコマンドです。つまり run bootcmd と同じ意味です。

```
=> env print loadaddr
loadaddr=0x82000000
=> env print bootcmd
bootcmd=run setup_mmcargs; ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load m
mc 0:2 0x83000000 /boot/${fdt_file}; bootm ${loadaddr} - 0x83000000;
=> run bootcmd
7229976 bytes read in 221 ms (31.2 MiB/s)
28291 bytes read in 55 ms (502 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at28
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7229912 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02e82 ... OK

Starting kernel ...
```

環境変数 bootcmd には複数のコマンドが ; で区切られて並んでいます。 U-Boot の run コマンドは、並んでいるコマンドを順次実行していきます。

```
bootcmd=ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load mmc 0:2 0x83000000 /boot/${fdt_file};
bootm ${loadaddr} - 0x83000000;
```

最初のコマンドは ext4load です。このコマンドは、**0** 番目の **MMC** デバイスにある、 **2** 番目パーティションにアクセスし **/boot/uImage** を、環境変数 **loadaddr** で指定されているメモリアドレスにロードします。コマンドで環境変数を参照するには、変数を ${…} でくくります。出荷状態では、オンボード eMMC の第 2 パーティションが EXT4 でフォーマットされており、 /boot/ ディレクトリに uImage というファイル名で Linux カーネルが配置されています。つまり最初のコマンドは、Linux カーネルをメモリにロードしているわけです。

```
=> help ext4load
ext4load - load binary file from a Ext4 filesystem

Usage:
ext4load <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]
    - load binary file 'filename' from 'dev' on 'interface'
      to address 'addr' from ext4 filesystem
=>
```

同じコマンドを U-Boot のプロンプトで手で実行しても、同じ動作結果になります。コマンドを 1 つだけ入力するときは ; を付けても付けなくても問題ありません。

```
=> ext4load mmc 0:2 ${loadaddr} /boot/uImage
6601520 bytes read in 206 ms (30.6 MiB/s)
=>
```

Armadillo-600 シリーズ では 512 MByte (0x20000000) の DRAM が 0x80000000 から 0xA0000000 までマップされています。この情報は bdinfo コマンドで確認することができます。

次のコマンドも同じく ext4load で、環境変数 fdt_file で指定されているファイルをメモリアドレス 0x83000000 にロードしている事が分ります。環境変数 fdt_file は、他の環境変数とは異なり、デフォルト値が Armadillo の起動時に決まります。Armadillo-640 では fdt_file=a640.dtb 、Armadillo-610 では fdt_file=a610.dtb となります。このファイルは「Device Tree Blob (dtb)」というもので、ボードのデバイス情報が記録されています。最近の Linux カーネルでは必須のファイルです。このファイルのロードアドレスは、 uImage と異なり変数になっておらず値 (0x83000000) がそのまま記載されています。ファイルがある場所は、 uImage と同じくオンボード eMMC の第 2 パーティションで /boot/ です。

```
=> ext4load mmc 0:2 0x83000000 /boot/${fdt_file}
27838 bytes read in 55 ms (494.1 KiB/s)
=>
```

これで、Linux を起動するために必要なファイル 2 つ (uImage と a640.dtb または a610.dtb) をメモリに配置することができました。次のコマンド bootm で実際に Linux をブートします。 bootm コマンドは、引数に 3 つのアドレスを取ります。

```
bootm ${loadaddr} - 0x83000000
```

1 つ目が、Linux カーネルを置いたアドレス (${loadaddr} == 0x82000000)、2 つ目が initrd のアドレスですが Armadillo-600 シリーズ では使用してないので - を書きます。3 つ目が Device Tree Blob を置いたアドレス (0x83000000) です。 U-Boot は Linux 以外も起動することができるので、 bootm のヘルプでは Linux カーネルを "Application image" と記載しています。

```
=> help bootm
bootm - boot application image from memory

Usage:
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image
        When booting a Linux kernel which requires a flat device-tree
        a third argument is required which is the address of the
        device-tree blob. To boot that kernel without an initrd image,
        use a '-' for the second argument. If you do not pass a third
        a bd_info struct will be passed instead
   :
   :
=>
```

実際に bootm コマンドを使って Linux を起動してみます。ここではあえて loadaddr ではなく 0x82000000 を入力してみます。

```
=> bootm 0x82000000 - 0x83000000
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at28
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7229912 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02e82 ... OK

Starting kernel ...
```

このように、手で入力しても手順さえ間違わなければ、Linux を起動することができます。もちろん uImage の場所やファイル名を変更しても、 U-Boot で正しく指定すれば同様に動作します。

9.5. U-Boot から見た eMMC / SD

起動コマンドから分るように Armadillo-600 シリーズ では、オンボード eMMC が 0 番目の MMC デバイスです。これは mmc list コマンドでも確認できます。

```
=> mmc list
FSL_SDHC: 0 (eMMC)
FSL_SDHC: 1
```

MMC デバイス情報は mmc info コマンドで表示できます。 mmc info コマンドは、引数でどのデバイスかを指定するのではなく、事前に mmc dev コマンドで使うデバイスを指定しておく必要があります。

```
=> mmc dev
switch to partitions #0, OK
mmc0(part 0) is current device
```

mmc dev コマンドでデバイス番号を指定しないと、現在指定されているデバイスを表示します。

```
=> mmc info
Device: FSL_SDHC
Manufacturer ID: 13
OEM: 14e
Name: S0J35
Bus Speed: 52000000
Mode : MMC High Speed (52MHz)
Rd Block Len: 512
MMC version 5.1
High Capacity: Yes
Capacity: 3.5 GiB
Bus Width: 8-bit
Erase Group Size: 512 KiB
HC WP Group Size: 8 MiB
User Capacity: 3.5 GiB ENH WRREL
User Enhanced Start: 0 Bytes
User Enhanced Size: 3.5 GiB
Boot Capacity: 31.5 MiB ENH
RPMB Capacity: 4 MiB ENH
GP1 Capacity: 8 MiB ENH WRREL
GP2 Capacity: 8 MiB ENH WRREL
GP3 Capacity: 8 MiB ENH WRREL
GP4 Capacity: 8 MiB ENH WRREL
```

これが、オンボード eMMC の情報です。次に 1 番目のデバイスを指定してみます。microSD カードが装着されていれば、次のように表示されます。

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> mmc info
Device: FSL_SDHC
Manufacturer ID: 2
OEM: 544d
Name: SA08G
Bus Speed: 50000000
Mode : SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
```

```
High Capacity: Yes
Capacity: 7.2 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
=>
```

カードの種類によって表示される値は異なります。もし SD カードに入れている Linux カーネルを起動する場合は、先の ext4load コマンドでデバイス 1 を指定すれば良いことが分ります。

```
=> ext4load mmc 1:1 ${loadaddr} /boot/uImage
```

上記の例は、 microSD の 1 番目のパーティションにある /boot/uImage をロードする例です。

9.6. Linux カーネル起動オプション

Linux カーネルは、ブートローダーから「カーネルパラメーター」と呼ばれる起動オプションを受けとる事ができます。U-Boot では 環境変数 bootargs に入っている文字列を Linux に渡します。

Armadillo-600 シリーズ では bootargs の値は setup_mmcargs 変数の内部で一旦 setenv することで以下のように設定しています

```
setup_mmcargs=setenv bootargs root=/dev/mmcblk0p2 rootwait ${optargs};
```

この文字列で、ルートファイルシステムが /dev/mmcblk0p2 であり、mmcblk0 がみつかるまで待つ (rootwait) ように指示しています。 前述の通り Armadillo-600 シリーズ では オンボード eMMC が 0 番目の MMC デバイスです。Armadillo-600 シリーズでは初期出荷時に、オンボード eMMC の第 2 パーティションに Debian をインストールして出荷しているので 「オンボード eMMC、つまり 0 番目の MMC デバイスの第 2 パーティション」を表わす /dev/mmcblk0p2 を指定しています。

もし microSD カードに、Debian を構築し、そのパーティションをルートファイルシステムとして指定したい場合、

```
=> setenv setup_mmcargs setenv bootargs root=/dev/mmcblk1p1 rootwait ${optargs};
```

としてください。 前述の通り microSD は 1 番目の MMC デバイスなので root=/dev/mmcblk1p1 で microSD の第 1 パーティションという意味になります。

Linux カーネル起動オプション

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。

10. ビルド手順

本章では、工場出荷イメージと同じイメージを作成する手順について説明します。

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、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-610 ソフトウェアダウンロード
>
> https://armadillo.atmark-techno.com/armadillo-610/resources/software

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。

10.1. 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「U-Boot」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。

1. ソースコードの準備

   U-Boot のソースコードアーカイブを準備し展開します。

   ```
   [ATDE ~]$ tar xf u-boot-a600-v2018.03-at[version].tar.gz
   ```

2. デフォルトコンフィギュレーションの適用

   U-Boot ディレクトリに入り、Armadillo-610 用のデフォルトコンフィギュレーションを適用します。

   ```
   [ATDE ~]$ cd u-boot-a600-v2018.03-at[version]
   [ATDE ~/u-boot-a600-v2018.03-at[version]]$ make ARCH=arm armadillo-640_defconfig
   ```

   デフォルトコンフィギュレーションは、Armadillo-610 と Armadillo-640 で共通です。

3. ビルド

ビルドには make コマンドを利用します。

```
[ATDE ~/u-boot-a600-v2018.03-at[version]]$ make CROSS_COMPILE=arm-linux-gnueabihf-
```

4. イメージファイルの生成確認

   ビルドが終了すると、U-Boot ディレクトリにイメージファイルが作成されています。

```
[ATDE ~/u-boot-a600-v2018.03-at[version]]$ ls u-boot.imx
u-boot.imx
```

10.2. Linux カーネルをビルドする

ここでは、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。

ビルドに必要なファイル
- linux-v4.14-at[version].tar.gz
- initramfs_a600-[version].cpio.gz

10.2.1. 手順：Linux カーネルをビルド

1. アーカイブの展開

   Linux カーネルのソースコードアーカイブを展開します。

```
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version].tar.gz
[ATDE ~]$ tar xf linux-v4.14-at[version].tar.gz
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version]  linux-v4.14-at[version].tar.gz
```

2. initramfs アーカイブへのシンボリックリンク作成

   Linux カーネルディレクトリに移動して、initramfs アーカイブへのシンボリックリンク作成します

```
[ATDE ~]$ cd linux-v4.14-at[version]
[ATDE ~/linux-v4.14-at[version]]$ ln -s ../initramfs_a600-[version].cpio.gz
initramfs_a600.cpio.gz
```

3. コンフィギュレーション

   コンフィギュレーションをします。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm armadillo-640_defconfig
```

デフォルトコンフィギュレーションは、Armadillo-610 と Armadillo-640 で共通です。

4. ビルド

   ビルドするには、次のようにコマンドを実行します。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

5. イメージファイルの生成確認

   ビルドが終了すると、 arch/arm/boot/ ディレクトリと、 arch/arm/boot/dts/ 以下にイメージファイル(Linux カーネルと DTB)が作成されています。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/uImage
arch/arm/boot/uImage
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-at-dtweb.dtb
arch/arm/boot/dts/armadillo-610-at-dtweb.dtb ❶
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dtb
arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dtb ❷
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dtb
arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dtb ❸
```

❶ at-dtweb を利用して Device Tree をカスタマイズする場合に使用します。詳しくは 「20.3. Device Tree をカスタマイズする」 を参照してください。

❷ Armadillo-610 拡張ボードの LCD インターフェースを利用する場合に使用します。

❸ Armadillo-610 拡張ボードの Grove インターフェースを利用する場合に使用します。

10.3. Debian GNU/Linux ルートファイルシステムをビルドする

ここでは、at-debian-builder を使って、Debian GNU/Linux ルートファイルシステムを構築する方法を示します。at-debian-builder は ATDE 等の PC で動作している Linux 上で Armadillo 用の armhf アーキテクチャに対応した Debian GNU/Linux ルートファイルシステムを構築することができるツールです。

Armadillo を一度起動した後のルートファイルシステム上には、使い方によっては ssh の秘密鍵や、動作ログ、シェルのコマンド履歴、ハードウェアの UUID に紐付く設定ファイル等が生成されています。そのまま、他の Armadillo にルートファイルシステムをコピーした場合は、鍵の流出や UUID の不一致による動作の相違が起きる可能性があります。そのため、量産等に使用するルートファイルシステムは新規に at-debian-builder を使って構築することをお勧めします。

Debian GNU/Linux 9(コードネーム stretch) ルートファイルシステムを構築するには、at-debian-builder のバージョンが v1.x.x (x は 0 以上の数字)である必要があります。

10.3.1. 出荷状態のルートファイルシステムアーカイブを構築する

出荷状態のルートファイルシステムアーカイブを構築する手順を次に示します。パッケージをインターネット上から取得するため回線速度に依存しますが、40 分程度かかります。

```
[ATDE ~]$ tar xf at-debian-builder-[VERSION].tar.gz
[ATDE ~]$ cd at-debian-builder-[VERSION]
[ATDE ~]$ sudo ./build.sh a600
```

図 10.1 出荷状態のルートファイルシステムアーカイブを構築する手順

10.3.2. カスタマイズされたルートファイルシステムアーカイブを構築する

at-debian-builder-**[VERSION]**/a600_resouces 内のファイルを変更し、build.sh を実行することで、ルートファイルシステムをカスタマイズすることができます。

10.3.2.1. ファイル/ディレクトリを追加する

a600_resources/ 以下に配置したファイルやディレクトリは resources ディレクトリを除いて、そのまま、ルートファイルシステムの直下にコピーされます。ファイルの UID と GID は共に root になります。

10.3.2.2. パッケージを変更する

a600_resources/resources/packages を変更することで、ルートファイルシステムにインストールするパッケージをカスタマイズすることができます。

パッケージ名は 1 行に 1 つ書くことができます。パッケージ名は Armadillo 上で"apt-get install"の引数に与えることのできる正しい名前で記載してください。

誤ったパッケージ名を指定した場合は、ビルドログに以下のようなエラーメッセージが表示されて当該のパッケージが含まれないアーカイブが生成されます。

```
E: Unable to locate package XXXXX
```

図 10.2 誤ったパッケージ名を指定した場合に起きるエラーメッセージ

パッケージに依存する他のパッケージは明記しなくても、apt によって自動的にインストールされます。また、apt や dpkg 等の Debian GNU/Linux の根幹となるパッケージも自動的にインストールされます。

openssh-server のような「パッケージのインストールの際に、自動的に秘密鍵を生成する」パッケージは、基本的に packages には追加せず、Armadillo を起動した後に "apt-get install" を使って個別にインストールしてください。

openssh-server を packages に追加した場合、構築したルートファイルシステムアーカイブを書き込んだ全ての Armadillo に、単一の公開鍵を使ってログインすることができてしまいます。もし、意図的に、複数の Armadillo で同一の秘密鍵を利用したい場合、脆弱性となり得ることを理解して適切な対策をとった上で利用してください。

11. イメージファイルの書き換え方法

本章では、Armadillo-610 の内蔵ストレージ(eMMC)に書き込まれているイメージファイルを書き換える手順について説明します。

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、"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、最新バージョンを利用することを推奨します。

> Armadillo サイト - Armadillo-610 ソフトウェアダウンロード
>
> https://armadillo.atmark-techno.com/armadillo-610/resources/software

11.1. インストールディスクを使用する

インストールディスクを使用すると、内蔵ストレージ上のすべてのイメージをまとめて書き換えることができます。 Armadillo がソフトウェアの問題により起動しなくなった場合の復旧方法としてもご使用頂けます。

内蔵ストレージに保存されている、すべてのイメージファイルが上書きされるため、既に保存されているデータやアプリケーションなどは削除されます。

特定のイメージのみ書き換えたい場合には「11.2. 特定のイメージファイルだけを書き換える」を参照してください。

インストールディスクは ATDE で作成します。インストールディスクの作成に使用するファイルを次に示します。

表 11.1 インストールディスク作成に使用するファイル

ファイル	ファイル名
インストールディスクイメージ	install-disk-sd-a600-**[version]**.img

11.1.1. インストールディスクの作成

1. 512 MB 以上の SD カードを用意してください。
2. ATDE に SD カードを接続します。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。
3. SD カードがマウントされている場合、アンマウントします。

```
[ATDE ~]$ mount
(省略)
/dev/sdb1 on /media/atmark/B18A-3218 type vfat
```

```
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=437,iocharset=u ⏎
tf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)
[ATDE ~]$ sudo umount /dev/sdb1
```

4. SD カードにインストールディスクイメージを書き込みます。

```
[ATDE ~]$ sudo dd if=install-disk-sd-a600-[version].img of=/dev/sdb bs=4M conv=fsync
127+1 レコード入力
127+1 レコード出力
536870400 バイト (537 MB) コピーされました、 65.6377 秒、 8.2 MB/秒
```

11.1.2. インストールの実行

1. Armadillo の電源が切断されていることを確認します。接続されていた場合は、電源を切断してください。
2. USB シリアル変換アダプタのスライドスイッチを確認します。スライドスイッチが「図 4.18. スライドスイッチの設定」の 1 側に設定されている事を確認してください。
3. インストールディスクを使用して SD ブートを行います。JP1 をジャンパでショートし、インストールディスクを Armadillo に接続してください。
4. Armadillo に電源を投入すると microSD カードからブートローダーが起動し、次に示すログが表示されます。

```
U-Boot 2018.03-at8 installer+ (Feb 17 2020 - 19:19:21 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC

=>
```

5. 次のように"boot"コマンドを実行するとインストールが始まり、自動的に eMMC が書き換えられます。

```
=> boot
6601520 bytes read in 344 ms (18.3 MiB/s)
25797 bytes read in 62 ms (406.3 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at19
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    6601456 Bytes = 6.3 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
```

```
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef024c4 ... OK

Starting kernel ...
:（省略）
*+*+* Install Start!! *+*+*
```

6. 以下のようにメッセージが表示されるとインストール完了です。電源を切断してください。

```
*+*+* Install Completed!! *+*+*
```

11.1.3. LED 点灯パターンによるインストールの進捗表示

LED 点灯パターンによって、インストールの進捗状況を確認することができます。

インストールの進捗と LED 点灯パターンの関係を次に示します。

表 11.2 インストールの進捗と LED 点灯パターン

進捗	ユーザー LED 黄
実行中	点滅
正常終了	点灯
異常終了	消灯

11.2. 特定のイメージファイルだけを書き換える

Armadillo-610 が起動した状態であれば、特定のイメージファイルだけを書き換えることができます。

イメージファイルと書き込み先の対応を次に示します。

表 11.3 イメージファイルと書き込み先の対応

名称	ファイル名	ストレージ	デバイスファイル
ブートローダーイメージ	u-boot-a600-v2018.03-at[version].imx	eMMC	/dev/mmcblk0
Linux カーネルイメージ	uImage-a600-v4.14-at[version]		/dev/mmcblk0p2
Device Tree Blob(LCD 用) [a]	armadillo-610-extboard-eva-lcd-v4.14-at[version].dtb		/dev/mmcblk0p2
Device Tree Blob(Grove 用) [a]	armadillo-610-extboard-eva-grove-v4.14-at[version].dtb		/dev/mmcblk0p2
Debian GNU/Linux ルートファイルシステム	debian-stretch-armhf-a600-[version].tar.gz		/dev/mmcblk0p2

[a]どちらか一方を書き込みます。工場出荷時の Armadillo-610 開発セットには Device Tree Blob(Grove 用)が書き込まれています。

11.2.1. ブートローダーイメージの書き換え

ブートローダーイメージの書き換え方法を次に示します。

```
[armadillo ~]# dd if=u-boot-a600-v2018.03-at[version].imx of=/dev/mmcblk0 bs=1k seek=1 conv=fsync ⏎
❶
```

```
275+0 records in
275+0 records out
281600 bytes (282 kB, 275 KiB) copied, 0.0310393 s, 9.1 MB/s
```

❶　MTD のブロックデバイスの先頭からブートローダーイメージを書き込みます。

製品アップデートでリリースされた最新のブートローダーイメージに書き換えを行った場合は「図 9.6. 全ての環境変数をデフォルト値に戻す」を参照して環境変数をデフォルト値に戻してください。

新しくリリースされた最新のブートローダーイメージは、環境変数のデフォルト値が更新されることがあります。書き替え前のブートローダーによって生成された環境変数で、最新のブートローダーを動作させると不具合が起こる可能性があります。

11.2.2. Linux カーネルイメージの書き換え

Linux カーネルイメージの書き換え方法を次に示します。

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt ❶
[armadillo ~]# cp uImage-a600-v4.14-at[version] /mnt/boot/uImage ❷
[armadillo ~]# umount /mnt ❸
```

❶　eMMC の第 2 パーティションを/mnt/ディレクトリにマウントします。

❷　Linux カーネルイメージを/mnt/boot/ディレクトリにコピーします。

❸　/mnt/ディレクトリにマウントした eMMC の第 2 パーティションをアンマウントします。

11.2.3. DTB の書き換え

DTB の書き換え方法を次に示します。

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt ❶
[armadillo ~]# cp armadillo-610-extboard-eva-grove-v4.14-at[version].dtb /mnt/boot/a610.dtb ❷
[armadillo ~]# umount /mnt ❸
```

❶　eMMC の第 2 パーティションを/mnt/ディレクトリにマウントします。

❷　DTB を a610.dtb にリネームして/mnt/boot/ディレクトリにコピーします。

LCD インターフェース(Armadillo-610 拡張ボード: CON11)を利用する場合は、 `armadillo-610-extboard-eva-grove-v4.14-at[version].dtb` の代わりに `armadillo-610-extboard-eva-lcd-v4.14-at[version].dtb` を使用してください。

❸　/mnt/ディレクトリにマウントした eMMC の第 2 パーティションをアンマウントします。

11.2.4. ルートファイルシステムの書き換え

eMMC 上のルートファイルシステムを書き換えるには、SD ブートを行う必要があります。ブートディスクの作成方法や SD ブートの実行方法ついては「14. SD ブートの活用」を参照してください。

SD ブートした Armadillo で、eMMC 上のルートファイルシステムを書き換える手順を次に示します。

1. Debian GNU/Linux ルートファイルシステムアーカイブを Armadillo にコピーします。例として USB メモリを利用する方法を記載します。

   ATDE にある Debian GNU/Linux ルートファイルシステムアーカイブを USB メモリの第 1 パーティションにコピーするには次のコマンドを実行します。

   ```
   [ATDE ~]$ sudo umount /dev/sdb1 ❶
   [ATDE ~]$ sudo mount /dev/sdb1 /media ❷
   [ATDE ~]$ sudo cp debian-stretch-armhf-a600-[version].tar.gz /media ❸
   [ATDE ~]$ sudo umount /media ❹
   ```

   ❶ USB メモリ接続時に自動で USB メモリの第 1 パーティションがマウントされることがあるため、一旦アンマウントします。

   ❷ USB メモリの第 1 パーティションを/media/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/media/ディレクトリ以下にコピーします。

   ❹ /media/ディレクトリにマウントした USB メモリの第 1 パーティションをアンマウントします。

   USB メモリの第 1 パーティションにある Debian GNU/Linux ルートファイルシステムアーカイブを Armadillo にコピーするには次のコマンドを実行します。

   ```
   [armadillo ~]# mount /dev/sda1 /media ❶
   [armadillo ~]# cp /media/debian-stretch-armhf-a600-[version].tar.gz . ❷
   [armadillo ~]# umount /media ❸
   ```

   ❶ USB メモリの第 1 パーティションを/media/ディレクトリにマウントします。

   ❷ ルートファイルシステムアーカイブをカレントディレクトリ以下にコピーします。

   ❸ /media/ディレクトリにマウントした USB メモリの第 1 パーティションをアンマウントします。

2. ルートファイルシステムを eMMC の第 2 パーティションに再構築します。

   ```
   [armadillo ~]# mount -t ext4 /dev/mmcblk0p2 /mnt ❶
   [armadillo ~]# cd /mnt
   [armadillo /mnt]# ls | grep -v -E 'boot|lost¥+found' | xargs rm -rf ❷
   [armadillo /mnt]# cd -
   [armadillo ~]# tar zxf debian-stretch-armhf-a600-[version].tar.gz -C /mnt ❸
   [armadillo ~]# umount /mnt ❹
   ```

❶ eMMC の第 2 パーティションを/mnt/ディレクトリにマウントします。

❷ eMMC の第 2 パーティションの boot、lost+found ディレクトリ以外のファイル・ディレクトリを削除します。

❸ ルートファイルシステムアーカイブを/mnt/ディレクトリに展開します。

❹ /mnt/ディレクトリにマウントした eMMC の第 2 パーティションをアンマウントします。

12. 開発の基本的な流れ

この章では Armadillo を使ったアプリケーションソフトウェアの開発方法について説明します。

Armadillo を使ったアプリケーションソフトウェア開発には、Ruby 等の軽量スクリプト言語を使うことができます。

もちろん、Ruby に限らず、Debian の提供する豊富なパッケージ群から Python や Go、Haskel といったスクリプト言語を自由にインストールして使うことも可能で、PC と同じように開発を進めることができます。

この章では、APT を使用して必要なソフトウェアを ATDE7 および Armadillo にインストールします。そのため、あらかじめ ATDE7 および Armadillo をインターネットに接続できる状態にしてください。

12.1. 軽量スクリプト言語によるデータの送信例(Ruby)

ここでは、サンプルとして Armadillo の現在時刻を定期的に HTTP POST でパラメータ名 "time" に格納した値として送信する例を示します。

現在時刻は date コマンド、もしくは Ruby の Time クラスを使用して取得します。

ここで作成するアプリケーションは Armadillo で動作するクライアントと ATDE で動作するテスト用のサーバーの 2 つです。ATDE で動作させるためのテスト用のサーバーは、典型的な HTTP プロトコルでアクセスできる Web API を持ったサービスを模擬しています。テスト用サーバーは単に入力された POST リクエストの内容を変数に格納してコンソールに出力し、クライアントには"Thanks!"という文字列を返します。

12.1.1. テスト用サーバーの実装

最初に ATDE7 にテスト用サーバーの動作に必要なパッケージをインストールします。

```
[ATDE ~]$ sudo apt-get install ruby
[ATDE ~]$ sudo gem install sinatra
```

図 12.1 ruby と sinatra のインストール

次にエディタで次のコードを入力して、server.rb として保存してください。

```
require 'sinatra'

post '/' do
  puts "Current Time is #{params[:time]}"
  "Thanks!¥n"
end
```

図 12.2 テスト用サーバー (server.rb)

12.1.2. テスト用サーバーの動作確認

Armadillo でクライアントアプリケーションを動かす前に、テスト用サーバーの動作確認を行います。動作確認は Armadillo から cURL コマンドを使って、クライアントアプリケーションと同等のリクエストを送ってみます。

まず、ATDE7 の IP アドレス を確認しておきます。下記の例では、ip コマンドで確認すると ATDE7 の IP アドレス が 172.16.2.117 であることがわかります。

```
[ATDE ~]$ ip addr show eth0
     2: eth0: &lt;BROADCAST,MULTICAST,UP,LOWER_UP&gt; mtu 1500 qdisc pfifo_fast state UNKNOWN group
default qlen 1000
    link/ether 00:0c:29:30:b0:e0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.117/16 brd 172.16.255.255 scope global dynamic eth0
       valid_lft 65913sec preferred_lft 65913sec
    inet6 fe80::20c:29ff:fe30:b0e0/64 scope link
       valid_lft forever preferred_lft forever
```

図 12.3 IP アドレスの確認 (ip コマンド)

次の例のように server.rb を実行すると、全ての IP アドレスからのリクエストを 8081 番ポートで Web サーバーとして待ち受けます。

```
[ATDE ~]$ ruby server.rb -p 8081 -o 0.0.0.0
[2018-07-18 17:47:21] INFO  WEBrick 1.3.1
[2018-07-18 17:47:21] INFO  ruby 2.3.3 (2016-11-21) [i386-linux-gnu]
== Sinatra (v2.0.3) has taken the stage on 8081 for development with backup from WEBrick
[2018-07-18 17:47:21] INFO  WEBrick::HTTPServer#start: pid=4610 port=8081
```

ここで、Armadillo から cURL を使ってテストデータを送ってみましょう。正しく通信できた場合は、"Thanks!" の文字列が表示されます。もし、"Connection refused" 等が表示された場合は、一旦 ATDE7 の IP アドレスに ping を送信してネットワークの設定に問題が無いか確認してください。

```
[armadillo ~]# apt-get install curl
```

図 12.4 curl のインストール

```
[armadillo ~]$ curl -d "time=$(date "+%H:%M:%S")" 172.16.2.117:8081
Thanks!
```

図 12.5 curl によるテストデータの送信

正しく受信できた場合は、ATDE7 で起動しているテスト用サーバーが起動しているコンソールに下記の文字列が出力されます。

```
Current Time is 07:44:19
```

図 12.6 ATDE7 におけるテストデータの受信表示

12.1.3. クライアントの実装

Armadillo で動作するクライアントを実装します。下記のコードをエディタで入力して、client.rb として保存してください。ファイルは、ATDE7 上で作成しても Armadillo 上で、vi 等を使って作成しても構いません。ATDE7 で作成した場合は次の手順で、Armadillo へ転送します。

```
require 'net/http'

uri = URI.parse(ARGV[0])
time = Time.now.strftime("%H:%M:%S")
response = Net::HTTP.post_form(uri, {"time" => time})

puts response.body
```

図 12.7 時刻送信クライアント(client.rb)

12.1.4. Armadillo へのファイルの転送

ATDE7 上で作成したソースコードを Armadillo に配置する方法の一例として、ここでは、SSH を使った転送方法を説明します。

```
[armadillo ~]# apt-get install openssh-server
```

図 12.8 Armadillo への SSH サーバーのインストール

```
[ATDE ~]$ scp client.rb atmark@[armadillo の IP アドレス]:~/
```

図 12.9 ATDE7 から Armadillo への client.rb の転送

12.1.5. クライアントの実行

Armadillo に Ruby をインストールしてから、作成した時刻送信クライアントを実行します。第一引数にはテスト用サーバーが動いている ATDE7 の IP アドレスとポートを HTTP スキーマの URI で記述してください。

```
[armadillo ~]# apt-get install ruby
```

図 12.10 ruby のインストール

```
[armadillo ~]# ruby client.rb http://172.16.2.117:8081
Thanks!
```

図 12.11 クライアントの実行方法

正しくクライアントとの通信ができた場合、ATDE7 で動作しているサーバーのコンソールには時刻が表示されます。

```
Current Time is 07:44:19
```

図 12.12 ATDE7 における時刻データの受信表示

12.2. C 言語による開発環境

C/C++等の資産がある場合は、Armadillo 上で gcc/g++を使ってアプリケーションをコンパイルする事もできます。

12.2.1. 開発環境の準備

アプリケーションをコンパイルするために、Armadillo に gcc 等を含むツールチェーンをインストールします。Armadillo のコンソールで次のコマンドを実行してください。

```
[armadillo ~]# apt-get install build-essential
```

図 12.13 ツールチェーンのインストール

これで、gcc, make, gdb 等が使えるようになりました。次に、アプリケーションのビルドに必要なライブラリとヘッダーファイルをインストールします。例えば libssl であれば次のコマンドでインストールすることができます。

```
[armadillo ~]# apt-get install libssl-dev
```

図 12.14 開発用パッケージのインストールの例 (libssl の場合)

例に示すように、コンパイルに必要なヘッダーファイルを含むパッケージは、普通 -dev という名前が付いています。

必要なヘッダファイルの名前や、共有ライブラリのファイル名がわかっている場合は、Debian プロジェクトサイトの「パッケージの内容を検索」からファイルの含まれるパッケージの名前を探す事ができます。

Debian — パッケージ パッケージの内容を検索

https://www.debian.org/distrib/packages#search_contents

また、パッケージの部分的な名前が分っている場合は「8.2. パッケージ管理」で紹介した、 `apt-cache search` コマンドを使って必要なパッケージを探す事もできます。

13. i.MX6ULL の電源制御方法

本章では、ソフトウェアによる i.MX6ULL の電源制御方法について説明します。

ハードウェアによる電源制御については、「15.6. 外部からの電源制御」を参照してください。

13.1. poweroff コマンドによる制御

poweroff コマンドを利用して、i.MX6ULL 自身で電源を OFF にすることができます。poweroff コマンドで電源を OFF にすると、ONOFF ピン(Armadillo-610 CON2 78 ピン)を制御することで電源を ON にすることができます。

電源を OFF にするには、次のようにコマンドを実行します。

```
[armadillo ~]# poweroff
```

図 13.1 poweroff コマンドによる電源 OFF

14. SD ブートの活用

本章では、microSD カードから直接起動(以降「SD ブート」と表記します)する手順を示します。SD ブートを活用すると、SD カードを取り替えることでシステムイメージを変更することができます。本章に示す手順を実行するためには、容量が 2Gbyte 以上の microSD カードを必要とします。例として Debian GNU/Linux 9(コードネーム stretch)を SD ブートする手順を示しますが、他の OS を SD ブートすることも可能です。

SD ブートを行った場合、ブートローダーの設定は eMMC に保存されます。

microSD カードに対する作業は、ATDE で行います。そのため、ATDE に microSD 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 microSD 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 microSD 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 ext2
(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
```

図 14.1 自動マウントされた microSD 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の最新版ファイルは、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、最新バージョンを利用することを推奨します。

Armadillo サイト - Armadillo-610 ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-610/resources/software

14.1. 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 14.1 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
ブートローダーイメージ	u-boot-a600-v2018.03-at[version].imx

「表 14.2. ブートディスクの構成例」に示します。

表 14.2 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	第 2 パーティションにルートファイルシステムを構築するため、第 1 パーティションを作成します。
2	残り全て	ext4	ルートファイルシステムを構築するために ext4 ファイルシステムを構築しておきます。

14.1.1. 手順：ブートディスクの作成例

1. ブートローダーイメージファイルを取得し、ATDE 内に配置しておきます。

```
[ATDE ~]$ ls
u-boot-a600-v2018.03-at[version].imx
```

1. 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

Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): o ❷
Created a new DOS disklabel with disk identifier 0x2b685734.

Command (m for help): n ❸
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): ❹

Using default response p.
Partition number (1-4, default 1): ❺
First sector (2048-7761919, default 2048): ❻
Last sector, +sectors or +size{K,M,G,T,P} (2048-7761919, default 7761919): +128M ❼

Created a new partition 1 of type 'Linux' and of size 128 MiB.

Command (m for help): n ❽
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): ❾

Using default response p.
Partition number (2-4, default 2): ❿
```

```
First sector (264192-7761919, default 264192):  ⓫
Last sector, +sectors or +size{K,M,G,T,P} (264192-7761919, default 7761919):  ⓬

Created a new partition 2 of type 'Linux' and of size 3.6 GiB.

Command (m for help): t  ⓭
Partition number (1,2, default 2): 1  ⓮
Hex code (type L to list all codes): b  ⓯

If you have created or modified any DOS 6.x partitions, please see the fdisk documentation
for additional information.
Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): w  ⓰
The partition table has been altered.
Calling ioctl() to re-read partition table.
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。

2. 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: 3.7 GiB, 3974103040 bytes, 7761920 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
Disklabel type: dos
Disk identifier: 0x2b685734

Device     Boot  Start     End Sectors  Size Id Type
/dev/sdb1         2048  264191  262144  128M  b W95 FAT32
/dev/sdb2       264192 7761919 7497728  3.6G 83 Linux
```

3. 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.fat 3.0.27 (2014-11-12)
[ATDE ~]$ sudo mkfs.ext4 /dev/sdb2  ❷
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 937216 4k blocks and 234320 inodes
Filesystem UUID: AAAAAAAA-BBBB-CCCC-DDDD-EEEEEEEEEEEE
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 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 ext4 ファイルシステムを構築します。

4. ブートローダーイメージファイルを microSD カードに書き込みます。

```
[ATDE ~]$ ls
u-boot-a600-v2018.03-at[version].imx
[ATDE ~]$ sudo dd if=u-boot-a600-v2018.03-at[version].imx of=/dev/sdb bs=1k seek=1 ⏎
conv=fsync
```

14.2. ルートファイルシステムの構築

「14.1. 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。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 14.3 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
Debian GNU/Linux	debian-stretch-armhf-a600-[version].tar.gz	ARM(armhf)アーキテクチャ用 Debian GNU/Linux 9(コードネーム stretch)のルートファイルシステムアーカイブ

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

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

14.2.1.1. 手順：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. 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 ext4 /dev/sdb2 sd ❷
[ATDE ~]$ sudo tar zxf debian-stretch-armhf-a600-[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。

14.3. Linux カーネルイメージと DTB の配置

「14.1. ブートディスクの作成」で作成したブートディスクに Linux カーネルイメージおよび DTB(Device Tree Blob)を配置します。使用するファイルを次に示します。以降、DTB(Device Tree Blob)を DTB と表記します。

表 14.4 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	uImage-a600-v4.14-at[version]
DTB	armadillo-610-extboard-eva-grove-v4.14-at[version].dtb

microSD カードに Linux カーネルイメージおよび DTB 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 カーネルイメージまたは DTB を検出することができなくなる場合があります。

表 14.5 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	ext4
圧縮形式	非圧縮
Linux カーネルイメージファイル名	uImage
DTB ファイル名	a610.dtb

Linux カーネルイメージおよび DTB をブートディスクに配置する手順を次に示します。

14.3.1. 手順：Linux カーネルイメージおよび DTB の配置

1. Linux カーネルイメージおよび DTB を準備しておきます。

```
[ATDE ~]$ ls
uImage-a600-v4.14-at[version]  armadillo-610-extboard-eva-grove-v4.14-at[version].dtb
```

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 ext4 /dev/sdb2 sd ❷
[ATDE ~]$ sudo cp uImage-a600-v4.14-at[version] sd/boot/uImage ❸
[ATDE ~]$ sudo cp armadillo-610-extboard-eva-grove-v4.14-at[version].dtb sd/boot/a610.dtb ⏎
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 sd/ディレクトリにコピーします。

❹ DTB を sd/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。

14.4. SD ブートの実行

「14.1. 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。

1. Armadillo に電源を投入する前に、ブートディスクを microSD スロット(Armadillo-610: CON1)に挿入します。また、Armadillo-610 拡張ボード上の JP1 をジャンパでショートします。

2. 電源を投入します。

```
U-Boot 2018.03-at8 (Feb 14 2020 - 10:21:57 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC

=>
```

3. ブートディスク上の Linux カーネルを起動します。

```
=> setenv bootcmd run setup_mmcargs¥; ext4load mmc 1:2 ¥${loadaddr} /boot/uImage¥; ext4load
mmc 1:2 0x83000000 /boot/¥${fdt_file}¥; bootm ¥${loadaddr} - 0x83000000¥;  ❶
=> setenv setup_mmcargs setenv bootargs root=/dev/mmcblk1p2 rootwait ¥${optargs}¥; ❷
=> saveenv  ❸
=> boot  ❹
```

❶ ブートディスク上の Linux カーネルイメージと DTB を使用するように bootcmd を設定します。

❷ ブートディスク上のルートファイルシステムを使用するように setup_mmcargs を設定します。

❸ 環境変数を保存します。

❹ 起動します。

bootcmd と setup_mmcargs をデフォルトの設定に戻す方法

bootcmd と setup_mmcargs をデフォルトの設定に戻すには、次のコマンドを実行します。

```
=> env default bootcmd setup_mmcargs
=> saveenv
```

15. 電気的仕様

15.1. 絶対最大定格

表 15.1 絶対最大定格

項目	記号	Min.	Max.	単位	備考
電源電圧	VIN	-0.3	4.8	V	—
入出力電圧(USB 信号以外)	VI,VO	-0.5	OVDD +0.3	V	OVDD=+3.3V_IO, VDD_SNVS_IN [a]
入力電圧(USB 信号)	VI_USB	-0.3	3.63	V	USB_OTG1_DP, USB_OTG1_DN, USB_OTG2_DP, USB_OTG2_DN
入力電圧(USB_VBUS)	VI_VBUS	—	5.5	V	USB_OTG1_VBUS, USB_OTG2_VBUS
RTC バックアップ電源電圧	RTC_BAT	-0.3	3.6	V	—
使用温度範囲	Topr	-20	70	℃	結露なきこと

[a]Armadillo-610: CON2 の 13 ピンのみ OVDD=VDD_SNVS_IN となります。VDD_SNVS_IN はダイオードを介して VSNVS と VDD_HIGH_IN の電源が供給されています。

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

15.2. 推奨動作条件

表 15.2 推奨動作条件

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VIN	3.6	—	4.5	V	—
入力電圧 (USB_VBUS)	VI_VBUS	4.4	—	5.5	V	USB_OTG1_VBUS, USB_OTG2_VBUS
RTC バックアップ電源電圧	RTC_BAT	2.75	—	3.3	V	Topr=+25℃
使用温度範囲	Ta	-20	25	70	V	結露なきこと

15.3. 入出力インターフェースの電気的仕様

表 15.3 入出力インターフェース(電源)の電気的仕様

項目	記号	Min.	Typ.	Max.	単位	備考
5V 電源電圧	+5V_IO	4.8	5	5.15	V	—
3.3V 電源電圧	+3.3V_IO	3.102	3.3	3.498	V	—
	VDD_HIGH _IN	3.201	3.3	3.399	V	—

項目	記号	Min.	Typ.	Max.	単位	備考
セキュア用電源電圧	VSNVS	2.85	3.0	3.21	V	3.2V < VIN < 4.5V, OFF mode
		2.85	3.0	3.15	V	3.2V < VIN < 4.5V, On mode
		RTC_BAT-0.1	—	RTC_BAT	V	2.84V < RTC_BAT < 3.3V, Coin cell mode

表 15.4 入出力インターフェースの電気的仕様(OVDD = +3.3V_IO、VDD_SNVS_IN)

項目	記号	Min.	Max.	単位	備考
ハイレベル出力電圧	VOH	OVDD-0.15	OVDD	V	IOH = -0.1mA, -1mA
ローレベル出力電圧	VOL	0	0.15	V	IOL = 0.1mA, 1mA
ハイレベル入力電圧[a]	VIH	0.7×OVDD	OVDD	V	—
ローレベル入力電圧[a]	VIL	0	0.3×OVDD	V	—
ローレベル入力電圧(ONOFF 信号)	VIL	0	0.9	V	—
ローレベル入力電圧 (PWRON 信号)	VIL	0	0.5	V	—
ローレベル入力電圧 (EXT_RESET_B 信号)	VIL	0	0.19	V	—
入力リーク電流(no Pull-up/ Pull-down)	IIN	-1	1	μA	—
Pull-up 抵抗(5kΩ)	—	4	6	kΩ	—
Pull-up 抵抗(47kΩ)	—	37.6	56.4	kΩ	—
Pull-up 抵抗(100kΩ)	—	80	120	kΩ	—
Pull-down 抵抗(100kΩ)	—	80	120	kΩ	—

[a]オーバーシュートとアンダーシュートは 0.6V 以下でかつ 4ns を超えないようにしてください。

15.4. 電源回路の構成

Armadillo-610 の電源回路の構成は「図 15.1. 電源回路の構成」のとおりです。

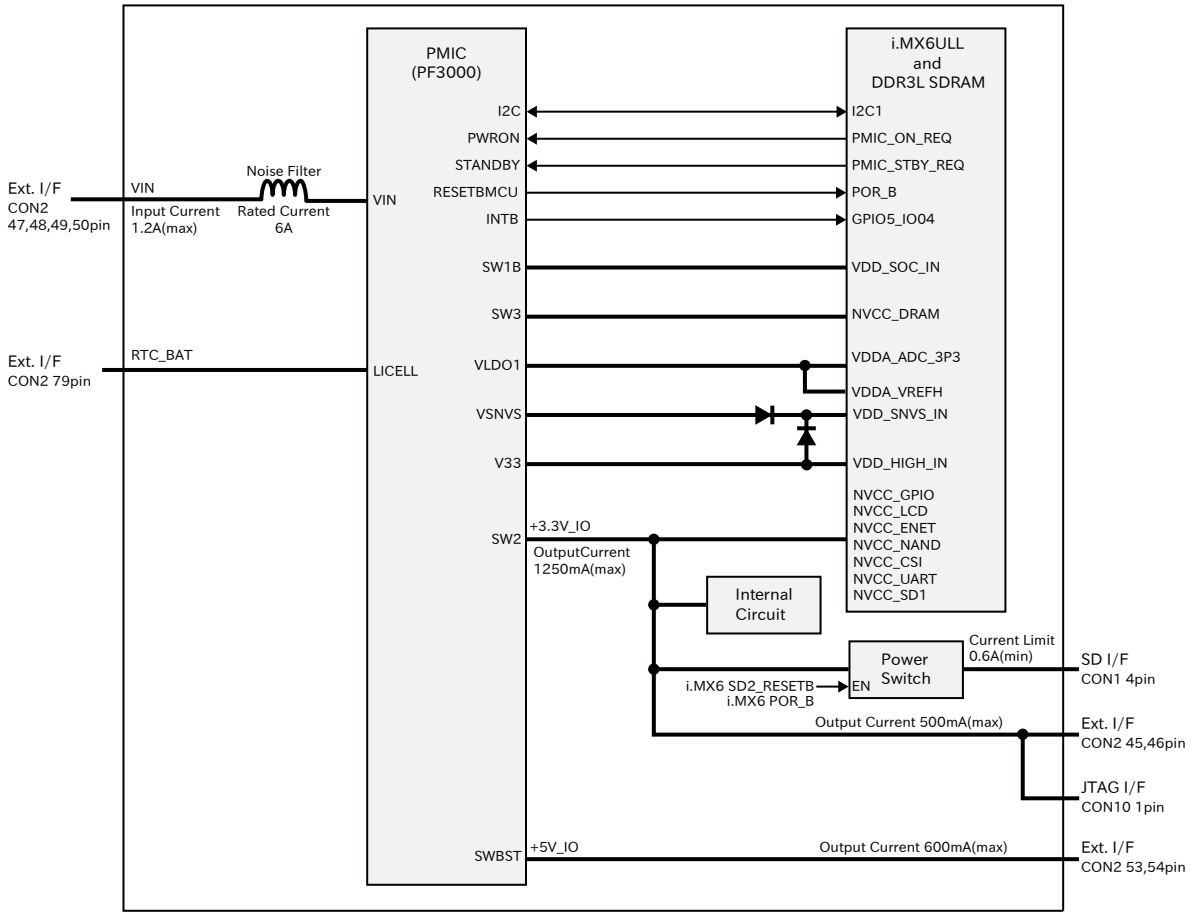


図 15.1 電源回路の構成

拡張インターフェース(Armadillo-610: CON2)からの入力電圧(VIN)をパワーマネジメント IC(PMIC)で各電圧に変換し、内部回路および各インターフェースに供給しています。各インターフェースやスイッチング・レギュレータの最大出力電流値を超えないように、外部機器の接続、供給電源の設計を行なってください。

電源シーケンスは次のとおりです。

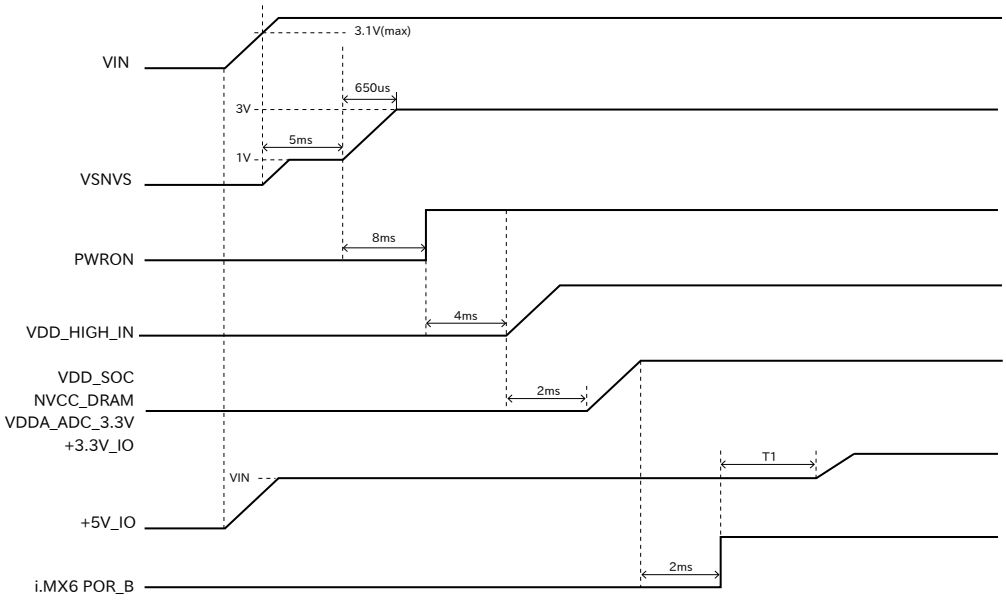


図 15.2 電源シーケンス [1]

USB_OTG1_VBUS, USB_OTG2_VBUS は電源シーケンスに関わらず、いつ電源を投入しても問題ありません。

15.5. リセット回路の構成

リセット回路の構成は「図 15.3. リセット回路の構成」のとおりです。

[1] T1: 任意のタイミング

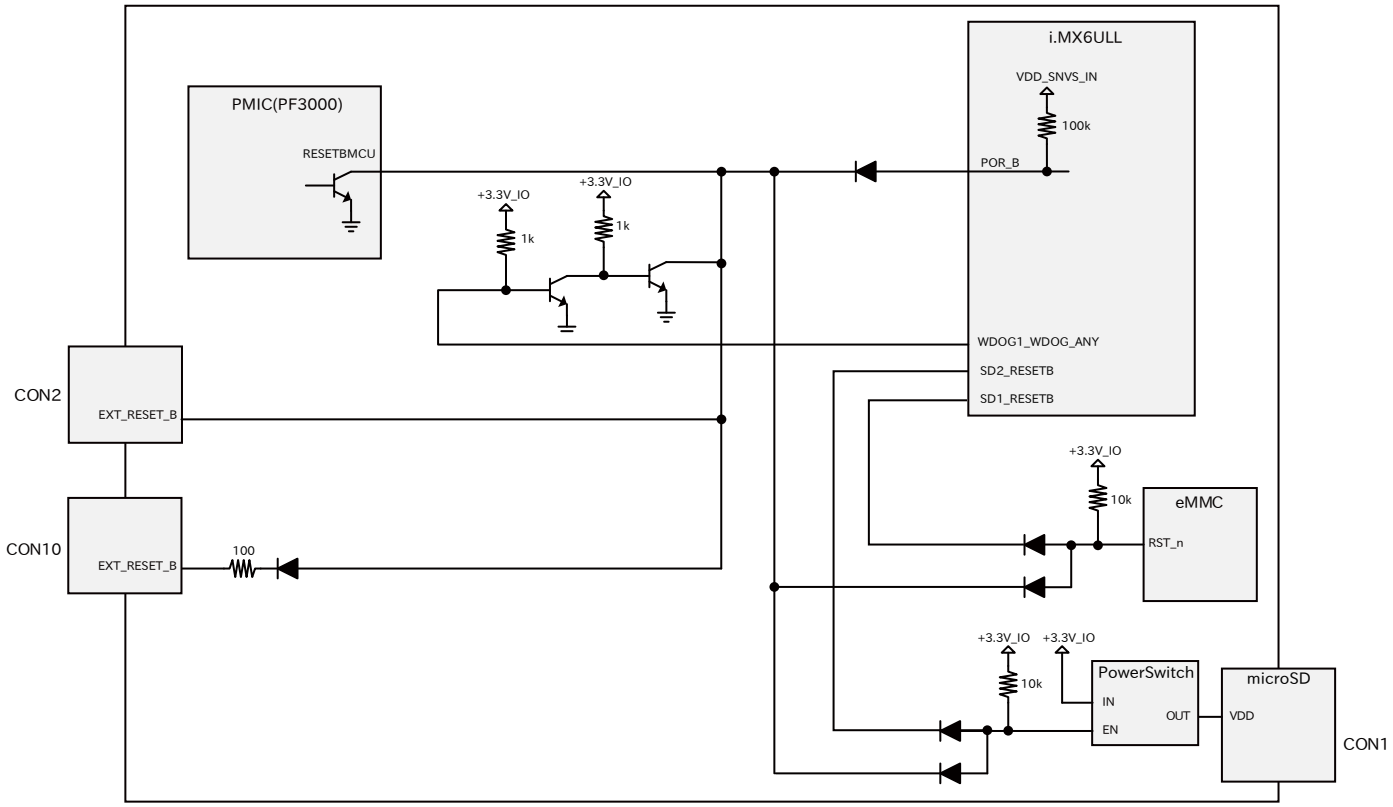


図 15.3 リセット回路の構成

拡張インターフェース(Armadillo-610: CON2)および JTAG インターフェース(Armadillo-610: CON10)の EXT_RESET_B ピンは i.MX6ULL の POR_B ピンに接続されています。EXT_RESET_B ピンから Low レベル出力することで、システムリセットすることができます。確実にシステムリセットするためには、20 ミリ秒以上 Low レベルを保持する必要があります。

図 15.4 システムリセットする場合の Low レベル保持時間

EXT_RESET_B ピンからシステムリセットする場合は、オープンドレイン出力等で GND とショートする回路を接続してください。

リセット信号が必要なデバイスを拡張する場合、拡張インターフェース(Armadillo-610: CON2)の EXT_RESET_B ピンを利用してリセットすることも可能です。

15.6. 外部からの電源制御

拡張インターフェース(Armadillo-610: CON2)の ONOFF ピンおよび PWRON ピンより、パワーマネジメント IC から i.MX6ULL への電源供給を制御することが可能です。

15.6.1. ONOFF ピンからの電源制御

拡張インターフェース(Armadillo-610: CON2)の ONOFF ピンは i.MX6ULL の ONOFF ピンに接続されています。ONOFF ピンから一定時間以上 Low レベル出力することで、i.MX6ULL の保持している電源のオン状態、オフ状態が切り替わります。

電源がオフ状態に切り替わった場合、i.MX6ULL からパワーマネジメント IC の PWRON ピンに Low レベルが出力され、パワーマネジメント IC からの電源が切断されます。

電源オン状態からオフ状態に切り替える場合は 5 秒以上、電源オフ状態からオン状態に切り替える場合は 500 ミリ秒以上、Low レベルを保持する必要があります。

連続して電源オンとオフを切り替える場合は、確実に動作させるために 5 秒以上の間隔を空けてください。

ONOFF ピンから電源制御する場合は、オープンドレイン出力等で GND とショートする回路を接続してください。

表 15.5 ONOFF ピンから電源オン、オフ切り替えする際の Low 保持時間

状態	Low 保持時間
電源オンからオフ	5 秒以上
電源オフからオン	500 ミリ秒以上

電源オンまたはオフの状態は、拡張インターフェース(Armadillo-610: CON2)の VIN ピンと RTC_BAT ピンのどちらか一方でも電源が供給されている限り、保持されます。

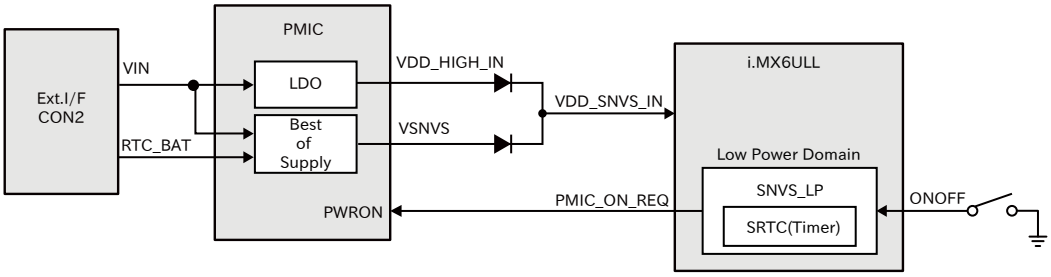


図 15.5 ONOFF 回路の構成 [2]

電源オフの状態にして拡張インターフェース(Armadillo-610: CON2)の VIN ピンからの電源を切断した場合、電荷が抜けるまでは電源オフであることが保持されます。電源オフを保持した状態で電源を投入したくない場合は、5 秒以上間隔を空けて電源を投入してください。

15.6.2. PWRON ピンからの電源制御

拡張インターフェース(Armadillo-610: CON2)の PWRON ピンはパワーマネジメント IC の PWRON ピンに接続されています。PWRON ピンから Low レベル出力することで、パワーマネジメント IC からの電源が切断されます。

[2]SNVS_LP：低消費電力ドメインです。詳細につきましては、NXP Semiconductors のホームページからダウンロード可能な『i.MX 6ULL Applications Processor Reference Manual』をご参照ください。

PWRON ピンから電源制御する場合は、オープンドレイン出力等で GND とショートする回路を接続してください。

16. インターフェース仕様

Armadillo-610 のインターフェース仕様について説明します。

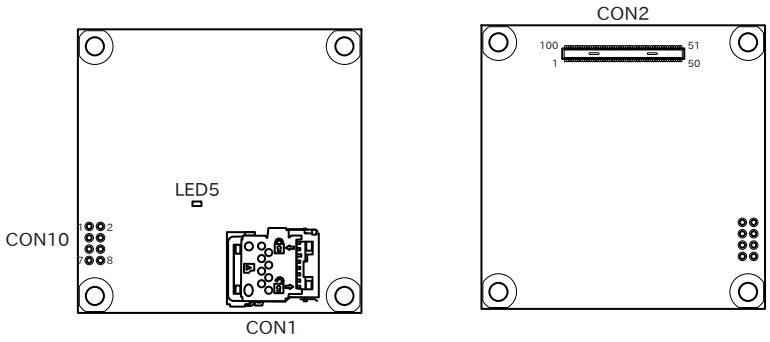


図 16.1 Armadillo-610 のインターフェース

表 16.1 Armadillo-610 インターフェース一覧 [a]

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	SDHK-8BNS-K-303-TB(HF)	J.S.T.Mfg.
CON2	拡張インターフェース	DF40C-100DP-0.4V(51)	HIROSE ELECTRIC
CON10	JTAG インターフェース	A3B-08PA-2DSA(51)	HIROSE ELECTRIC
LED5	ユーザー LED(黄)	SML-310YTT86	ROHM

[a]部品の実装、未実装を問わず、搭載可能な部品型番を記載しています。

「表 16.1. Armadillo-610 インターフェース一覧 」には搭載可能な代表型番を記載しており、実際に搭載されている部品型番と違う場合があります。お手元の製品に搭載されている部品型番や部品の実装、未実装の情報については、「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロードできる納入仕様書および変更履歴表にてご確認ください。

16.1. CON1(SD インターフェース)

CON1 はハイスピード(最大クロック周波数: 49.5MHz)に対応した SD インターフェースです。信号線は i.MX6ULL の SD ホストコントローラ(uSDHC2)に接続されています。

SD カードに供給される電源は i.MX6ULL の NAND_ALE ピン(GPIO4_IO10)で制御が可能です。High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

CON1 は活線挿抜に対応していません。microSD カードの挿抜は、電源を切断してから行ってください。

SD コントローラ(uSDHC2)は CON2(拡張インターフェース)でも利用可能ですが、排他利用となります。

表 16.2 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	DAT2	In/Out	SD データバス(bit2)、i.MX6ULL の NAND_DATA02 ピンに接続
2	CD/DAT3	In/Out	SD データバス(bit3)、i.MX6ULL の NAND_DATA03 ピンに接続
3	CMD	In/Out	SD コマンド/レスポンス、i.MX6ULL の NAND_WE_B ピンに接続
4	VDD	Power	電源(+3.3V_IO)
5	CLK	Out	SD クロック、i.MX6ULL の NAND_RE_B ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	SD データバス(bit0)、i.MX6ULL の NAND_DATA00 ピンに接続
8	DAT1	In/Out	SD データバス(bit1)、i.MX6ULL の NAND_DATA01 ピンに接続

16.1.1. microSD カードの挿抜方法

1. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックを解除します。

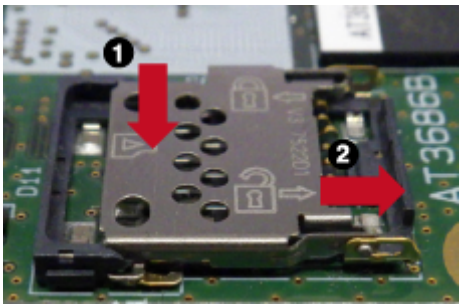


図 16.2 カバーのロックを解除する

2. カバーを開けます。

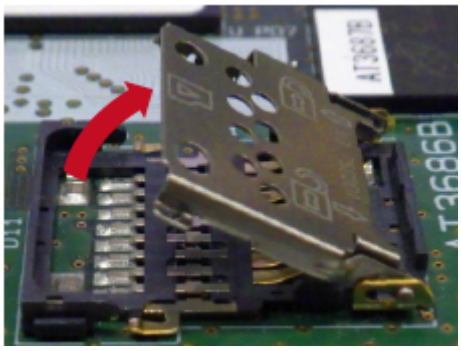


図 16.3 カバーを開ける

カバーは過度な力で回転させたり、回転方向以外の方向へ力を加えると、破損の原因となりますので、ご注意ください。

3. 任意の角度までトレイを開いた状態で、microSD カードを挿抜します。

図 16.4 microSD カードの挿抜

microSD カード挿入方向については、カバーに刻印されているカードマークを目安にしてください。

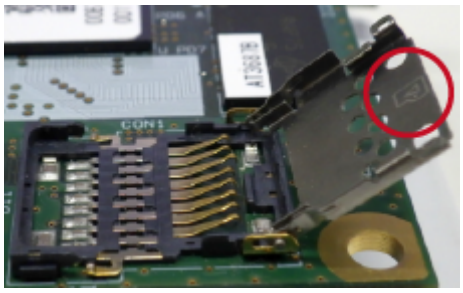


図 16.5 カードマークの確認

4. カバーを閉めます。

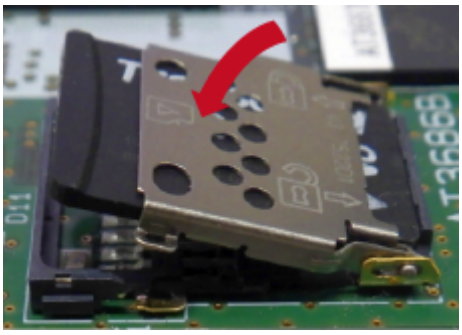


図 16.6 カバーを閉める

5. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックします。

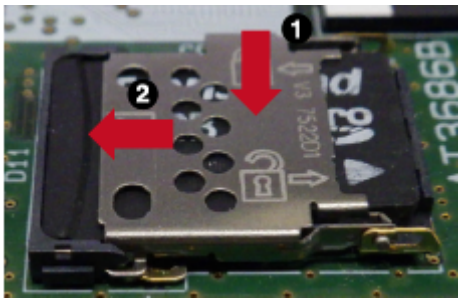


図 16.7 カバーをロックする

microSD カード装着後のカードの抜き取り手順は挿入時と同じです。

16.2. CON2(拡張インターフェース)

CON2 は Armadillo-610 拡張用のインターフェースです。電源、リセット、複数の機能(マルチプレクス)をもった i.MX6ULL の信号線、USB、Ethernet PHY の信号線等、Armadillo-610 を拡張するために必要な信号線がすべて接続されています。

Armadillo-610 の電源は VIN ピンから供給します。

RTC_BAT ピンは、i.MX6ULL の低消費電力ドメインにある SRTC(Secure Real Time Clock)の外部バックアップインターフェースで、長時間電源が切断されても i.MX6ULL の一部データ(時刻データ等)を保持させたい場合にご使用ください。

プルアップ/ダウン抵抗が接続されている拡張入出力ピンは、i.MX6ULL の内蔵 ROM によるブートモード設定ピンを兼用しており、ブートモード設定のため、プルアップ/ダウン抵抗で電源投入時に High/Low レベルの状態を保持しています。意図しない動作を引き起こす原因となるため、電源投入時から U-Boot が動作するまでは、各々のピンを High/Low レベルに保持した状態でご使用ください。

表 16.3 CON2 信号配列

ピン番号	ピン名	I/O	説明
1	USB_OTG1_DP	In/Out	USB_OTG1 のプラス側信号、i.MX6ULL の USB_OTG1_DP ピンに接続
2	USB_OTG1_DN	In/Out	USB_OTG1 のマイナス側信号、i.MX6ULL の USB_OTG1_DN ピンに接続
3	GND	Power	電源(GND)
4	USB_OTG2_DN	In/Out	USB_OTG2 のマイナス側信号、i.MX6ULL の USB_OTG2_DN ピンに接続
5	USB_OTG2_DP	In/Out	USB_OTG2 のプラス側信号、i.MX6ULL の USB_OTG2_DP ピンに接続
6	GND	Power	電源(GND)
7	USB_OTG1_VBUS	Power	電源(USB_OTG1_VBUS)、i.MX6ULL の USB_OTG1_VBUS ピンに接続、1uF のバイパスコンデンサが接続されています。

ピン番号	ピン名	I/O	説明
8	USB_OTG2_VBUS	Power	電源(USB_OTG2_VBUS)、i.MX6ULL の USB_OTG2_VBUS ピンに接続、1uF のバイパスコンデンサが接続されています。
9	SPEEDLED	Out	LAN スピード LED 用信号、Ethernet PHY の LED2 ピンに接続
10	LINK_ACTLED	Out	LAN リンクアクティビティ LED 用信号、Ethernet PHY の LED1 ピンに接続
11	GPIO1_IO19	In/Out	拡張入出力、i.MX6ULL の UART1_RTS_B ピンに接続、基板上で 10kΩ プルダウンされています。
12	GPIO4_IO17	In/Out	拡張入出力、i.MX6ULL の CSI_MCLK ピンに接続
13	GPIO5_IO00	In/Out	拡張入出力、i.MX6ULL の SNVS_TAMPER0 ピンに接続、オープンドレインでの使用推奨 [a]
14	GPIO1_IO04	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続
15	GPIO1_IO03	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続
16	GPIO1_IO02	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続
17	GPIO1_IO01	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続
18	LCD_DATA00	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続、基板上で 10kΩ プルダウンされています。
19	LCD_DATA01	In/Out	拡張入出力、i.MX6ULL の LCD_DATA01 ピンに接続、基板上で 10kΩ プルアップ(+3.3V_IO)されています。
20	LCD_DATA02	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続、基板上で 10kΩ プルダウンされています。
21	LCD_DATA03	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続、基板上で 10kΩ プルダウンされています。
22	LCD_DATA04	In/Out	拡張入出力、i.MX6ULL の LCD_DATA04 ピンに接続、基板上で 10kΩ プルダウンされています。
23	LCD_DATA05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続、BJP1 が Low レベル時 10kΩ プルアップ(+3.3V_IO)、High レベル時 10kΩ プルダウンされます。
24	LCD_DATA06	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続、基板上で 10kΩ プルアップ(+3.3V_IO)されています。
25	LCD_DATA07	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続、基板上で 10kΩ プルダウンされています。
26	LCD_DATA08	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続、基板上で 10kΩ プルダウンされています。
27	LCD_DATA09	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続、基板上で 10kΩ プルダウンされています。
28	LCD_DATA10	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続、基板上で 10kΩ プルダウンされています。
29	LCD_DATA11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続、BJP1 が Low レベル時 10kΩ プルダウン、High レベル時 10kΩ プルアップ(+3.3V_IO)されます。
30	LCD_DATA12	In/Out	拡張入出力、i.MX6ULL の LCD_DATA12 ピンに接続、基板上で 10kΩ プルダウンされています。
31	LCD_DATA13	In/Out	拡張入出力、i.MX6ULL の LCD_DATA13 ピンに接続、基板上で 10kΩ プルダウンされています。
32	LCD_DATA14	In/Out	拡張入出力、i.MX6ULL の LCD_DATA14 ピンに接続、基板上で 10kΩ プルダウンされています。
33	LCD_DATA15	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続、基板上で 10kΩ プルダウンされています。
34	LCD_DATA16	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続、基板上で 10kΩ プルダウンされています。
35	LCD_DATA17	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続、基板上で 10kΩ プルダウンされています。
36	GND	Power	電源(GND)
37	LCD_CLK	In/Out	拡張入出力、i.MX6ULL の LCD_CLK ピンに接続
38	LCD_HSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_HSYNC ピンに接続
39	LCD_VSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_VSYNC ピンに接続
40	LCD_ENABLE	In/Out	拡張入出力、i.MX6ULL の LCD_ENABLE ピンに接続
41	PWM5_OUT	In/Out	拡張入出力、i.MX6ULL の NAND_DQS ピンに接続

ピン番号	ピン名	I/O	説明
42	BJP1	In	起動デバイス設定用信号、ロジック IC を経由して i.MX6ULL の LCD_DATA05 ピン、LCD_DATA11 ピンに接続、基板上で 47kΩ プルダウンされています。 (Low: LCD_DATA05 ピンは 10kΩ プルアップ(+3.3V_IO)、LCD_DATA11 ピンは 10kΩ プルダウンされます。High: LCD_DATA05 ピンは 10kΩ プルダウン、LCD_DATA11 ピンは 10kΩ プルアップ(+3.3V_IO)されます。)
43	JTAG_MOD	In	SJC モード設定ピン、i.MX6ULL の JTAG_MOD ピンに接続、基板上で 11kΩ プルダウンされています。 [b]
44	EXT_RESET_B	In	システムリセット、i.MX6ULL の POR_B ピンに接続、オープンドレイン入力
45	+3.3V_IO	Power	電源(+3.3V_IO)
46	+3.3V_IO	Power	電源(+3.3V_IO)
47	VIN	Power	電源(VIN)
48	VIN	Power	電源(VIN)
49	VIN	Power	電源(VIN)
50	VIN	Power	電源(VIN)
51	GND	Power	電源(GND)
52	GND	Power	電源(GND)
53	+5V_IO	Power	電源(+5V_IO)
54	+5V_IO	Power	電源(+5V_IO)
55	GPIO4_IO19	In/Out	拡張入出力、i.MX6ULL の CSI_VSYNC ピンに接続
56	GPIO4_IO20	In/Out	拡張入出力、i.MX6ULL の CSI_HSYNC ピンに接続
57	GPIO4_IO25	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
58	GPIO4_IO26	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
59	GPIO4_IO27	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
60	GPIO4_IO28	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
61	GPIO4_IO23	In/Out	拡張入出力、i.MX6ULL の CSI_DATA02 ピンに接続
62	GPIO4_IO22	In/Out	拡張入出力、i.MX6ULL の CSI_DATA01 ピンに接続
63	GPIO4_IO24	In/Out	拡張入出力、i.MX6ULL の CSI_DATA03 ピンに接続
64	GPIO4_IO21	In/Out	拡張入出力、i.MX6ULL の CSI_DATA00 ピンに接続
65	GPIO4_IO18	In/Out	拡張入出力、i.MX6ULL の CSI_PIXCLK ピンに接続
66	GPIO4_IO09	In/Out	拡張入出力、i.MX6ULL の NAND_DATA07 ピンに接続
67	GPIO4_IO08	In/Out	拡張入出力、i.MX6ULL の NAND_DATA06 ピンに接続
68	GPIO4_IO07	In/Out	拡張入出力、i.MX6ULL の NAND_DATA05 ピンに接続
69	GPIO4_IO06	In/Out	拡張入出力、i.MX6ULL の NAND_DATA04 ピンに接続
70	GPIO3_IO28	In/Out	拡張入出力、i.MX6ULL の LCD_DATA23 ピンに接続、基板上で 47kΩ プルダウンされています。
71	GPIO3_IO27	In/Out	拡張入出力、i.MX6ULL の LCD_DATA22 ピンに接続
72	GPIO3_IO26	In/Out	拡張入出力、i.MX6ULL の LCD_DATA21 ピンに接続
73	GPIO3_IO25	In/Out	拡張入出力、i.MX6ULL の LCD_DATA20 ピンに接続
74	GPIO3_IO24	In/Out	拡張入出力、i.MX6ULL の LCD_DATA19 ピンに接続
75	GND	Power	電源(GND)
76	GPIO3_IO23	In/Out	拡張入出力、i.MX6ULL の LCD_DATA18 ピンに接続
77	PWRON	In	パワーマネジメント IC の PWRON 信号、オープンドレイン入力、パワーマネジメント IC の PWRON ピンと i.MX6ULL の PMIC_ON_REQ ピンに接続、i.MX6ULL 内部で 100kΩ プルアップ(VDD_SNVS_IN)されています。
78	ONOFF	In	i.MX6ULL の ON/OFF 信号、オープンドレイン入力、i.MX6ULL の ONOFF ピンに接続、i.MX6ULL 内部で 100kΩ プルアップ(VDD_SNVS_IN)されています。
79	RTC_BAT	Power	電源(RTC_BAT)、パワーマネジメント IC の LICELL ピンに接続
80	GPIO1_IO08	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO08 ピンに接続
81	GPIO1_IO05	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO05 ピンに接続
82	GPIO1_IO30	In/Out	拡張入出力、i.MX6ULL の UART5_TX_DATA ピンに接続
83	GPIO1_IO16	In/Out	拡張入出力、i.MX6ULL の UART1_TX_DATA ピンに接続
84	GPIO1_IO31	In/Out	拡張入出力、i.MX6ULL の UART5_RX_DATA ピンに接続

ピン番号	ピン名	I/O	説明
85	GPIO1_IO17	In/Out	拡張入出力、i.MX6ULL の UART1_RX_DATA ピンに接続
86	GPIO1_IO23	In/Out	拡張入出力、i.MX6ULL の UART2_RTS_B ピンに接続
87	GPIO1_IO22	In/Out	拡張入出力、i.MX6ULL の UART2_CTS_B ピンに接続
88	GPIO1_IO21	In/Out	拡張入出力、i.MX6ULL の UART2_RX_DATA ピンに接続
89	GPIO1_IO20	In/Out	拡張入出力、i.MX6ULL の UART2_TX_DATA ピンに接続
90	GPIO1_IO00	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO00 ピンに接続
91	GPIO1_IO26	In/Out	拡張入出力、i.MX6ULL の UART3_CTS_B ピンに接続
92	GPIO1_IO27	In/Out	拡張入出力、i.MX6ULL の UART3_RTS_B ピンに接続
93	GPIO1_IO25	In/Out	拡張入出力、i.MX6ULL の UART3_RX_DATA ピンに接続
94	GPIO1_IO24	In/Out	拡張入出力、i.MX6ULL の UART3_TX_DATA ピンに接続
95	GND	Power	電源(GND)
96	Ethrer_RXN	In/Out	Ethernet 送信/受信データ(+) CH1、Ethernet PHY(LAN8720AI)の RXN ピンに接続
97	Ethrer_RXP	In/Out	Ethernet 送信/受信データ(-) CH1、Ethernet PHY(LAN8720AI)の RXP ピンに接続
98	GND	Power	電源(GND)
99	Ethrer_TXN	In/Out	Ethernet 送信/受信データ(-) CH2、Ethernet PHY(LAN8720AI)の TXN ピンに接続
100	Ethrer_TXP	In/Out	Ethernet 送信/受信データ(+) CH2、Ethernet PHY(LAN8720AI)の TXP ピンに接続

[a]GPIO5_IO00 のみ VDD_SNVS_IN 系の拡張入出力ピンとなります。電圧レベルにご注意ください。
[b]Armadillo-610 拡張ボードでは GPIO で使用していますが、JTAG モード設定ピンですので、GPIO での使用は推奨しません。動作を理解した上でご使用ください。

16.2.1. 拡張可能な機能

CON2 から拡張可能な機能の概要は次のとおりです。

拡張入出力となっている信号線のほとんどが、複数の機能をもっています。拡張できる機能の詳細につきましては、「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロードできる『Armadillo-610 マルチプレクス表』をご参照ください。

複数箇所に割り当て可能な信号(USDHC2、UART1、ESPI1、I2C2 等)がありますが、同じ信号は複数ピンで同時利用できません。

16.2.1.1. LAN(Ethernet)

LAN を 1 ポート拡張することが可能です。

信号線は Ethernet PHY(LAN8720AI-CP/Microchip Technology) を経由して i.MX6ULL の Ethernet コントローラ(ENET1)に接続されています。

- 通信速度: 10BASE-T/100BASE-TX(AUTO-MDIX 対応)

16.2.1.2. USB

USB を 2 ポート拡張可能で、Host は最大 2 ポート、OTG は最大 1 ポート拡張することが可能です。信号線は i.MX6ULL の USB コントローラ(USB_OTG1、USB_OTG2)に接続されています。

- USB 2.0
  - High Speed(480Mbps)
  - Full Speed(12Mbps)
  - Low Speed(1.5Mbps)

16.2.1.3. UART

シリアル(UART)を最大 8 ポート拡張することが可能です。信号線は i.MX6ULL の UART(UART1、UART2、UART3、UART4、UART5、UART6、UART7、UART8)に接続されています。

- 最大データ転送レート: 4Mbps
- 信号レベル: +3.3V_IO

16.2.1.4. SD/SDIO/MMC

CON1(SD インターフェース)と排他で SD/SDIO/MMC を 1 ポート拡張することが可能です。信号線は i.MX6ULL の SD ホストコントローラ(uSDHC2)に接続されています。

- 最大クロック周波数: 49.5MHz
- 信号レベル: +3.3V_IO

16.2.1.5. LCD

LCD を最大 1 ポート拡張することが可能です。信号線は i.MX6ULL の LCD インターフェース(eLCDIF)に接続されています。

- 最大解像度: WXGA(1366 x 768/24bpp)
- 信号レベル: +3.3V_IO

16.2.1.6. I2S(SAI)

I2S を最大 2 ポート拡張することが可能です。信号線は i.MX6ULL の同期式オーディオインターフェース(SAI1、SAI3)に接続されています。

- 信号レベル: +3.3V_IO

16.2.1.7. MQS

MQS を最大 1 ポート拡張することが可能です。信号線は i.MX6ULL の Medium Quality Sound(MQS)に接続されています。

- 信号レベル: +3.3V_IO

16.2.1.8. S/PDIF

S/PDIF を最大 1 ポート拡張することが可能です。信号線は i.MX6ULL の Sony/Philips デジタルインターフェース(SPDIF)に接続されています。

- 信号レベル: +3.3V_IO

16.2.1.9. I2C

I2C を最大 2 ポート拡張することが可能です。信号線は i.MX6ULL の I2C コントローラ(I2C2、I2C4)に接続されています。

- 最大データ転送レート: 400kbps
- 信号レベル: +3.3V_IO

16.2.1.10. SPI

SPI を最大 4 ポート拡張することが可能です。信号線は i.MX6ULL の ESPI(ECSPI1、ECSPI2、ECSPI3、ECSPI4)に接続されています。

- 信号レベル : +3.3V_IO

16.2.1.11. CAN

CAN を最大 2 ポート拡張することが可能です。信号線は i.MX6ULL の FLEXCAN(FLEXCAN1、FLEXCAN2)に接続されています。

- プロトコルバージョン 2.0B アクティブ対応
- 信号レベル: +3.3V_IO

16.2.1.12. A/D

A/D を最大 8 ポート拡張することが可能です。信号線は i.MX6ULL の AD コンバーター(ADC1、ADC2)に接続されています。

- 分解能: 最大 12 ビット
- サンプリングレート: 最大 1MS/s
- 測定電圧範囲: DC 0〜3.3V

16.2.1.13. PWM

PWM を最大 8 ポート拡張することが可能です。

- 最大周波数: 66MHz
- 信号レベル: +3.3V_IO

16.2.1.14. GPIO

GPIO を最大 66 ポート拡張することが可能です。

- 信号レベル : +3.3V_IO

16.2.1.15. リアルタイムクロック

i.MX6ULL 内蔵のリアルタイムクロックを使用可能です。バックアップ用のピン(RTC_BAT)が接続されていますので、Armadillo-610 の電源(VIN)が切断されても時刻データを保持したい場合にご使用ください。

- 平均月差: 約 70 秒@25℃(参考値)

・バックアップ時間: 約 4 か月(CR2032 使用時の参考値)

16.3. CON10(JTAG インターフェース)

CON10 は JTAG デバッガを接続することのできる JTAG インターフェースです。信号線は i.MX6ULL のシステム JTAG コントローラ(SJC)に接続されています。

JTAG のセキュリティ状態は eFuse で変更することが可能です。

システム JTAG コントローラの詳細につきましては、NXP Semiconductors のホームページからダウンロード可能な『i.MX 6ULL Applications Processor Reference Manual』をご参照ください。モード設定に必要な i.MX6ULL の JTAG_MOD ピンは CON2(拡張インターフェース)に接続されています。

表 16.4 CON10 信号配列

ピン番号	ピン名	I/O	説明
1	+3.3V_IO	Power	電源(+3.3V_IO)
2	JTAG_TRST_B	In	テストリセット、i.MX6ULL の JTAG_TRST_B ピンに接続、i.MX6ULL 内部で 47kΩ プルアップ(+3.3V_IO)されています。
3	JTAG_TDI	In	テストデータ入力、i.MX6ULL の JTAG_TDI ピンに接続、i.MX6ULL 内部で 47kΩ プルアップ(+3.3V_IO)されています。
4	JTAG_TMS	In	テストモード選択、i.MX6ULL の JTAG_TMS ピンに接続、i.MX6ULL 内部で 47kΩ プルアップ(+3.3V_IO)されています。
5	JTAG_TCK	In	テストクロック、i.MX6ULL の JTAG_TCK ピンに接続、i.MX6ULL 内部で 47kΩ プルアップ(+3.3V_IO)されています。
6	JTAG_TDO	Out	テストデータ出力、i.MX6ULL の JTAG_TDO ピンに接続
7	EXT_RESET_B	In	システムリセット、i.MX6ULL の POR_B ピンに接続、オープンドレイン入力
8	GND	Power	電源(GND)

16.4. LED5(ユーザー LED)

LED5 は、ユーザー側で自由に利用できる LED です。

表 16.5 LED5

部品番号	名称(色)	説明
LED5	ユーザー LED(黄)	i.MX6ULL の UART1_CTS_B ピンに接続、(Low: 消灯、High: 点灯)

17. 基板形状図

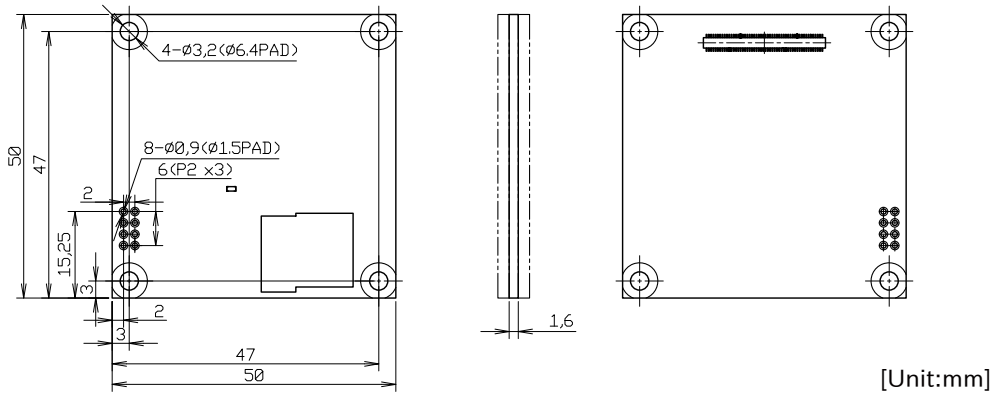


図 17.1 基板形状および固定穴寸法

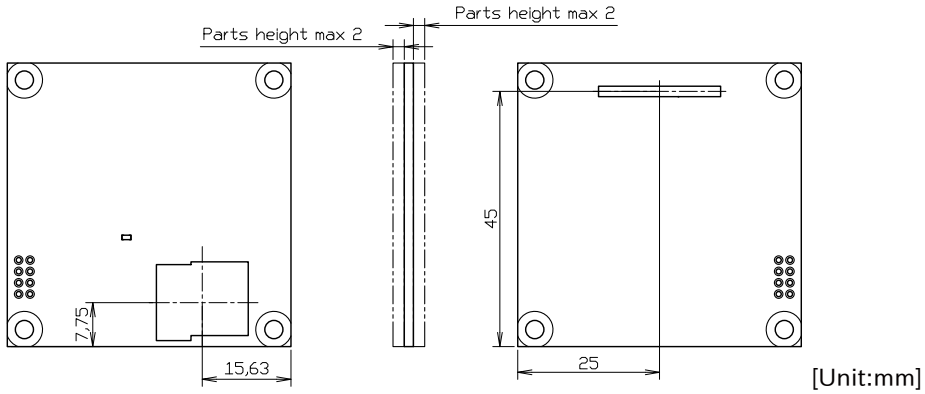


図 17.2 コネクタ中心寸法

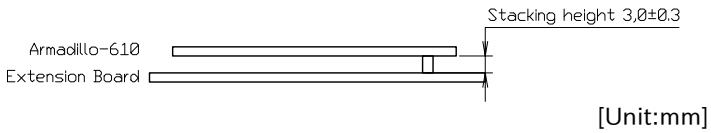


図 17.3 Armadillo-610 のスタッキング高さ

DXF 形式の基板形状図を、購入者向けの限定データとして「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロード可能です。

18. オプション品

本章では、Armadillo-610 のオプション品について説明します。

表 18.1 Armadillo-610 関連のオプション品

名称	型番	備考
Armadillo-610 拡張ボード	ー	Armadillo-610 開発セットに同梱
USB シリアル変換アダプタ	SA-SCUSB-00	Armadillo-610 開発セットに同梱
LCD オプションセット(7 インチタッチパネル WVGA 液晶)	OP-LCD70EXT-L00	7 インチタッチパネル WVGA 液晶が付属
Armadillo-400 シリーズ LCD オプションセット	OP-A400-LCD43EXT-L01	4.3 インチタッチパネル WQVGA 液晶が付属
Armadillo-WLAN(AWL13)	AWL13-U00Z	
無線 LAN 用外付けアンテナセット 01	OP-ANT-WLAN-01W	

Armadillo-WLAN(AWL13)、無線 LAN 用外付けアンテナセット 01 の詳細につきましては、「Armadillo-WLAN 製品ページ」 [https://armadillo.atmark-techno.com/armadillo-wlan/awl13/] をご参照ください。

18.1. USB シリアル変換アダプタ

18.1.1. 概要

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。

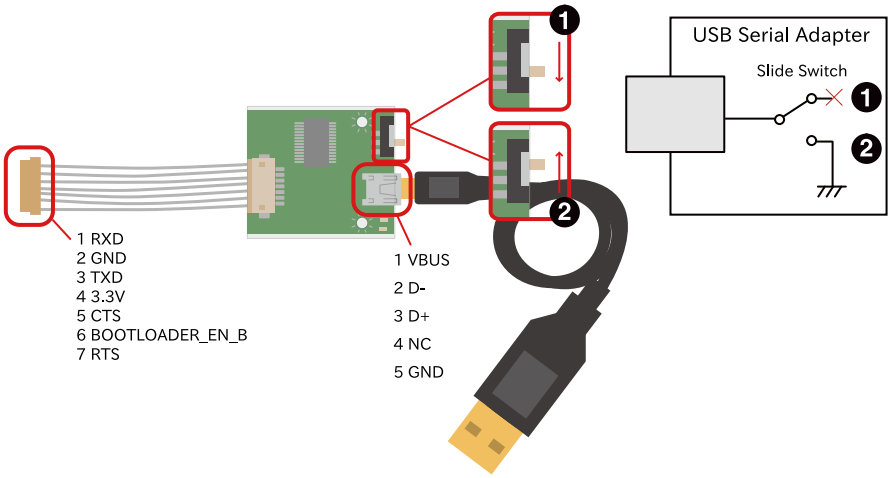


図 18.1 USB シリアル変換アダプタの配線

❶ オープン

❷ GND ショート

シリアルインターフェース(Armadillo-610 拡張ボード: CON3)に接続した場合、USB シリアル変換アダプタのスイッチで、電源投入時の起動モードを設定することが可能です。スライドスイッチの状態に対応した起動モードは以下のとおりです。

表 18.2 USB シリアル変換アダプタのスライドスイッチによる起動モードの設定

スライドスイッチ	起動モード
オープン	オートブートモード
GND ショート	保守モード

USB シリアル変換アダプタは、Armadillo-610 の電源を切断した状態で接続してください。故障の原因となる可能性があります。

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

18.2. Armadillo-610 拡張ボード

18.2.1. 概要

Armadillo-610 拡張ボードは Armadillo-610 を搭載する拡張ボードを設計開発するためのリファレンスボードです。電源、LAN、無線 LAN [1]、USB、SD、LCD [1]、RS485、オーディオ、絶縁デジタル入出力、リアルタイムクロック、スイッチ、LED 等の動作を確認することが可能です。Armadillo-610 拡張ボードは Armadillo-610 開発セットに同梱されます。

Armadillo-610 拡張ボードの回路図、部品表は購入者向けの限定データとして「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロード可能です。

Armadillo-610 拡張ボードは Armadillo-610 がないと機能しない製品ですので、Armadillo-610 を搭載した状態での仕様を説明します。

18.2.2. 仕様

Armadillo-610 拡張ボードの主な仕様は次のとおりです。

[1]動作を確認するためには、別途オプション品を購入する必要があります。

表 18.3 Armadillo-610 拡張ボードの仕様

LAN(Ethernet)	100BASE-TX/10BASE-T x 1、AUTO-MDIX 対応
無線 LAN	Armadillo-WLAN(AWL13)搭載可能 [a]
シリアル(UART)	3.3V CMOS レベル x 1、RS485 x 1
USB	USB 2.0 Host(High Speed) x 2、USB 2.0 OTG(High Speed) x 1
SD	SD スロット x 1 [b]
カレンダ時計	リアルタイムクロック搭載、バックアップ用コネクタ搭載 [c]
オーディオ	モノラルスピーカー出力 x 1
ビデオ	LCD オプションセット(7 インチタッチパネル WVGA 液晶)接続可能 [d]
接点入出力	入力 x 2、出力 x 2
Grove インターフェース	Grove コネクタ x 4 [e] UART x 1、I2C x 1、A/D x 2
拡張インターフェース	UART、SD、LCD、I2S、S/PDIF、MQS、I2C、SPI、CAN、A/D、PWM、GPIO 等 [f]
スイッチ	ユーザースイッチ x 1、 リセットスイッチ x 1、パワースイッチ x 1
LED	ユーザー LED x 1
電源電圧	DC 9〜24V±10%(メイン電源)、DC 2.0〜3.5V(RTC バックアップ)、DC 2.75〜3.3V(i.MX6ULL 内蔵 RTC バックアップ)
消費電力	約 1.2W(待機時)、約 1.8W(LAN 通信時) [g]
使用温度範囲	+10〜+40℃(結露なきこと)
外形サイズ	115 x 160 mm(突起部を除く)

[a]Armadillo-610 拡張ボードに Armadillo-WLAN(AWL13)は付属しません。
[b]Armadillo-610 上の microSD スロットと排他利用となります。
[c]電池は付属しません。
[d]Armadillo-610 拡張ボードに LCD オプションセット(7 インチタッチパネル WVGA 液晶)は付属しません。
[e]LCD インターフェースと排他利用となります。
[f]シリアル、SD スロット、オーディオ、LCD 等のインターフェースと排他利用となります。
[g]電源電圧 DC 12V 時の消費電力です。外部接続機器の消費分は含みません。

Armadillo-610 開発ボードは設計開発用のリファレンスボードです。仕様や外観を予告なく変更する場合があります。

18.2.3. ブロック図

Armadillo-610 拡張ボードのブロック図は次のとおりです。

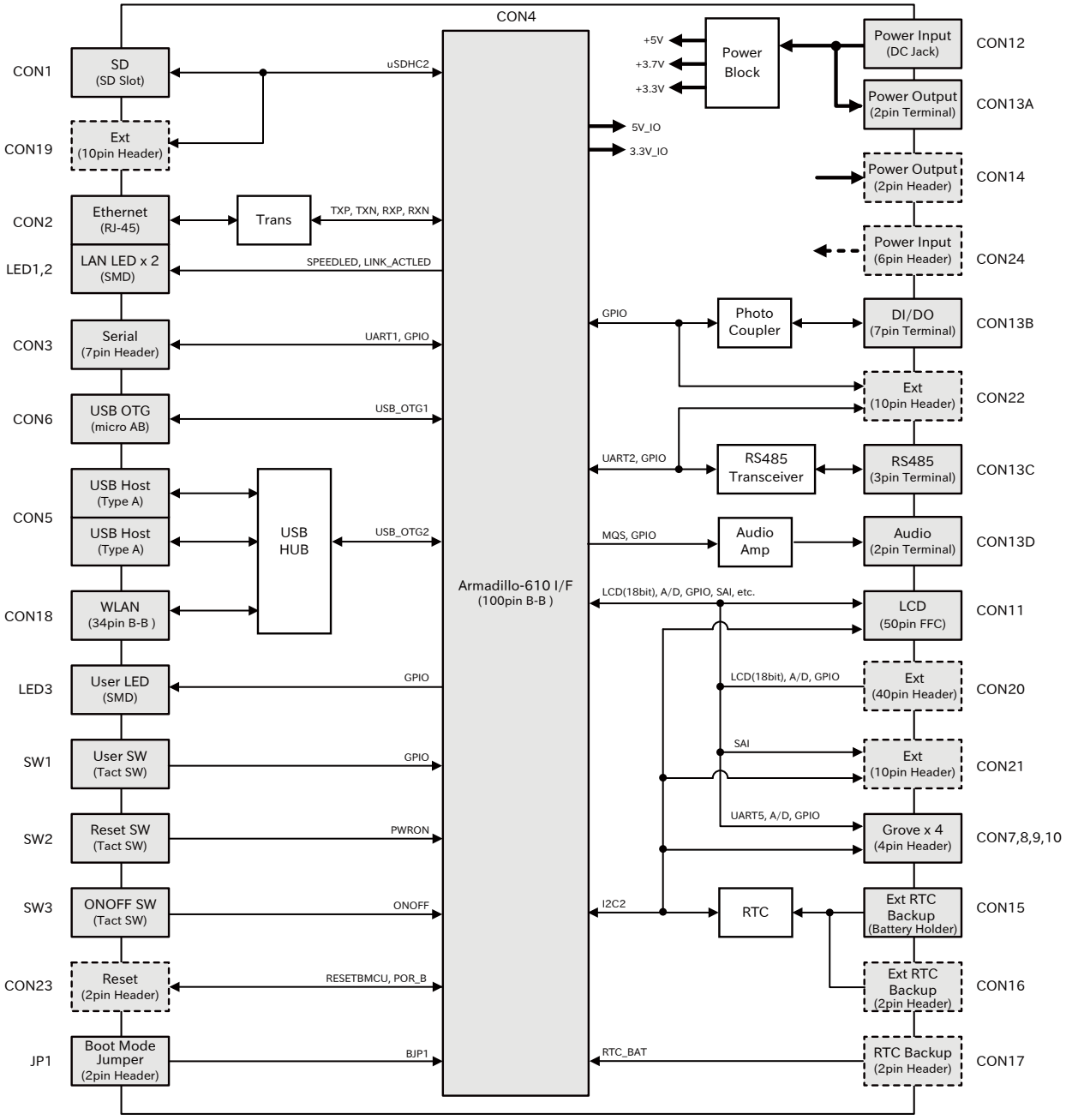


図 18.2 Armadillo-610 拡張ボードのブロック図

Armadillo-610 拡張ボードの電源回路の構成は次のとおりです。

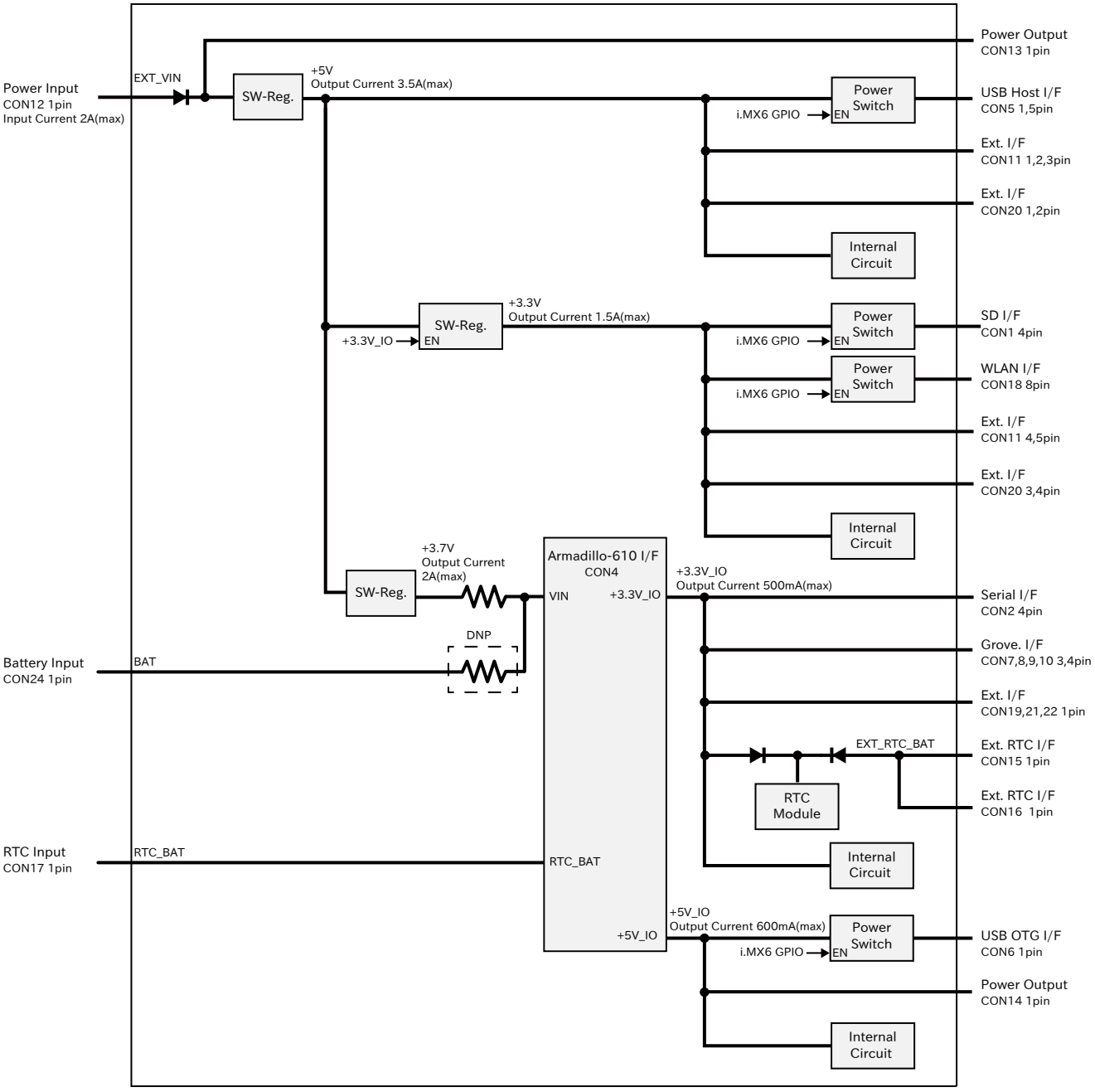


図 18.3 Armadillo-610 拡張ボードの電源回路の構成

18.2.4. インターフェース仕様

Armadillo-610 拡張ボードのインターフェース仕様について説明します。

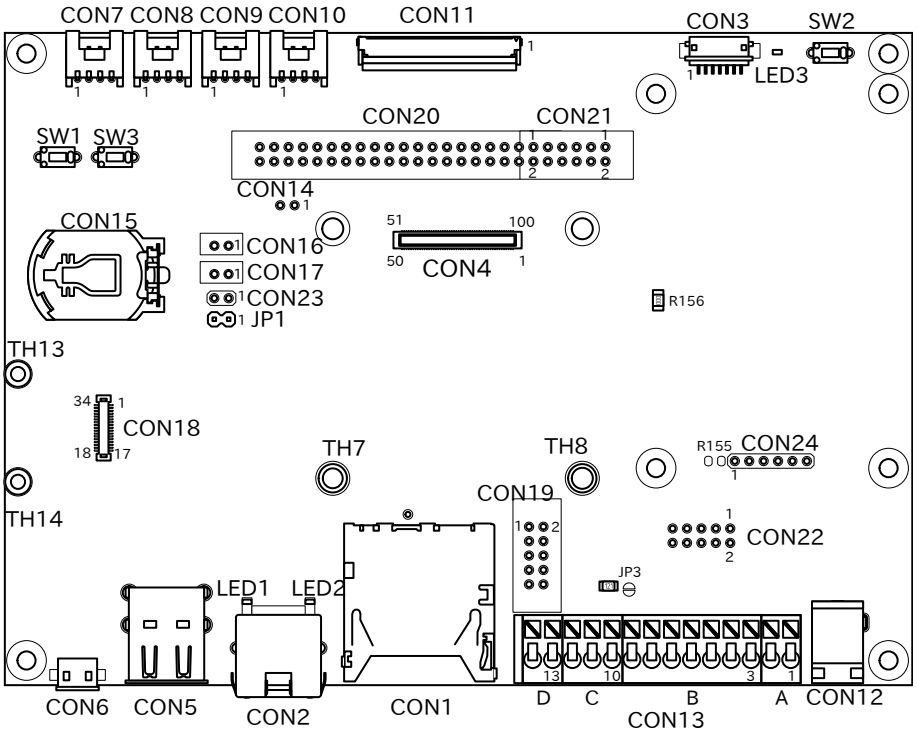


図 18.4 Armadillo-610 拡張ボードのインターフェース

表 18.4 Armadillo-610 拡張ボードのインターフェース一覧 [a]

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	CIM-K03NS	MITSUMI ELECTRIC
CON2	LAN インターフェース	TM11R-5M2-88-LP	HIROSE ELECTRIC
CON3	シリアルインターフェース	DF13A-7P-1.25H(51)	HIROSE ELECTRIC
CON4	Armadillo-610 インターフェース	DF40HC(3.0)-100DS-0.4V(51)	HIROSE ELECTRIC
CON5	USB ホストインターフェース	UBA-4RS-D14T-4D(LF)(SN)	J.S.T. Mfg.
CON6	USB OTG インターフェース	UB-MC5ABR3-SD204-4S-1	J.S.T. Mfg.
CON7	Grove インターフェース	1125R-4P	Shenzhen NS-TECH Co.,Ltd
CON8		1125R-4P	Shenzhen NS-TECH Co.,Ltd
CON9		1125R-4P	Shenzhen NS-TECH Co.,Ltd
CON10		1125R-4P	Shenzhen NS-TECH Co.,Ltd
CON11	LCD インターフェース	XF2M-5015-1A	OMRON
CON12	電源入力インターフェース	PJ-102AH	CUI
CON13A	電源出力インターフェース	TBL002A-350-14GY-2GY	CUI
CON13B	DIDO インターフェース		
CON13C	RS485 インターフェース		
CON13D	オーディオインターフェース		
CON14	電源出力インターフェース	A2-2PA-2.54DSA(71)	HIROSE ELECTRIC
CON15	RTC バックアップインターフェース	CH7410-2032LF	TAKACHI
CON16		B2B-EH(LF)(SN)	J.S.T. Mfg.
CON17	内蔵 RTC バックアップインターフェース	B2B-EH(LF)(SN)	J.S.T. Mfg.
CON18	WLAN インターフェース	AXK6F34347YG	Panasonic

部品番号	インターフェース名	型番	メーカー
CON19	拡張インターフェース	XG4C-1031	OMRON
CON20		XG4C-4031	OMRON
CON21		XG4C-1031	OMRON
CON22		XG4C-1031	OMRON
CON23	リセットインターフェース	A2-2PA-2.54DSA(71)	HIROSE ELECTRIC
CON24	電源入力インターフェース	A2-6PA-2.54DSA(71)	HIROSE ELECTRIC
JP1	起動デバイス設定ジャンパ	A2-2PA-2.54DSA(71)	HIROSE ELECTRIC
SW1	ユーザースイッチ	SKHLACA010	ALPS ELECTRIC
SW2	リセットスイッチ	SKHLACA010	ALPS ELECTRIC
SW3	ONOFF スイッチ	SKHLACA010	ALPS ELECTRIC
LED1	LAN スピード LED	SML-310MTT86	ROHM
LED2	LAN リンクアクティビティ LED	SML-310YTT86	ROHM
LED3	ユーザー LED	SML-310MTT86	ROHM
TH7	Armadillo-610 用スタッド	TH-1.6-3.0-M3	Mac-Eight
TH8		TH-1.6-3.0-M3	Mac-Eight
TH13	WLAN モジュール用スタッド	TH-1.6-1.5-M2	Mac-Eight
TH14		TH-1.6-1.5-M2	Mac-Eight

[a]部品の実装、未実装を問わず、搭載可能な部品型番を記載しています。

「表 18.4. Armadillo-610 拡張ボードのインターフェース一覧 」には搭載可能な代表型番を記載しており、実際に搭載されている型番と違うことがあります。

18.2.4.1. CON1 (SD インターフェース)

CON1 はハイスピード(最大クロック周波数: 49.5MHz)に対応した SD インターフェースです。信号線は i.MX6ULL の SD ホストコントローラ(uSDHC2)に接続されます。

SD カードに供給される電源は i.MX6ULL の UART2_RTS_B ピン(GPIO1_IO23)で制御が可能です。High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

SD コントローラ(uSDHC2)は SD インターフェース(Armadillo-610: CON1)でも使用しており、同時に使用することはできません。こちらの SD を有効にした場合、SD インターフェース(Armadillo-610: CON1)はブート時のみ利用され、ブート以降はこちらが利用されます。

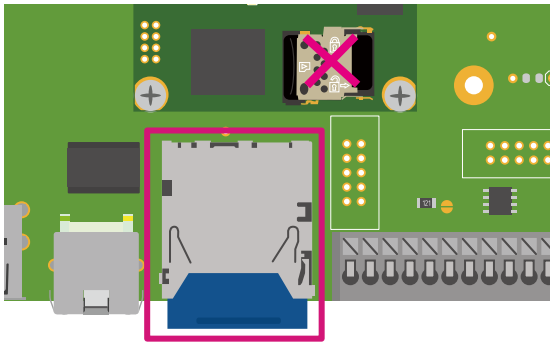


図 18.5 Armadillo-610 拡張ボード CON1

表 18.5 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	CD/DAT3	In/Out	SD データバス(bit3)、i.MX6ULL の LCD_DATA23 ピンに接続
2	CMD	In/Out	SD コマンド/レスポンス、i.MX6ULL の LCD_DATA18 ピンに接続
3	VSS	Power	電源(GND)
4	VDD	Power	電源(+3.3V)
5	CLK	Out	SD クロック、i.MX6ULL の LCD_DATA19 ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	SD データバス(bit0)、i.MX6ULL の LCD_DATA20 ピンに接続
8	DAT1	In/Out	SD データバス(bit1)、i.MX6ULL の LCD_DATA21 ピンに接続
9	DAT2	In/Out	SD データバス(bit2)、i.MX6ULL の LCD_DATA22 ピンに接続
10	CD1	In	カード検出、i.MX6ULL の UART3_RTS_B ピンに接続 (Low: カード挿入、High: カード未挿入)
11	CD2		
12	WP1	Power	電源(GND)
13			
14	WP2	In	ライトプロテクト検出、i.MX6ULL の UART3_CTS_B ピンに接続 (Low: 書き込み可能、High: 書き込み不可能)
15	GND	Power	電源(GND)
16			
17			
18			
19			

18.2.4.2. CON2(LAN インターフェース)

CON2 は 10BASE-T/100BASE-TX に対応した LAN インターフェースです。カテゴリ 5 以上の Ethernet 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、ストレートケーブルまたはクロスケーブルを自動認識して送受信端子を切り替えます。

信号線は Ethernet PHY(LAN8720AI-CP/Microchip Technology) を経由して i.MX6ULL の Ethernet コントローラ(ENET1)に接続されます。

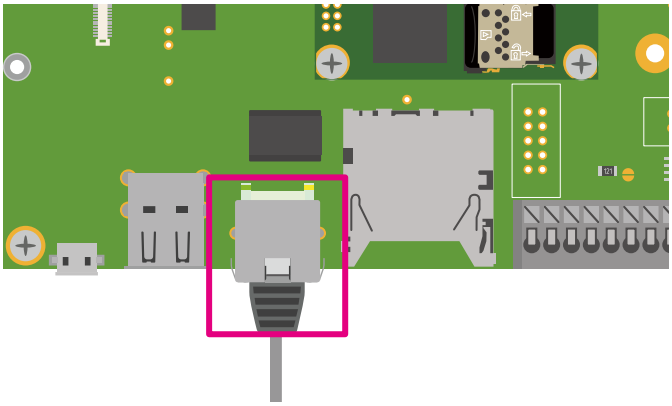


図 18.6 Armadillo-610 拡張ボード CON2

表 18.6 CON2 信号配列

ピン番号	ピン名	I/O	説明
1	TX+	In/Out	送信データ(+)
2	TX-	In/Out	送信データ(-)
3	RX+	In/Out	受信データ(+)
4	—	—	CON2 の 5 ピンと接続後に 75Ω 終端

ピン番号	ピン名	I/O	説明
5	—	—	CON2 の 4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	受信データ(-)
7	—	—	CON2 の 8 ピンと接続後に 75Ω 終端
8	—	—	CON2 の 7 ピンと接続後に 75Ω 終端

18.2.4.3. CON3(シリアルインターフェース)

CON3 は非同期(調歩同期)シリアルインターフェースです。信号は i.MX6ULL の UART コントローラ(UART1)に接続されます。CON3 の 6 ピンは i.MX6ULL の UART2_CTS_B ピン(GPIO1_IO22)に接続されており、Low レベル入力で保守モード、High レベル入力でオートブートモードで起動します。

シリアルインターフェース(Armadillo-610 拡張ボード: CON3)に USB シリアル変換アダプタを接続する際は、ケーブルの根本を軽く握り、指先でコネクタを押すようにして挿入してください。取り外しの際は、全ケーブルが均等に引きぬかれるようにケーブルをつかみ、引き抜いてください。また、両コネクタを水平にして挿入・抜去してください。30°以上傾けた状態での斜め挿入・抜去は、端子変形、ケース破損の原因となります。

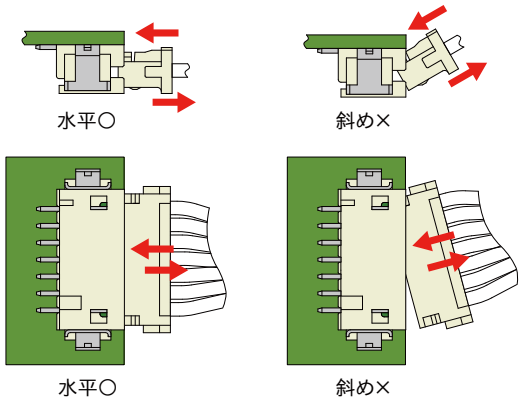


図 18.7 USB シリアル変換アダプタの挿抜角度

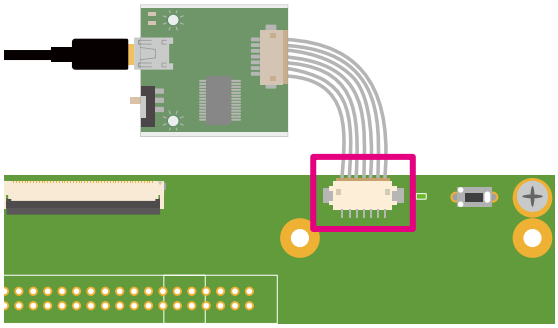


図 18.8 Armadillo-610 拡張ボード CON3

表 18.7 CON3 信号配列

ピン番号	ピン名	I/O	説明
1	UART_RXD	In	受信データ、i.MX6ULL の UART1_RX_DATA ピンに接続

ピン番号	ピン名	I/O	説明
2	GND	Power	電源(GND)
3	UART_TXD	Out	送信データ、i.MX6ULL の UART1_TX_DATA ピンに接続
4	+3.3V_IO	Power	電源(+3.3V_IO)
5	UART_CTS	In	送信可能、CON3 の 7 ピンと接続
6	BOOTLOADER_EN_B	In	起動モード設定、i.MX6ULL の UART2_CTS_B ピンに接続 (Low: 保守モード、High: オートブートモード)
7	UART_RTS	Out	送信要求、CON3 の 5 ピンと接続

18.2.4.4. CON4(Armadillo-610 インターフェース)

CON4 は Armadillo-610 と接続するためのインターフェースです。

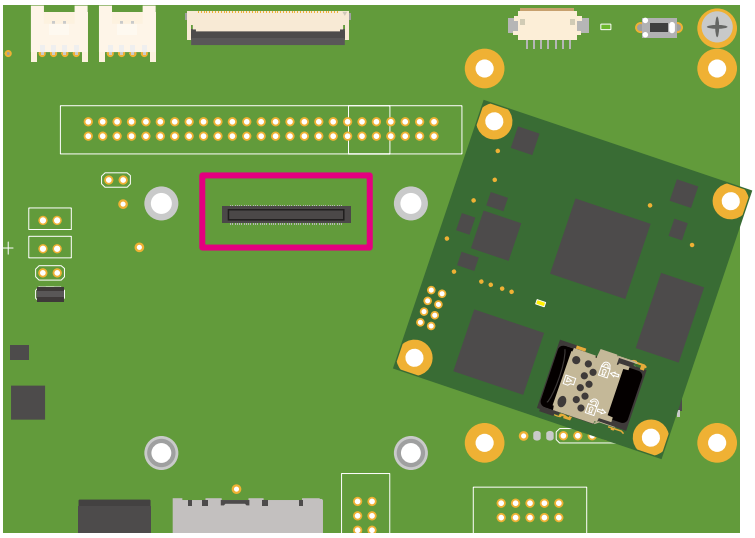


図 18.9 Armadillo-610 拡張ボード CON4

表 18.8 CON4 信号配列

ピン番号	ピン名	I/O	説明
1	USB_OTG1_DP	In/Out	USB1 のプラス側信号、CON6 の 3 ピンに接続
2	USB_OTG1_DN	In/Out	USB1 のマイナス側信号、CON6 の 2 ピンに接続
3	GND	Power	電源(GND)
4	USB_OTG2_DN	In/Out	USB2 のマイナス側信号、USB HUB の USBUP_DM ピンに接続
5	USB_OTG2_DP	In/Out	USB2 のプラス側信号、USB HUB の USBUP_DP ピンに接続
6	GND	Power	電源(GND)
7	USB_OTG1_VBUS	Power	電源(USB_OTG1_VBUS)、CON6 の 1 ピンに接続
8	USB_OTG2_VBUS	Power	電源(USB_OTG2_VBUS)、CON5 の 1 ピンと 5 ピンに接続
9	SPEEDLED	In	LAN スピード LED 用信号、Ethernet PHY の LED2 ピンに接続
10	LINK_ACTLED	In	LAN リンクアクティビティ LED 用信号、Ethernet PHY の LED1 ピンに接続
11	USB1_PWREN	In	USB1 用パワースイッチ切り替え信号、パワースイッチのイネーブルピンに接続 (High: 電源供給、Low: 電源切断)
12	USB2_PWREN	In	USB2 用パワースイッチ切り替え信号、パワースイッチのイネーブルピンに接続 (High: 電源供給、Low: 電源切断)
13	RTC_INT_B	Out	リアルタイムクロック割り込み信号、リアルタイムクロックの割り込みピンに接続

ピン番号	ピン名	I/O	説明
14	ADC_IN4	In/Out	拡張入出力、CON11 の 34 ピン、CON20 の 37 ピン、CON10 の 1 ピンに接続
15	ADC_IN3	In/Out	拡張入出力、CON11 の 35 ピン、CON20 の 38 ピン、CON9 の 1 ピンに接続
16	ADC_IN2	In/Out	拡張入出力、CON11 の 36 ピン、CON20 の 39 ピン、CON9 の 2 ピンに接続
17	ADC_IN1	In/Out	拡張入出力、CON11 の 37 ピン、CON20 の 40 ピン、CON10 の 2 ピンに接続
18	LCD_DATA00	In/Out	拡張入出力、CON11 の 13 ピン、CON20 の 13 ピンに接続
19	LCD_DATA01	In/Out	拡張入出力、CON11 の 14 ピン、CON20 の 14 ピンに接続
20	LCD_DATA02	In/Out	拡張入出力、CON11 の 15 ピン、CON20 の 15 ピンに接続
21	LCD_DATA03	In/Out	拡張入出力、CON11 の 16 ピン、CON20 の 16 ピンに接続
22	LCD_DATA04	In/Out	拡張入出力、CON11 の 17 ピン、CON20 の 17 ピンに接続
23	LCD_DATA05	In/Out	拡張入出力、CON11 の 18 ピン、CON20 の 18 ピンに接続
24	LCD_DATA06	In/Out	拡張入出力、CON11 の 20 ピン、CON20 の 20 ピンに接続
25	LCD_DATA07	In/Out	拡張入出力、CON11 の 21 ピン、CON20 の 21 ピンに接続
26	LCD_DATA08	In/Out	拡張入出力、CON11 の 22 ピン、CON20 の 22 ピンに接続
27	LCD_DATA09	In/Out	拡張入出力、CON11 の 23 ピン、CON20 の 23 ピンに接続
28	LCD_DATA10	In/Out	拡張入出力、CON11 の 24 ピン、CON20 の 24 ピンに接続
29	LCD_DATA11	In/Out	拡張入出力、CON11 の 25 ピン、CON20 の 25 ピンに接続
30	LCD_DATA12	In/Out	拡張入出力、CON11 の 27 ピン、CON20 の 27 ピンに接続
31	LCD_DATA13	In/Out	拡張入出力、CON11 の 28 ピン、CON20 の 28 ピンに接続
32	LCD_DATA14	In/Out	拡張入出力、CON11 の 29 ピン、CON20 の 29 ピンに接続
33	LCD_DATA15	In/Out	拡張入出力、CON11 の 30 ピン、CON20 の 30 ピンに接続
34	LCD_DATA16	In/Out	拡張入出力、CON11 の 31 ピン、CON20 の 31 ピンに接続
35	LCD_DATA17	In/Out	拡張入出力、CON11 の 32 ピン、CON20 の 32 ピンに接続
36	GND	Power	電源(GND)
37	LCD_CLK	In/Out	拡張入出力、CON11 の 8 ピン、CON20 の 7 ピンに接続
38	LCD_HSYNC	In/Out	拡張入出力、CON11 の 9 ピン、CON20 の 8 ピンに接続
39	LCD_VSYNC	In/Out	拡張入出力、CON11 の 10 ピン、CON20 の 9 ピンに接続
40	LCD_ENABLE	In/Out	拡張入出力、CON11 の 11 ピン、CON20 の 10 ピンに接続
41	PWM5_OUT	In/Out	拡張入出力、CON11 の 12 ピン、CON20 の 11 ピンに接続
42	BJP1	Out	起動デバイス設定用信号、JP1 に接続
43	EXT_SW1	Out	ユーザースイッチ、SW1 に接続
44	EXT_RESET_B	Out	システムリセット、CON23 の 1 ピンに接続
45	+3.3V_IO	Power	電源(+3.3V_IO)
46	+3.3V_IO	Power	電源(+3.3V_IO)
47	VIN	Power	電源(VIN)
48	VIN	Power	電源(VIN)
49	VIN	Power	電源(VIN)
50	VIN	Power	電源(VIN)
51	GND	Power	電源(GND)
52	GND	Power	電源(GND)
53	+5V_IO	Power	電源(+5V_IO)
54	+5V_IO	Power	電源(+5V_IO)
55	I2C2_SDA	In/Out	I2C データ信号、CON11 の 49 ピン、CON21 の 10 ピン、CON8 の 2 ピン、リアルタイムクロックの SDA ピンに接続
56	I2C2_SCL	In	I2C クロック信号、CON11 の 48 ピン、CON21 の 9 ピン、CON8 の 1 ピン、リアルタイムクロックの SCL ピンに接続
57	SAI1_TX_SYNC	In/Out	拡張入出力、CON11 の 47 ピン、CON21 の 8 ピンに接続
58	SAI1_TX_BCLK	In/Out	拡張入出力、CON11 の 46 ピン、CON21 の 7 ピンに接続
59	SAI1_RX_DATA	In/Out	拡張入出力、CON11 の 45 ピン、CON21 の 6 ピンに接続
60	SAI1_TX_DATA	In/Out	拡張入出力、CON11 の 44 ピン、CON21 の 5 ピンに接続
61	SAI1_RX_SYNC	In/Out	拡張入出力、CON11 の 43 ピン、CON21 の 4 ピンに接続

ピン番号	ピン名	I/O	説明
62	SAI1_MCLK	In/Out	拡張入出力、CON11 の 42 ピン、CON21 の 3 ピン、CON7 の 1 ピンに接続
63	GPIO4_IO24	In/Out	拡張入出力、CON11 の 41 ピン、CON20 の 36 ピンに接続
64	GPIO4_IO21	In/Out	拡張入出力、CON11 の 40 ピン、CON20 の 35 ピンに接続
65	GPIO4_IO18	In/Out	拡張入出力、CON11 の 39 ピン、CON20 の 34 ピンに接続
66	RS485_DE	In	RS485 送信イネーブル信号、RS485 トランシーバの DE ピン、CON22 の 6 ピンに接続
67	RS485_RE_N	In	RS485 受信イネーブル信号、RS485 トランシーバの RE ピン、CON22 の 5 ピンに接続
68	RS485_RX	Out	RS485 受信データ、RS485 トランシーバの RO ピン、CON22 の 4 ピンに接続
69	RS485_TX	In	RS485 送信データ、RS485 トランシーバの DI ピン、CON22 の 3 ピンに接続
70	SD2_DATA3	In/Out	SD データバス(bit3)、CON1 の 1 ピンに接続
71	SD2_DATA2	In/Out	SD データバス(bit2)、CON1 の 9 ピンに接続
72	SD2_DATA1	In/Out	SD データバス(bit1)、CON1 の 8 ピンに接続
73	SD2_DATA0	In/Out	SD データバス(bit0)、CON1 の 7 ピンに接続
74	SD2_CLK	In	SD クロック、CON1 の 5 ピンに接続
75	GND	Power	電源(GND)
76	SD2_CMD	In/Out	SD コマンド/レスポンス、CON1 の 2 ピンに接続
77	PWRON	Out	パワーマネジメント IC の PWRON 信号、SW2 に接続
78	ONOFF	Out	i.MX6ULL の ON/OFF 信号、SW3 に接続
79	RTC_BAT	Power	電源(RTC_BAT)、CON17 の 1 ピンに接続
80	EXT_LED1	In	LED3 に接続(High: 点灯、Low: 消灯)
81	AMP_SD_B	In	オーディオアンプのシャットダウンピンに接続 (High: オーディオ開始、Low: オーディオ停止)
82	DO1	In	CON13 の DO1 制御ピンに接続 (High: DO1 ショート、Low: DO1 オープン)
83	DEBUG_UART_TX	In	送信データ、CON3 の 3 ピンに接続
84	DO2	In	CON13 の DO 制御ピンに接続 (High: DO2 ショート、Low: DO2 オープン)
85	DEBUG_UART_RX	Out	受信データ、CON3 の 1 ピンに接続
86	SD_PWREN	In	SD 用パワースイッチ切り替え信号、パワースイッチのイネーブルピンに接続 (High: 電源供給、Low: 電源切断)
87	MAINT_EN_B	Out	起動モード設定、CON3 の 6 ピンに接続
88	DI2	Out	DI2 入力、CON13 の 4 ピンに接続
89	DI1	Out	DI1 入力、CON13 の 5 ピンに接続
90	MQS	In	オーディオ入力、オーディオアンプに接続
91	SD2_WP	Out	ライトプロテクト検出、CON1 の 14 ピン、CON19 の 10 ピンに接続
92	SD2_CD_B	Out	カード検出、CON1 の 10 ピン、CON19 の 9 ピンに接続
93	WLAN_PWREN	In	WLAN 用パワースイッチ切り替え信号、パワースイッチのイネーブルピンに接続 (High: 電源供給、Low: 電源切断)
94	USB1_OTG_ID	Out	CON6 の 4 ピンに接続
95	GND	Power	電源(GND)
96	Ether_RXN	In/Out	Ethernet 送信/受信データ(-) CH2、Ethernet トランスに接続
97	Ether_RXP	In/Out	Ethernet 送信/受信データ(+) CH2、Ethernet トランスに接続
98	GND	Power	電源(GND)
99	Ether_TXN	In/Out	Ethernet 送信/受信データ(-) CH1、Ethernet トランスに接続
100	Ether_TXP	In/Out	Ethernet 送信/受信データ(+) CH1、Ethernet トランスに接続

18.2.4.5. CON5(USB ホストインターフェース)

CON5 は USB ホストインターフェースです。2 段のコネクタを実装しており、信号線は USB HUB を経由して i.MX6ULL の USB コントローラ(USB OTG2)に接続されます。

供給される電源は i.MX6ULL の CSI_MCLK ピン(GPIO4_IO17)で制御が可能で、High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

- データ転送モード
  - High Speed(480Mbps)
  - Full Speed(12Mbps)
  - Low Speed(1.5Mbps)

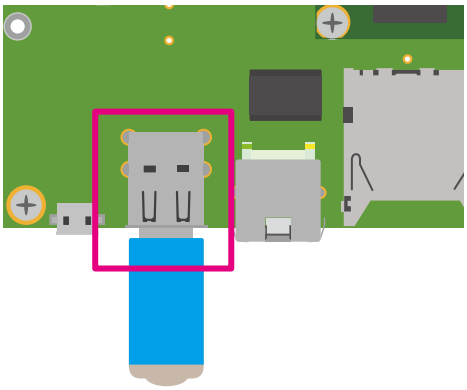


図 18.10 Armadillo-610 拡張ボード CON5

表 18.9 CON5 信号配列

ピン番号	ピン名	I/O	説明
1	+5V	Power	電源(+5V)
2	USB_L_DN	In/Out	USB 下段のマイナス側信号、USB HUB(Port2)を経由して i.MX6ULL の USB_OTG2_DN ピンに接続
3	USB_L_DP	In/Out	USB 下段のプラス側信号、USB HUB(Port2)を経由して i.MX6ULL の USB_OTG2_DP ピンに接続
4	GND	Power	電源(GND)
5	+5V	Power	電源(+5V)
6	USB_U_DN	In/Out	USB 上段のマイナス側信号、USB HUB(Port3)を経由して i.MX6ULL の USB_OTG2_DN ピンに接続
7	USB_U_DP	In/Out	USB 上段のプラス側信号、USB HUB(Port3)を経由して i.MX6ULL の USB_OTG2_DP ピンに接続
8	GND	Power	電源(GND)

18.2.4.6. CON6(USB OTG インターフェース)

CON6 は USB OTG インターフェースです。信号線は i.MX6ULL の USB コントローラ(USB OTG1)に接続されます。

供給される電源は i.MX6ULL の UART1_RTS_B ピン(GPIO1_IO19)および CON6 の 4 ピン(USB_ID)により制御が可能です。i.MX6ULL のの UART1_RTS_B ピン(GPIO1_IO19)から High レベル出力かつ CON6 の 4 ピン(USB_ID)から Low レベル入力で電源が供給され、CON6 の 4 ピン(USB_ID)がオープンで電源切断されます。

- データ転送モード
  - High Speed(480Mbps)

- Full Speed(12Mbps)
- Low Speed(1.5Mbps)

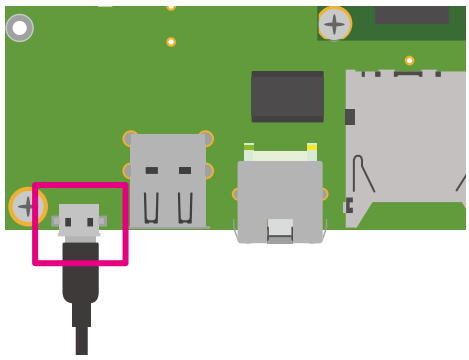


図 18.11 Armadillo-610 拡張ボード CON6

表 18.10 CON6 信号配列

ピン番号	ピン名	I/O	説明
1	+5V	Power	電源(+5V_IO)
2	USB_DN	In/Out	USB のマイナス側信号、USB HUB を経由して i.MX6ULL の USB_OTG1_DN ピンに接続
3	USB_DP	In/Out	USB のプラス側信号、USB HUB を経由して i.MX6ULL の USB_OTG1_DP ピンに接続
4	USB_ID	In	USB の ID 信号、i.MX6ULL の GPIO1_IO24 ピンに接続
5	GND	Power	電源(GND)

18.2.4.7. CON7、CON8、CON9、CON10(Grove インターフェース)

CON7、CON8、CON9、CON10 は Seeed 社が推奨するコネクタ規格「Grove システム」に対応した Grove モジュール接続用のインターフェースです。

マルチプレクスの設定で機能を割り当てることで、GPIO、UART、I2C、A/D で拡張する Grove モジュールを接続することができます。

CON11(LCD インターフェース)、CON20(拡張インターフェース)、CON21(拡張インターフェース)と共通の信号線が接続されているため、同時に使用できません。また、CON8 の I2C 信号は基板上のリアルタイムクロックにも接続されておりますので、マルチプレクスの設定を変更する際には、ご注意ください。

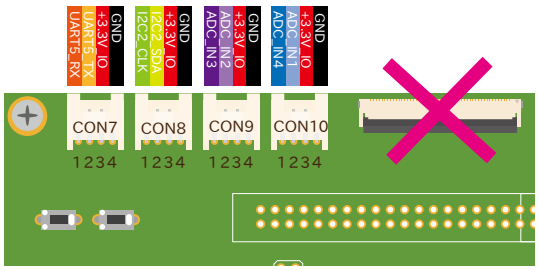


図 18.12 Armadillo-610 拡張ボード CON7、CON8、CON9、CON10

表 18.11 CON7 信号配列

ピン番号	ピン名	I/O	説明
1	UART5_RX	I/O	拡張入出力、i.MX6ULL の CSI_DATA01 ピンに接続
2	UART5_TX	I/O	拡張入出力、i.MX6ULL の CSI_DATA00 ピンに接続
3	+3.3V_IO	Power	電源(+3.3V_IO)
4	GND	Power	電源(GND)

表 18.12 CON8 信号配列

ピン番号	ピン名	I/O	説明
1	I2C2_CLK	I/O	拡張入出力、i.MX6ULL の CSI_HSYNC ピンに接続
2	I2C2_SDA	I/O	拡張入出力、i.MX6ULL の CSI_VSYNC ピンに接続
3	+3.3V_IO	Power	電源(+3.3V_IO)
4	GND	Power	電源(GND)

表 18.13 CON9 信号配列

ピン番号	ピン名	I/O	説明
1	ADC_IN3	I/O	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続
2	ADC_IN2	I/O	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続
3	+3.3V_IO	Power	電源(+3.3V_IO)
4	GND	Power	電源(GND)

表 18.14 CON10 信号配列

ピン番号	ピン名	I/O	説明
1	ADC_IN4	I/O	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続
2	ADC_IN1	I/O	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続
3	+3.3V_IO	Power	電源(+3.3V_IO)
4	GND	Power	電源(GND)

18.2.4.8. CON11(LCD インターフェース)

CON11 はデジタル RGB 入力を持つ液晶パネルモジュールなどを接続することができる、LCD インターフェースです。信号線は i.MX6ULL の LCD インターフェース(eLCDIF)等に接続されます。

LCD オプションセット(7 インチタッチパネル WVGA 液晶)(型番: OP-LCD70EXT-00)、Armadillo-400 シリーズ LCD オプションセット(4.3 インチタッチパネル WQVGA 液晶)(型番: OP-A400-LCD43EXT-L01)を接続可能です。

オプションセットの詳細につきましては「18.3. LCD オプションセット(7 インチタッチパネル WVGA 液晶)」、「18.4. Armadillo-400 シリーズ LCD オプションセット」をご確認ください。

LCD オプションセット(7 インチタッチパネル WVGA 液晶)を使用する場合、I2C2_SCL 信号がバックライト用の PWM 信号として使用されるため、基板上のリアルタイムクロックが使用できなくなります。

Armadillo-400 シリーズ LCD オプションセットを使用する場合、LCD オプションセット側にもリアルタイムクロックが接続されており、アドレスが被っているため、アクセスすると信号が衝突します。リアルタイムクロックにアクセスしない、もしくはどちらかのリアルタイムクロックを切り離してご使用ください。Armadillo-610 拡張ボード側は R146、

R147(基板裏)の抵抗を未実装にすることでリアルタイムクロックを切り離すことが可能です。

CON7、CON8、CON9、CON10(Groveインターフェース)、CON20(拡張インターフェース)、CON21(拡張インターフェース)と共通の信号線が接続されているため、同時に使用できません。また、48、49ピンの信号線は基板上のリアルタイムクロックにも接続されており、I2Cで使用しておりますので、マルチプレクスの設定を変更する際には、ご注意ください。

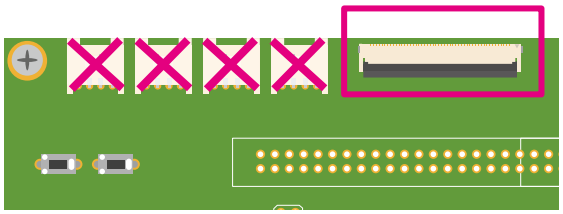


図 18.13 Armadillo-610 拡張ボード CON11

表 18.15 CON11 信号配列

ピン番号	ピン名	I/O	説明
1	+5V	Power	電源出力(+5V)
2	+5V	Power	電源出力(+5V)
3	+5V	Power	電源出力(+5V)
4	+3.3V	Power	電源出力(+3.3V)
5	+3.3V	Power	電源出力(+3.3V)
6	GND	Power	電源(GND)
7	GND	Power	電源(GND)
8	LCD_CLK	In/Out	拡張入出力、i.MX6ULL の LCD_CLK ピンに接続
9	LCD_HSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_HSYNC ピンに接続
10	LCD_VSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_VSYNC ピンに接続
11	LCD_ENABLE	In/Out	拡張入出力、i.MX6ULL の LCD_ENABLE ピンに接続
12	PWM5_OUT	In/Out	拡張入出力、i.MX6ULL の NAND_DQS ピンに接続
13	LCD_DATA00	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続
14	LCD_DATA01	In/Out	拡張入出力、i.MX6ULL の LCD_DATA01 ピンに接続
15	LCD_DATA02	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続
16	LCD_DATA03	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続
17	LCD_DATA04	In/Out	拡張入出力、i.MX6ULL の LCD_DATA04 ピンに接続
18	LCD_DATA05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続
19	GND	Power	電源(GND)
20	LCD_DATA06	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続
21	LCD_DATA07	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続
22	LCD_DATA08	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続
23	LCD_DATA09	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続
24	LCD_DATA10	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続
25	LCD_DATA11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続
26	GND	Power	電源(GND)
27	LCD_DATA12	In/Out	拡張入出力、i.MX6ULL の LCD_DATA12 ピンに接続

ピン番号	ピン名	I/O	説明
28	LCD_DATA13	In/Out	拡張入出力、i.MX6ULL の LCD_DATA13 ピンに接続
29	LCD_DATA14	In/Out	拡張入出力、i.MX6ULL の LCD_DATA14 ピンに接続
30	LCD_DATA15	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続
31	LCD_DATA16	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続
32	LCD_DATA17	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続
33	GND	Power	電源(GND)
34	ADC_IN4	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続、0.01uF のコンデンサが接続されています。
35	ADC_IN3	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続、0.01uF のコンデンサが接続されています。
36	ADC_IN2	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続、0.01uF のコンデンサが接続されています。
37	ADC_IN1	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続、0.01uF のコンデンサが接続されています。
38	GND	Power	電源(GND)
39	GPIO4_IO18	In/Out	拡張入出力、i.MX6ULL の CSI_PIXCLK ピンに接続
40	GPIO4_IO21	In/Out	拡張入出力、i.MX6ULL の CSI_DATA00 ピンに接続
41	GPIO4_IO24	In/Out	拡張入出力、i.MX6ULL の CSI_DATA03 ピンに接続
42	SAI1_MCLK	In/Out	拡張入出力、i.MX6ULL の CSI_DATA01 ピンに接続
43	SAI1_RX_SYNC	In/Out	拡張入出力、i.MX6ULL の CSI_DATA02 ピンに接続
44	SAI1_TX_DATA	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
45	SAI1_RX_DATA	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
46	SAI1_TX_BCLK	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
47	SAI1_TX_SYNC	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
48	I2C2_SCL	In/Out	拡張入出力、i.MX6ULL の CSI_HSYNC ピンに接続
49	I2C2_SDA	In/Out	拡張入出力、i.MX6ULL の CSI_VSYNC ピンに接続
50	GND	Power	電源(GND)

18.2.4.9. CON12(電源入力インターフェース)

CON12 は電源供給用のインターフェースです。DC ジャックが実装されており、「図 18.14. AC アダプタの極性マーク」と同じ極性マークのある AC アダプタが使用できます。

図 18.14 AC アダプタの極性マーク

AC アダプタから電源を供給する際、DC プラグを Armadillo-610 拡張ボードの DC ジャックに接続してから、AC プラグをコンセントに接続してください。突入電流により、故障する可能性があります。

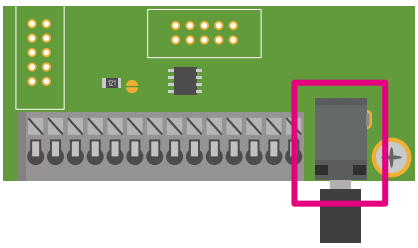


図 18.15 Armadillo-610 拡張ボード CON12

表 18.16 CON12 信号配列

ピン番号	ピン名	I/O	説明
1	EXT_VIN	Power	電源(EXT_VIN)
2	GND	Power	電源(GND)
3	GND	Power	電源(GND)

18.2.4.10. CON13A(電源出力インターフェース)

CON13A は電源出力用インターフェースです。端子台が実装されています。CON12 から入力した電源が出力されます。

電源入力として使用することも可能ですが、同時に CON12 から電源供給しないようご注意ください。

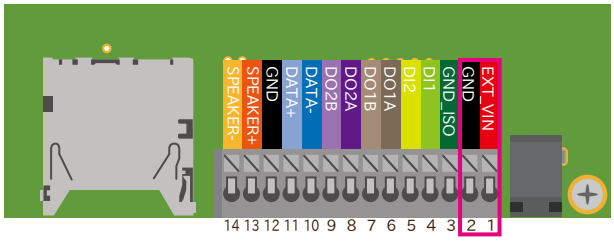


図 18.16 CON13A の配置

表 18.17 CON13A 信号配列

ピン番号	ピン名	I/O	説明
1	EXT_VIN	Power	電源(EXT_VIN)
2	GND	Power	電源(GND)

18.2.4.11. CON13B(DIDO インターフェース)

CON13B は絶縁デジタル入出力インターフェースです。

デジタル入力部はフォトカプラによる絶縁入力となっています。入力部を駆動するためには外部に電源(定格電圧 DC 3.3～12V)が必要となります。

デジタル出力部はフォトリレーによる絶縁出力(無極性)となっています。出力部を駆動するためには外部に電源が必要となります。出力 1 点につき最大電流 200mA(最大電圧 DC 48V)まで駆動可能です。

動作確認時には CON13A(電源出力インターフェース)から電源を取るのが便利です。

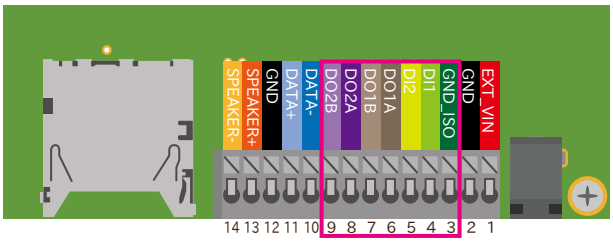


図 18.17 CON13B の配置

表 18.18 CON13B 信号配列

ピン番号	ピン名	I/O	説明
3	GND_ISO	Power	電源(GND_ISO)
4	DI1	In	デジタル入力 1
5	DI2	In	デジタル入力 2
6	DO1A	—	デジタル出力 1A
7	DO1B	—	デジタル出力 1B
8	DO2A	—	デジタル出力 2A
9	DO2B	—	デジタル出力 2B

CON13 の 3 ピンの GND_ISO は絶縁されています。

18.2.4.12. CON13C(RS485 インターフェース)

CON13C は RS485(半二重)のシリアルインターフェースです。信号線は RS485 トランシーバを経由して、i.MX6ULL の UART コントローラ(UART2)に接続されています。

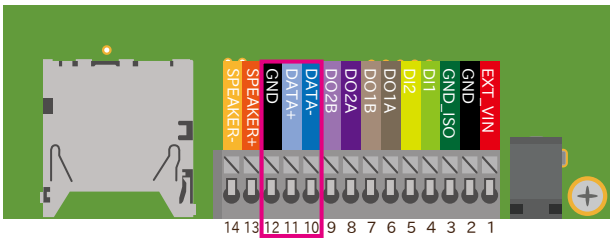


図 18.18 CON13C の配置

表 18.19 CON13C 信号配列

ピン番号	ピン名	I/O	説明
10	DATA-	In/Out	送受信データ(-)
11	DATA+	In/Out	送受信データ(+)
12	GND	Power	電源(GND)

18.2.4.13. CON13D(オーディオインターフェース)

CON13D はモノラルのオーディオ出力インターフェースです。1.4W オーディオアンプを経由して i.MX6ULL の Medium Quality Sounc(MQS)に接続されています。8Ω スピーカーが接続可能です。

Armadillo-610 開発セットに付属しているのは、8Ω スピーカーです。

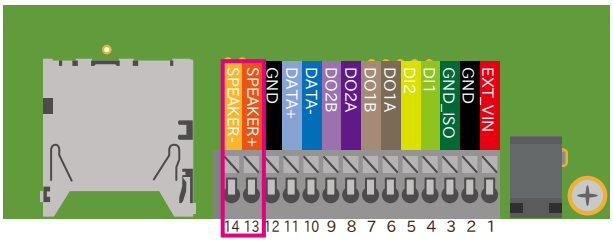


図 18.19 CON13D の配置

表 18.20 CON13D 信号配列

ピン番号	ピン名	I/O	説明
13	SPEAKER+	Out	スピーカー出力(+)
14	SPEAKER-	Out	スピーカー出力(-)

18.2.4.14. CON14(電源出力インターフェース)

CON14 は電源出力インターフェースです。Armadillo-610 で生成する+5V_IO が接続されています。

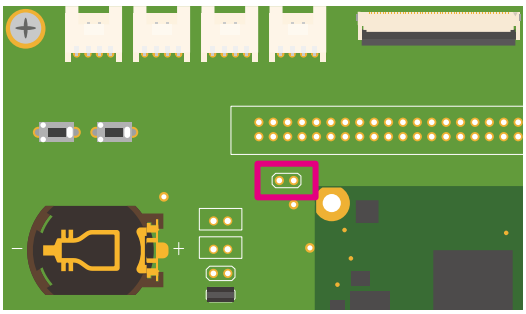


図 18.20 Armadillo-610 拡張ボード CON14

表 18.21 CON14 信号配列

ピン番号	ピン名	I/O	説明
1	+5V_IO	Power	電源(+5V_IO)
2	GND	Power	電源(GND)

18.2.4.15. CON15、CON16(RTC バックアップインターフェース)

CON15、CON16 は Armadillo-610 拡張ボードに搭載しているリアルタイムクロックのバックアップ用インターフェースです。

別途バックアップ用の電源を接続することで、Armadillo-610 の電源(VIN)が切断された場合でも、時刻データを保持することが可能です。

Armadillo-610 拡張ボードに搭載しているリアルタイムクロックは、i.MX6ULL 内蔵のリアルタイムクロックよりも消費電力が少なく、精度が良いものとなっております。

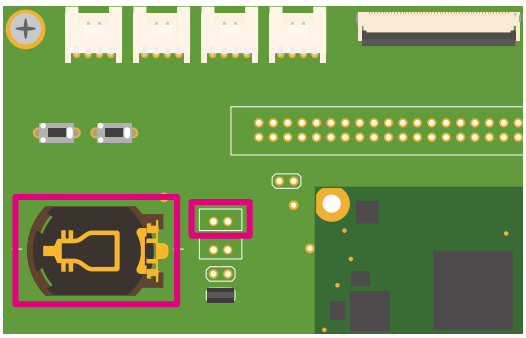


図 18.21 Armadillo-610 拡張ボード CON15、CON16

表 18.22 CON16 信号配列

ピン番号	ピン名	I/O	説明
1	EXT_RTC_BAT	Power	電源(EXT_RTC_BAT)
2	GND	Power	電源(GND)

表 18.23 CON15、CON16 対応電池の例とバックアップ時間

対応電池	バックアップ時間(参考値)
CR2032	6.2 年

CON15、CON16 は共通の端子に接続されており、同時に使用できません。

リアルタイムクロックの平均月差は周囲温度 25℃で±8 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。

CON15、CON16 はリチウムコイン電池からの電源供給を想定しています。リチウムコイン電池以外から電源を供給する場合、回路図、部品表にて搭載部品をご確認の上、絶対定格値を超えない範囲でご使用ください。

18.2.4.16. CON17(内蔵 RTC バックアップインターフェース)

CON17 は i.MX6ULL の低消費電力ドメインにある SRTC(Secure Real Time Clock)のデータ等を保持するためのバックアップ用インターフェースです。

別途バックアップ用の電源を接続することで、Armadillo-610 の電源(VIN)が切断された場合でも、データを保持することが可能です。

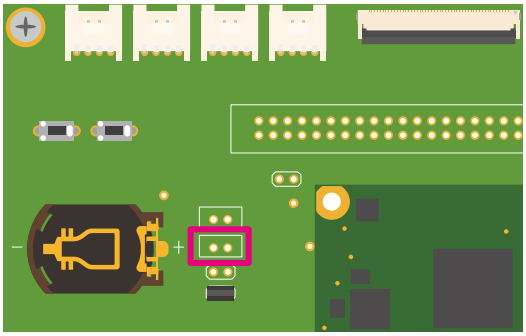


図 18.22 Armadillo-610 拡張ボード CON17

表 18.24 CON17 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	電源(RTC_BAT)、Armadillo-610 のパワーマネジメント IC の LICELL ピンに接続
2	GND	Power	電源(GND)

表 18.25 CON17 対応電池の例とバックアップ時間

対応電池	バックアップ時間(参考値)
CR2032	約 4 か月 [a]

[a]内蔵リアルタイムクロックは、一般的なリアルタイムクロック IC よりも消費電力が高いため、外付けバッテリの消耗が速くなります。

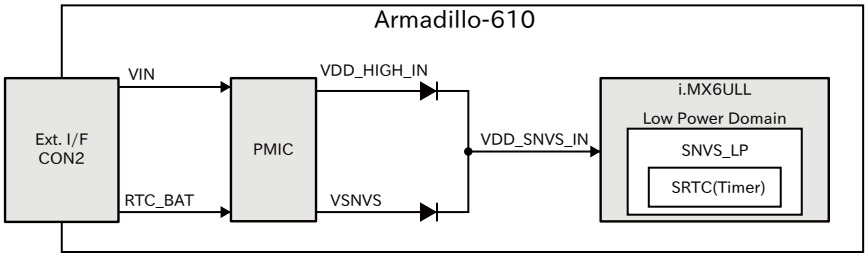


図 18.23 バックアップ電源供給回路

低消費電力モードに速やかに移行するためには、バックアップ電源(RTC_BAT)を供給した直後に一度、Armadillo-610 への供給電源(VIN)を 100 ミリ秒以上供給する必要があります。

RTC_BAT の入力電圧範囲は 2.75～3.3V です。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ので、入力電圧範囲内でご使用ください。

内蔵リアルタイムクロックの平均月差は周囲温度 25℃で±70 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。

18.2.4.17. CON18(WLAN インターフェース)

CON18 は Armadillo-WLAN モジュール(AWL13)接続用のインターフェースです。Armadillo-WLAN(AWL13)は USB 起動モードに設定されます。

Armadillo-WLAN モジュール(AWL13)の仕様については、Armadillo サイトで公開している Armadillo-WLAN(AWL13)の各種ドキュメントをご確認ください。

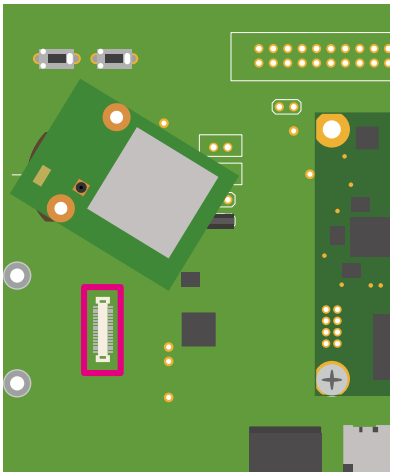


図 18.24 Armadillo-610 拡張ボード CON18

表 18.26 CON18 信号配列

ピン番号	ピン名	I/O	説明
1	SDDATA1	-	未接続
2	SDDATA0	-	未接続
3	GND	Power	電源(GND)
4	GND	Power	電源(GND)
5	USB_DM	In/Out	USB マイナス側信号
6	USB_DP	In/Out	USB プラス側信号
7	SDCLK	-	未接続
8	+3.3V	Power	電源(+3.3V)
9	NC	-	未接続
10	SDCMD	-	未接続
11	SDDATA3	-	未接続
12	SDDATA2	-	未接続
13	UART_RXD	-	未接続
14	UART_TXD	-	未接続
15	BOOT_SEL1	Out	起動モード設定、USB 起動モードに設定
16	BOOT_SEL0	Out	
17	HOST_SEL	Out	
18	FLASH_RXD	-	未接続
19	FLASH_CSB	-	未接続
20	FLASH_CLK	-	未接続
21	FLASH_TXD	Out	47kΩ プルダウン

ピン番号	ピン名	I/O	説明
22	FLASH_SEL	-	未接続
23	GPIO0	-	未接続
24	GPIO1	-	未接続
25	M_ANA	-	未接続
26	GPIO2	-	未接続
27	GPIO6	-	未接続
28	HRST	Out	+3.3V に接続
29	PRST	-	未接続
30	TMS	-	未接続
31	TCK	-	未接続
32	TDI	-	未接続
33	TDO	-	未接続
34	TRSTB	-	未接続

18.2.4.18. CON19(拡張インターフェース)

CON19 は SD 用の機能を割り当て可能な信号線を接続した、SD 拡張用のインターフェースです。

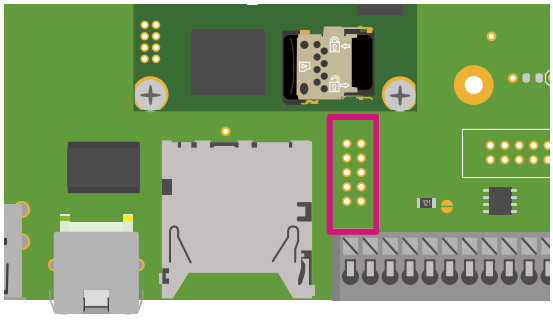


図 18.25 Armadillo-610 拡張ボード CON19

CON1(SD インターフェース)と共通の信号線が接続されているため、同時に使用できません。R78～R83、R86、R87(CON1 周辺、R79 以外は基板裏)の抵抗を未実装にすることで、CON1(SD インターフェース)と切り離すことが可能です。

表 18.27 CON19 信号配列

ピン番号	ピン名	I/O	説明
1	+3.3V_IO	Power	電源(+3.3V_IO)
2	GND	Power	電源(GND)
3	SD2_CLK	In/Out	拡張入出力、i.MX6ULL の LCD_DATA19 ピンに接続
4	SD2_CMD	In/Out	拡張入出力、i.MX6ULL の LCD_DATA18 ピンに接続
5	SD2_DATA0	In/Out	拡張入出力、i.MX6ULL の LCD_DATA20 ピンに接続
6	SD2_DATA1	In/Out	拡張入出力、i.MX6ULL の LCD_DATA21 ピンに接続
7	SD2_DATA2	In/Out	拡張入出力、i.MX6ULL の LCD_DATA22 ピンに接続
8	SD2_DATA3	In/Out	拡張入出力、i.MX6ULL の LCD_DATA23 ピンに接続
9	SD2_CD_B	In/Out	拡張入出力、i.MX6ULL の UART3_RTS_B ピンに接続
10	SD2_WP	In/Out	拡張入出力、i.MX6ULL の UART3_CTS_B ピンに接続

18.2.4.19. CON20(拡張インターフェース)

CON20 は主に LCD やタッチパネル用の機能を割り当て可能な信号線を接続した、LCD 拡張用インターフェースです。

CON7、CON9、CON10(Grove インターフェース)、CON11(LCD インターフェース)と共通の信号線が接続されているため、同時に使用できません。

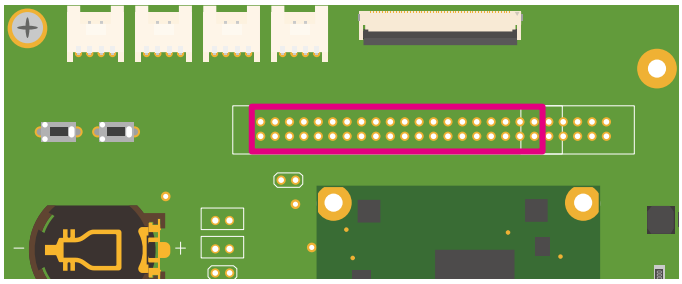


図 18.26 Armadillo-610 拡張ボード CON20

表 18.28 CON20 信号配列

ピン番号	ピン名	I/O	説明
1	+5V	Power	電源出力(+5V)
2	+5V	Power	電源出力(+5V)
3	+3.3V	Power	電源出力(+3.3V)
4	+3.3V	Power	電源出力(+3.3V)
5	GND	Power	電源(GND)
6	GND	Power	電源(GND)
7	LCD_CLK	In/Out	拡張入出力、i.MX6ULL の LCD_CLK ピンに接続
8	LCD_HSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_HSYNC ピンに接続
9	LCD_VSYNC	In/Out	拡張入出力、i.MX6ULL の LCD_VSYNC ピンに接続
10	LCD_ENABLE	In/Out	拡張入出力、i.MX6ULL の LCD_ENABLE ピンに接続
11	PWM5_OUT	In/Out	拡張入出力、i.MX6ULL の NAND_DQS ピンに接続
12	GND	Power	電源(GND)
13	LCD_DATA00	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続
14	LCD_DATA01	In/Out	拡張入出力、i.MX6ULL の LCD_DATA01 ピンに接続
15	LCD_DATA02	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続
16	LCD_DATA03	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続
17	LCD_DATA04	In/Out	拡張入出力、i.MX6ULL の LCD_DATA04 ピンに接続
18	LCD_DATA05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続
19	GND	Power	電源(GND)
20	LCD_DATA06	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続
21	LCD_DATA07	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続
22	LCD_DATA08	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続
23	LCD_DATA09	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続
24	LCD_DATA10	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続
25	LCD_DATA11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続
26	GND	Power	電源(GND)
27	LCD_DATA12	In/Out	拡張入出力、i.MX6ULL の LCD_DATA12 ピンに接続
28	LCD_DATA13	In/Out	拡張入出力、i.MX6ULL の LCD_DATA13 ピンに接続

ピン番号	ピン名	I/O	説明
29	LCD_DATA14	In/Out	拡張入出力、i.MX6ULL の LCD_DATA14 ピンに接続
30	LCD_DATA15	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続
31	LCD_DATA16	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続
32	LCD_DATA17	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続
33	GND	Power	電源(GND)
34	GPIO4_IO18	In/Out	拡張入出力、i.MX6ULL の CSI_PIXCLK ピンに接続
35	GPIO4_IO21	In/Out	拡張入出力、i.MX6ULL の CSI_DATA00 ピンに接続
36	GPIO4_IO24	In/Out	拡張入出力、i.MX6ULL の CSI_DATA03 ピンに接続
37	ADC_IN4	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続、0.01uF のコンデンサが接続されています。
38	ADC_IN3	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続、0.01uF のコンデンサが接続されています。
39	ADC_IN2	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続、0.01uF のコンデンサが接続されています。
40	ADC_IN1	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続、0.01uF のコンデンサが接続されています。

18.2.4.20. CON21(拡張インターフェース)

CON21 は主にオーディオ用の機能を割り当て可能な信号線を接続した、オーディオ拡張用インターフェースです。

CON7(Grove インターフェース)、CON11(LCD インターフェース)と共通の信号線が接続されているため、同時に使用できません。また、9、10 ピンの信号線は基板上のリアルタイムクロックにも接続されており、I2C で使用しておりますので、マルチプレクスの設定を変更する際には、ご注意ください。

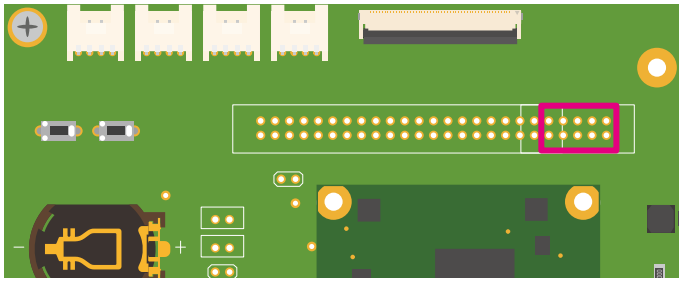


図 18.27 Armadillo-610 拡張ボード CON21

表 18.29 CON21 信号配列

ピン番号	ピン名	I/O	説明
1	+3.3V_IO	Power	電源出力(+3.3V_IO)
2	GND	Power	電源(GND)
3	SAI1_MCLK	In/Out	拡張入出力、i.MX6ULL の CSI_DATA01 ピンに接続
4	SAI1_RX_SYNC	In/Out	拡張入出力、i.MX6ULL の CSI_DATA02 ピンに接続
5	SAI1_TX_DATA	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
6	SAI1_RX_DATA	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
7	SAI1_TX_BCLK	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
8	SAI1_TX_SYNC	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
9	I2C2_SCL	In/Out	拡張入出力、i.MX6ULL の CSI_HSYNC ピンに接続

ピン番号	ピン名	I/O	説明
10	I2C2_SDA	In/Out	拡張入出力、i.MX6ULL の CSI_VSYNC ピンに接続

18.2.4.21. CON22(拡張インターフェース)

CON22 は SPI、I2C、UART 等に割り当て可能な信号線を接続した、拡張用インターフェースです。

CON13B(DIDO インターフェース)、CON13C(RS485 インターフェース)と共通の信号線が接続されているため、同時に使用できません。R93～R96、R105、R108、R110、R112(CON13 周辺)の抵抗を未実装にすることで、CON13 と切り離すことが可能です。

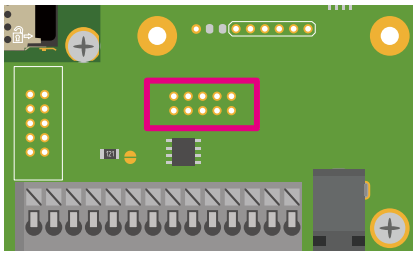


図 18.28 Armadillo-610 拡張ボード CON22

表 18.30 CON22 信号配列

ピン番号	ピン名	I/O	説明
1	+3.3V_IO	Power	電源出力(+3.3V_IO)
2	GND	Power	電源(GND)
3	RS485_TX	In/Out	拡張入出力、i.MX6ULL の NAND_DATA04 ピンに接続
4	RS485_RX	In/Out	拡張入出力、i.MX6ULL の NAND_DATA05 ピンに接続
5	RS485_RE_N	In/Out	拡張入出力、i.MX6ULL の NAND_DATA06 ピンに接続
6	RS485_DE	In/Out	拡張入出力、i.MX6ULL の NAND_DATA07 ピンに接続
7	DI1	In/Out	拡張入出力、i.MX6ULL の UART2_TX_DATA ピンに接続
8	DI2	In/Out	拡張入出力、i.MX6ULL の UART2_RX_DATA ピンに接続
9	DO1	In/Out	拡張入出力、i.MX6ULL の UART5_TX_DATA ピンに接続
10	DO2	In/Out	拡張入出力、i.MX6ULL の UART5_RX_DATA ピンに接続

18.2.4.22. CON23(リセットインターフェース)

CON23 はリセット入出力用のインターフェースです。EXT_RESET_N ピンは Armadillo-610 上のパワーマネジメント IC の RESETBMCU ピン、i.MX6ULL の POR_B ピンに接続されています。

リセット回路の詳細につきましては、「図 15.3. リセット回路の構成」をご参照ください。

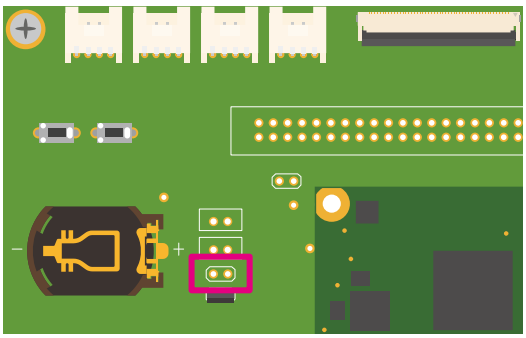


図 18.29 Armadillo-610 拡張ボード CON23

表 18.31 CON23 信号配列

ピン番号	ピン名	I/O	説明
1	EXT_RESET_N	In/Out	リセット信号、パワーマネジメント IC の RESETBMCU ピン、i.MX6ULL の POR_B ピンに接続
2	GND	Power	電源(GND)

18.2.4.23. CON24(電源入力インターフェース)

CON24 は電源供給用のインターフェースです。電源回路の詳細につきましては、「図 18.3. Armadillo-610 拡張ボードの電源回路の構成」をご参照ください。

出荷時、CON24(電源入力インターフェース)から電源供給することはできません。R155 を実装、R156 未実装にすることで、CON24 の 1 ピンから Armadillo-610 の電源(VIN)に電源を供給することが可能になります。回路図をご確認の上、絶対定格値を超えない範囲でご使用ください。Armadillo-610 の入力電圧範囲(VIN)は 3.6～4.5V です。

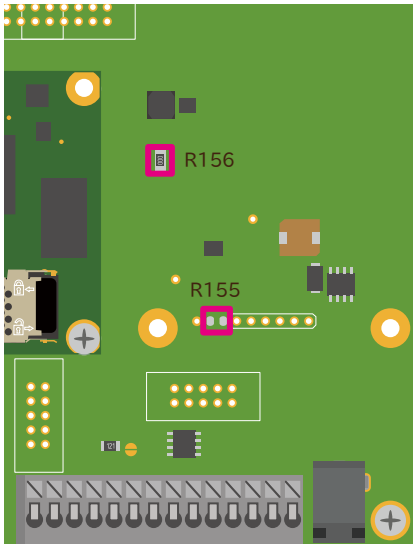


図 18.30 Armadillo-610 拡張ボード CON24

表 18.32 CON24 信号配列

ピン番号	ピン名	I/O	説明
1	BAT_IN	Power	電源(BAT_IN)、R155(出荷時未実装)を経由してパワーマネジメント IC の VIN ピンに接続
2	GND	Power	電源(GND)
3	+5V	Power	電源(+5V)
4	GND	Power	電源(GND)
5	NC	-	未接続
6	NC	-	未接続

18.2.4.24. JP1(起動デバイス設定ジャンパ)

JP1 は起動デバイス設定ジャンパです。JP1 の状態で、起動デバイスを設定することができます。

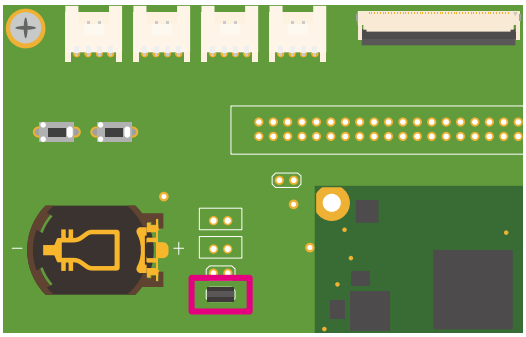


図 18.31 Armadillo-610 拡張ボード JP1

表 18.33 JP1 信号配列

部品番号	説明
JP1	ロジック IC を経由して、i.MX6ULL の LCD_DATA05 ピン、LCD_DATA_DATA11 ピンに接続

表 18.34 ジャンパの設定と起動デバイス

JP1	起動デバイス
オープン	eMMC
ショート	microSD

eFUSE で起動デバイスを設定している場合、JP1 の設定は無視されます。JP1 をショート状態にすると、プルアップ抵抗により消費電流が増加するため、JP1 はオープン状態で使用することをお勧めします。

eFUSE は一度書き込むと元に戻すことができません。eFUSE の設定によっては Armadillo-610 が正常に動作しなくなる可能性がありますので、細心の注意を払って書き込みを行うようお願いいたします。eFUSE の設定によって異常が起こった場合は保証対象外となります。

18.2.4.25. SW1(ユーザースイッチ)

SW1 はユーザー側で自由に利用できる押しボタンスイッチです。

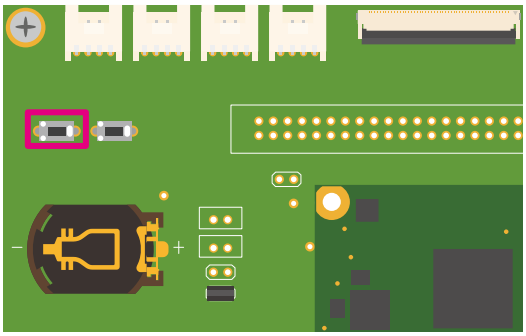


図 18.32 Armadillo-610 拡張ボード SW1

表 18.35 SW1 信号配列

部品番号	説明
SW1	i.MX6ULL の JTAG_MOD ピンに接続(押されていない状態: Low、押された状態: High)

18.2.4.26. SW2(リセットスイッチ)

SW2 はリセット用の押しボタンスイッチです。ボタンを押すとパワーマネジメント IC からの電源が切断されます。

動作の詳細については、「15.6.2. PWRON ピンからの電源制御」をご確認ください。

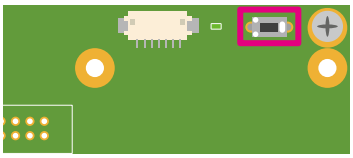


図 18.33 Armadillo-610 拡張ボード SW2

表 18.36 SW2 信号配列

部品番号	説明
SW2	パワーマネジメント IC の PWRON ピンと i.MX6ULL の PMIC_ON_REQ ピンに接続(押されていない状態: リセット解除、 押された状態: リセット状態)

18.2.4.27. SW3(ONOFF スイッチ)

SW3 はボタンの長押しで電源を制御する押しボタンスイッチです。

動作の詳細については、「15.6.1. ONOFF ピンからの電源制御」をご確認ください。

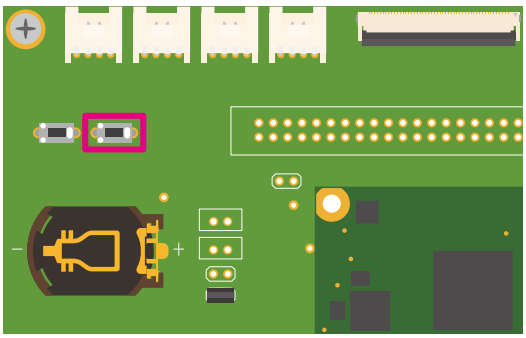


図 18.34 Armadillo-610 拡張ボード SW3

表 18.37 SW3 信号配列

部品番号	説明
SW3	i.MX6ULL の ONOFF ピンに接続

18.2.4.28. LED1、LED2(LAN LED)

LED1、LED2 は CON2(LAN インターフェース)のステータス LED です。CON2(LAN インターフェース)の上部に表示されます。信号線は Ethernet PHY(LAN8720AI-CP/Microchip Technology)の LED ピンに接続されます。

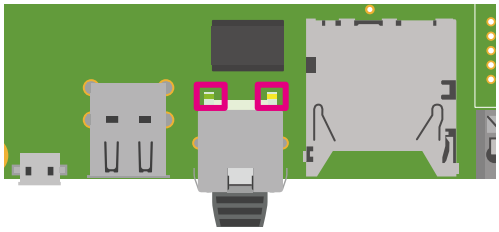


図 18.35 Armadillo-610 拡張ボード LED1、LED2

表 18.38 LAN LED の動作

LED	名称(色)	状態	説明
LED1	LAN スピード LED(緑)	消灯	10Mbps で接続されている、もしくは Ethernet ケーブル未接続
		点灯	100Mbps で接続されている
LED2	LAN リンクアクティビティ(黄)	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、データを送受信している

18.2.4.29. LED3(ユーザー LED)

LED3 ユーザー側で自由に利用できる LED です。

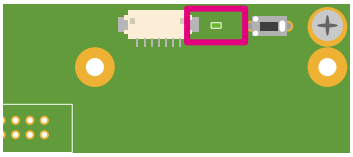


図 18.36 Armadillo-610 拡張ボード LED3

表 18.39 LED3

部品番号	名称(色)	説明
LED3	ユーザー LED(緑)	i.MX6ULL の GPIO1_IO08 ピンに接続、(Low: 消灯、High: 点灯)

18.2.5. 基板形状図

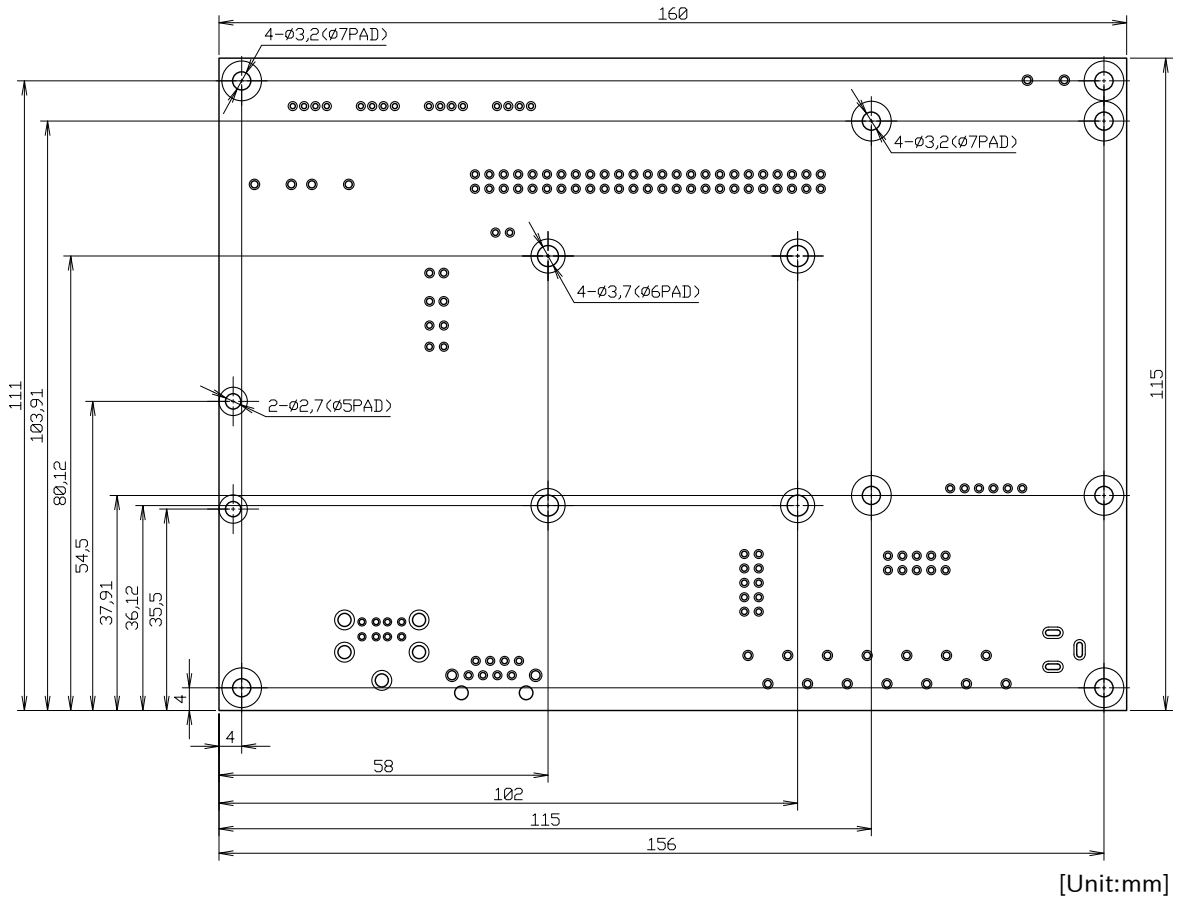


図 18.37 Armadillo-610 拡張ボード 基板形状および固定穴寸法

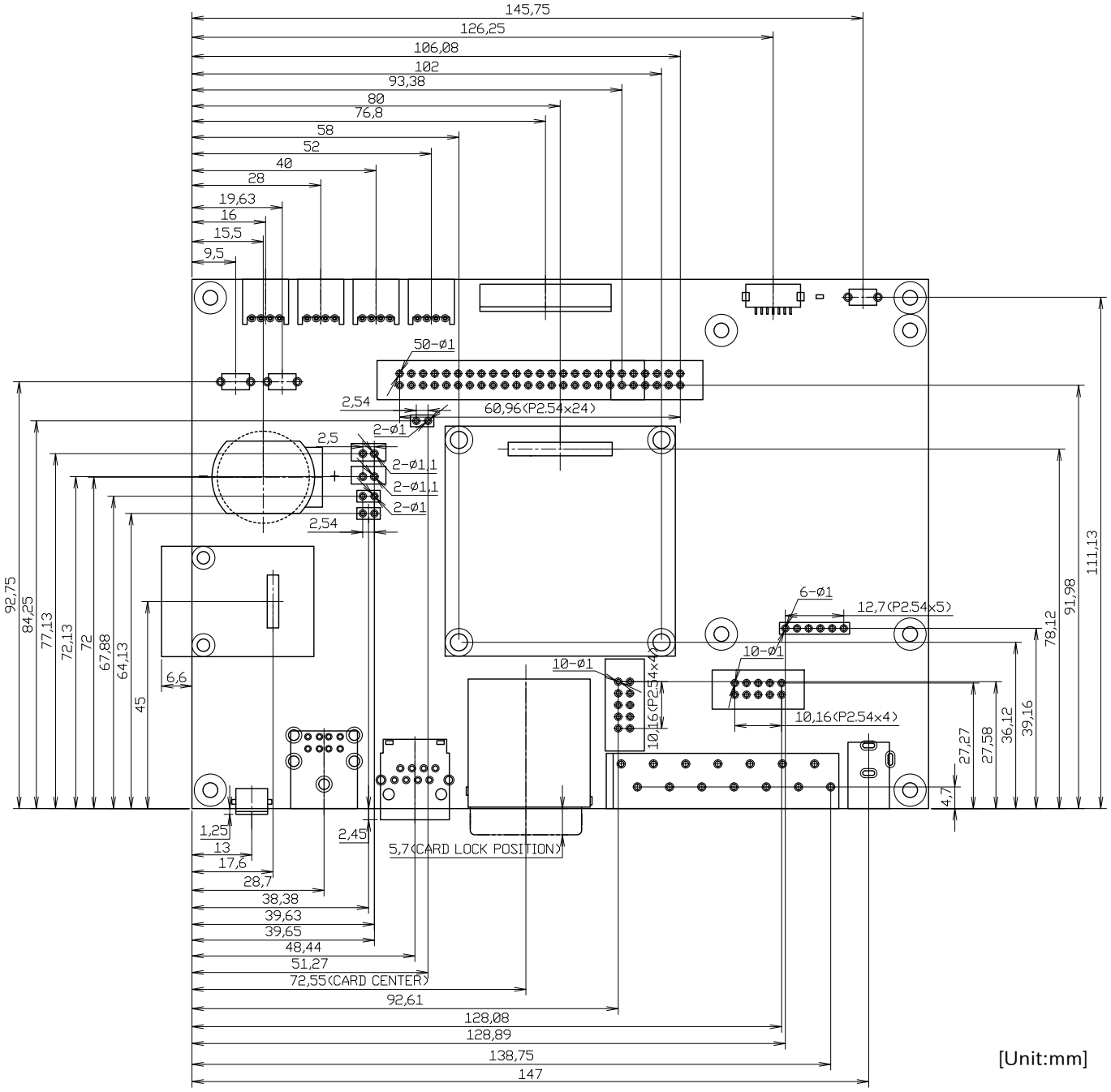


図 18.38 Armadillo-610 拡張ボード コネクタ中心寸法および穴寸法

18.3. LCD オプションセット(7 インチタッチパネル WVGA 液晶)

18.3.1. 概要

ノリタケ伊勢電子製のタッチパネル LCD とフレキシブルフラットケーブル(FFC)のセットです。LCD インターフェース(Armadillo-610 拡張ボード: CON11)に接続して使用することが可能です。

LCD オプションセット(7 インチタッチパネル WVGA 液晶)を使用する場合、Armadillo-610 拡張ボード搭載のリアルタイムクロックが使用できません。

表 18.40 LCD オプションセット(7 インチタッチパネル WVGA 液晶)について

商品名	LCD オプションセット(7 インチタッチパネル WVGA 液晶)
型番	OP-LCD70EXT-L00
内容	7インチ タッチパネル LCD、FFC

表 18.41 LCD の仕様

型番	GT800X480A-1013P
メーカー	ノリタケ伊勢電子
タイプ	TFT-LCD
表示サイズ	7 インチ
外形サイズ	164.8 x 99.8 mm
解像度	800 x 480 pixels
表示色数	約 1677 万色
使用温度範囲	-20～+70℃
輝度	850cd/m^2 (Typ.) 25℃
電源	DC 5V±5%/500mA (Typ.), DC 3.3V±3%/35mA (Typ.)
映像入力インターフェース	RGB パラレル(18bit/24bit)[a]
タッチパネルインターフェース	I2C(HID 準拠)
タッチ方式	投影型静電容量方式
マルチタッチ	最大 10 点対応

[a]LCD インターフェース(Armadillo-610 拡張ボード: CON11)は 18bit 対応です。

タッチパネル LCD をご使用になる前に、『GT800X480A-1013P 製品仕様書』にて注意事項、詳細仕様、取扱方法等をご確認ください。

『GT800X480A-1013P 製品仕様書』は「アットマークテクノ Armadillo サイト」の「[オプション] LCD オプションセット (7 インチタッチパネル WVGA 液晶) 製品仕様書」からダウンロード可能です。

18.3.2. 組み立て

「図 18.39. LCD の接続方法」を参考にし、タッチパネル LCD の CN4 の 1 ピンと Armadillo-610 拡張ボードの CON11 の 1 ピンが対応するように、FFC を接続します。

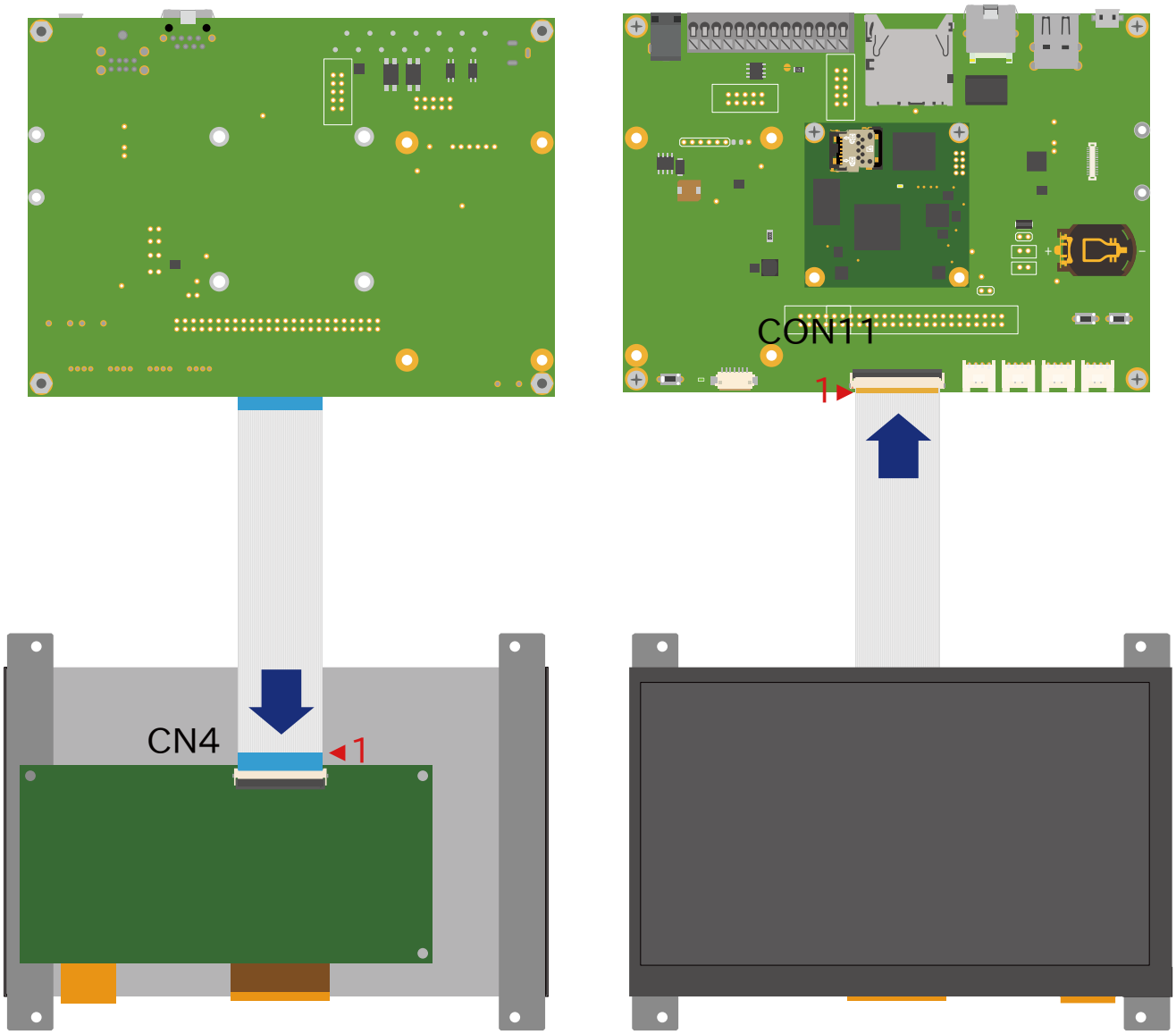


図 18.39 LCD の接続方法

図 18.40 フレキシブルフラットケーブルの形状

必ず 1 ピンと 1 ピンが対応するように、接続してください。1 ピンと 50 ピンが対応するように接続した場合、電源と GND がショートし、故障の原因となります。

18.4. Armadillo-400 シリーズ LCD オプションセット

18.4.1. 概要

4.3 インチタッチパネル LCD、LCD 拡張ボード、フレキシブルフラットケーブル(FFC)のセットです。

LCD 拡張ボードには LCD 接続インターフェースの他にオーディオコーデック、リアルタイムクロック(RTC)を搭載しています。LCD インターフェース(Armadillo-610 拡張ボード: CON11)に接続して使用することが可能です。

LCD 拡張ボードの回路図、部品表は「アットマークテクノ Armadillo サイト」からダウンロード可能です。

本製品は Armadillo-400 シリーズ用に準備された LCD オプションセットですが、Armadillo-600 シリーズでも利用可能です。

表 18.42 Armadillo-400 シリーズ LCD オプションセットについて

商品名	Armadillo-400 シリーズ LCD オプションセット
型番	OP-A400-LCD43EXT-L01
内容	4.3 インチ タッチパネル LCD、FFC、LCD 固定用両面テープ

表 18.43 LCD 拡張ボードの仕様

LCD	型番	FG040346DSSWBG04 [a]
	メーカー	Data Image
LCD バックライト	バックライト用 LED ドライバ搭載	
オーディオ	型番	WM8978GEFL/V
	メーカー	Wolfson
リアルタイムクロック	型番	S-35390A
	メーカー	ABLIC
RTC バックアップ	300 秒(Typ.)、60 秒(Min.)、電池ホルダ搭載、外部バッテリ接続コネクタ搭載可能(対応電池: CR2032 等)	
RTC 平均月差(参考値)	約 30 秒@25℃	
電源電圧	DC 3.3±0.2V(メイン電源)、DC 2.8～5.5V(LCD バックライト)、DC 2.0～3.5V(RTC バックアップ)	
消費電力	約 0.8W(LCD 消費分を含む)	
使用温度範囲	-20～+70℃(結露なきこと)	

外形サイズ	106 x 82 mm(突起部を除く)

[a]LCD 拡張インターフェース(汎用)により、その他の LCD も接続可能です。

RTC の時間精度や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。ご使用の際には十分に特性の確認をお願いします。

表 18.44 LCD の仕様

型番	FG040346DSSWBG04
メーカー	Data Image
タイプ	TFT-LCD
表示サイズ	4.3 インチ
外形サイズ	105.5(W) x 67.2(H) x 4.3(D) mm
解像度	480 x 272 pixels
表示色数	約 1677 万色
使用温度範囲	-20〜+70℃
輝度	320cd/m^2 (Typ.)
映像入力インターフェース	RGB パラレル(24bit)[a]
タッチ方式	4 線抵抗膜方式

[a]Armadillo-610 拡張ボードの CON11 は 18bit 対応です。

LCD オプションセットは製品リビジョンによって一部仕様が違いますのでご注意ください。本章には、「製品リビジョン F」以降についての情報を記載しています。製品リビジョンにつきましては、「アットマークテクノ Armadillo サイト」[https://armadillo.atmark-techno.com/]からダウンロードできる「Armadillo-400 シリーズ LCD オプションセット変更履歴表」にてご確認ください。

18.4.2. ブロック図

LCD 拡張ボードのブロック図は次のとおりです。

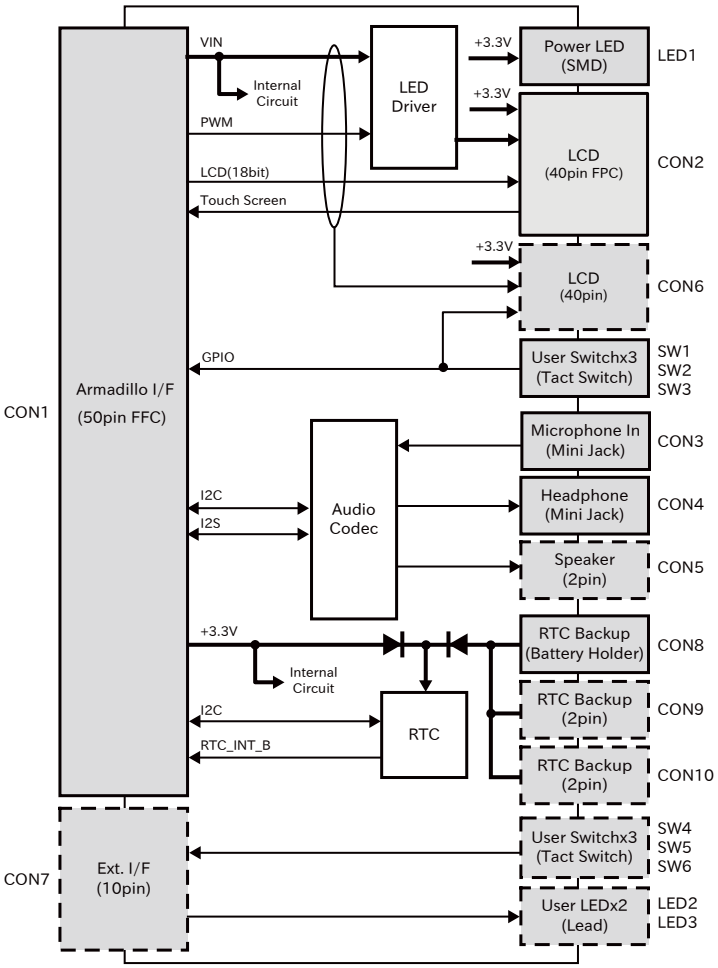


図 18.41 LCD 拡張ボードのブロック図

18.4.3. インターフェース仕様

LCD 拡張ボードのインターフェース仕様について説明します。

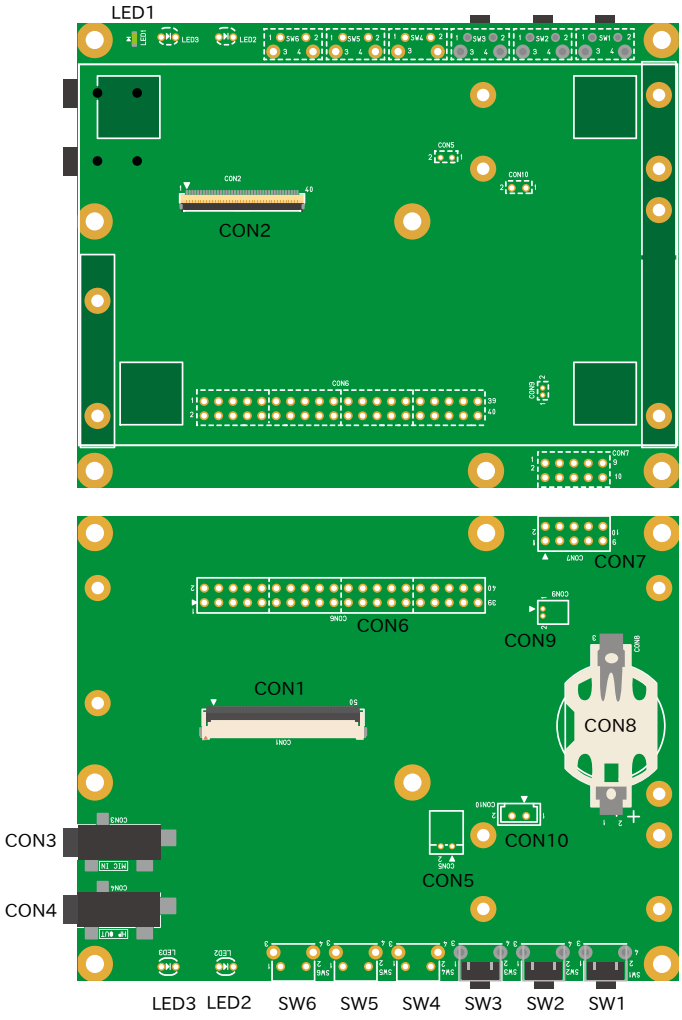


図 18.42 LCD 拡張ボードのインターフェース

表 18.45 LCD 拡張ボード インターフェース一覧 [a]

部品番号	インターフェース名	型番	メーカー
CON1	Armadillo 接続インターフェース	XF2M-5015-1A	OMRON
CON2	LCD 拡張インターフェース(専用)	FH19SC-40S-0.5SH(05)	HIROSE ELECTRIC
CON3	モノラルマイク入力	SJ-3524-SMT	CUI Inc
CON4	ステレオヘッドホン出力	SJ-3524-SMT	CUI Inc
CON5	スピーカー出力	S2B-PH-K-S(LF)(SN)	J.S.T.Mfg.
CON6	LCD 拡張インターフェース(汎用)	RF-H402TD-1130	J.S.T.Mfg.
CON7	拡張インターフェース	RF-H102TD-1130	J.S.T.Mfg.
CON8	RTC バックアップインターフェース	HU2032	TAKACHI
CON9		DF13-2P-1.25DS(20)	HIROSE ELECTRIC
CON10		B2B-EH(LF)(SN)	J.S.T.Mfg.
SW1	ユーザースイッチ	SKHHLMA010	ALPS
SW2		SKHHLMA010	ALPS
SW3		SKHHLMA010	ALPS
SW4		SKHHLMA010	ALPS
SW5		SKHHLMA010	ALPS
SW6		SKHHLMA010	ALPS
LED1	電源 LED	SML-310MT	ROHM

部品番号	インターフェース名	型番	メーカー
LED2	ユーザー LED	OSNG3133A	OPTO SUPPLY
LED3		OSNG3133A	OPTO SUPPLY

[a]部品の実装、未実装を問わず、搭載可能な部品型番を記載しています。

18.4.3.1. CON1 (Armadillo 接続インターフェース)

CON1 は Armadillo と接続するためのインターフェースです。50 ピン(0.5mm ピッチ)のフレキシブルフラットケーブルにより、LCD インターフェース(Armadillo-610 拡張ボード: CON11)と接続可能です。

LCD、タッチパネル、オーディオコーデック、リアルタイムクロックの各信号線が接続されています。

表 18.46 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	I2C3_SDA	In/Out	コーデック I2C データ、コーデックの SDIN ピンに接続、1kΩ プルアップ(+3.3V)
3	I2C3_SCL	In	コーデック I2C クロック、コーデック IC の SCLK ピンに接続、1kΩ プルアップ(+3.3V)
4	AUD5_TXFS	In	コーデック TXFS、コーデックの LRC ピンに接続
5	AUD5_TXC	In	コーデック TXC、コーデックの BCLK ピンに接続
6	AUD5_RXD	Out	コーデック RXD、コーデックの ADCDAT ピンに接続
7	AUD5_TXD	In	コーデック TXD、コーデックの DACDAT ピンに接続
8	AUD_RXFS	In/Out	R50 の実装によりリアルタイムクロックの割り込みピンに接続
9	AUD5_SYSCLK	In	コーデック SYSCLK、コーデックの MCLK ピンに接続
10	GPIO2_30	In/Out	タクトスイッチ出力、SW3 の 2 ピン、CON6 の 7 ピンに接続
11	GPIO2_29	In/Out	タクトスイッチ出力、SW2 の 2 ピン、CON6 の 8 ピンに接続
12	GPIO2_20	In/Out	タクトスイッチ出力、SW1 の 2 ピン、CON6 の 9 ピンに接続
13	GND	Power	電源(GND)
14	TOUCH_YN	In/Out	タッチパネル YN、CON2 の 40 ピン、CON6 の 10 ピンに接続
15	TOUCH_YP	In/Out	タッチパネル YP、CON2 の 38 ピン、CON6 の 11 ピンに接続
16	TOUCH_XN	In/Out	タッチパネル XN、CON2 の 39 ピン、CON6 の 12 ピンに接続
17	TOUCH_XP	In/Out	タッチパネル XP、CON2 の 37 ピン、CON6 の 13 ピンに接続
18	GND	Power	電源(GND)
19	LCD_LD17	In	LCD 拡張 I/F LD17、CON2 の 12 ピン、CON6 の 15 ピンに接続
20	LCD_LD16	In	LCD 拡張 I/F LD16、CON2 の 11 ピン、CON6 の 16 ピンに接続
21	LCD_LD15	In	LCD 拡張 I/F LD15、CON2 の 10 ピン、CON6 の 17 ピンに接続
22	LCD_LD14	In	LCD 拡張 I/F LD14、CON2 の 9 ピン、CON6 の 18 ピンに接続
23	LCD_LD13	In	LCD 拡張 I/F LD13、CON2 の 8 ピン、CON6 の 19 ピンに接続
24	LCD_LD12	In	LCD 拡張 I/F LD12、CON2 の 7 ピン、CON6 の 20 ピンに接続
25	GND	Power	電源(GND)
26	LCD_LD11	In	LCD 拡張 I/F LD11、CON2 の 20 ピン、CON6 の 22 ピンに接続
27	LCD_LD10	In	LCD 拡張 I/F LD10、CON2 の 19 ピン、CON6 の 23 ピンに接続
28	LCD_LD9	In	LCD 拡張 I/F LD9、CON2 の 18 ピン、CON6 の 24 ピンに接続
29	LCD_LD8	In	LCD 拡張 I/F LD8、CON2 の 17 ピン、CON6 の 25 ピンに接続
30	LCD_LD7	In	LCD 拡張 I/F LD7、CON2 の 16 ピン、CON6 の 26 ピンに接続
31	LCD_LD6	In	LCD 拡張 I/F LD6、CON2 の 15 ピン、CON6 の 27 ピンに接続
32	GND	Power	電源(GND)
33	LCD_LD5	In	LCD 拡張 I/F LD4、CON2 の 28 ピン、CON6 の 29 ピンに接続
34	LCD_LD4	In	LCD 拡張 I/F LD4、CON2 の 27 ピン、CON6 の 30 ピンに接続
35	LCD_LD3	In	LCD 拡張 I/F LD3、CON2 の 26 ピン、CON6 の 31 ピンに接続
36	LCD_LD2	In	LCD 拡張 I/F LD2、CON2 の 25 ピン、CON6 の 32 ピンに接続
37	LCD_LD1	In	LCD 拡張 I/F LD1、CON2 の 24 ピン、CON6 の 33 ピンに接続
38	LCD_LD0	In	LCD 拡張 I/F LD0、CON2 の 23 ピン、CON6 の 34 ピンに接続

ピン番号	ピン名	I/O	説明
39	PWMO1	In	LCD 拡張 I/F PWMO1、CON6 の 36 ピン、LED ドライバの FB ピンに接続
40	LCD_OE_ACD	In	LCD 拡張 I/F OE_ACD、CON2 の 34 ピン、CON6 の 37 ピンに接続
41	LCD_VSYN	In	LCD 拡張 I/F VSYN、CON2 の 33 ピン、CON6 の 38 ピンに接続
42	LCD_HSYN	In	LCD 拡張 I/F HSYN、CON2 の 32 ピン、CON6 の 39 ピンに接続
43	LCD_LSCLK	In	LCD 拡張 I/F LSCLK、CON2 の 30 ピン、CON6 の 40 ピンに接続
44	GND	Power	電源(GND)
45	GND	Power	電源(GND)
46	+3.3V	Power	電源(+3.3V)
47	+3.3V	Power	電源(+3.3V)
48	VIN	Power	電源(VIN)
49	VIN	Power	電源(VIN)
50	VIN	Powe	電源(VIN)

18.4.3.2. CON2(LCD 拡張インターフェース)

CON2 は LCD(FG040346DSSWBG04/Data Image)と接続するためのインターフェースです。

CON2 と CON6 には、共通の信号が接続されていますので同時に使用できません。CON6 を使用する場合は、CON2 から LCD を取り外してください。

表 18.47 CON2 信号配列

ピン番号	ピン名	I/O	説明
1	BL_LED_K	Power	LED ドライバ電源(ー端子)
2	BL_LED_A	Power	LED ドライバ電源(＋端子)
3	GND	Power	電源(GND)
4	+3.3V	Power	電源(+3.3V)
5	GND	Power	電源(GND)
6	GND	Power	電源(GND)
7	LCD_LD12	Out	LCD 拡張 I/F LD12、CON1 の 24 ピン、CON6 の 20 ピンに接続
8	LCD_LD13	Out	LCD 拡張 I/F LD13、CON1 の 23 ピン、CON6 の 19 ピンに接続
9	LCD_LD14	Out	LCD 拡張 I/F LD14、CON1 の 22 ピン、CON6 の 18 ピンに接続
10	LCD_LD15	Out	LCD 拡張 I/F LD15、CON1 の 21 ピン、CON6 の 17 ピンに接続
11	LCD_LD16	Out	LCD 拡張 I/F LD16、CON1 の 20 ピン、CON6 の 16 ピンに接続
12	LCD_LD17	Out	LCD 拡張 I/F LD17、CON1 の 19 ピン、CON6 の 15 ピンに接続
13	GND	Power	電源(GND)
14	GND	Power	電源(GND)
15	LCD_LD6	Out	LCD 拡張 I/F LD6、CON1 の 31 ピン、CON6 の 27 ピンに接続
16	LCD_LD7	Out	LCD 拡張 I/F LD7、CON1 の 30 ピン、CON6 の 26 ピンに接続
17	LCD_LD8	Out	LCD 拡張 I/F LD8、CON1 の 29 ピン、CON6 の 25 ピンに接続
18	LCD_LD9	Out	LCD 拡張 I/F LD9、CON1 の 28 ピン、CON6 の 24 ピンに接続
19	LCD_LD10	Out	LCD 拡張 I/F LD10、CON1 の 27 ピン、CON6 の 23 ピンに接続
20	LCD_LD11	Out	LCD 拡張 I/F LD11、CON1 の 26 ピン、CON6 の 22 ピンに接続
21	GND	Power	電源(GND)
22	GND	Power	電源(GND)
23	LCD_LD0	Out	LCD 拡張 I/F LD0、CON1 の 38 ピン、CON6 の 34 ピンに接続
24	LCD_LD1	Out	LCD 拡張 I/F LD1、CON1 の 37 ピン、CON6 の 33 ピンに接続
25	LCD_LD2	Out	LCD 拡張 I/F LD2、CON1 の 36 ピン、CON6 の 32 ピンに接続
26	LCD_LD3	Out	LCD 拡張 I/F LD3、CON1 の 35 ピン、CON6 の 31 ピンに接続
27	LCD_LD4	Out	LCD 拡張 I/F LD4、CON1 の 34 ピン、CON6 の 30 ピンに接続

ピン番号	ピン名	I/O	説明
28	LCD_LD5	Out	LCD 拡張 I/F LD5、CON1 の 33 ピン、CON6 の 29 ピンに接続
29	GND	Power	電源(GND)
30	DISP	In/Out	10kΩ プルアップ(+3.3V)
31	LCD_LSCLK	Out	LCD 拡張 I/F LSCLK、CON1 の 43 ピン、CON6 の 40 ピンに接続
32	LCD_HSYN	Out	LCD 拡張 I/F HSYN、CON1 の 42 ピン、CON6 の 39 ピンに接続
33	LCD_VSYN	Out	LCD 拡張 I/F VSYN、CON1 の 41 ピン、CON6 の 38 ピンに接続
34	LCD_OE_ACD	Out	LCD 拡張 I/F OE_ACD、CON1 の 40 ピン、CON6 の 37 ピンに接続
35	NC	—	—
36	GND	Power	電源(GND)
37	TOUCH_XP	In/Out	タッチパネル XP、 CON1 の 17 ピン、CON6 の 13 ピンに接続
38	TOUCH_YP	In/Out	タッチパネル YP、 CON1 の 15 ピン、CON6 の 11 ピンに接続
39	TOUCH_XN	In/Out	タッチパネル XN、 CON1 の 16 ピン、CON6 の 12 ピンに接続
40	TOUCH_YN	In/Out	タッチパネル YN、 CON1 の 14 ピン、CON6 の 10 ピンに接続

18.4.3.3. CON3(モノラルマイク入力)

CON3 はモノラルマイク入力です。

表 18.48 CON3 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	MIC_IN	In	コーデックの LIP ピンに接続
3	—	—	—
10	—	—	—

18.4.3.4. CON4(ステレオヘッドホン出力)

CON4 はステレオヘッドホン出力です。

表 18.49 CON4 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	HP_L_OUT	Out	コーデックの LOUT1 ピンに接続
3	HP_R_OUT	Out	コーデックの ROUT1 ピンに接続
10	HP_DET	In	コーデックの L2/GPIO2 ピンに接続

18.4.3.5. CON5(スピーカー出力)

CON5 はスピーカー出力です。部品は実装されていません。

表 18.50 CON5 信号配列

ピン番号	ピン名	I/O	説明
1	SPK_N	Out	コーデックの LOUT2 ピンに接続
2	SPK_P	Out	コーデックの ROUT2 ピンに接続

18.4.3.6. CON6(LCD 拡張インターフェース)

CON6 は LCD と接続するためのインターフェースです。部品は実装されていません。

CON2 と CON6 には、共通の信号が接続されていますので同時に使用できません。CON6 を使用する場合は、CON2 から LCD を取り外してください。

表 18.51 CON6 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源(VIN)
2	VIN	Power	電源(VIN)
3	+3.3V	Power	電源(+3.3V)
4	+3.3V	Power	電源(+3.3V)
5	GND	Power	電源(GND)
6	GND	Power	電源(GND)
7	GPIO2_30	In/Out	タクトスイッチ出力、SW3 の 2 ピン、CON1 の 10 ピンに接続
8	GPIO2_29	In/Out	タクトスイッチ出力、SW2 の 2 ピン、CON1 の 11 ピンに接続
9	GPIO2_20	In/Out	タクトスイッチ出力、SW1 の 2 ピン、CON1 の 12 ピンに接続
10	TOUCH_YN	In/Out	タッチパネル YN、 CON1 の 14 ピン、CON1 の 40 ピンに接続
11	TOUCH_YP	In/Out	タッチパネル YP、 CON1 の 15 ピン、CON1 の 38 ピンに接続
12	TOUCH_XN	In/Out	タッチパネル XN、 CON1 の 16 ピン、CON1 の 39 ピンに接続
13	TOUCH_XP	In/Out	タッチパネル XP、 CON1 の 17 ピン、CON1 の 37 ピンに接続
14	GND	Power	電源(GND)
15	LCD_LD17	Out	LCD 拡張 I/F LD17、CON2 の 12 ピン、CON1 の 19 ピンに接続
16	LCD_LD16	Out	LCD 拡張 I/F LD16、CON2 の 11 ピン、CON1 の 20 ピンに接続
17	LCD_LD15	Out	LCD 拡張 I/F LD15、CON2 の 10 ピン、CON1 の 21 ピンに接続
18	LCD_LD14	Out	LCD 拡張 I/F LD14、CON2 の 9 ピン、CON1 の 22 ピンに接続
19	LCD_LD13	Out	LCD 拡張 I/F LD13、CON2 の 8 ピン、CON1 の 23 ピンに接続
20	LCD_LD12	Out	LCD 拡張 I/F LD12、CON2 の 7 ピン、CON1 の 24 ピンに接続
21	GND	Power	電源(GND)
22	LCD_LD11	Out	LCD 拡張 I/F LD11、CON2 の 20 ピン、CON1 の 26 ピンに接続
23	LCD_LD10	Out	LCD 拡張 I/F LD10、CON2 の 19 ピン、CON1 の 27 ピンに接続
24	LCD_LD9	Out	LCD 拡張 I/F LD9、CON2 の 18 ピン、CON1 の 28 ピンに接続
25	LCD_LD8	Out	LCD 拡張 I/F LD8、CON2 の 17 ピン、CON1 の 29 ピンに接続
26	LCD_LD7	Out	LCD 拡張 I/F LD7、CON2 の 16 ピン、CON1 の 30 ピンに接続
27	LCD_LD6	Out	LCD 拡張 I/F LD6、CON2 の 15 ピン、CON1 の 31 ピンに接続
28	GND	Power	電源(GND)
29	LCD_LD5	Out	LCD 拡張 I/F LD5、CON2 の 28 ピン、CON1 の 33 ピンに接続
30	LCD_LD4	Out	LCD 拡張 I/F LD4、CON2 の 27 ピン、CON1 の 34 ピンに接続
31	LCD_LD3	Out	LCD 拡張 I/F LD3、CON2 の 26 ピン、CON1 の 35 ピンに接続
32	LCD_LD2	Out	LCD 拡張 I/F LD2、CON2 の 25 ピン、CON1 の 36 ピンに接続
33	LCD_LD1	Out	LCD 拡張 I/F LD1、CON2 の 24 ピン、CON1 の 37 ピンに接続
34	LCD_LD0	Out	LCD 拡張 I/F LD0、CON2 の 23 ピン、CON1 の 38 ピンに接続
35	GND	Power	電源(GND)
36	PWMO1	Out	LCD 拡張 I/F PWMO1、CON1 の 39 ピン、LED ドライバの FB ピンに接続
37	LCD_OE_ACD	Out	LCD 拡張 I/F OE_ACD、CON1 の 40 ピン、CON1 の 34 ピンに接続
38	LCD_VSYN	Out	LCD 拡張 I/F VSYN、CON1 の 41 ピン、CON1 の 33 ピンに接続
39	LCD_HSYN	Out	LCD 拡張 I/F HSYN、CON1 の 42 ピン、CON1 の 32 ピンに接続
40	LCD_LSCLK	Out	LCD 拡張 I/F LSCLK、CON1 の 43 ピン、CON1 の 31 ピンに接続

18.4.3.7. CON7(拡張インターフェース)

CON7 はユーザー側で自由に利用できるスイッチ、LED 拡張用のインターフェースです。部品は実装されていません。

表 18.52 CON7 信号配列

ピン番号	ピン名	I/O	説明
1	LED2	In	LED2 に接続 (Low:消灯、 High:点灯)
2	NC	ー	ー
3	LED3	In	LED3 に接続 (Low:消灯、 High:点灯)

ピン番号	ピン名	I/O	説明
4	NC	—	—
5	SW4	In/Out	SW4 の 2 ピンに接続
6	NC	—	—
7	SW5	In/Out	SW5 の 2 ピンに接続
8	NC	—	—
9	SW6	In/Out	SW6 の 2 ピンに接続
10	NC	—	—

18.4.3.8. CON8、CON9、CON10(RTC バックアップインターフェース)

CON8、CON9、CON10 は RTC のバックアップ電源供給用のインターフェースです。電源(+3.3V)が切断されても長期間時刻データを保持させたい場合に、別途バックアップ用のバッテリを接続することで、時刻データを保持することが可能です。3 つの形状のインターフェースがありますので、お使いのバッテリに合わせてご使用ください。CON9、CON10 に部品は実装されていません。

表 18.53 CON8 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	電源(RTC_BAT)、リアルタイムクロックの電源ピンに接続
2	RTC_BAT	Power	電源(RTC_BAT)、リアルタイムクロックの電源ピンに接続
3	GND	Power	電源(GND)

CON8、CON9、CON10 は共通の端子に接続されており、同時に使用することはできません。

CON8、CON9、CON10 はリチウムコイン電池からの電源供給を想定したインターフェースです。リチウムコイン電池以外から電源を供給する場合、回路図、部品表にて搭載部品をご確認の上、絶対定格値を超えない範囲でご使用ください。

表 18.54 CON9、CON10 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	電源(RTC_BAT)、リアルタイムクロックの電源ピンに接続
2	GND	Power	電源(GND)

18.4.3.9. SW1、SW2、SW3、SW4、SW5、SW6(ユーザースイッチ)

SW1、SW2、SW3、SW4、SW5、SW6 はユーザー側で自由に利用できるスイッチです。SW4、SW5、SW6 に部品は実装されていません。

表 18.55 SW1、SW2、SW3、SW4、SW5、SW6 の機能

部品番号	説明
SW1	CON1 の 12 ピン、CON6 の 9 ピンに接続 (Low:押された状態、 Open:押されていない状態)
SW2	CON1 の 11 ピン、CON6 の 8 ピンに接続 (Low:押された状態、 Open:押されていない状態)
SW3	CON1 の 10 ピン、CON6 の 7 ピンに接続 (Low:押された状態、 Open:押されていない状態)
SW4	CON7 の 5 ピンに接続 (Low:押された状態、 Open:押されていない状態)

部品番号	説明
SW5	CON7 の 7 ピンに接続 (Low:押された状態、 Open:押されていない状態)
SW6	CON7 の 9 ピンに接続 (Low:押された状態、 Open:押されていない状態)

18.4.3.10. LED1(電源 LED)

LED1 は電源 LED です。

表 18.56 LED1 の機能

部品番号	名称(色)	説明
LED1	電源 LED(緑)	電源(+3.3V)ON: 点灯、電源(+3.3V)OFF: 消灯

18.4.3.11. LED2、LED3 ユーザー(LED)

LED2、LED3 はユーザー側で自由に利用できる LED です。部品は実装されていません。

表 18.57 LED2、LED3 の機能

部品番号	名称(色)	説明
LED2	ユーザー LED(一)	CON7 の 1 ピンと接続(High: 点灯、Low: 消灯)
LED3		CON7 の 3 ピンと接続(High: 点灯、Low: 消灯)

18.4.4. 組み立て

タッチパネル LCD と LCD 拡張ボードは「図 18.43. タッチパネル LCD と LCD 拡張ボードの接続」のように接続します。

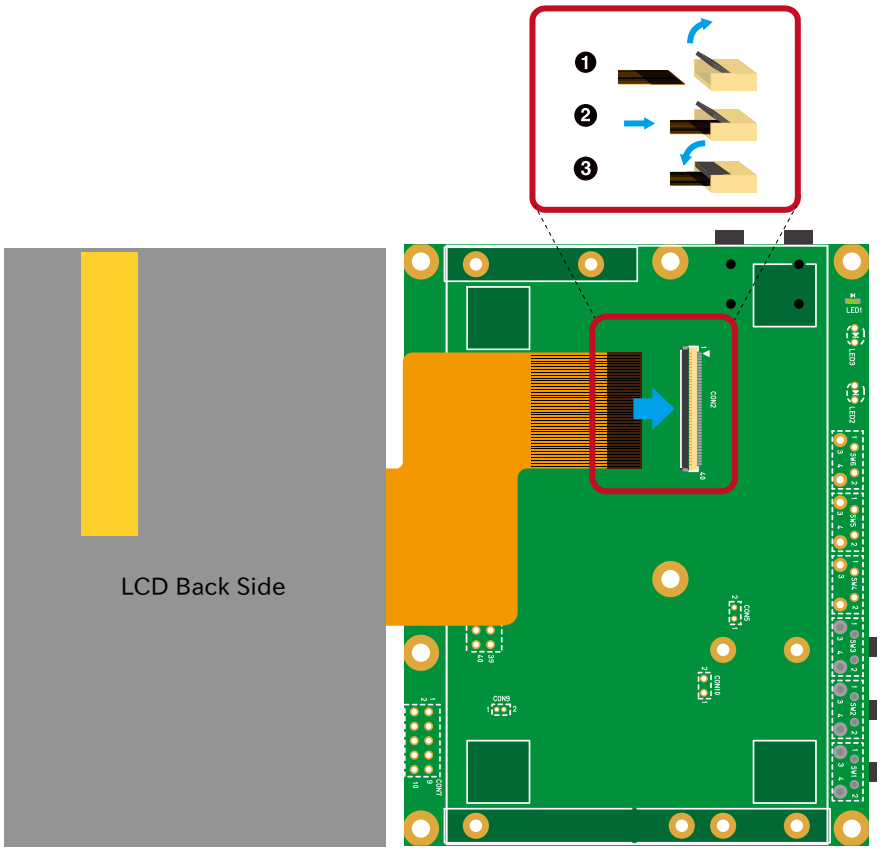


図 18.43 タッチパネル LCD と LCD 拡張ボードの接続

❶ LCD 拡張ボード CON2 のロックレバーを上げる

❷ タッチパネル LCD のフレキシブル基板(FPC)が止まるところまで挿入

❸ LCD 拡張ボード CON2 のロックレバーを下げる

LCD 拡張ボード CON2 のロックレバーは指でつまむ等の操作は避け、親指や人指し指の爪により、跳ね上げる感じで操作してください。強い力を加えると、コネクタが破損する恐れがあります。

タッチパネル LCD を LCD 拡張ボードに両面テープで固定する場合、「図 18.44. 両面テープの貼付位置」の位置に貼付するのがおすすめです。

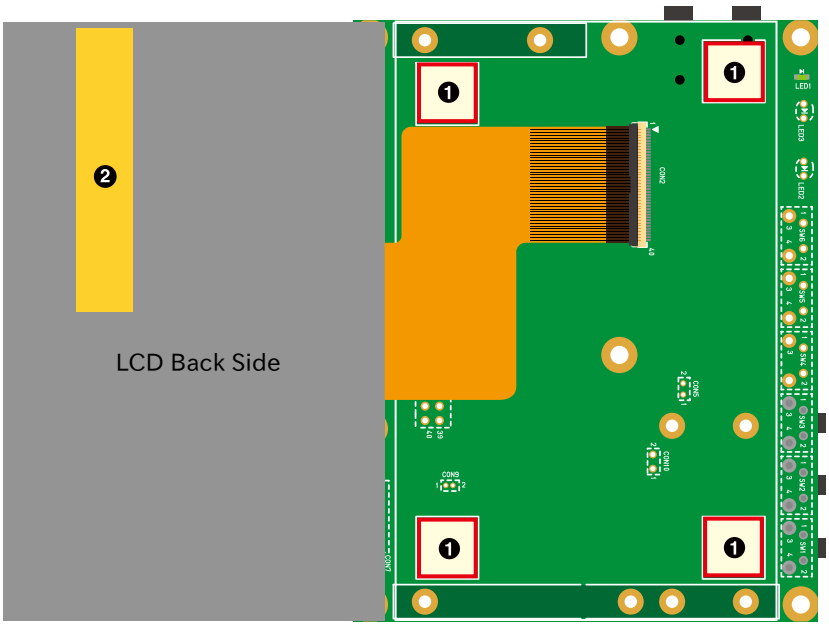


図 18.44 両面テープの貼付位置

❶ 両面テープ貼付位置

❷ 絶縁テープ

付属の両面テープは以下の理由から量産時の使用を推奨しておりません。

- 経年劣化により両面テープの粘着力が低下し、タッチパネル LCD が剥がれる可能性があります。
- 弾力性のある両面テープでタッチパネル LCD を LCD 拡張ボードに固定した場合、タッチパネルに強い力が加わった際に、タッチパネル LCD のフレームと LCD 拡張ボードの基板配線が接触し、故障の原因となる可能性があります。

LCD 拡張ボードには LCD の固定に利用可能な穴を複数設けておりますので、ご利用ください。

タッチパネル LCD 裏面の絶縁テープは LCD 拡張ボード CON2 とのショートを防止するために貼付していますので、剥がさないでください。

Armaidllo-610 拡張ボードと LCD 拡張ボードは「図 18.45. Armadillo-610 と LCD 拡張ボードの接続」を参考にし、Armadillo-610 拡張ボードの CON11 の 1 ピンと LCD 拡張ボードの CON1 の 50 ピンが対応するように、FFC を接続します。FFC は電極が上になるようにします。

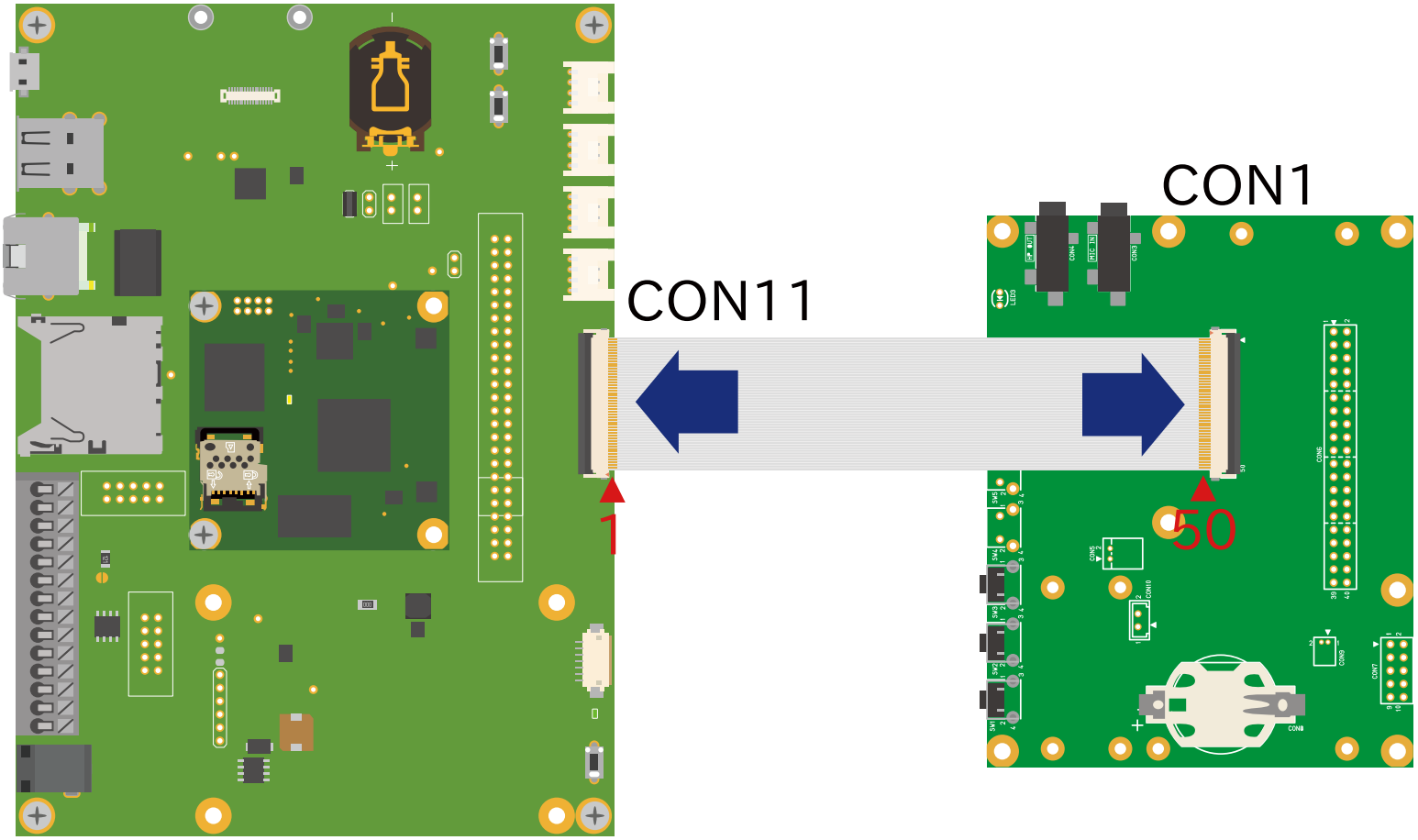


図 18.45 Armadillo-610 と LCD 拡張ボードの接続

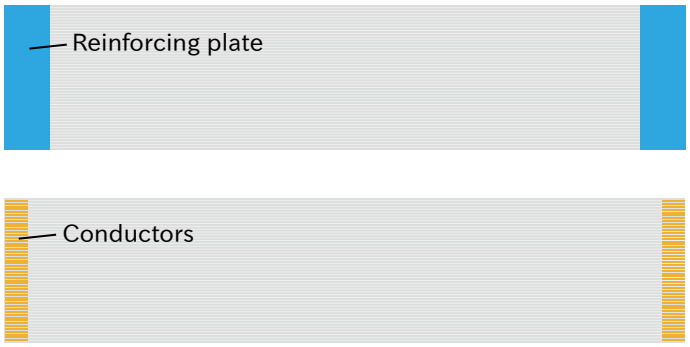


図 18.46 フレキシブルフラットケーブルの形状

必ず 1 ピンと 50 ピンが対応するように、接続してください。1 ピンと 1 ピンが対応するように接続した場合、電源と GND がショートし、故障の原因となります。

FFC の電極を上下逆に接続した場合、実装部品と電極が接触し、故障する可能性があります。

18.4.5. 形状図

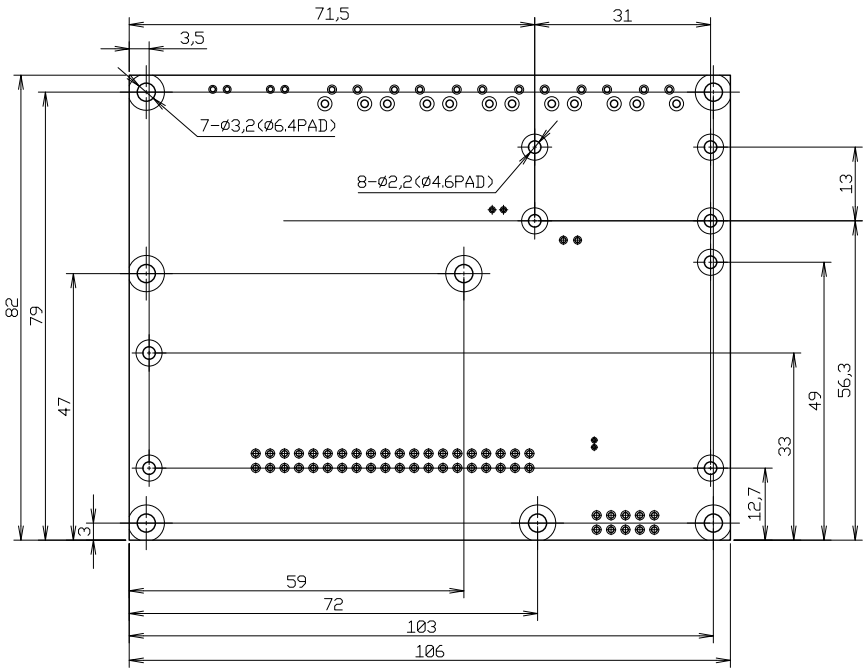


図 18.47 LCD 拡張ボードの基板形状および固定穴寸法

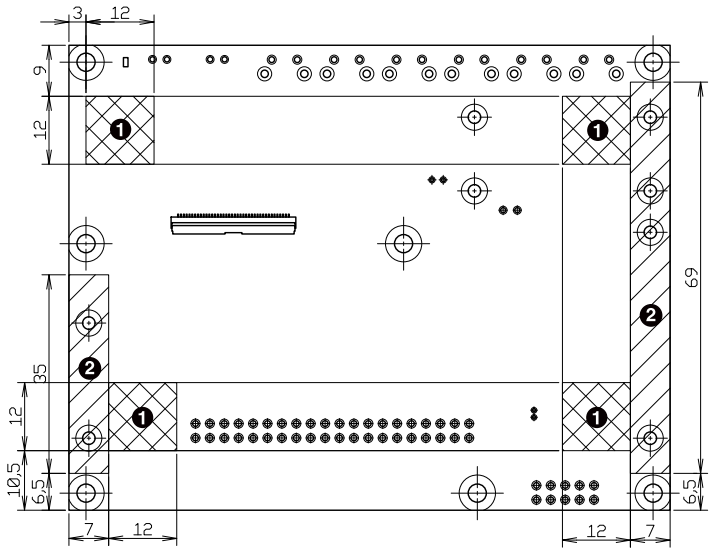


図 18.48 LCD 拡張ボードの両面テープと固定部品配置可能位置

❶ 両面テープ

❷ 固定部品

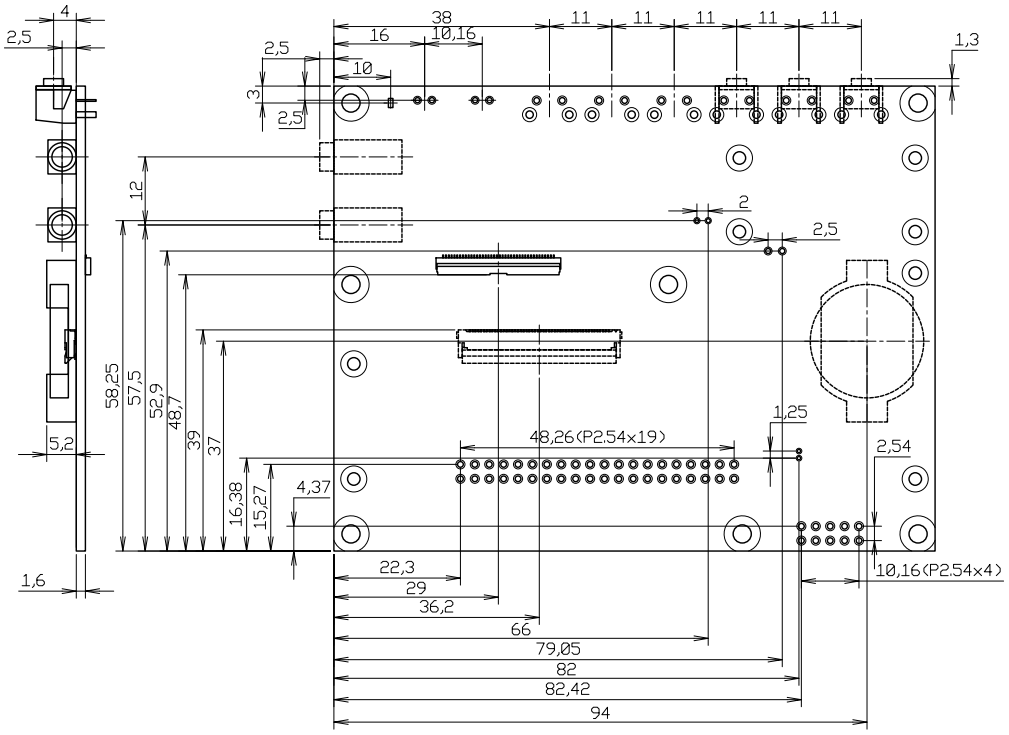


図 18.49 LCD 拡張ボードのコネクタ位置寸法

19. 設計情報

本章では、Armadillo-610 の機能拡張や信頼性向上のための設計情報について説明します。

19.1. 拡張ボードの設計

Armadillo-610 は拡張インターフェース(Armadillo-610: CON2)から拡張します。電源、リセット、複数の機能をもった i.MX6ULL の信号線等、Armadillo-610 を拡張するために必要な信号線はすべて、CON2 の 100 ピンコネクタに接続されています。

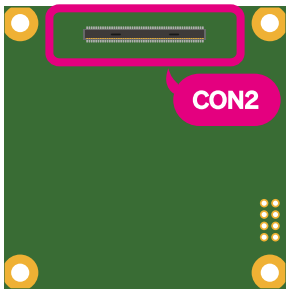


図 19.1 Afmadillo-610 の CON2

Armadillo-610 では、「表 3.3. 仕様」の拡張インターフェースの欄にあるとおりの機能が拡張できます。ただし、ここに記載の拡張数は、優先的に機能を割り当てた場合の最大数ですので、必要な機能がすべて実現できるかは、『Armadillo-610 マルチプレクス表』で検討する必要があります。

マルチプレクス表では、CON2 の各ピンに割り当て可能な機能の他に、リセット後の信号状態、プルアップ/ダウン抵抗の有無等の情報を確認することができます。

Armadillo-610 マルチプレクス表

CON2 ピン番号	信号名	i.MX6ULL ピン名	電圧グループ	リセット時の信号状態			基板上のPull-Up/Pull-Down とコンデンサ	GPIO	USDHC2	
				初期	In/Out	Value				
1	USB_OTG1_DP	USB_OTG1_DP	-	USB_OTG1_DP	-	-				
2	USB_OTG1_DN	USB_OTG1_DN	-	USB_OTG1_DN	-	-				
3	GND									
4	USB_OTG2_DN	USB_OTG2_DN	-	USB_OTG2_DN	-	-				
5	USB_OTG2_DP	USB_OTG2_DP	-	USB_OTG2_DP	-	-				
6	GND									
7	USB_OTG1_VBUS	USB_OTG1_VBUS	-	USB_OTG1_VBUS	-	-	1uF Bypass Capacitor			
8	USB_OTG2_VBUS	USB_OTG2_VBUS	-	USB_OTG2_VBUS	-	-	1uF Bypass Capacitor			
9	SPEEDLED	-	-	-	-	-				
10	LINK_ACTLED	-	-	-	-	-				
11	GPIO1_IO19	UART1_RTS_B	+3.3V_IO	GPIO1_IO19	Input	Keeper	10kΩ Pull-Down	GPIO1_IO19	USDHC2_CD_B	
12	GPIO4_IO17	CSI_MCLK	+3.3V_IO	GPIO4_IO17	Input	Keeper		GPIO4_IO17	USDHC2_CD_B	
13	GPIO5_IO00	SNVS_TAMPER0	VDD_SNVS_IN	GPIO5_IO00	Input	Keeper		GPIO5_IO00		
14	GPIO1_IO04	GPIO1_IO04	+3.3V_IO	GPIO1_IO04	Input	Keeper		GPIO1_IO04		
15	GPIO1_IO03	GPIO1_IO03	+3.3V_IO	GPIO1_IO03	Input	Keeper		GPIO1_IO03		
16	GPIO1_IO02	GPIO1_IO02	+3.3V_IO	GPIO1_IO02	Input	Keeper		GPIO1_IO02		
17	GPIO1_IO01	GPIO1_IO01	+3.3V_IO	GPIO1_IO01	Input	Keeper		GPIO1_IO01		
18	LCD_DATA00	LCD_DATA00	+3.3V_IO	GPIO3_IO05	Input	Keeper	10kΩ Pull-Down	GPIO3_IO05		LCD
19	LCD_DATA01	LCD_DATA01	+3.3V_IO	GPIO3_IO06	Input	Keeper	10kΩ Pull-Up	GPIO3_IO06		LCD
20	LCD_DATA02	LCD_DATA02	+3.3V_IO	GPIO3_IO07	Input	Keeper	10kΩ Pull-Down	GPIO3_IO07		LCD
21	LCD_DATA03	LCD_DATA03	+3.3V_IO	GPIO3_IO08	Input	Keeper	10kΩ Pull-Down	GPIO3_IO08		LCD
22	LCD_DATA04	LCD_DATA04	+3.3V_IO	GPIO3_IO09	Input	Keeper	10kΩ Pull-Down	GPIO3_IO09		LCD
23	LCD_DATA05	LCD_DATA05	+3.3V_IO	GPIO3_IO10	Input	Keeper	10kΩ Pull-Up or 10kΩ Pull-Down	GPIO3_IO10		LCD
24	LCD_DATA06	LCD_DATA06	+3.3V_IO	GPIO3_IO11	Input	Keeper	10kΩ Pull-Up	GPIO3_IO11		LCD
25	LCD_DATA07	LCD_DATA07	+3.3V_IO	GPIO3_IO12	Input	Keeper	10kΩ Pull-Down	GPIO3_IO12		LCD
26	LCD_DATA08	LCD_DATA08	+3.3V_IO	GPIO3_IO13	Input	Keeper	10kΩ Pull-Down	GPIO3_IO13		LCD
27	LCD_DATA09	LCD_DATA09	+3.3V_IO	GPIO3_IO14	Input	Keeper	10kΩ Pull-Down	GPIO3_IO14		LCD
28	LCD_DATA10	LCD_DATA10	+3.3V_IO	GPIO3_IO15	Input	Keeper	10kΩ Pull-Down	GPIO3_IO15		LCD
29	LCD_DATA11	LCD_DATA11	+3.3V_IO	GPIO3_IO16	Input	Keeper	10kΩ Pull-Up or 10kΩ Pull-Down	GPIO3_IO16		LCD
30	LCD_DATA12	LCD_DATA12	+3.3V_IO	GPIO3_IO17	Input	Keeper	10kΩ Pull-Down	GPIO3_IO17		LCD
31	LCD_DATA13	LCD_DATA13	+3.3V_IO	GPIO3_IO18	Input	Keeper	10kΩ Pull-Down	GPIO3_IO18	USDHC2_RESET_B	LCD
32	LCD_DATA14	LCD_DATA14	+3.3V_IO	GPIO3_IO19	Input	Keeper	10kΩ Pull-Down	GPIO3_IO19	USDHC2_DATA4	LCD

割り当て可能な機能

図 19.2 Afmadillo-610 のマルチプレクス表

本書には各機能の概要しか記載していませんので、詳細な仕様が必要な場合は、NXP Semiconductors のホームページからダウンロード可能な、『i.MX 6ULL Applications Processor Reference Manual』、『i.MX 6ULL Applications Processors for Industrial Products』をご確認ください。Armadillo-610 固有の情報を除いて、回路設計に必要な情報はこれらのマニュアルに、すべて記載されています。検索しやすいように、マルチプレクス表や「16.2. CON2(拡張インターフェース)」に i.MX6ULL のピン名やコントローラー名を記載しておりますので、是非ご活用ください。

Armadillo-610 マルチプレクス表は「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロードしてください。

Armadillo-610 の拡張ボードを設計開発するためのリファレンスボードとして、Armadillo-610 拡張ボードの回路図を公開しています。Armadillo-610 拡張ボードでは、以下の機能の拡張方法を確認することが可能です。

- 電源
- 起動デバイス設定
- LAN(Ethernet)、無線 LAN
- USB HUB、USB Host、USB OTG
- SD
- LCD [1]
- RS485、UART(デバッグ用)
- MQS、SAI [1]
- 絶縁デジタル入出力
- リアルタイムクロック
- スイッチ、LED
- A/D、I2C

Armadillo-610 拡張ボードの回路図/部品表[2]は「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロードしてください。

[1]Armadillo-400 シリーズ LCD オプションセットの LCD 拡張ボードの回路図も合わせてご確認ください。
[2]Armadillo-610 拡張ボードの回路図、部品表は購入者向けの限定データです。

19.1.1. 回路設計

19.1.1.1. 電源

Armadillo-610 の電源電圧は 3.6〜4.5V です。拡張ボード側に USB Host を搭載するのであれば、USB デバイスに供給するための 5V が必要となりますので、拡張ボード側の主電源を 5V にして USB への供給電源とし、5V を 3.6〜4.5V の電圧に降圧して、Armadillo-610 に供給するのがおすすめです。

Armadillo-610 拡張ボードでは、5V を 3.7V に降圧して Armadillo-610 に供給する回路を確認することができます。抵抗値で出力電圧を変更できるタイプの DC/DC コンバータを採用しておりますので、別の電圧値が必要な場合は、抵抗値を変更することで対応可能です。

電源は拡張インターフェース(Armadillo-610: CON2)の VIN ピンから供給します。CON2 に搭載している 100 ピンのコネクタは、1 ピンあたり流せる電流値が最大 0.3A です。VIN ピンは全部で 4 本ありますので、CON2 から供給できる電流値は最大 1.2A となります。

CON2 から、5V 電源(+5V_IO)、3.3V 電源(+3.3V_IO)が拡張ボード用に出力されています。+5V_IO は最大 600mA、+3.3V_IO は最大 500mA まで供給可能ですが、Armadillo-610 への最大供給電流が 1.2A であるため、必ずしも最大まで出力することはできません。各最大電流値を超えないように外部機器の接続、供給電源の設計を行ってください。

全体の消費電力が少ないのであれば、Armadillo-610 へは 3.6〜4.5V 電源を直接供給し、5V 電源、3.3V 電源を Armadillo-610 からの出力で賄うという構成も可能です。

Armadillo-610 の電源投入時、+5V_IO は Armadillo-610 の電源(VIN)とほぼ同時に立ち上がり、一定時間 VIN と同電位を維持した後、ソフトウェアから有効にされたタイミングで 5V まで立ち上がります。+5V_IO は、VIN を昇圧して 5V 電源を生成しているため、+5V_IO からの出力電圧を VIN 以下にすることはできません。+5V_IO の出力無効時は VIN と同電位の電圧が出力され、+5V_IO の出力有効時は +5V が出力されます。+5V_IO の出力を 0V にしたい場合は、電源制御のためのパワースイッチの搭載をおすすめします。

「15. 電気的仕様」で Armadillo-610 の電気的仕様について説明しておりますので、詳細についてはこちらでご確認ください。

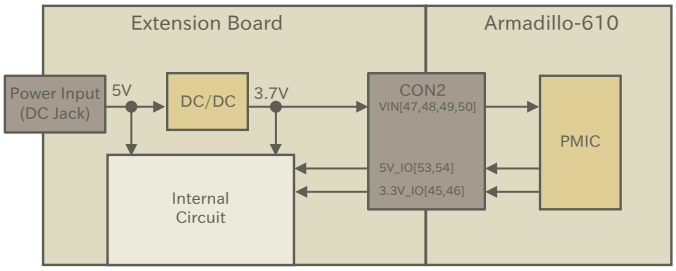


図 19.3 電源回路例

19.1.1.2. 起動デバイスの設定

Armadillo-610 は下記 2 つのデバイスから起動が可能です。

- オンボード eMMC
- microSD カード(Armadillo-610: CON1)

どちらのデバイスから起動するかは、eFUSE もしくは拡張インターフェース(Armadillo-610: CON2)の BJP1 ピンで設定します。eFUSE で設定する方法については、付録 A eFuse をご確認ください。BJP1 ピンの状態は、Armadillo-610 の電源(VIN)投入時に読み出され、起動デバイスが選択されます。

表 19.1 BJP1 の状態と起動デバイス

BJP1	起動デバイス
Low	eMMC
High	microSD カード

BJP1 ピンは Armadillo-610 上で 47kΩ でプルダウンされているため、eMMC から起動したい場合は BJP1 ピンをオープン、microSD カードから起動したい場合は、BJP1 ピンを+3.3V_IO にプルアップ(抵抗値は 1kΩ 程度)してください。

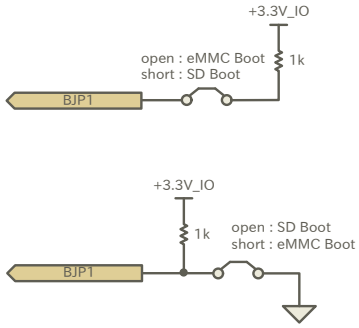


図 19.4 起動設定ジャンパー例

起動デバイスを microSD カードに設定した場合でも、microSD スロットに microSD カードが挿さっていなかった場合は、eMMC から起動します。 [3]

出荷時、i.MX6ULL の起動デバイスに関する eFUSE は未設定です。

eFUSE を設定した場合、BJP1 の設定は無視されます。

eFUSE は一度書き込むと元に戻すことができません。eFUSE の設定によっては Armadillo-610 が正常に動作しなくなる可能性がありますので、書き込みを行う際には細心の注意を払うようお願いいたします。eFUSE の設定によって異常が起こった場合は、保証対象外となります。

[3]eFUSE で eMMC からの起動を禁止した場合を除きます

19.1.1.3. LAN(Ethernet)

LAN を拡張する場合は、拡張ボード側にトランスと LAN コネクタ、LED を搭載してください。拡張インターフェース(Armadillo-610: CON2)には Ethernet PHY の送受信の信号線および LAN のスピード LED、アクティビティ LED 用の信号線が接続されています。

回路の詳細は Armadillo-610 拡張ボードで確認することが可能です。

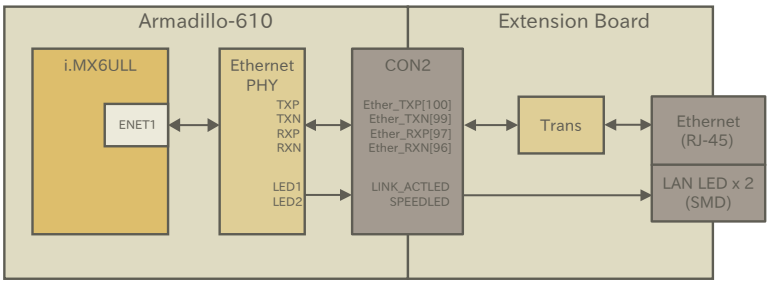


図 19.5 LAN(Ethernet)接続例

19.1.1.4. 無線 LAN

無線 LAN を拡張する場合、Armadillo-WLAN(AWL13)がおすすめです。Armadillo-WLAN(AWL13)は USB 起動モードで Armadillo-610 に接続することが可能です。

Armadillo-WLAN(AWL13)を搭載する場合は、電源制御のためのパワースイッチも搭載してください。パワースイッチのイネーブルピン制御のための GPIO は電源投入時、プルアップされていないものを選定してください。5V 電源が必要ない USB デバイスを使用する場合でも、USB PHY への電源供給のため USB_OTGx_VBUS ピンへダイオード経由で 5V を入力する必要があります。

回路の詳細は Armadillo-610 拡張ボードで確認することが可能です。Armadillo-610 開発セットに Armadillo-WLAN(AWL13)は付属しませんので、動作を確認する場合は別途購入が必要となります。

Armadillo-WLAN モジュール(AWL13)の仕様については、Armadillo サイトで公開している Armadillo-WLAN(AWL13)の各種ドキュメントをご確認ください。

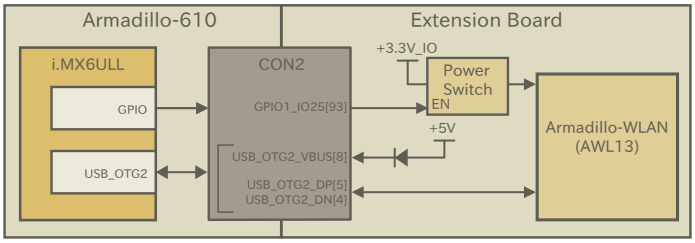


図 19.6 Armadillo-WLAN(AWL13)接続例

19.1.1.5. USB

USB は 2 ポート拡張可能で、Host は最大 2 ポート、OTG は最大 1 ポート拡張することが可能です。

USB Host を拡張する場合は、USB デバイスへ供給する電源制御のためのパワースイッチを搭載してください。パワースイッチのイネーブルピン制御のための GPIO は電源投入時、プルアップされていないものを選定してください。USB_OTGx_VBUS ピンを使用しない場合でも、デバイス検出のためにダイオード経由で 5V を入力する必要があります。

USB ポートが 3 ポート以上必要な場合は、USB ハブを接続してください。

Armadillo-610 拡張ボードでは、USB ハブ、Host、OTG の回路を確認することが可能です。拡張ボードで採用している USB ハブは 3 ポート品ですが、ピンコンパチで 4 ポート品もラインアップされているので、そちらもご検討ください。

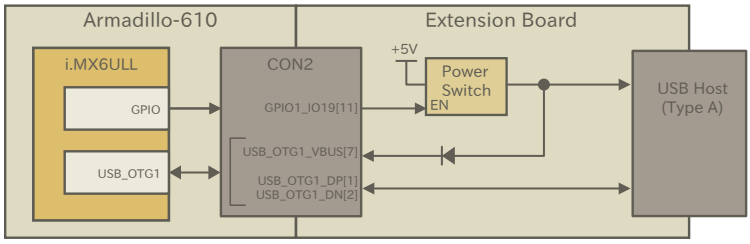


図 19.7 USB Host 接続例

19.1.1.6. シリアル(UART)

Armadillo-610 のシリアル(UART)の信号レベルは+3.3V_IO ですので、必要なインターフェースの規格に合わせて、レベル変換 IC 等を拡張ボード側に搭載してください。

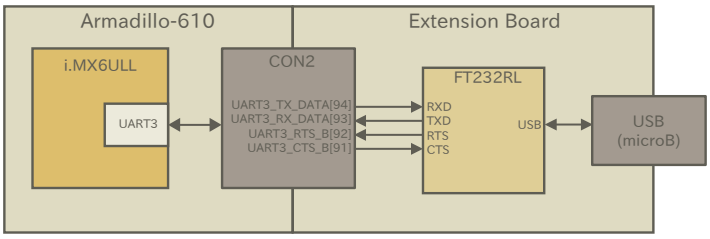


図 19.8 シリアル(UART)接続例

i.MX6ULL の CTS、RTS 信号は一般的な UART の信号と名前が逆になっています。誤接続にご注意ください。

デバッグやメンテナンス用途であれば、拡張ボード上にレベル変換 IC を搭載せずに、外付けのレベル変換アダプタを使用するのもおすすめです。レベル変換アダプタは、弊社からもオプションで購入することが可能です。

19.1.1.7. SD

SD ホストコントローラ(uSDHC2)を使用できる信号線が SD インターフェース(Armadillo-610: CON1)と拡張インターフェース(Armadillo-610: CON2)に接続されており、SD ホストコントローラはどちらか一方でしか使用することができません。

SD スロットを基板端に配置したい場合や SDIO 接続のデバイスを拡張したい場合などにご使用ください。

SD スロットを拡張する場合、SD カード検出、ライトプロテクト検出は GPIO で行うことが可能ですので、専用ピンを割り当てる必要はありません。

BJP1 ピンを High レベルにして Armadillo-610 の電源(VIN)を投入した場合、SD インターフェース(Armadillo-610: CON1)に接続された microSD カードがブートデバイスに設定されます。電源投入時、SD ホストコントローラ(uSDHC2)は SD インターフェース(Armadillo-610: CON1)に接続されており、microSD カードに書き込まれたイメージファイルから起動します。起動後、マルチプレクスの設定により、SD ホストコントローラ(uSDHC2)の接続先が変更になるため、拡張インターフェース(Armadillo-610: CON2)側のデバイスがブートデバイスになることはできません。

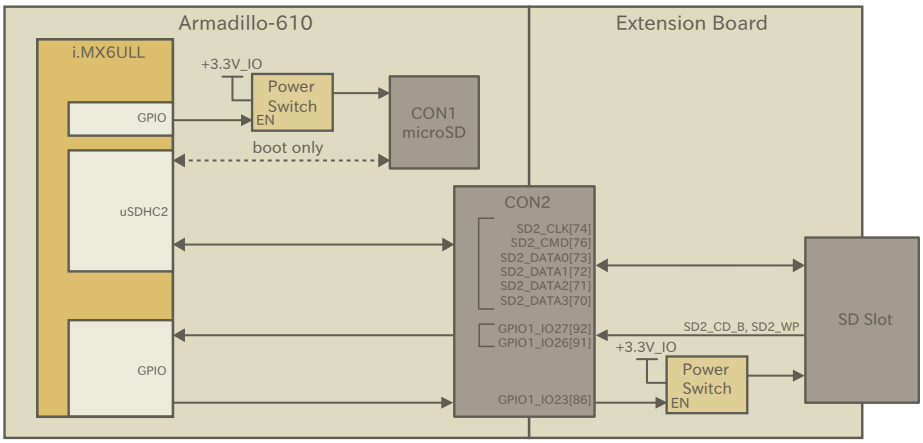


図 19.9 SD 接続例

SD2_DATA3 として使用可能な GPIO3_IO28 ピンはブートモード設定ピンを兼用しています。Armadillo-610 の電源(VIN)投入時に Low レベルを保持する必要があり、47kΩ のプルダウン抵抗が接続されています。ただし、SD カード等が正常動作するには、SD2_DATA3 にプルアップ抵抗が必要となるため、Armadillo-610 の電源(VIN)投入時は立ち上がらない電源でプルアップ(抵抗値は 15kΩ 程度)し、起動後にプルアップ抵抗の接続された電源を立ち上げる等の対処が必要となります。

SD スロットを拡張する回路の詳細は、Armadillo-610 拡張ボードで確認することが可能です。

19.1.1.8. スイッチ、LED、リレー

スイッチや LED、リレーを拡張する場合は、GPIO を割り当てます。GPIO に割り当て可能なピンは多数ありますので、プルアップ/プルダウン抵抗の有無と電圧レベルを確認して、使用するピンを決定してください。

拡張インターフェース(Armadillo-610: CON2)には、i.MX6ULL の信号線が直接接続されています。スイッチは人の手で操作するインターフェースですので、静電気等による内部回路の故障を防ぐため、電流制限抵抗を接続することをおすすめします。

LED、リレーは GPIO ピンで直接駆動せずにトランジスタ等を経由して駆動してください。

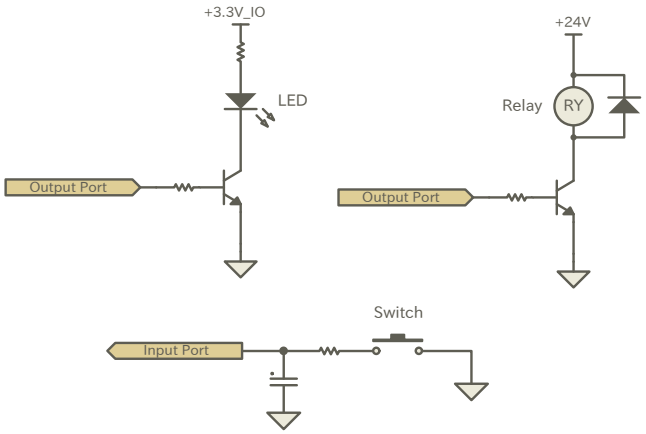


図 19.10 スイッチ、LED、リレー接続例

19.1.1.9. リアルタイムクロック

i.MX6ULL 内蔵のリアルタイムクロックのバックアップ用のピン(RTC_BAT)が拡張インターフェース(Armadillo-610: CON2)に接続されていますので、Armadillo-610 の電源(VIN)を切断しても時刻データを保持したい場合にバッテリー等を接続してください。

i.MX6ULL 内蔵のリアルタイムクロックは、一般的なリアルタイムクロック IC よりも消費電力が高いため、外付けバッテリーの消耗が速くなります。

バッテリーの消耗が製品の運用に支障をきたす場合は、消費電力が少ないリアルタイムクロックを拡張ボード側に搭載してください。

19.1.1.10. LCD

デジタル RGB 入力を持つ液晶パネルモジュールなどを接続することができます。Armadillo-610 拡張ボードでは、以下のタッチパネル LCD を接続するための回路を確認することが可能です。

- LCD オプションセット(7 インチタッチパネル WVGA 液晶)(型番: OP-LCD70EXT-00)、
- Armadillo-400 シリーズ LCD オプションセット(4.3 インチタッチパネル WQVGA 液晶)(型番: OP-A400-LCD43EXT-L01)

Armadillo-610 開発セットに LCD オプションセットは付属しませんので、動作を確認する場合は別途購入が必要となります。

オプションセットの詳細につきましては「18.3. LCD オプションセット(7 インチタッチパネル WVGA 液晶)」、「18.4. Armadillo-400 シリーズ LCD オプションセット」をご確認ください。

19.1.1.11. オーディオ

I2S で最大 2 ポート、MQS で最大 1 ポート、S/PDIF で最大 1 ポートオーディオを拡張することが可能です。MQS は拡張ボード側にオーディオアンプを搭載するだけで良いので、高品質な音が必要でない場合にはおすすめです。SAI、S/PDIF を使用する場合は、オーディオコーデックとオーディオアンプ等を拡張ボード側に搭載してください。

MQS の回路は Armadillo-610 拡張ボード、I2S の回路は Armadillo-400 シリーズ LCD オプションセットの LCD 拡張ボードで確認することが可能です。

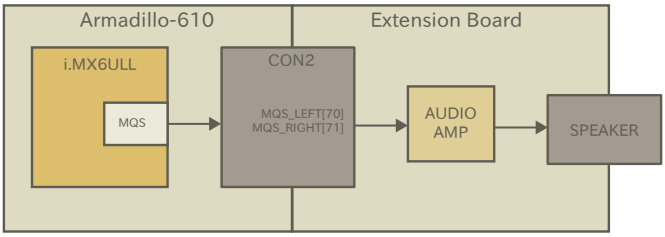


図 19.11 MQS 接続例

19.1.1.12. リセットスイッチ

拡張インターフェース(Armadillo-610: CON2)の EXT_RESET_B ピン、PWRON ピンによりリセットスイッチを拡張することが可能です。押していない時はオープン、押した時は GND とショートするスイッチを接続してください。

スイッチは人の手で操作するインターフェースですので、静電気等による内部回路の故障を防ぐため、電流制限抵抗を接続することをおすすめします。

EXT_RESET_B ピンではシステムリセット、PWRON ピンではパワーマネジメント IC からの電源の供給/切断の制御が可能です。それぞれの機能の詳細については、「15.5. リセット回路の構成」、「15.6. 外部からの電源制御」をご確認ください。

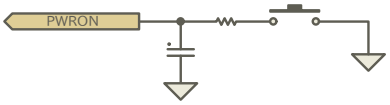


図 19.12 PWRON 回路例

19.1.1.13. ONOFF スイッチ

拡張インターフェース(Armadillo-610: CON2)の ONOFF ピンにより、長押しで電源の制御を行うスイッチを拡張することが可能です。押していない時はオープン、押した時は GND とショートするスイッチを接続してください。

スイッチは人の手で操作するインターフェースですので、静電気等による内部回路の故障を防ぐため、電流制限抵抗を接続することをおすすめします。

ONOFF ピンの機能の詳細については「15.6. 外部からの電源制御」をご確認ください。

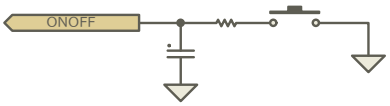


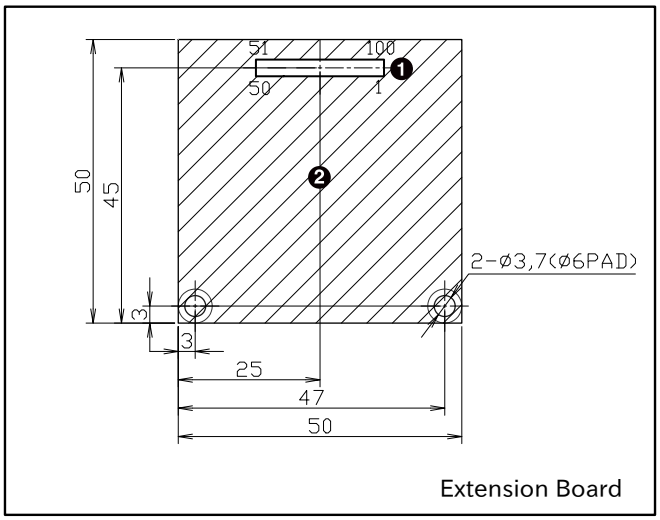
図 19.13 ONOFF 回路例

19.1.2. 基板形状

Armadillo-610 の拡張ボードを設計する際の、推奨レイアウトは「図 19.14. 拡張ボード推奨レイアウト」のとおりです。

Armadillo-610 との接続コネクタは、HIROSE ELECTRIC 製の DF40HC(3.0)-100DS-0.4V(51)を搭載してください。嵌合高さは 3mm となりますので、Armadillo-610 の下に部品を配置する場合、部品高さにご注意ください。

固定穴径とパッド寸法はマックエイト製のスルーホールタップ(TH-1.6-3.0-M3)を実装する場合の推奨となります。別の方法で固定する場合は適宜寸法を変更してください。



[Unit:mm]

図 19.14 拡張ボード推奨レイアウト

❶ DF40HC(3.0)-100DS-0.4V(51)/HIROSE ELECTRIC

❷ Armadillo-610 側の最大部品高さ: 2mm

Armadillo-610 の固定穴は GND に接続されています。GND 強化のため、拡張ボード側の固定穴も GND に接続し、金属製のスペーサーとねじで固定してください。スペーサーの長さはコネクタの嵌合高さと同じ 3mm としてください。

Armadillo-610 の固定例は「図 19.15. Armadillo-610 の固定例」のとおりです。

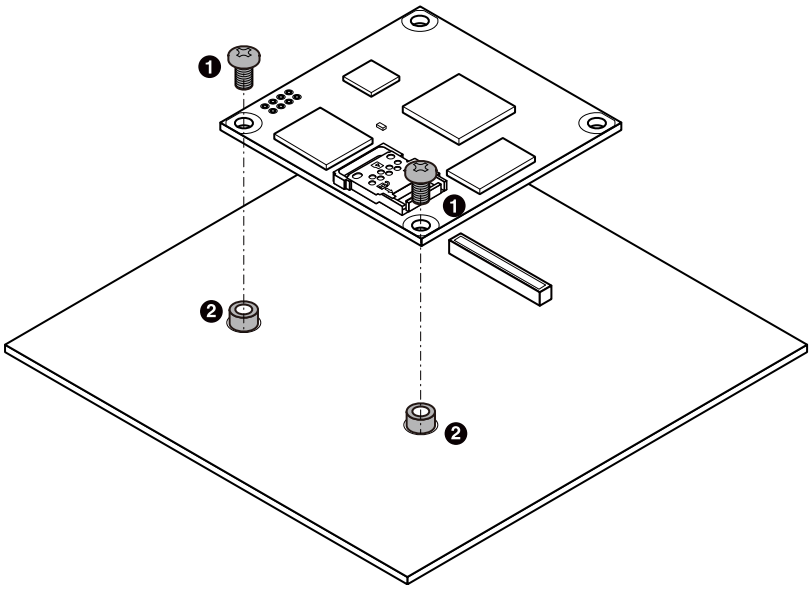


図 19.15 Armadillo-610 の固定例

❶ なべ小ねじ(M3、L=5mm)

❷ スルーホールタップ(TH-1.6-3.0-M3/Mac-Eight)

Armadillo-610 の固定穴は 4 箇所ありますが、コネクタから離れた位置の 2 箇所のみ、スペーサーとねじで固定して各種試験を行い、動作に異常がないことを確認しております。試験の詳細につきましては、「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/] の「Armadillo-610 信頼性試験報告書」にてご確認ください。

19.2. 製品化に向けて

Armadillo-610 を組み込んだ基板からの放射ノイズを減らしたり、ESD 耐性を向上させるさせるための情報を紹介します。

19.2.1. 放射ノイズ

フレキシブルフラットケーブル(FFC)を使用して拡張を行った場合、放射ノイズが問題になる場合があります。特に、オーディオアンプのような電力が大きく変動するデバイスを FFC で接続する拡張ボードに搭載している場合、FFC の GND 線の接続のみでは強い放射ノイズが発生する可能性があります。放射ノイズを減らすために、以下の対策が効果的です。

- シールド付 FFC を使用する
  - 長さが余る場合は、ケーブルを折りたたむ
  - シールドは拡張ボードの GND に接続する
- 固定穴を GND と接続し、固定穴同士を太い導線や金属スペーサー等で接続する
- 未使用の拡張ピンは Low レベル出力とする

・使用する拡張ピンはコンデンサ(1000pF 程度)を介して GND と接続する

19.2.2. ESD/雷サージ

Armadillo-610 を組み込んだ機器の ESD 耐性を向上させるために、以下の対策が効果的です。

・金属筐体に組み込み、GND(固定穴等)を金属ねじ等で接続する

・金属筐体を接地する

Armadillo-610 を組み込んだ機器に接続されたケーブルが屋外に露出するような設置環境では、ケーブルに侵入した雷サージ等のストレスによりインターフェース回路が破壊される場合があります。ストレスへの耐性を向上させるために、以下の対策が効果的です。

・通信対向機との GND 接続を強化する

・シールド付きのケーブルを使用する

20. Howto

本章では、Armadillo-610 のソフトウェアをカスタマイズする方法などについて説明します。

20.1. Device Tree とは

Device Tree とは、ハードウェア情報を記述したデータ構造体です。ハードウェアの差分を Device Tree に記述することによって、1 つの Linux カーネルイメージを複数のハードウェアで利用することができるようになります。

Device Tree に対応しているメリットの 1 つは、ハードウェアの変更に対するソフトウェアの変更が容易になることです。例えば、拡張インターフェース(Armadillo-610: CON2)に接続する拡張基板を作成した場合、主に C 言語で記述された Linux カーネルのソースコードを変更する必要はなく、やりたいことをより直感的に記述できる DTS(Device Tree Source)の変更で対応できます。

ただし、Device Tree は「データ構造体」であるため、ハードウェアの制御方法などの「処理」を記述することができない点に注意してください。Device Tree には、CPU アーキテクチャ、RAM の容量、各種デバイスのベースアドレスや割り込み番号などのハードウェアの構成情報のみが記述されます。

Device Tree のより詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(`Documentation/devicetree/`)、devicetree.org で公開されている「Device Tree Specification」を参照してください。

> **DeviceTree: The Devicetree Specification**
>
> https://www.devicetree.org/

Linux カーネルのソースコードに含まれている初期出荷状態での DTS、及び関連するファイルを次に示します。

初期出荷状態での DTS、及び関連するファイル

- `arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dts`
- `arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dts`
- `arch/arm/boot/dts/armadillo-610-extboard-eva-common.dtsi`
- `arch/arm/boot/dts/armadillo-610.dtsi`
- `arch/arm/boot/dts/imx6ull.dtsi`
- `arch/arm/boot/dts/imx6ul.dtsi`

20.2. 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 カーネルイメージをカスタマイズする方法を説明します。

20.2.1. イメージをカスタマイズ

1. Linux カーネルアーカイブの展開

Linux カーネルのソースコードアーカイブを準備し、Linux カーネルのソースコードアーカイブを展開します。

```
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version].tar.gz
[ATDE ~]$ tar xf linux-v4.14-at[version].tar.gz
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version]  linux-v4.14-at[version].tar.gz
```

2. initramfs アーカイブへのシンボリックリンク作成

   Linux カーネルディレクトリに移動して、initramfs アーカイブへのシンボリックリンク作成します。

```
[ATDE ~]$ cd linux-v4.14-at[version]
[ATDE ~/linux-v4.14-at[version]]$ ln -s ../initramfs_a600-[version].cpio.gz
initramfs_a600.cpio.gz
```

3. コンフィギュレーション

   コンフィギュレーションをします。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm armadillo-640_defconfig
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm menuconfig
```

4. カーネルコンフィギュレーションの変更

   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 4.14-at1 Kernel Configuration
--------------------------------------------------------------------------------
 ---------------- Linux/arm 4.14-at1 Kernel Configuration ----------------
    Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
    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  [ ]
   ------------------------------------------------------------------------
            General setup  --->
        [ ] Enable loadable module support  ----
        [*] Enable the block layer  --->
            System Type  --->
            Bus support  --->
            Kernel Features  --->
            Boot options  --->
            CPU Power Management  --->
            Floating point emulation  --->
            Userspace binary formats  --->
            Power management options  --->
        [*] Networking support  --->
            Device Drivers  --->
```

```
            Firmware Drivers  --->
            File systems  --->
            Kernel hacking  --->
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
        [ ] Virtualization  ----



  ------------------------------------------------------------------------
------------------------------------------------------------------------------
        <Select>    < Exit >    < Help >    < Save >    < Load >
```

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。

1. ビルド

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

2. イメージファイルの生成確認

   ビルドが終了すると、arch/arm/boot/ディレクトリと、arch/arm/boot/dts/以下にイメージファイル(Linux カーネルと DTB)が作成されています。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/uImage
arch/arm/boot/uImage
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dtb
arch/arm/boot/dts/armadillo-610-extboard-eva-lcd.dtb
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dtb
arch/arm/boot/dts/armadillo-610-extboard-eva-grove.dtb
```

20.3. Device Tree をカスタマイズする

at-dtweb を利用して Device Tree をカスタマイズする方法を説明します。at-dtweb では、Web ブラウザ上のマウス操作で DTS および DTB を生成することができます。

Armadillo-610 のカスタマイズの対象は拡張インターフェース(Armadillo-610: CON2)です。

20.3.1. at-dtweb のインストール

ATDE7 に at-dtweb パッケージをインストールします。

```
[ATDE ~]$ sudo apt-get update
[ATDE ~]$ sudo apt-get install at-dtweb
```

インストール済みの場合は、以下のコマンドを実行し最新版への更新を行ってください。

```
[ATDE ~]$ sudo apt-get update
[ATDE ~]$ sudo apt-get upgrade
```

20.3.2. at-dtweb の起動

1. at-dtweb の起動開始

   at-dtweb の起動を開始するには、デスクトップ左上のアクティビティから「at-dtweb」と入力し、at-dtweb のアイコンをクリックしてください。

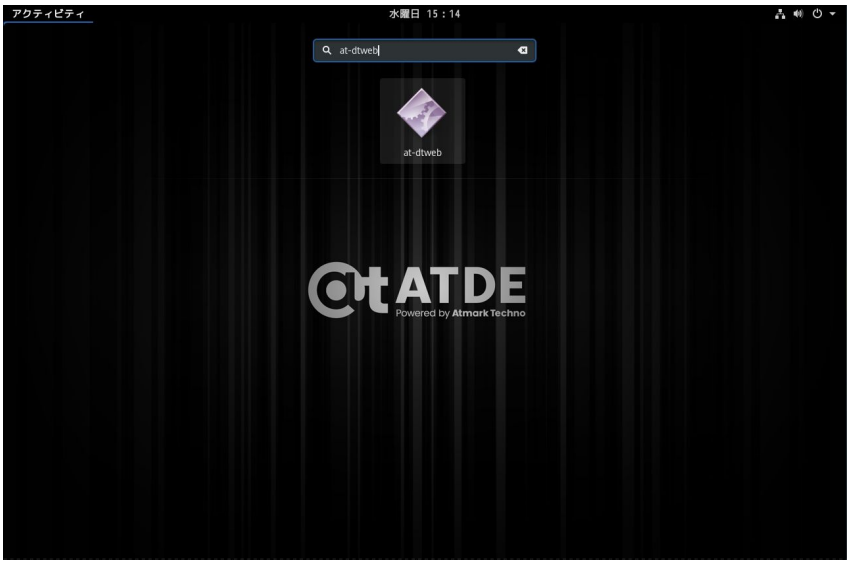


図 20.1 at-dtweb の起動開始

2. ボードの選択

   ボードを選択します。Armadillo-610 を選択して、「OK」をクリックします。

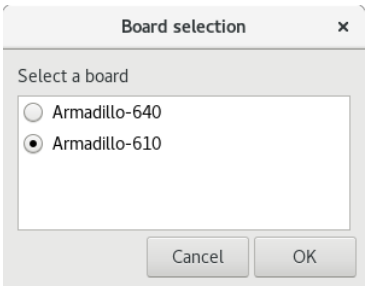


図 20.2 ボード選択画面

3. Linux カーネルディレクトリの選択

Linux カーネルディレクトリを選択します。コンフィギュレーション済みの Linux カーネルディレクトリを選択して、「OK」をクリックします。

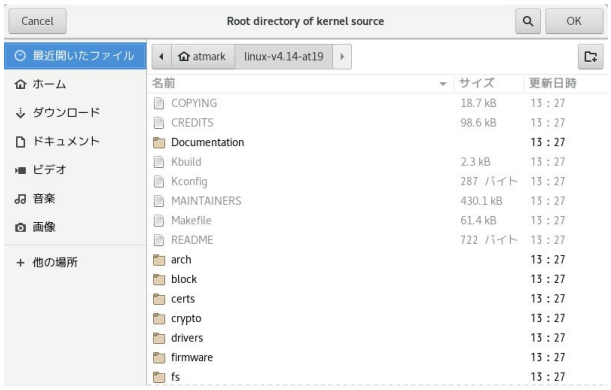


図 20.3 Linux カーネルディレクトリ選択画面

4. at-dtweb の起動完了

   at-dtweb が起動し、次のように画面が表示されます。

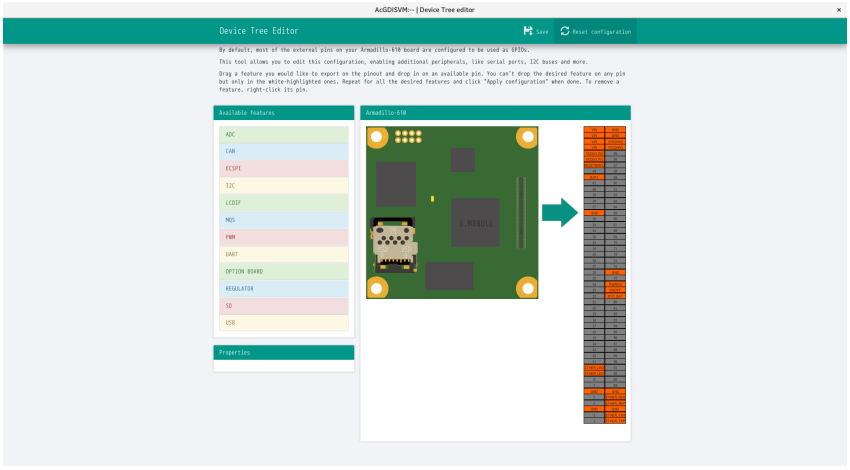


図 20.4 at-dtweb 起動画面

Linux カーネルは、事前にコンフィギュレーションされている必要があります。コンフィギュレーションの手順については「10.2.1. 手順：Linux カーネルをビルド」を参照してください。

20.3.3. Device Tree をカスタマイズ

20.3.3.1. 機能の選択

機能の選択は、ドラッグ&ドロップで行います。画面左上の「Available features」から有効にしたい機能をドラッグし、画面右側の「Armadillo-610」の白色に変化したピンにドロップします。例として CON2 83/85 ピンを UART1(RXD/TXD)に設定します。

何も機能が選択されていないピンには GPIO の機能が割り当てられます。

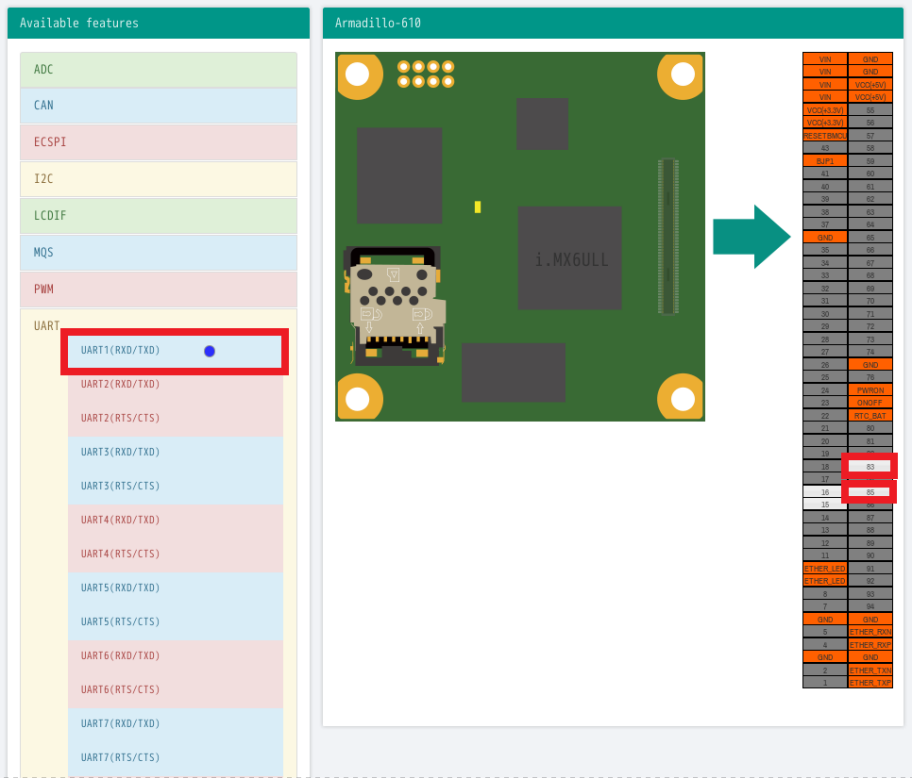


図 20.5 UART1(RXD/TXD)のドラッグ

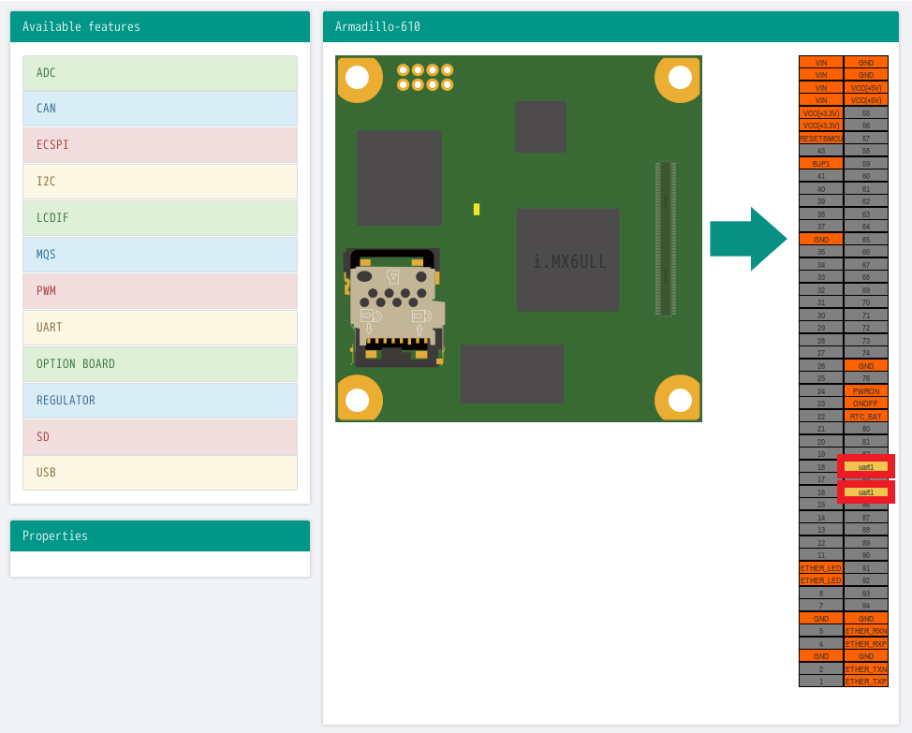


図 20.6 CON2 83/85 ピンへのドロップ

20.3.3.2. 信号名の確認

画面右側の「Armadillo-610」にドロップして設定したピンを左クリックすると信号名が表示されます。どのピンがどの信号に対応しているのかを確認することができます。

例として UART1(RXD/TXD) の信号名を確認します。

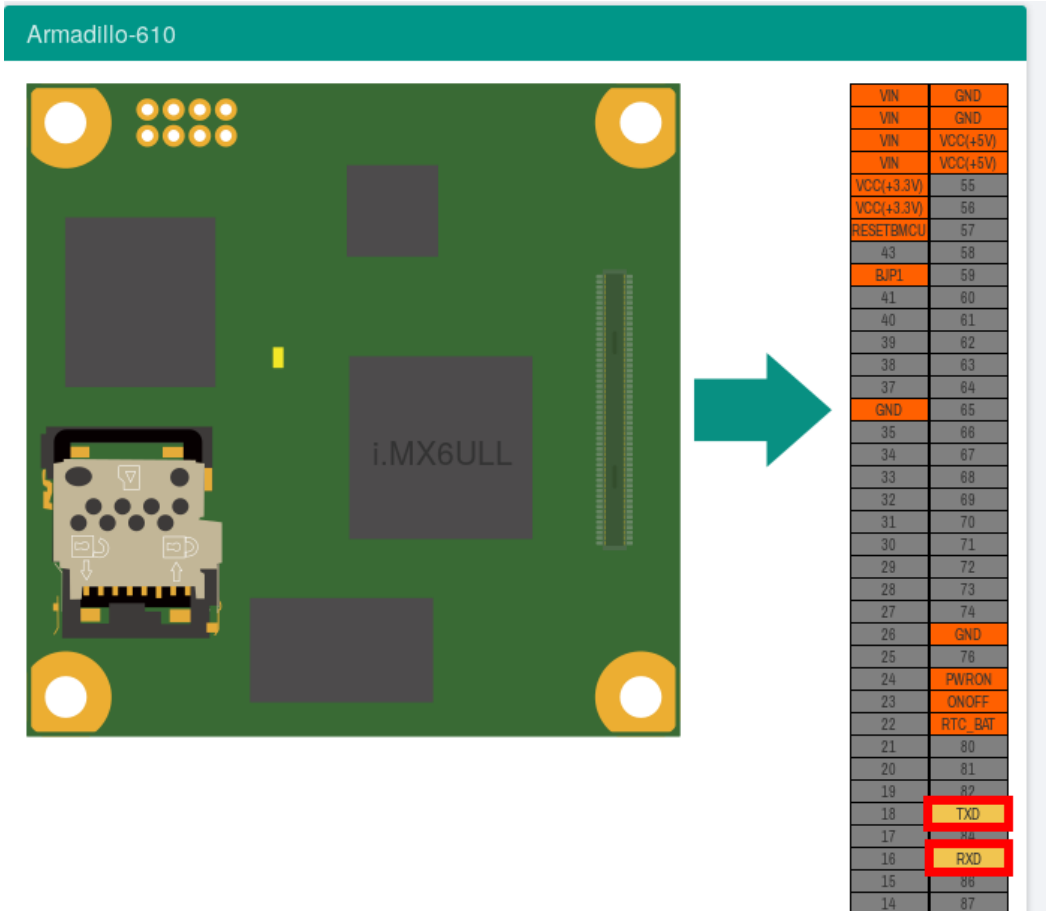


図 20.7 信号名の確認

再度ピンを左クリックすると機能名の表示に戻ります。

20.3.3.3. プロパティの設定

いくつかの機能にプロパティを設定することができます。画面右側の「Armadillo-610」に選択した機能を左クリックすると、画面左下の「Properties」からプロパティを選択することができます。

例として CON2 88/89 ピンの I2C4(SCL/SDA)の clock-frequency プロパティを設定します。

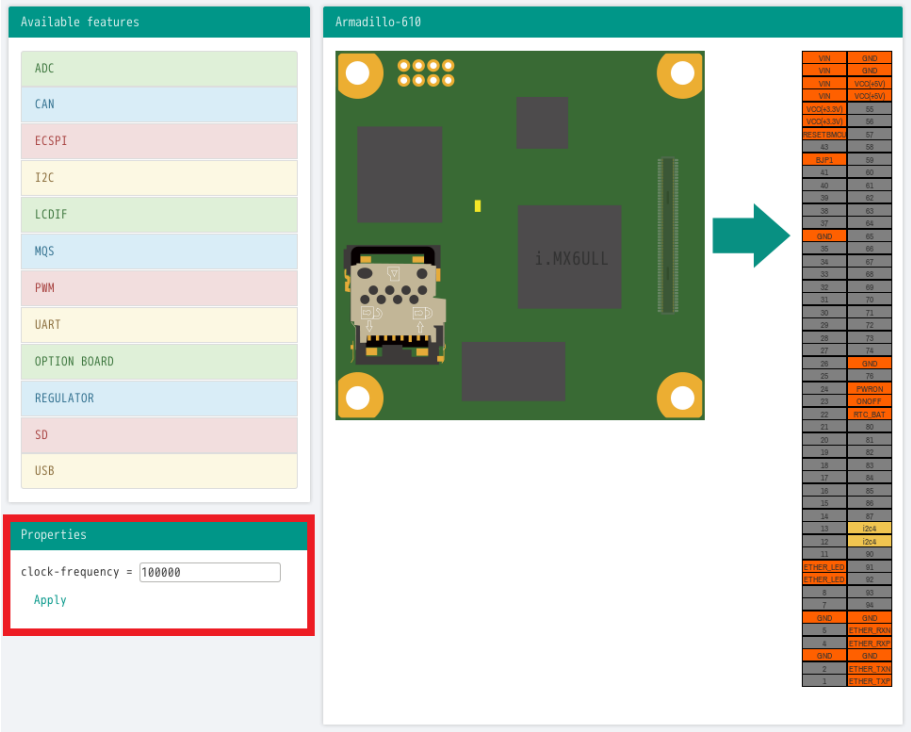


図 20.8 プロパティの設定

設定したプロパティを確定させるには「Apply」をクリックします。

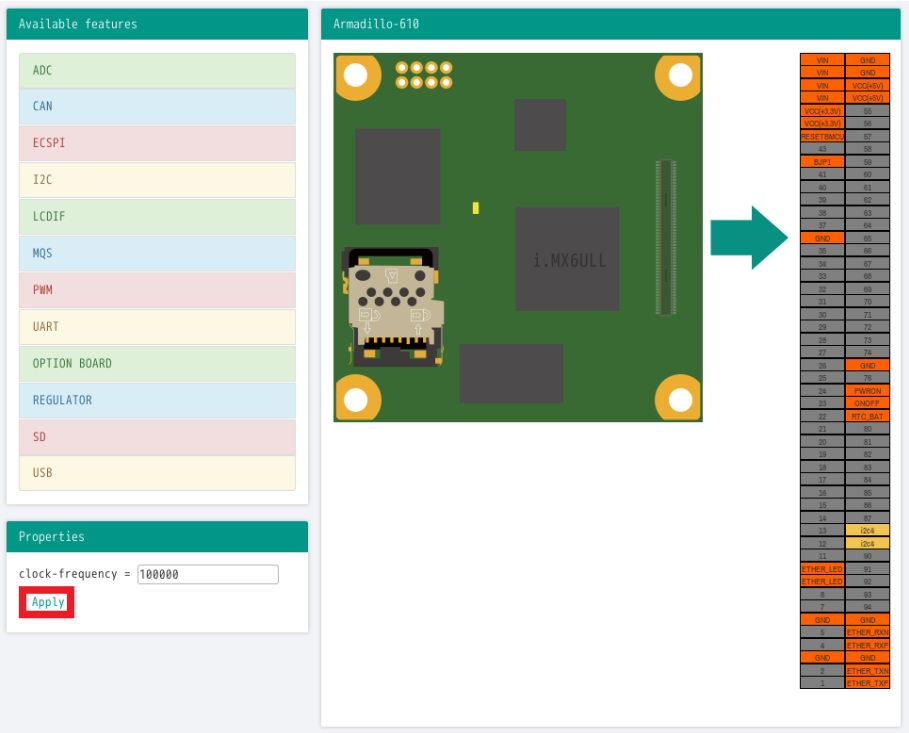


図 20.9 プロパティの保存

20.3.3.4. 機能の削除

全ての機能を削除する場合は、画面右上の「Reset configuration」をクリックします。機能ごとに削除する場合は、画面右側の「Armadillo-610」のピンを右クリックして「Remove」をクリックします。

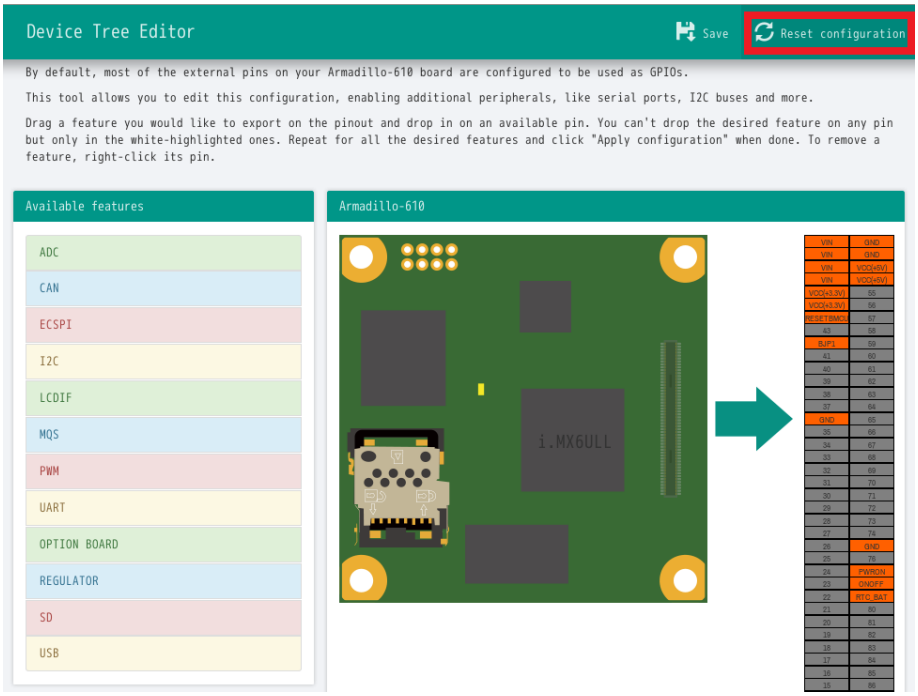


図 20.10 全ての機能の削除

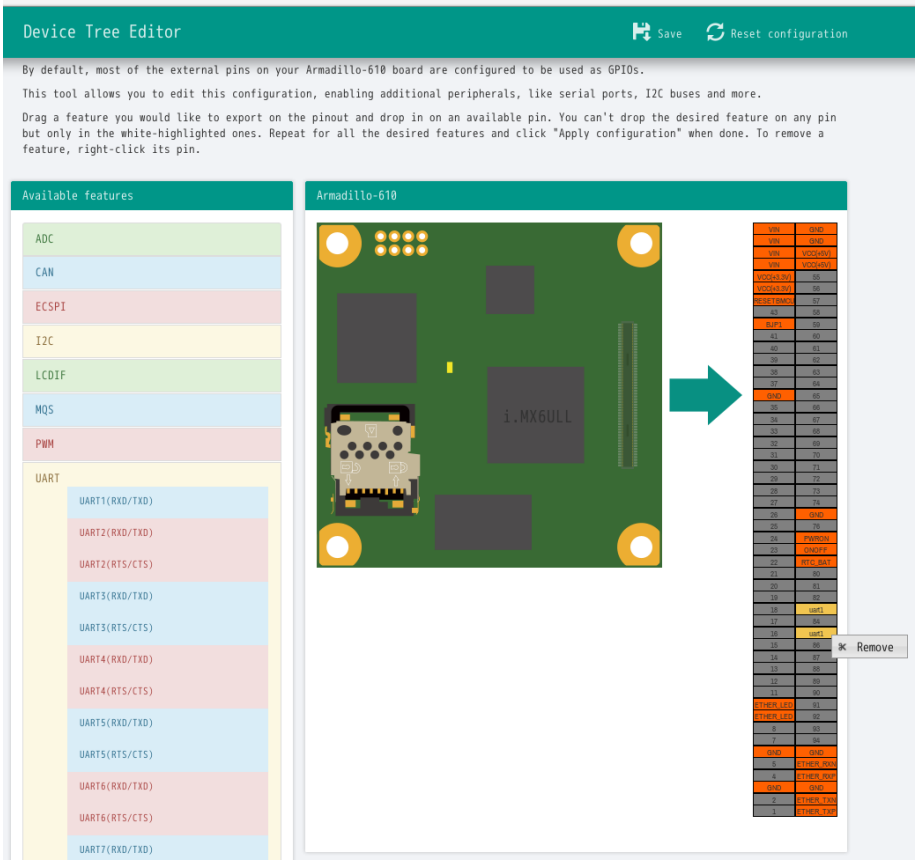


図 20.11 UART1(RXD/TXD)の削除

20.3.3.5. DTS/DTB の生成

DTS および DTB を生成するには、画面右上の「Save」をクリックします。

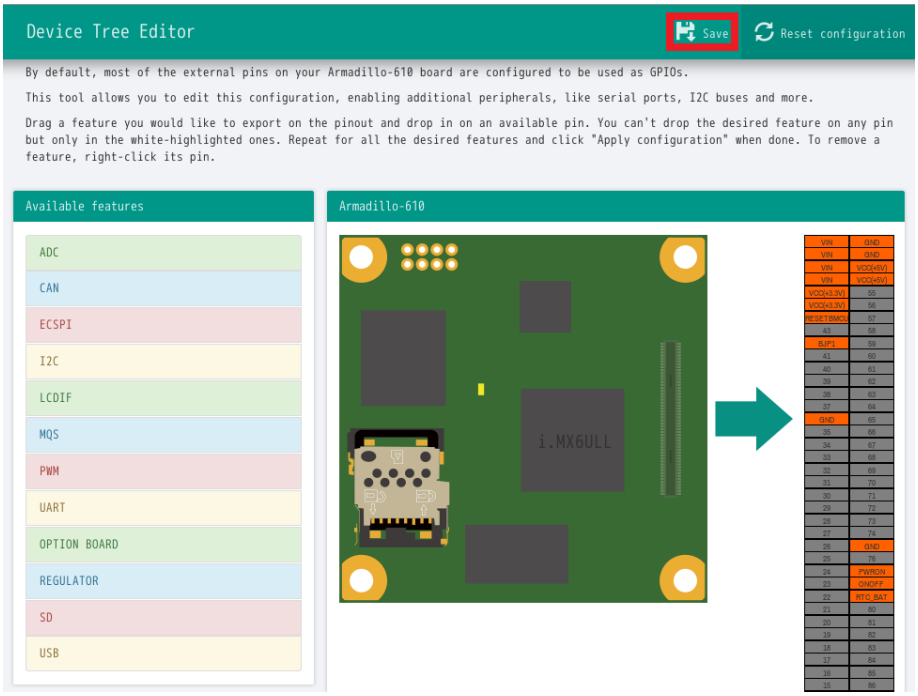


図 20.12 DTS/DTB の生成

「Device tree built!」と表示されると、DTS および DTB の生成は完了です。

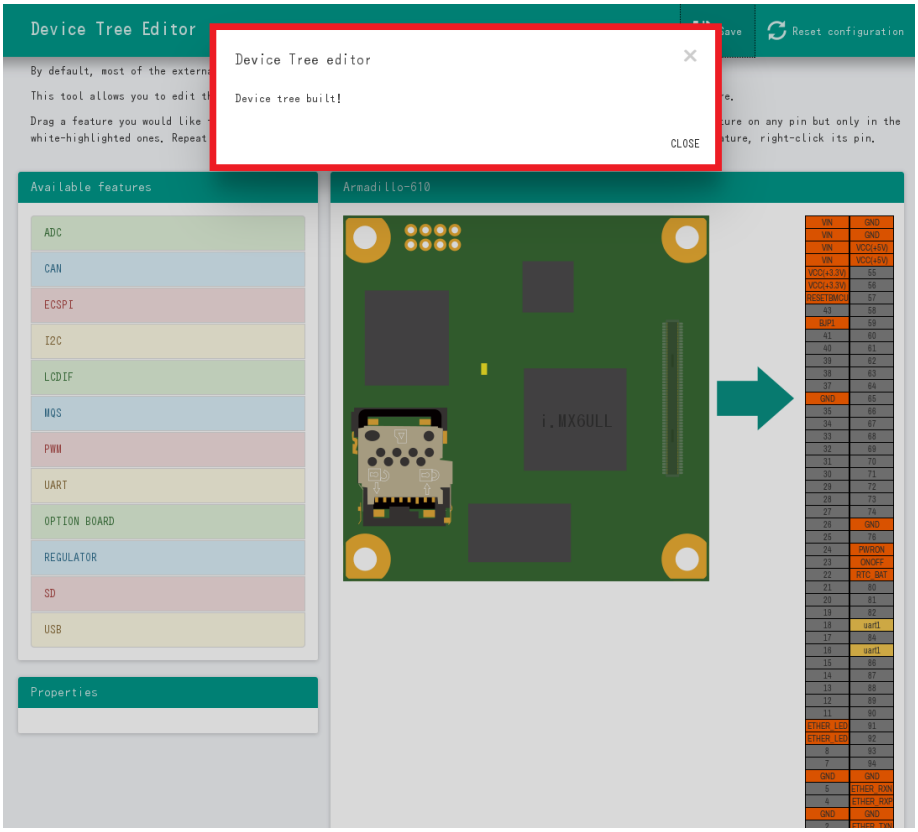


図 20.13 DTS/DTB の生成完了

ビルドが終了すると、arch/arm/boot/dts/以下に DTS/DTB が作成されています。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-expansion-interface.dtsi
arch/arm/boot/dts/armadillo-610-expansion-interface.dtsi
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-610-at-dtweb.dtb
arch/arm/boot/dts/armadillo-610-at-dtweb.dtb
```

20.4. ルートファイルシステムへの書き込みと電源断からの保護機能

Armadillo-610 のルートファイルシステムは、標準で eMMC に配置されます。Linux が稼働している間は、ログや、設定ファイル、各種アプリケーションによるファイルへの書き込みが発生します。もし、停電等で終了処理を実行できずに電源を遮断した場合は、RAM 上に残ったキャッシュが eMMC に書き込まれずに、ファイルシステムの破綻やファイルの内容が古いままになる状況が発生します。

また、eMMC 内部の NAND Flash Memory には消去回数に上限があるため、書き込み回数を制限することを検討する必要がある場合もあります。

そこで、Armadillo-610 では、overlayfs を利用して、eMMC への書き込み保護を行う機能を提供しています。

20.4.1. 保護機能の使用方法

eMMC への書き込み保護を行うには、kernel の起動オプションに"overlay=50%"("=50%"は省略可、"overlay"のみ書くと RAM を 256MByte 使用)というパラメータを追加するだけです。

パラメータを追加すると、debian の起動前に initramfs よってルートファイルシステムが upper=RAM ディスク(tmpfs)、lower=eMMC(ext4)とした overlayfs に切り替えられて、Debian が起動します。

overlayfs の機能によって、起動後のルートファイルシステムに対する差分は、全て RAM ディスク(/overlay/ramdisk にマウント)に記録されるようになります。そのため、起動後の情報は保存されませんが、電源を遮断した場合でも、eMMC は起動前と変わらない状態のまま維持されています。

kernel の起動オプションの指定を行うには Armadillo-610 を保守モードで起動し、次のようにコマンドを実行してください。

```
=> setenv optargs overlay
=> saveenv
```

また、オプションの指定を解除するには次のようにコマンドを実行してください。

```
=> setenv optargs
=> saveenv
```

20.4.2. 保護機能を使用する上での注意事項

overlayfs は差分を ファイル単位で管理するため、予想以上に RAM ディスクを消費する場合があります。 単に、新しいファイルやディレクトリを作れば、その分 RAM ディスクが消費されるのは想像に難くないと思います。

しかし、「lower=eMMC に既に存在していたファイルの書き換え」をする場合は、upper=RAM ディスク に対象のファイル全体をコピーして書き換え」ます。

具体的に、問題になりそうな例を紹介します。 例えば、sqlite は DB 毎に 1 つのファイルでデータ格納します。ここで、1GB の DB を作って eMMC に保存した後、 overlayfs による保護を有効にして起動した後に、たった 10 バイトのレコードを 追加しただけで RAM ディスクは 1GB + 10 バイト消費されます。実際には、 Armadillo に 1GB も RAM は無いので、追記を開始した時点で RAM ディスクが不足します。

overlayfs による、eMMC への書き込み保護を行う場合、 必ず実際の運用状態でのテストを行い、RAM ディスクが不足しないか確認してください。 動作中に書き込むファイルを必要最小限に留めると共に、追記を行う大きなファイルを作らない実装の検討を行ってください。

Armadillo-610 の eMMC の記録方式は出荷時に SLC に設定しており、MLC 方式の eMMC よりも消去回数の上限が高くなっています。そのため、開発するシステムの構成によっては eMMC への書き込み保護機能を必要としない可能性があります。

eMMC への書き込み保護機能を有効にすると、eMMC を安全に使用できるというメリットがありますが、その分、使用できる RAM サイズが減る、システム構成が複雑になる、デメリットもあります。 開発・運用したいシステムの構成、eMMC への書き込み保護機能のメリット・デメリットを十分に考慮・評価したうえで、保護機能を使用する、しないの判断を行ってください。

20.5. eMMC の GPP(General Purpose Partition) を利用する

GPP に squashfs イメージを書き込み、Armadillo の起動時に自動的にマウントする方法を紹介します。

20.5.1. squashfs イメージを作成する

この作業は ATDE7 上で行います。

squashfs-tools パッケージに含まれている mksquashfs コマンドを使用して squashfs イメージを作成します。

```
[ATDE]$ mkdir sample
[ATDE]$ echo "complete mounting squashfs on eMMC(GPP)" > sample/README
[ATDE]$ mksquashfs sample squashfs.img
```

図 20.14 squashfs イメージの作成

20.5.2. squashfs イメージを書き込む

以降の作業は Armadillo 上で行います。

「20.5.1. squashfs イメージを作成する」で作成した squashfs イメージを、USB メモリ利用するなどして Armadillo-610 にコピーし、GPP に書き込みます。

ユーザー領域として使用可能な GPP は /dev/mmcblk0gp2 および /dev/mmcblk0gp3 です。

GPP への書き込みを行う際は、誤って /dev/mmcblk0gp0 や /dev/mmcblk0gp1 に書き込みを行わないよう、十分に注意してください。

```
[armadillo]# mount /dev/sda1 /mnt
[armadillo]# dd if=/mnt/squashfs.img of=/dev/mmcblk0gp2 conv=fsync
[armadillo]# umount /mnt
```

20.5.3. GPP への書き込みを制限する

GPP の全ブロックに対して Temporary Write Protection をかけることにより、GPP への書き込みを制限することができます。Temporary Write Protection は電源を切断しても解除されません。

Temporary Write Protection をかけるには、mmc-utils パッケージに含まれている mmc コマンドを使用します。

```
[armadillo]# apt-get install mmc-utils
```

図 20.15 mmc-utils のインストール

GPP の全ブロックに対して Temporary Write Protection をかけるには、次のようにコマンドを実行します。

```
[armadillo]# mmc writeprotect user get /dev/mmcblk0gp2 ❶
Write Protect Group size in blocks/bytes: 16384/8388608
Write Protect Groups 0-0 (Blocks 0-16383), No Write Protection
[armadillo]# mmc writeprotect user set temp 0 16384 /dev/mmcblk0gp2 ❷
```

図 20.16 eMMC の GPP に Temporary Write Protection をかける

❶ /dev/mmcblk0gp2 のブロック数を確認します。コマンドの出力を見ると /dev/mmcblk0gp2 が 16384 ブロックあることがわかります。

❷ /dev/mmcblk0gp2 の全ブロックに Temporary Write Protection をかけます。

Temporary Write Protection を解除するには、次のコマンド実行します。

```
[armadillo]# mmc writeprotect user set none 0 16384 /dev/mmcblk0gp2
```

20.5.4. 起動時に squashfs イメージをマウントされるようにする

/etc/fstab を変更し、起動時に squashfs イメージがマウントされるようにします。

```
[armadillo]# mkdir -p /opt/sample ❶
[armadillo]# vi /etc/fstab
:
:(省略)
:
/dev/mmcblk0gp2 /opt/sample squashfs defaults,nofail 0 0 ❷
```

❶ squashfs イメージをマウントするディレクトリを作成します

❷ 最終行にこの行を追加します。これで、/dev/mmcblk0gp2 が /opt/sample にマウントされるようになります。

Armadillo の再起動後、 /opt/sample/README の内容が正しければ完了です。

```
[armadillo]# reboot
:
: (省略)
:
Debian GNU/Linux 9 armadillo ttymxc0

armadillo login:
[armadillo]# ls /opt/sample
README
[armadillo]# cat /opt/sample/README
complete mounting squashfs on eMMC(GPP)
```

20.6. Armadillo-610 拡張ボードの SD インターフェースを利用する

Armadillo-610 拡張ボードの SD インターフェース(Armadillo-610 拡張ボード: CON1)を利用する方法を紹介します。

Armadillo-610 開発セットの標準状態では、SD インターフェース(Armadillo-610: CON1)が有効になっています。そのため、SD インターフェース(Armadillo-610 拡張ボード: CON1)を利用するためには Device Tree をカスタマイズする必要があります。Device Tree をカスタマイズする方法については、まず「20.3. Device Tree をカスタマイズする」 を参照してください。

1. コンソールの選択

   コンソールとして、UART1(RXD/TXD)をシリアルインターフェース(Armadillo-610 拡張ボード: CON3)に該当する 83,85 ピンに割り当てます。

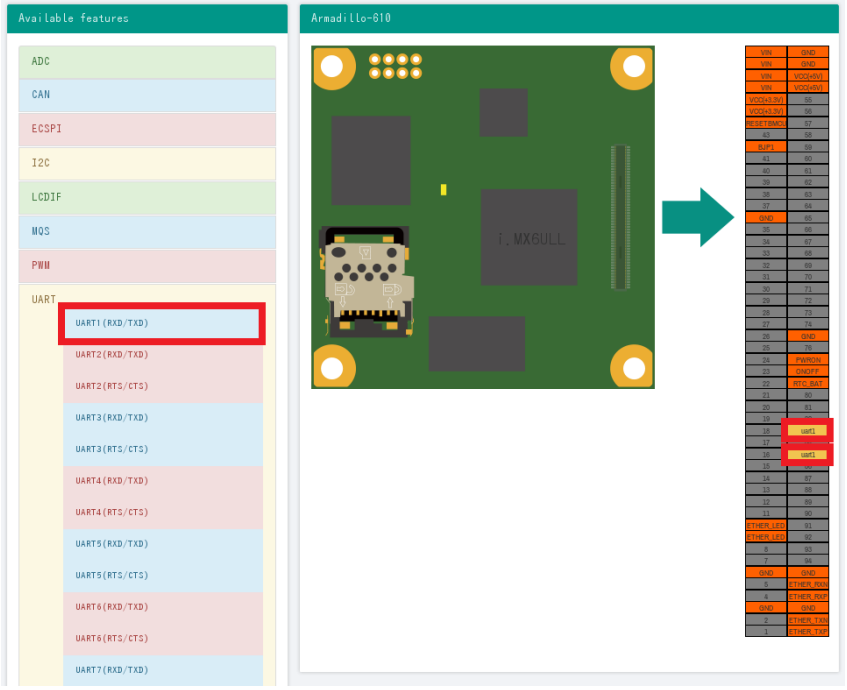


図 20.17 コンソールの選択

2. uSDHC2 の選択

   USDHC2 を SD インターフェース(Armadillo-610 拡張ボード: CON1)に該当する 70〜74,76,91,92 ピンに割り当てます。

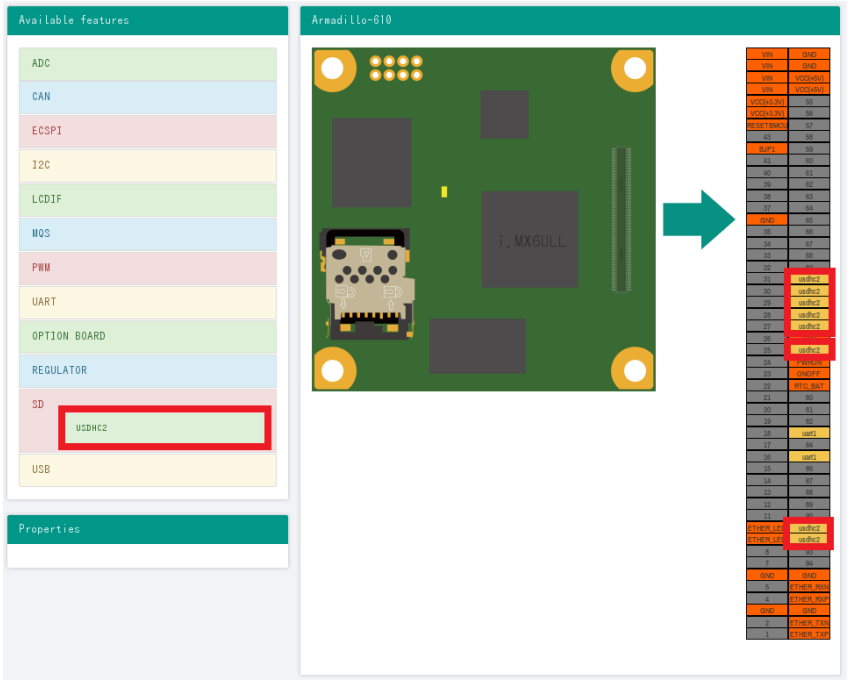


図 20.18 uSDHC2 の選択

3. レギュレータの選択

   レギュレータを Armadillo-610 拡張ボードに搭載されたパワースイッチに該当する 86 ピンに割り当てます。

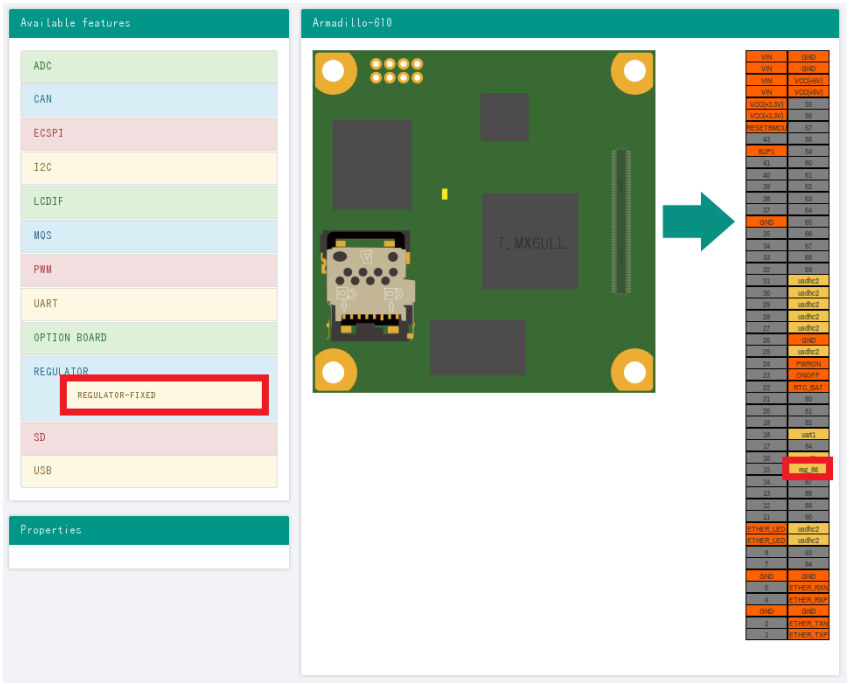


図 20.19 レギュレータの選択

4. レギュレータの割り当て

uSDHC2 のレギュレータを割り当てます。「vmcc-supply」に reg_86 と入力した後、「Apply」をクリックします。

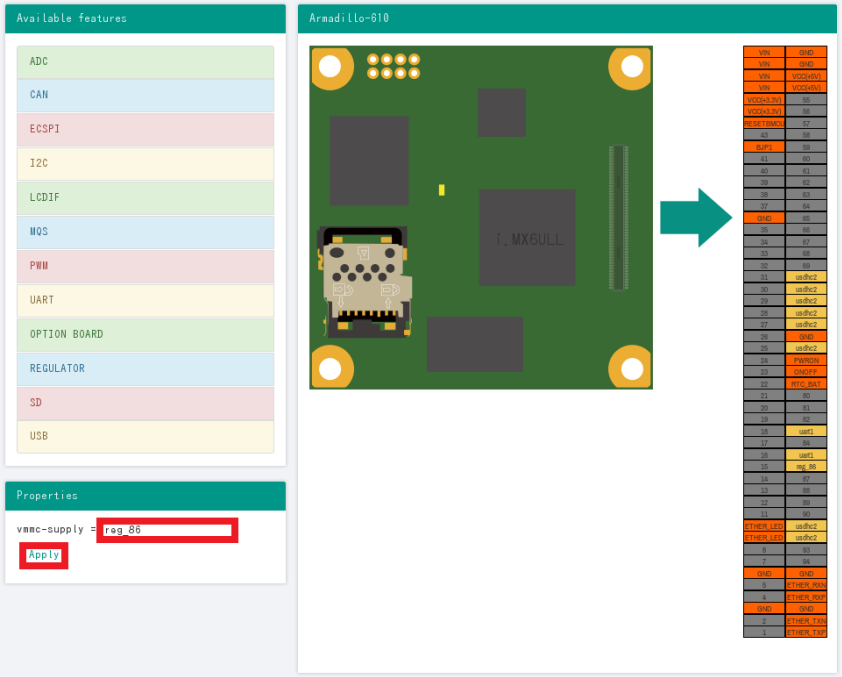


図 20.20 レギュレータの割り当て

生成した DTB(arch/arm/boot/dts/armadillo-610-at-dtweb.dtb)を Armadillo-610 に書き込むと、Armadillo-610 拡張ボードの SD インターフェースが利用可能になります。

20.7. Armadillo-610 拡張ボードの LCD インターフェースを利用する

Armadillo-610 拡張ボードの LCD インターフェース(Armadillo-610 拡張ボード: CON11)を利用する方法を紹介します。ここでは例として「18.3. LCD オプションセット(7 インチタッチパネル WVGA 液晶)」を利用します。

「11.2.3. DTB の書き換え」を参照して、DTB を armadillo-610-extboard-eva-lcd-v4.14-at[version].dtb に書き換えてください。

Device Tree Blob(LCD 用)は、「アットマークテクノ Armadillo サイト」[https://armadillo.atmark-techno.com/]からダウンロード可能です。

再起動後、LCD が利用可能になります。

「18.4. Armadillo-400 シリーズ LCD オプションセット」を利用する場合は、「20.3. Device Tree をカスタマイズする」を参照して DTB を作成してください。

20.8. wxWidgets を利用して GUI アプリケーションを開発する

wxWidgets は C++で開発されているクロスプラットフォームの GUI ツールキットです。Python、Perl、Ruby などのスクリプト言語のラッパーも用意されており、これらを使って開発することも可能です。

ここでは wxWidgets で開発したサンプルアプリケーションと、Python ラッパーである wxPython で開発したサンプルアプリケーションを紹介します。

事前に「20.7. Armadillo-610 拡張ボードの LCD インターフェースを利用する」を参照して LCD を利用可能にする必要があります。

GUI アプリケーションの動作を確認するためには、LCD オプションセット(7 インチタッチパネル WVGA 液晶)が必要です。

20.8.1. wxWidgets を直接利用して GUI アプリケーションを開発する

ラッパーを利用せず、wxWidgets を直接利用して開発したサンプルアプリケーションを紹介します。

20.8.1.1. wxWidgets のインストール

開発に必要なパッケージをインストールします。

```
[armadillo ~]# apt-get update
[armadillo ~]# apt-get install build-essential xorg libwxgtk3.0-dev
```

20.8.1.2. サンプルアプリケーションのダウンロード

次に示すコマンドでサンプルアプリケーションをダウンロードします。

```
[armadillo ~]# wget https://download.atmark-techno.com/sample/a600-wxwidgets-howto/wxwidgets_led-
v1.1.tar.gz
[armadillo ~]# tar zxf wxwidgets_led.tar.gz
```

20.8.1.3. サンプルアプリケーションのビルドと起動

ダウンロードしたサンプルアプリケーションのディレクトリへ移動しビルドを実行します。

```
[armadillo ~]# cd wxwidgets_led
[armadillo ~/wxwidgets_led]# make
```

```
[armadillo ~/wxwidgets_led]# ls
Makefile  led.h               led_controller.h  main_frame.h
led       led.o               led_controller.o  main_frame.o
led.cpp   led_controller.cpp  main_frame.cpp    resources
```

次に示すコマンドでサンプルアプリケーションを起動します。LCD オプションセット(7 インチタッチパネル WVGA 液晶)が接続済みであるとします。

```
[armadillo ~/wxwidgets_led]# export DISPLAY=:0
[armadillo ~/wxwidgets_led]# X -retro -r &
[armadillo ~/wxwidgets_led]# ./led
```

サンプルアプリケーション上の YELLOW ボタンを押すと、Armadillo-610 上の LED5 が点灯・消灯します。

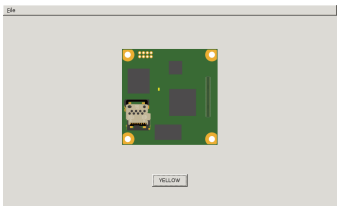


図 20.21 wxWidgets サンプルアプリケーション

20.8.1.4. ソースファイル構成

```
[armadillo ~/wxwidgets_led]# ls
Makefile  led.h               led_controller.h  main_frame.h
led.cpp ❶   led_controller.cpp ❷  main_frame.cpp ❸   resources
```

❶ アプリケーション本体です。

❷ Armadillo-610 上の LED を操作するための処理を実装しているソースです。

❸ UI に関する処理を実装しているソースです。

wxWidgets には公式に多くのサンプルアプリケーションがあり、それらを参考にしながら開発することもできます。サンプルアプリケーションを取得するには、次のコマンドを実行します。

```
[armadillo ~]# apt-get install wx3.0-examples
[armadillo ~]# cd /usr/share/doc/wx3.0-examples/examples
[armadillo /usr/share/doc/wx3.0-examples/examples]# ./
unpack_examples.sh samples/ ~/wx3.0-examples
[armadillo /usr/share/doc/wx3.0-examples/examples]# cd ~/wx3.0-examples/
samples
[armadillo ~/wx3.0-examples/samples]#
```

上記のディレクトリに多種多様なサンプルアプリケーションが入っており、ビルド後に起動して動作を確認することができます。

20.8.2. wxPython を利用して GUI アプリケーションを開発する

先に説明したとおり wxWidgets には Python 向けラッパーがあり、それが wxPython です。スクリプト言語という特性を活かして、wxWidgets を直接利用するよりも素早く GUI アプリケーションの開発ができます。さらに、 C 言語と組み合わせることにより、ハードウェアの制御も可能となります。

ここでは wxPython で開発したサンプルアプリケーションを紹介します。(アプリケーションの内容は「20.8.1.3. サンプルアプリケーションのビルドと起動」で説明したものと同じです。)

20.8.2.1. Python と wxPython のインストール

開発に必要なパッケージをインストールします。

```
[armadillo ~]# apt-get update
[armadillo ~]# apt-get install build-essential xorg python-dev python-wxgtk3.0-dev
```

20.8.2.2. サンプルアプリケーションのダウンロード

次に示すコマンドでサンプルアプリケーションをダウンロードします。

```
[armadillo ~]# wget https://download.atmark-techno.com/sample/a600-wxwidgets-howto/wxpython_led-
v1.1.tar.gz
[armadillo ~]# tar zxf wxpython_led.tar.gz
```

20.8.2.3. サンプルアプリケーションのビルドと起動

ダウンロードしたサンプルアプリケーションのディレクトリへ移動し、必要なモジュールをビルドします。ここでビルドするモジュールとは C 言語で記述された Python からハードウェアを制御するためのプログラムです。

```
[armadillo ~]# cd wxpython_led
[armadillo ~/wxpython_led]# make
[armadillo ~/wxpython_led]# ls
Makefile  led.o   led_wrapper.c  ledmodule.so
led.c     led.py  led_wrapper.o  resources
```

次に示すコマンドでサンプルアプリケーションを起動します。LCD オプションセット(7 インチタッチパネル WVGA 液晶)が接続済みであるとします。

```
[armadillo ~/wxpython_led]# export DISPLAY=:0
[armadillo ~/wxpython_led]# X -retro -r &
[armadillo ~/wxpython_led]# python led.py
```

サンプルアプリケーション上の YELLOW ボタンを押すと、Armadillo-610 上の LED5 が点灯・消灯します。

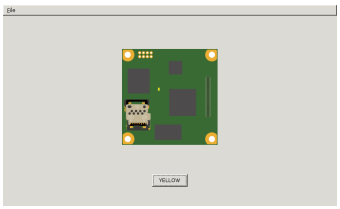


図 20.22 wxPython サンプルアプリケーション

20.8.2.4. ソースファイル構成

ハードウェア(LED5)を制御する部分を C 言語で実装しているため、C 言語のソースファイルも含んでいます。

```
[armadillo ~/wxpython_led]# ls
Makefile led.c ❶  led.py ❷  led_wrapper.c ❸ resources
```

❶ Armadillo-610 上の LED を操作するための処理を実装している C 言語プログラムです

❷ Python で記述されたアプリケーション本体です

❸ led.c で実装している処理を Python 側から呼び出すためのブリッジとなる処理を実装している C 言語プログラムです

wxPython パッケージには多くのサンプルアプリケーションが含まれており、それらを参考にしながら開発することもできます。サンプルアプリケーションを取得するには、次のコマンドを実行します。

```
[armadillo ~]# apt-get source python-wxgtk3.0
[armadillo ~]# cd wxpython3.0-3.0.2.0+dfsg/wxPython/
[armadillo ~/wxpython3.0-3.0.2.0+dfsg/wxPython]#
```

上記のディレクトリ内にある demo 、 samples ディレクトリに多種多様なサンプルアプリケーションが入っており、起動して動作を確認することができます。

21. 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 Armadillo 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 Armadillo サイト」をご覧ください。

アットマークテクノ Armadillo サイト

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

21.1. 購入製品登録

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

購入製品登録の詳しい手順は以下の URL をご参照ください。

Armadillo-610 購入製品登録

https://armadillo.atmark-techno.com/armadillo-610/register

付録 A eFuse

Armadillo-610 で採用している CPU (i.MX6ULL) には、一度しか書き込むことのできない eFuse が搭載されています。 eFuse には、 CPU がブートする時の設定や MAC アドレスなどが書かれます。Armadillo-610 は組み込み機器を作り込むエンジニアを対象にした製品ですので、eFuse もユーザーに開放し、細かな制御を可能にしています。しかし eFuse はその性質上、一度書き間違うと直すことができません。十分に注意してください。

eFUSE は一度書き込むと元に戻すことができません。eFUSE の設定によっては Armadillo-610 が正常に動作しなくなる可能性がありますので、書き込みを行う際には細心の注意を払うようお願いいたします。eFUSE の設定によって異常が起こった場合は保証対象外となります。

MAC アドレスは Armadillo-610 の出荷時に書き込まれているので、新たに書き込む必要はありません。この章では U-Boot を使って eFuse の書き換えを行い、ブートモードを制御する方法を説明します。

eFuse を変更する場合は、必ず「i.MX 6ULL Applications Processor Reference Manual [https://www.nxp.com/docs/en/reference-manual/IMX6ULLRM.pdf]」を参照してください。重要な章は、以下の 4 つです。

- Chapter 5: Fusemap
- Chatper 8: System Boot
- Chapter 37: On-Chip OTP Controller
- Chapter 58: Ultra Secured Digital Host Controller

以降、本章では i.MX 6ULL Applications Processor Reference Manual を「リファレンスマニュアル」と呼びます。

章番号や章タイトルは、i.MX 6ULL Applications Processor Reference Manual Rev. 1, 11/2017 現在の情報です。異るリビジョンのリファレンスマニュアルでは、章番号およびタイトルが異なる場合があります。

A.1. ブートモード

i.MX6ULL にはブートモードを決める BOOT_MODE0 と BOOT_MODE1 というピンがあります。Armadillo-610 では、BOOT_MODE0 は 0 、BOOT_MODE1 は 1 となるよう回路が設計されており、ブートモードは必ず **Internal Boot** モードとなります。

A.1.1. Internal Boot モード

Internal Boot モードでは、 on-chip boot ROM に書き込まれているコードが実行し、ブート可能なデバイスを検索します。リファレンスマニュアル「8.5 Boot devices (internal boot)」に、i.MX6ULL

がブートできるデバイスの一覧が記載されています。Armadillo-610 では、そのうちオンボード eMMC と microSD カードに対応しています。

Internal Boot モードでは、GPIO によって eFuse の設定を上書き (override) できるようになっています。この機能は eFuse の BT_FUSE_SEL が 0 の場合のみ有効となります。eFuse の設定とは異なり何度も再設定できる点では便利ですが、override に対応したピンには i.MX6ULL の電源投入時に決まった信号を入力しておかなければいけないため、ハードウェア設計上は不便になります。

Armadillo-610 では、GPIO による override を利用することで、仕様が確定していない段階ではブートデバイスを自由に何度も切り替えることを可能にしつつ、BT_FUSE_SEL を 1 にして GPIO による override を無効化することで、仕様が確定した段階では自由なハードウェア設計が可能になるよう配慮しています。また、GPIO による override を無効化することで、フィールドに出した製品が悪意ある人によって意図していないブートをし、被害が出ることを防ぐことができます。(もちろん、ブート後に root アカウントを乗っ取られるような作りでは、意味がありませんが…)

A.2. ブートデバイス

Internal Boot モードでは、GPIO によって eFuse の設定を上書き (override) できるようになってると紹介しましたが、Armadillo-610 では、Armadillo-610 拡張ボードの JP1 はまさにこの機能を使っています。 JP1 は BJP1(Armadillo-610 CON2_42 ピン) に接続されており、 LCD1_DATA05 と LCD1_DATA11 の制御をしていますが、これらのピンはそれぞれ BOOT_CFG1[5] と BOOT_CFG2[3] を override しています。「8.3.2 GPIO boot overrides」の 表「8-3. GPIO override contact assignments」を確認してください。

ややこしい事に、この BOOT_CFG で始まる eFUSE は、リファレンスマニュアルの中では eFuse のアドレスでも表記されています。 BOOT_CFG1 は eFuse のアドレスで言うと 0x450 の下位 8 bit つまり 0x450[7:0] であり、 BOOT_CFG2 は上位 8 bit つまり 0x450[15:8] にあたります。これは「5.1 Boot Fusemap」の表「5-5. SD/eSD Boot Fusemap」または表「5-6. MMC/eMMC Boot Fusemap」を確認することでわかります。

さらにややこしい事に、eFuse を書き込む場合にはこれら全ての値が使えず、On-Chip OTP Controller の bank と word の値が必要になります。これらの値は リファレンスマニュアルの「On-Chip OTP Controller」を参照してください。後で出てきますが Boot From Fuses で使用する BT_FUSE_SEL という eFuse のように GPIO による override ができないものもあります。

表 A.1 GPIO override と eFuse

信号名	eFuse 名	eFuse アドレス	OCOTP 名	Bank	Word
LCD1_DATA05	BOOT_CFG1[5]	0x450[5]	OCOTP_CFG4	0	5
LCD1_DATA11	BOOT_CFG2[3]	0x450[11]	OCOTP_CFG4	0	5
N/A	BT_FUSE_SEL	0x460[4]	OCOTP_CFG5	0	6

Armadillo-610 では SD カード または eMMC からのブートになるので、ブートデバイスを選択する eFuse BOOT_CFG1[7:4] は、010x または 011x になります。

リファレンスマニュアル「8.5.3.1 Expansion device eFUSE configuration」には、さらに詳しく SD/MMC デバイスの設定について記載されています。テーブル「8-15. USDHC boot eFUSE descriptions」によれば、eFuse の 0x450[7:6] が 01 の場合に SD/MMC デバイスからブートすることを決めています。さらに 0x450[5] が 0 なら SD が、 0x450[5] が 1 なら MMC が選択されます。つまり、4 から 7 bit までの間で 5 bit 目だけが MMC か SD かを決めています。 BOOT_CFG1[5] が 0 の場合はコントローラーは SD デバイスが繋がっている前提で、 BOOT_CFG1[5] が 1 の場合は MMC デバイスが繋っている前提で動作します。

i.MX6ULL には、SD/MMC のコントローラーである uSDHC が 2 つ搭載されています。Armadillo-610 では、eMMC が uSDHC1 に、 microSD カードが uSDHC2 に接続されています。ブー

ト時にどちらのコントローラーからブートするかを決めている eFuse が 0x450[12:11] です。0x450[12:11] が 00 であれば uSDHC1 つまりオンボード eMMC から、01 であれば uSDHC2 つまり microSD カードからブートします。言い換えると Armadillo-610 でオンボード eMMC からブートしたい場合は、0x450[5] を 1 に、 0x450[12:11] を 00 にします。逆に microSD カードから起動したい場合は 0x450[5] を 0 に、0x450[12:11] を 01 にします。

表 A.2 ブートデバイスと eFuse

ブートデバイス	eFuse 0x450[5]	0x450[12:11]
オンボード eMMC	1	00
microSD カード	0	01

A.3. eFuse の書き換え

Armadillo-610 では、U-Boot のコマンドによって eFuse の書き換えをサポートしています。 U-Boot については 「9. ブートローダー (U-Boot) 仕様」 を参照してください。

eFuse の書き換えは、 fuse コマンドを使います。

U-Boot の fuse コマンドのソースコードは、以下の 2 つです。

- cmd/fuse.c
- drivers/misc/mxc_ocotp.c

```
=> help fuse
fuse - Fuse sub-system

Usage:
fuse read <bank> <word> [<cnt>] - read 1 or 'cnt' fuse words,
    starting at 'word'
fuse sense <bank> <word> [<cnt>] - sense 1 or 'cnt' fuse words,
    starting at 'word'
fuse prog [-y] <bank> <word> <hexval> [<hexval>...] - program 1 or
    several fuse words, starting at 'word' (PERMANENT)
fuse override <bank> <word> <hexval> [<hexval>...] - override 1 or
    several fuse words, starting at 'word'
=>
```

fuse read　eFuse の値を Shadow Register から読み出します。i.MX6ULL の eFuse は、すべて Shadow Register を持ち、起動時に eFuse から Shadow Register に値がコピーされます。詳しくはリファレンスマニュアル「37.3.1.1 Shadow Register Reload」を確認してください。

fuse sense　eFuse の値を eFuse から読み出します

fuse prog　eFuse の値を書き換えます

fuse コマンドは、 bank 、 word 、cnt 、 hexval を引数に取ります。

bank　eFuse のバンク番号

word　　eFuse のワード番号

cnt　　eFuse を読み出す個数

hexval　　書き込む値

A.4. eFuse の設定によるブートデバイスの選択

A.4.1. BT_FUSE_SEL

eFuse の設定によるブートデバイスの選択を可能にするには、 eFuse に書き込んだ値が正しいことを i.MX6ULL に教える必要があります。そのための eFuse が BT_FUSE_SEL (0x460[4]) です。Armadillo-610 では、このビットが 1 であれば、GPIO による override が無効になり eFuse の設定にしたがってブートデバイスが選択されるようになります。

A.4.2. eMMC からのブートに固定

オンボード eMMC からだけブートさせたい場合は、ブートデバイスの種類で MMC と、コントローラーで uSDHC1 を選択することで可能です。忘れずに BT_FUSE_SEL を 1 にします。

オンボード eMMC のスペックは、以下の通りです。リファレンスマニュアル 8.5.3 Expansion device および 表「5-6. MMC/eMMC Boot Fusemap」を確認してください。「可変」列が「不」となっている値は、変更しないでください。例えば、オンボード eMMC は 1.8 V に対応していません。 bit 9 の SD Voltage Selection で 1 の 1.8 V では動作しません。

表 A.3 オンボード eMMC のスペック

名前	Bit	eFuse	値	bit 列	可変
BOOT_CFG2	[15:13]	Bus Width	8 bit	010	不
	[12:11]	Port Select	uSDHC1	00	不
	[10]	Boot Frequencies	500 / 400 MHz	00	可
	[9]	SD Voltage Selection	3.3 V	0	不
	[8]	-	-	0	-
BOOT_CFG1	[7:5]	eMMC	-	011	不
	[4]	Fast Boot	Regular	0	可
	[3]	SD/MMC Speed	High	0	不
	[2]	Fast Boot Acknowledge Disable	Enabled	0	可
	[1]	SD Power Cycle Enable	Enabled	1	可
	[0]	SD Loopback Clock Source Sel	SD Pad	0	不

値を見易いように、 BOOT_CFG2 を上にしています。 BOOT_CFG1 と BOOT_CFG2 は、OCOTP_CFG4 にマップされており Bank 0 Word 5 です。つまり 010000000 01100010 の 16 bit (0x4062) を Bank 0 Word 5 に書き込めば良いことが分ります。 BOOT_CFG3 と BOOT_CFG4 はここでは無視します。

BT_FUSE_SEL は Bank 0 Word 6 の 4 bit 目になるので 0x10 を書き込みます。

```
=> fuse read 0 5
Reading bank 0:
```

```
Word 0x00000005: 00000000
=> fuse prog 0 5 0x4060
Programming bank 0 word 0x00000005 to 0x00004060...
Warning: Programming fuses is an irreversible operation!
         This may brick your system.
         Use this command only if you are sure of what you are doing!

Really perform this fuse programming? <y/N>
y
=> fuse read 0 6
Reading bank 0:

Word 0x00000006: 00000000
=> fuse prog -y 0 6 0x10
Programming bank 0 word 0x00000006 to 0x00000010...
=> fuse read 0 6
Reading bank 0:

Word 0x00000006: 00000010

（電源入れなおしても、SD からブートしない）
```

fuse prog にオプション -y を付けると 「 Really perform this fuse programming? <y/N> 」と聞かれません。

これで eMMC からしか起動しない Armadillo-610 ができあがりました。

eMMC からしか起動しないので、あやまって eMMC に書き込まれている U-Boot を消してしまうと、二度と起動しないようになります。注意してください。

eMMC Fast Boot 機能を使う場合や Power Cycle を Enable にする場合は、当該ビットを 1 に変更してください。

同じ要領で、SD からだけしかブートしないようにすることも可能です。しかし eFuse によるブートデバイスの固定は、意図しないブートを防ぐことが目的です。 Armadillo-610 で microSD からのブートに固定することは可能ですが、別の microSD カードを挿入されてしまうと、その別の microSD カードからブートしてしまうので目的を達成できません。理解してお使いください。

A.4.3. eFuse のロック

書き込んだ eFuse の値を変更されてしまっては、Boot From Fuse モードにしている意味がありません。i.MX6ULL では eFuse を変更できなくするビットも用意されています。

リファレンスマニュアル「5.3 Fusemap Descriptions Table」を確認してください。

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2020/02/27	・初版発行
1.0.1	2020/03/24	・他の Armadillo 製品に共通する記述については Armadillo で表記を統一 ・他の Armadillo-600 シリーズの製品に共通する記述については Armadillo-600 シリーズで表記を統一 ・「2.4. 電波障害について」の VCCI に関する記述を修正 ・誤記修正
1.0.2	2020/05/14	・ユーザーズサイトへの誘導を Armadillo サイトに変更 ・B-B コネクタの嵌合/抜去時の注意を追加 ・誤記修正
1.1.0	2020/09/16	・「図 18.2. Armadillo-610 拡張ボードのブロック図」を修正 ・「9.1. U-Boot の起動モード」にユーザースイッチによる起動モードの変更について追加
1.2.0	2020/12/24	・Debian GNU/Linux 9 (stretch) 向けのドキュメントであることを明記 ・at-debian-builder のバージョンについての注記を追加
1.3.0	2021/01/28	・「21.1. 購入製品登録」を追加
1.4.0	2021/06/15	・「6.6. RTC」に、systemd-timesyncd.service に関する情報を追加 ・「表 3.5. eMMC メモリマップ」で、パーティション番号のかわりにディスクデバイスを表記するよう変更
1.5.0	2022/01/27	・「16.1. CON1(SD インターフェース)」に、「16.1.1. microSD カードの挿抜方法」 を追加 ・「20.3. Device Tree をカスタマイズする」に、「20.3.3.2. 信号名の確認」 を追加
1.5.1	2022/02/24	・HTML 版の検索機能を修正
1.5.2	2022/04/27	・電波障害についての文章を修正 ・AC アダプタの極性マークを変更

Armadillo-610 製品マニュアル
Version 1.5.2
2022/04/26