Armadillo-640 製品マニュアル

A6400-U00Z
A6400-D00Z
A6400-B00Z

Version 1.12.2
2020/01/29

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

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

Armadillo-640 製品マニュアル

株式会社アットマークテクノ

Version 1.12.2
2020/01/29

目次

図目次

表目次

1. はじめに

このたびは Armadillo-640 をご利用いただき、ありがとうございます。

Armadillo-640 は、NXP Semiconductors 製アプリケーションプロセッサ「i.MX6ULL」を採用し、標準インターフェースとして、USB2.0 ホストポートやイーサネットポート、microSD を搭載した小型シングルボードコンピューターです。

Armadillo-640 は、Armadillo-440 の形状を継承しつつ、処理能力や搭載メモリなどの基本機能を向上したモデルです。また、標準インターフェース以外に多くの拡張インターフェースを搭載しており、お客様の用途に合わせた柔軟な機能拡張に対応することができます。

Armadillo-640 では、Debian GNU/Linux がプリインストールされているため、オープンソースソフトウェアを含む多くのソフトウェア資産を活用し、自由にオリジナルのアプリケーションを開発することができます。開発言語としては、C/C++言語だけでなく、Oracle Java や Ruby なども利用することができるため、PC ライクな開発が可能です。

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

1.1. 本書で扱うこと扱わないこと

1.1.1. 扱うこと

本書では、Armadillo-640 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 サイトやユーザーズサイトをはじめ、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-640 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-640 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-640 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-640 は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

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

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

1.3. ユーザー限定コンテンツ

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

- 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「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-640 ドキュメント・ダウンロード**
>
> http://armadillo.atmark-techno.com/armadillo-640/downloads

1.5. 本書の構成

本書には、Armadillo-640 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-640 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」

- **拡張基板の開発や、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」

- **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。

役に立つ情報を記載します。

1.7. 謝辞

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

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

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

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

2.2. 取扱い上の注意事項

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

破損しやすい箇所
microSD コネクタ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(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。

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

2.5. 無線モジュールの安全規制について

Armadillo-600 シリーズ WLAN オプションモジュール、Armadillo-600 シリーズ Thread オプションモジュールに搭載している無線モジュールは、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

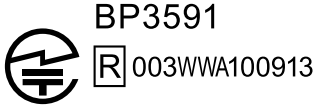


図 2.1 Armadillo-WLAN モジュール(AWL13) 認証マーク

表 2.2 Thread 通信モジュール(EYSKBNZWB) 適合証明情報

項目	内容
型式又は名称	EYSKBN
電波法に基づく工事設計認証における認証番号	001-A14398

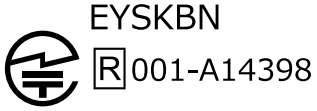


図 2.2 Thread 通信モジュール(EYSKBNZWB) 認証マーク

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-640 とは

Armadillo-600 シリーズは、フィールド向けの機器・端末のプラットフォームとして豊富な採用実績を持つ小型・省電力で Linux 採用の組み込み CPU ボード「Armadillo-400 シリーズ」の思想を継承しつつ、処理能力と搭載メモリをともに大幅にグレードアップさせた、次世代の Linux 組み込みプラットフォームです。高性能ながら省電力性能を向上、さらに耐環境性を追求するなど、量産時の使いやすさを重視した堅実な設計が特長です。

図 3.1 Armadillo-640 とは

- i.MX6ULL 搭載/Armadillo-440 と形状互換で性能向上

  Armadillo-640 は、従来の Armadillo-440 のコネクタ配置を踏襲したシングルボード型モデルです。CPU コアクロックは 528MHz にアップ、メモリは Armadillo-440 の約 4 倍の 512MB(DDR3-800)、オンボードストレージは約 4GB(eMMC)を搭載し、microSD カードスロットも備えています。従来の Armadillo-400 シリーズに比べてハードウェア性能が大幅に向上し、アプリケーション開発の自由度が高くなりました。Armadillo-440 向けと同じ形状のオプションケース(樹脂製・金属製)もラインアップしているので、Armadillo-440 から乗り換えるときも筐体を新規設計する必要がありません。

- 省電力モード搭載・バッテリー駆動の機器にも最適

  省電力モードを搭載し、「アプリケーションから Armadillo-640 本体の電源を OFF にする」「RTC(リアルタイムクロック)のアラームで決まった時間に本体の電源を ON にする」といった細かな電源制御が可能です [1]。必要な時だけ本体を起動するといった運用が可能なので、バッテリーで稼動させるような機器にも適しています。

- 使用温度範囲-20℃〜+70℃対応

  Armadillo-640 は使用温度範囲-20℃〜+70℃をカバーしています。

- シングルボード型ながら拡張性にも十分に配慮

  Armadillo-640 は、シングルボード型ながら多くの拡張インターフェースを搭載しており、USB、LCD、シリアル、GPIO、I2C、I2S、SPI などの拡張に対応します。量産向けに、リード部品コネクタを搭載したモデルの他、リード部品非搭載のモデルも提供する予定です。

- Debian GNU/Linux に標準対応

[1]順次ソフトウェアアップデートにて対応予定です

Armadillo-640 は、標準ルートファイルシステムに Debian GNU/Linux を採用し、PC ライクな開発が可能です。カーネルやデバイスドライバなどの基本アプリケーションは Web サイトで無償公開されているので、Linux の豊富な開発資産も利用できます。

3.2. 製品ラインアップ

Armadillo-640 の製品ラインアップは次のとおりです。

表 3.1 Armadillo-640 ラインアップ

名称	型番
Armadillo-640 ベーシックモデル開発セット	A6400-D00Z
Armadillo-640 量産ボード	A6400-U00Z
Armadillo-640 量産ボード(リード部品未実装・部品付)	A6400-B00Z
Armadillo-640 量産ボード(リード部品未実装)	A6400-N00Z

3.2.1. Armadillo-640 ベーシックモデル開発セット

Armadillo-640 ベーシックモデル開発セット(型番: A6400-D00Z)は、Armadillo-640 を使った開発がすぐに開始できるように、開発に必要なものを一式含んだセットです。

- Armadillo-640
- Armadillo-600 シリーズオプションケース(樹脂製)
- USB(A オス-miniB)ケーブル
- USB シリアル変換アダプタ [2]
- AC アダプタ(5V/2.0A, EIAJ#2 準拠)
- ジャンパソケット
- ねじ
- ゴム足
- スペーサ

3.2.2. Armadillo-640 量産ボード

Armadillo-640 量産ボードは、Armadillo-640 ベーシックモデル開発セットのセット内容を必要最小限に絞った量産向けのラインアップです。リード部品実装(A6400-U00Z)、リード部品未実装、部品付(型番: A6400-B00Z)、リード部品未実装(型番: A6400-N00Z)の 3 種類あります。

3.3. 仕様

Armadillo-640 の主な仕様は次のとおりです。

[2]Armadillo-800 シリーズ、Armadillo-IoT シリーズなどで使用していた USB シリアル変換アダプタ(型番: SA-SCUSB-00)とはコネクタ形状が違いますので、ご注意ください。

表 3.2 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)
LAN(Ethernet)	100BASE-TX/10BASE-T x 1 AUTO-MDIX 対応
シリアル(UART)	RS232C レベル x 1 3.3V CMOS レベル 最大 6 ポート拡張可能 [a]
USB	USB 2.0 Host x 2
SD	microSD スロット x 1
ビデオ	LCD インターフェース 最大 1 ポート拡張可能 [a] 最大解像度: WXGA (1366 x 768), 18bpp タッチパネル対応可能
オーディオ	I2S 最大 3 ポート拡張可能 [a] S/PDIF 最大 1 ポート拡張可能 [a]
GPIO	最大 65 bit 拡張可能
I2C	最大 3 ポート拡張可能 [a]
SPI	最大 4 ポート拡張可能 [a]
CAN	最大 2 ポート拡張可能 [a]
PWM	最大 8 ポート拡張可能 [a]
カレンダー時計	SoC 内蔵リアルタイムクロック [b] I2C 拡張可能
スイッチ	ユーザースイッチ x 1
LED	ユーザー LED x 3
電源電圧	DC 5V±5%
消費電力(参考値)	約 1.1W(待機時)、約 1.5W(LAN 通信時) [c]
使用温度範囲	-20〜+70℃(結露なきこと) [d]
外形サイズ	75×50mm(突起部を除く)

[a]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。
[b]電池は付属していません。
[c]外部接続機器の消費分は含みません。
[d]Armadillo-600 シリーズ オプションケースセット(樹脂製)を組み合わせた場合の使用温度範囲は-10℃〜+50℃です。

3.4. ブロック図

Armadillo-640 のブロック図は次のとおりです。

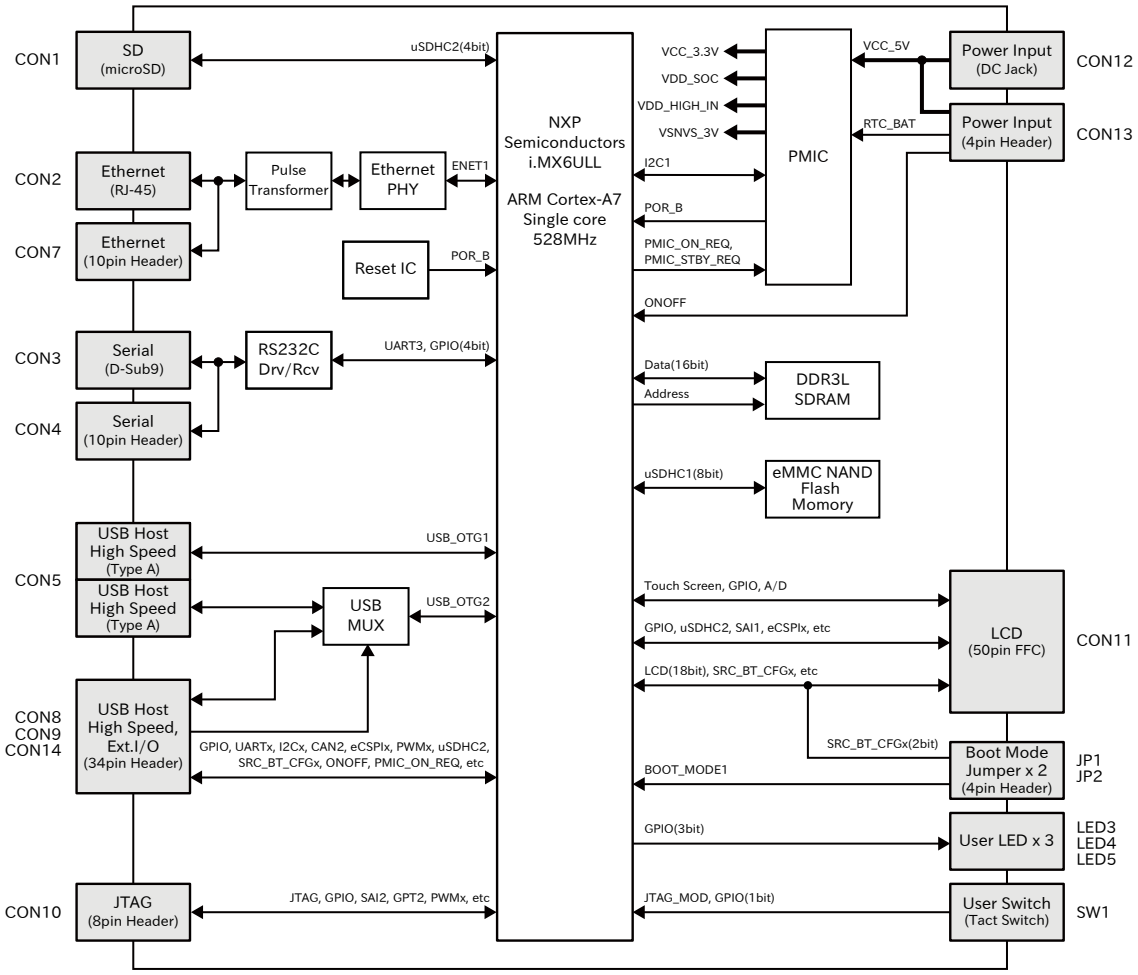


図 3.2 Armadillo-640 ブロック図

3.5. ソフトウェア構成

Armadillo-640 で動作するソフトウェアの構成について説明します。Armadillo-640 で利用可能なソフトウェアを「表 3.3. Armadillo-640 で利用可能なソフトウェア」に示します。

表 3.3 Armadillo-640 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。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。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、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。

Armadillo-640 の eMMC のメモリマップを「表 3.4. eMMC メモリマップ」に示します。

表 3.4 eMMC メモリマップ

パーティション	サイズ	説明
1	30.6MByte	予約領域
2	3.4GByte	Linux カーネルイメージ, Device Tree Blob, Debian GNU/Linux
3	122.1MByte	予約領域

Armadillo-640 の eMMC(GPP)のメモリマップを「表 3.5. eMMC(GPP)メモリマップ」に示します。

表 3.5 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。
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 v20180401 以降です。

ATDE7 は Debian GNU/Linux 9 (コードネーム Stretch) をベースに、Armadillo-640 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-640 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 v20180401 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。

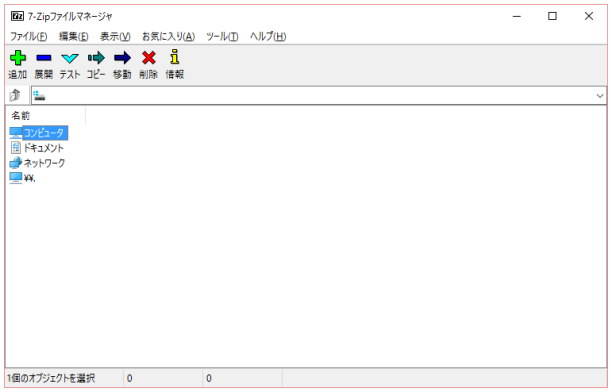
4.2.1.4. Windows で ATDE のアーカイブ展開する

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。

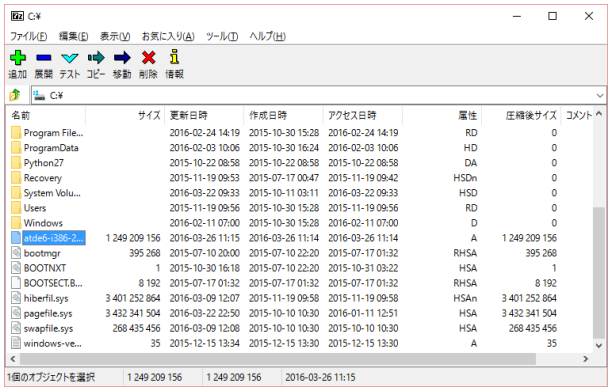
2. 7-Zip の起動

   7-Zip を起動します。



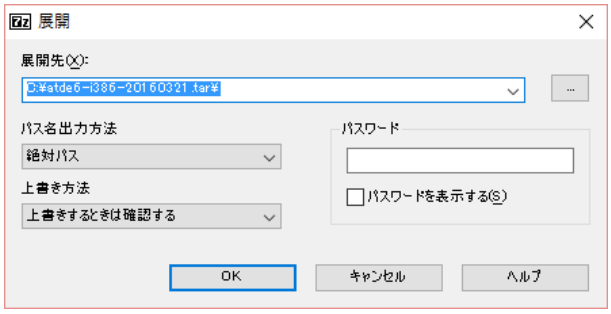
3. xz 圧縮ファイルの選択

   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。

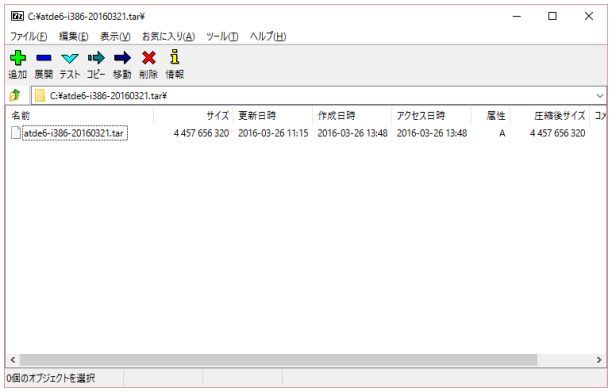


5. xz 圧縮ファイルの展開

   展開が始まります。

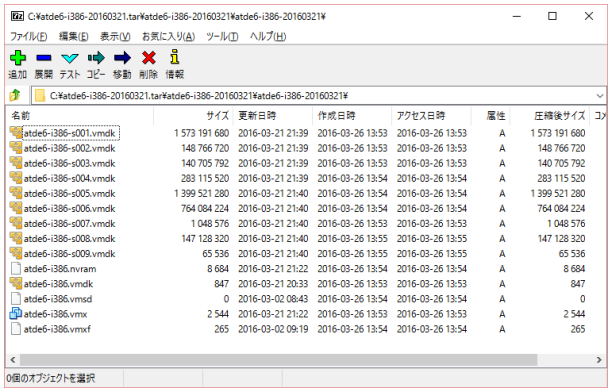
6. tar アーカイブファイルの選択

   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-640 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。

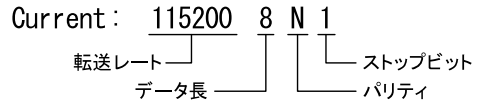


図 4.8 minicom シリアルポートの設定値

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-640 のインターフェースレイアウトです。各インターフェースの配置場所等を確認してください。

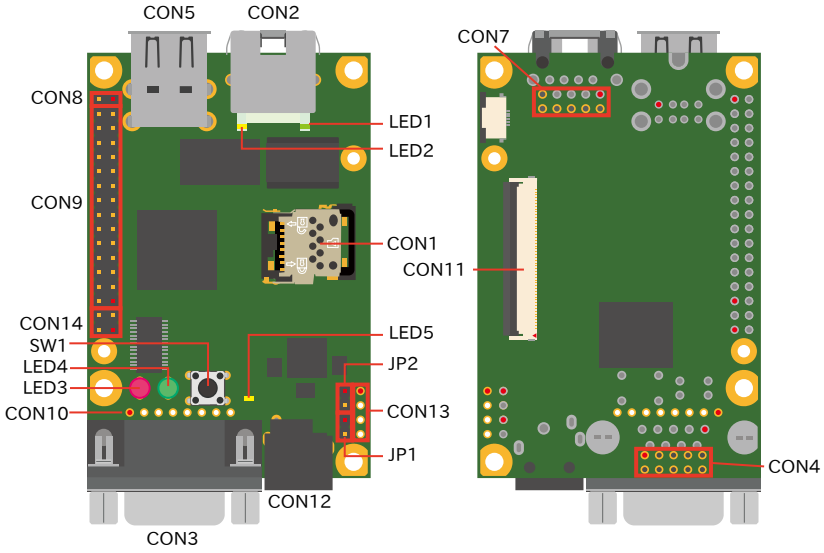


図 4.11 インターフェースレイアウト

表 4.4 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	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	D-Sub9 ピン(オス)	
CON4	シリアルインターフェース	ピンヘッダ 10 ピン(2.54mm ピッチ)	コネクタ非搭載
CON5	USB インターフェース	Type-A コネクタ(2 ポートスタック)	
CON7	LAN 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
CON8	拡張インターフェース	ピンヘッダ 2 ピン(2.54mm ピッチ)	
CON9	拡張インターフェース	ピンヘッダ 28 ピン(2.54mm ピッチ)	
CON10	JTAG インターフェース	ピンヘッダ 8 ピン(2.54mm ピッチ)	コネクタ非搭載
CON11	LCD 拡張インターフェース	FFC コネクタ 50 ピン(0.5mm ピッチ)	挿抜寿命: 20 回 [a]
CON12	電源入力インターフェース	DC ジャック	対応プラグ: EIAJ#2
CON13	電源入力インターフェース	ピンヘッダ 4 ピン(2.54mm ピッチ)	コネクタ非搭載
CON14	拡張インターフェース	ピンヘッダ 4 ピン(2.54mm ピッチ)	
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)	
JP2	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)	
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(赤)	LED(赤色,φ3mm)	
LED4	ユーザー LED(緑)	LED(緑色,φ3mm)	
LED5	ユーザー LED(黄)	LED(黄色,面実装)	
SW1	ユーザースイッチ	タクトスイッチ(h=17mm)	

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

4.4. 接続方法

Armadillo-640 と周辺装置の接続例を次に示します。

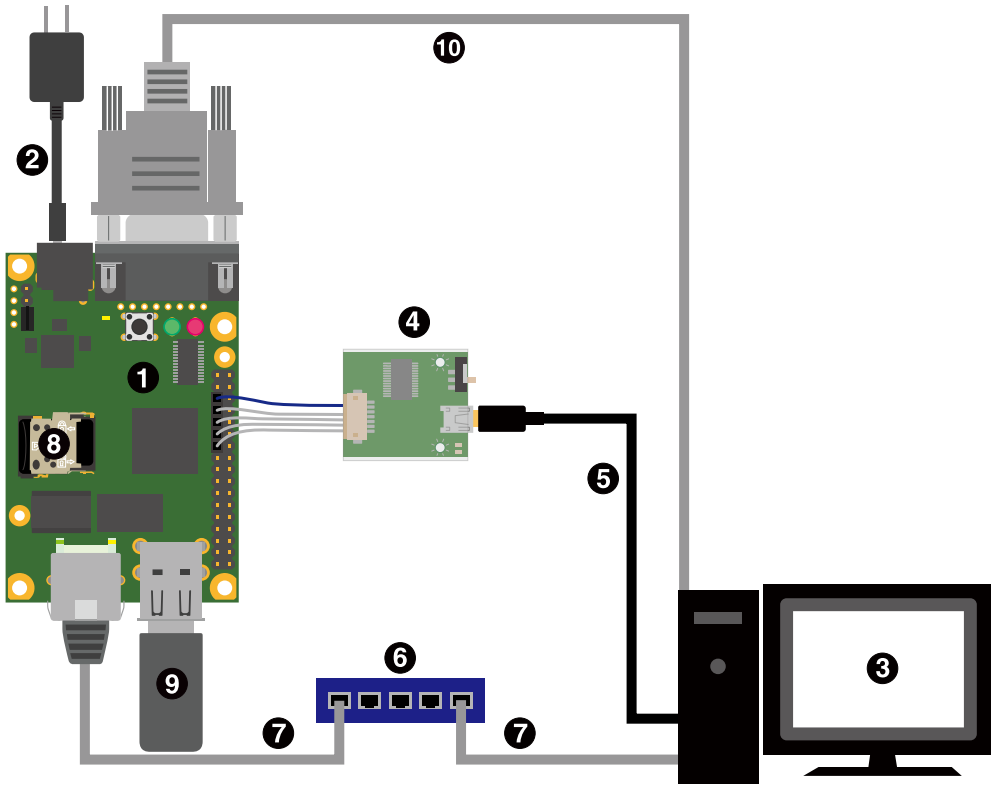


図 4.12 Armadillo-640 の接続例

❶ Armadillo-640

❷ AC アダプタ(5V/2A)

❸ 作業用 PC

❹ USB シリアル変換アダプタ

❺ USB2.0 ケーブル(A-miniB タイプ)

❻ LAN HUB

❼ Ethernet ケーブル

❽ microSD カード

❾ USB メモリ

❿ シリアルクロスケーブル

4.4.1. USB シリアル変換アダプタの接続方法

USB シリアル変換アダプタは、青色のケーブルを 1 ピンとして、Armadillo-640 の CON9 1,3,5,7,9 ピンと接続します。

USB シリアル変換アダプタを接続するピンの隣だけ、CON9,CON14 を囲っているシルクが太くなっているのでそれを目印にして、下図のように接続してください。

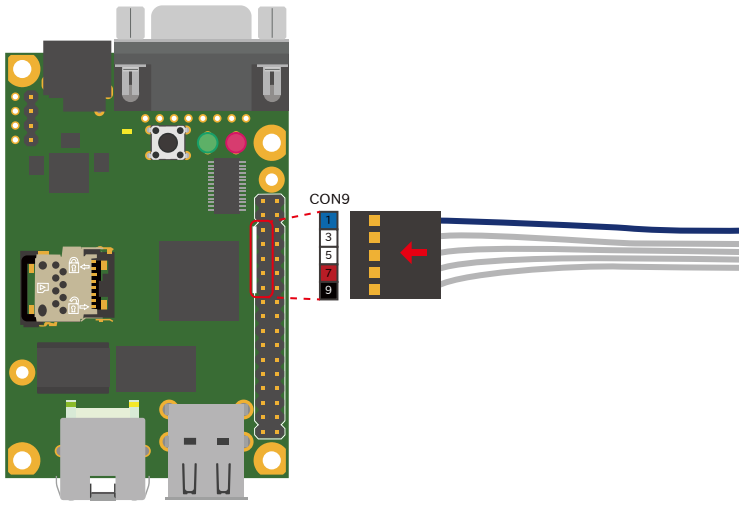


図 4.13 CON9-USB シリアル変換アダプタ接続図

4.5. ジャンパピンの設定について

ジャンパの設定を変更することで、Armadillo-640 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「16.12. JP1、JP2(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 4.14. JP1、JP2 の位置」で確認してください。

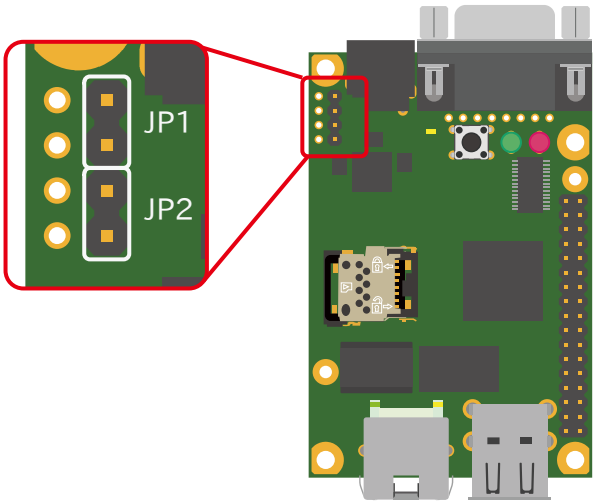


図 4.14 JP1、JP2 の位置

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、JP2 をショート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。

4.6. 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。

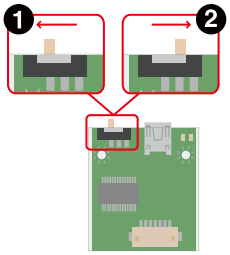


図 4.15 スライドスイッチの設定

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。

4.7. vi エディタの使用方法

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

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

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

4.7.1. vi の起動

vi を起動するには、以下のコマンドを入力します。

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

図 4.16 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。

4.7.2. 文字の入力

文字を入力するにはコマンドモードから入力モードへ移行する必要があります。コマンドモードから入力モードに移行するには、「表 4.5. 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 4.5 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 4.17. 入力モードに移行するコマンドの説明」に示します。

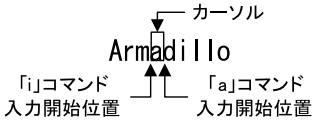


図 4.17 入力モードに移行するコマンドの説明

vi での文字削除

コンソールの環境によっては BS(Backspace)キーで文字が削除できず、「^H」文字が入力される場合があります。その場合は、「4.7.4. 文字の削除」で説明するコマンドを使用し、文字を削除してください。

4.7.3. カーソルの移動

方向キーでカーソルの移動ができますが、コマンドモードで「表 4.6. 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 4.6 カーソルの移動コマンド

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

4.7.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.7. 文字の削除コマンド」に示すコマンドを入力します。

表 4.7 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 4.18. 文字を削除するコマンドの説明」に示します。

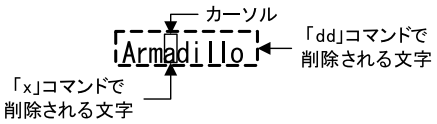


図 4.18 文字を削除するコマンドの説明

4.7.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.8. 保存・終了コマンド」に示します。

表 4.8 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

CON12 に電源を接続すると、Armadillo-640 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によって起動モードが変わります。 詳しくは 「4.4. 接続方法」 や 「4.6. スライドスイッチの設定について」 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、上記のような Warning が表示されます。

```
U-Boot 2018.03-at1 (Apr 04 2018 - 12:12:16 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
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
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-at1 (Apr 04 2018 - 12:12:16 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
```

```
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
3345800 bytes read in 128 ms (24.9 MiB/s)
23185 bytes read in 54 ms (418.9 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3345736 Bytes = 3.2 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 9eeff000, end 9ef07a90 ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.14-at1 (atmark@atde7) (gcc version 6.3.0 20170516 (Debian 6.3.0-18)) ⏎
#1 Wed Apr 4 12:30:18 JST 2018
[    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-640
[    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: 495096K/524288K available (4096K kernel code, 205K rwdata, 932K rodata, 1024K ⏎
init, 222K bss, 12808K 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 - 0xc0500000   (5088 kB)
[    0.000000]       .init : 0xc0700000 - 0xc0800000   (1024 kB)
[    0.000000]       .data : 0xc0800000 - 0xc0833500   ( 206 kB)
[    0.000000]        .bss : 0xc0837ea4 - 0xc086f90c   ( 223 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
[    0.000017] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 89478484971ns
[    0.000055] clocksource: mxc_timer1: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: ⏎
```

```
79635851949 ns
[    0.001566] Console: colour dummy device 80x30
[    0.002150] console [tty0] enabled
[    0.002214] Calibrating delay loop (skipped), value calculated using timer frequency.. 48.00 ⏎
BogoMIPS (lpj=240000)
[    0.002278] pid_max: default: 32768 minimum: 301
[    0.002676] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.002732] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.003859] CPU: Testing write buffer coherency: ok
[    0.004801] Setting up static identity map for 0x80100000 - 0x8010003c
[    0.006511] devtmpfs: initialized
[    0.014845] random: get_random_u32 called from bucket_table_alloc+0x84/0x1bc with crng_init=0
[    0.015271] VFP support v0.3: implementor 41 architecture 2 part 30 variant 7 rev 5
[    0.015551] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: ⏎
19112604462750000 ns
[    0.015625] futex hash table entries: 256 (order: -1, 3072 bytes)
[    0.016914] pinctrl core: initialized pinctrl subsystem
[    0.017887] NET: Registered protocol family 16
[    0.019636] DMA: preallocated 256 KiB pool for atomic coherent allocations
[    0.026662] vdd3p0: supplied by regulator-dummy
[    0.027342] cpu: supplied by regulator-dummy
[    0.027984] vddsoc: supplied by regulator-dummy
[    0.037305] imx6ul-pinctrl 20e0000.iomuxc: initialized IMX pinctrl driver
[    0.060697] SCSI subsystem initialized
[    0.061040] usbcore: registered new interface driver usbfs
[    0.061167] usbcore: registered new interface driver hub
[    0.061306] usbcore: registered new device driver usb
[    0.063106] clocksource: Switched to clocksource mxc_timer1
[    0.074787] NET: Registered protocol family 2
[    0.075860] TCP established hash table entries: 4096 (order: 2, 16384 bytes)
[    0.076007] TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
[    0.076131] TCP: Hash tables configured (established 4096 bind 4096)
[    0.076328] UDP hash table entries: 256 (order: 0, 4096 bytes)
[    0.076396] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[    0.076763] NET: Registered protocol family 1
[    0.077447] RPC: Registered named UNIX socket transport module.
[    0.077498] RPC: Registered udp transport module.
[    0.077526] RPC: Registered tcp transport module.
[    0.077554] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.078867] workingset: timestamp_bits=14 max_order=17 bucket_order=3
[    0.094152] squashfs: version 4.0 (2009/01/31) Phillip Lougher
[    0.095627] NFS: Registering the id_resolver key type
[    0.095731] Key type id_resolver registered
[    0.095767] Key type id_legacy registered
[    0.095872] fuse init (API version 7.26)
[    0.097413] random: fast init done
[    0.103392] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 253)
[    0.103456] io scheduler noop registered
[    0.103486] io scheduler deadline registered
[    0.103879] io scheduler cfq registered (default)
[    0.106845] 2020000.serial: ttymxc0 at MMIO 0x2020000 (irq = 18, base_baud = 5000000) is a IMX
[    0.106926] Console IMX rounded baud rate from 114943 to 114900
[    0.571197] console [ttymxc0] enabled
[    0.576206] 21ec000.serial: ttymxc2 at MMIO 0x21ec000 (irq = 49, base_baud = 5000000) is a IMX
[    0.586159] libphy: Fixed MDIO Bus: probed
[    0.591838] fec 2188000.ethernet: 2188000.ethernet supply phy not found, using dummy regulator
[    0.601779] libphy: fec_enet_mii_bus: probed
[    0.607654] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
```

```
[    0.614318] ehci-mxc: Freescale On-Chip EHCI Host driver
[    0.619872] usbcore: registered new interface driver usb-storage
[    0.631466] ci_hdrc ci_hdrc.0: EHCI Host Controller
[    0.636510] ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus number 1
[    0.673145] ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00
[    0.679823] hub 1-0:1.0: USB hub found
[    0.683770] hub 1-0:1.0: 1 port detected
[    0.692494] ci_hdrc ci_hdrc.1: EHCI Host Controller
[    0.697538] ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus number 2
[    0.733143] ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
[    0.739800] hub 2-0:1.0: USB hub found
[    0.743742] hub 2-0:1.0: 1 port detected
[    0.749035] i2c /dev entries driver
[    0.752682] IR NEC protocol handler initialized
[    0.757322] IR RC5(x/sz) protocol handler initialized
[    0.762409] IR RC6 protocol handler initialized
[    0.766991] IR JVC protocol handler initialized
[    0.771550] IR Sony protocol handler initialized
[    0.776223] IR SANYO protocol handler initialized
[    0.780955] IR Sharp protocol handler initialized
[    0.785707] IR MCE Keyboard/mouse protocol handler initialized
[    0.791568] IR XMP protocol handler initialized
[    0.796752] sdhci: Secure Digital Host Controller Interface driver
[    0.802981] sdhci: Copyright(c) Pierre Ossman
[    0.807432] sdhci-pltfm: SDHCI platform and OF driver helper
[    0.873143] mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
[    0.943550] mmc1: SDHCI controller on 2194000.usdhc [2194000.usdhc] using ADMA
[    0.952328] usbcore: registered new interface driver usbhid
[    0.958033] usbhid: USB HID core driver
[    0.963378] NET: Registered protocol family 17
[    0.968018] Key type dns_resolver registered
[    0.973564] ThumbEE CPU extension supported.
[    0.986466] mmc0: new DDR MMC card at address 0001
[    0.992696] input: gpio-keys as /devices/soc0/gpio-keys/input/input0
[    0.999756] mmcblk0: mmc0:0001 Q2J55L 3.53 GiB
[    1.005960] mmcblk0boot0: mmc0:0001 Q2J55L partition 1 2.00 MiB
[    1.012395] mmcblk0boot1: mmc0:0001 Q2J55L partition 2 2.00 MiB
[    1.018791] mmcblk0gp0: mmc0:0001 Q2J55L partition 4 8.00 MiB
[    1.024976] mmcblk0gp1: mmc0:0001 Q2J55L partition 5 8.00 MiB
[    1.031084] mmcblk0gp2: mmc0:0001 Q2J55L partition 6 8.00 MiB
[    1.038315] mmcblk0gp3: mmc0:0001 Q2J55L partition 7 8.00 MiB
[    1.045551] mmcblk0rpmb: mmc0:0001 Q2J55L partition 3 4.00 MiB
[    1.052754]  mmcblk0: p1 p2 p3
[    1.070413] EXT4-fs (mmcblk0p2): couldn't mount as ext3 due to feature incompatibilities
[    1.087888] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[    1.096168] VFS: Mounted root (ext4 filesystem) readonly on device 179:2.
[    1.104879] Freeing unused kernel memory: 1024K
[    1.332987] systemd[1]: System time before build time, advancing clock.
[    1.346222] systemd[1]: Failed to insert module 'autofs4': No such file or directory
[    1.376710] 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)
[    1.395593] systemd[1]: Detected architecture arm.

Welcome to Debian GNU/Linux 9 (stretch)!

[    1.434096] systemd[1]: Set hostname to <armadillo>.
[    1.983620] systemd[1]: Listening on /dev/initctl Compatibility Named Pipe.
```

```
[  OK  ] Listening on /dev/initctl Compatibility Named Pipe.
[    2.023802] systemd[1]: Listening on Journal Socket (/dev/log).
[  OK  ] Listening on Journal Socket (/dev/log).
[    2.053468] systemd[1]: Reached target Remote File Systems.
[  OK  ] Reached target Remote File Systems.
[    2.084241] systemd[1]: Listening on fsck to fsckd communication Socket.
[  OK  ] Listening on fsck to fsckd communication Socket.
[    2.123898] systemd[1]: Started Dispatch Password Requests to Console Directory Watch.
[  OK  ] Started Dispatch Password Requests to Console Directory Watch.
[    2.163705] systemd[1]: Listening on udev Kernel Socket.
[  OK  ] Listening on udev Kernel Socket.
[    2.194815] systemd[1]: Created slice User and Session Slice.
[  OK  ] Created slice User and Session Slice.
[  OK  ] Listening on Journal Socket.
[  OK  ] Reached target Swap.
[UNSUPP] Starting of Arbitrary Executable Fi…tem Automount Point not supported.
[  OK  ] Listening on Syslog Socket.
[  OK  ] Created slice System Slice.
[  OK  ] Reached target Slices.
[  OK  ] Created slice system-getty.slice.
         Starting File System Check on Root Device...
         Starting Create Static Device Nodes in /dev...
[  OK  ] Created slice system-serial\x2dgetty.slice.
         Starting Journal Service...
[  OK  ] Started Forward Password Requests to Wall Directory Watch.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Encrypted Volumes.
[  OK  ] Listening on udev Control Socket.
         Starting Load Kernel Modules...
[  OK  ] Started File System Check on Root Device.
[  OK  ] Started Create Static Device Nodes in /dev.
[  OK  ] Started Load Kernel Modules.
[  OK  ] Started Journal Service.
[  OK  ] Started File System Check Daemon to report status.
         Starting Apply Kernel Variables...
         Mounting FUSE Control File System...
         Starting udev Kernel Device Manager...
         Starting Remount Root and Kernel File Systems...
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Started Apply Kernel Variables.
[    3.335346] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[  OK  ] Started Remount Root and Kernel File Systems.
[  OK  ] Started udev Kernel Device Manager.
[  OK  ] Reached target Local File Systems (Pre).
         Starting udev Coldplug all Devices...
         Starting Load/Save Random Seed...
         Starting Flush Journal to Persistent Storage...
[  OK  ] Reached target Local File Systems.
         Starting Raise network interfaces...
[  OK  ] Started Load/Save Random Seed.
[    3.942404] systemd-journald[68]: 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 udev Coldplug all Devices.
```

```
[  OK  ] Started Network Time Synchronization.
[  OK  ] Found device /dev/ttymxc0.
[  OK  ] Reached target System Time Synchronized.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Daily apt download activities.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Started Daily apt upgrade and clean activities.
[  OK  ] Reached target Timers.
[  OK  ] Reached target Basic System.
         Starting Login Service...
[    5.533524] SMSC LAN8710/LAN8720 2188000.ethernet-1:00: attached PHY driver [SMSC LAN8710/
LAN8720] (mii_bus:phy_addr=2188000.ethernet-1:00, irq=POLL)
         Starting System Logging Service...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Regular background program processing daemon.
[  OK  ] Started System Logging Service.
[  OK  ] Started Login Service.
[    7.683515] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[  OK  ] Started Raise network interfaces.
[  OK  ] Reached target Network.
         Starting Permit User Sessions...
[  OK  ] Reached target Network is Online.
         Starting /etc/rc.local Compatibility...
         Starting LSB: exim Mail Transport Agent...
[  OK  ] Started Permit User Sessions.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttymxc0.
[  OK  ] Reached target Login Prompts.
[  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

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-640 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...
[  OK  ] Stopped target Login Prompts.
         Stopping Getty on tty1...
         Stopping Serial Getty on ttymxc0...
         Stopping LSB: exim Mail Transport Agent...
         Stopping Regular background program processing daemon...
         Stopping User Manager for UID 0...
         Stopping System Logging Service...
[  OK  ] Stopped target Timers.
[  OK  ] Stopped Daily apt upgrade and clean activities.
[  OK  ] Stopped Daily apt download activities.
[  OK  ] Stopped Daily Cleanup of Temporary Directories.
[  OK  ] Stopped System Logging Service.
[  OK  ] Stopped D-Bus System Message Bus.
[  OK  ] Stopped Regular background program processing daemon.
[  OK  ] Stopped Getty on tty1.
[  OK  ] Stopped Serial Getty on ttymxc0.
[  OK  ] Stopped User Manager for UID 0.
[  OK  ] Stopped Session c1 of user root.
[  OK  ] Removed slice User Slice of root.
         Stopping Login Service...
[  OK  ] Removed slice system-serial\x2dgetty.slice.
[  OK  ] Stopped /etc/rc.local Compatibility.
[  OK  ] Removed slice system-getty.slice.
```

```
         Stopping Permit User Sessions...
[  OK  ] Stopped Login Service.
[  OK  ] Stopped LSB: exim Mail Transport Agent.
[  OK  ] Stopped Permit User Sessions.
[  OK  ] Stopped target Network is Online.
[  OK  ] Stopped target Network.
         Stopping Raise network interfaces...
[  OK  ] Stopped target Remote File Systems.
[  OK  ] Stopped target System Time Synchronized.
[  OK  ] Stopped target Basic System.
[  OK  ] Stopped target Slices.
[  OK  ] Stopped target Paths.
[  OK  ] Stopped target Sockets.
[  OK  ] Closed Syslog Socket.
[  OK  ] Closed D-Bus System Message Bus Socket.
[  OK  ] Removed slice User and Session Slice.
[  OK  ] Stopped target System Initialization.
         Stopping Network Time Synchronization...
[  OK  ] Stopped target Swap.
         Stopping Load/Save Random Seed...
         Stopping Update UTMP about System Boot/Shutdown...
[  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 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 Apply Kernel Variables.
[  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.
[ 1012.593800] systemd-shutdow: 21 output lines suppressed due to ratelimiting
[ 1012.650219] systemd-shutdown[1]: Sending SIGTERM to remaining processes...
[ 1012.667975] systemd-journald[68]: Received SIGTERM from PID 1 (systemd-shutdow).
[ 1012.684489] systemd-shutdown[1]: Sending SIGKILL to remaining processes...
[ 1012.700722] systemd-shutdown[1]: Unmounting file systems.
[ 1012.707655] systemd-shutdown[1]: Remounting '/' read-only with options 'data=ordered'.
[ 1012.727070] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[ 1012.739616] systemd-shutdown[1]: Remounting '/' read-only with options 'data=ordered'.
[ 1012.748129] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[ 1012.754170] systemd-shutdown[1]: All filesystems unmounted.
[ 1012.759786] systemd-shutdown[1]: Deactivating swaps.
[ 1012.765075] systemd-shutdown[1]: All swaps deactivated.
[ 1012.770350] systemd-shutdown[1]: Detaching loop devices.
[ 1012.778377] systemd-shutdown[1]: All loop devices detached.
[ 1012.811460] ci_hdrc ci_hdrc.1: remove, state 1
[ 1012.816006] usb usb2: USB disconnect, device number 1
[ 1012.822050] ci_hdrc ci_hdrc.1: USB bus 2 deregistered
[ 1012.827394] ci_hdrc ci_hdrc.0: remove, state 1
[ 1012.831879] usb usb1: USB disconnect, device number 1
```

```
[ 1012.838029] ci_hdrc ci_hdrc.0: USB bus 1 deregistered
[ 1012.843458] 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-640 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-640 の状態が異なります。

poweroff の場合、Armadillo-640 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、5V 電源を切ったのち 5V 電源を再入力しても Armadillo-640 が起動しません。詳しくは「15.5.1. ONOFF ピンの制御について」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-640 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-640 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-640 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-640 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-640 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-640 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	microSD スロット(CON1)
USB メモリ	/dev/sd* [a]	/dev/sd*1	USB ホストインターフェース (CON5)

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

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-640 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 (CON1) に SDHC カードを挿入し、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。

```
[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-640 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/red/	ユーザー LED 赤	default-on

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-640 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
```

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「トリガー」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 トリガーの種類

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

設定	説明
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ーと、現在有効のトリガーが表示されます。[] 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
trllock kbd-ctrlrlock mmc0 mmc1 timer oneshot heartbeat default-on
```

図 6.20 対応している LED トリガーを表示

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-640 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-640 は、i.MX6ULL の RTC 機能を利用しています。

電源が切断されても時刻を保持させたい場合は、電源入力インターフェース(CON13)に外付けバッテリーを接続することができます。

「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」、「18.5. Armadillo-600 シリーズ RTC オプションモジュール」を Armadillo-640 に接続することで温度補償高精度リアルタイムクロック(RTC)を使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-640 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
```

Armadillo-640 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。

「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」または「18.5. Armadillo-600 シリーズ RTC オプションモジュール」の接続時に 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
```

6.7. GPIO

Armadillo-640 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。

「表 6.9. CON9 ピンと GPIO 番号の対応」の各ピンは GPIO として利用することができます。

表 6.9 CON9 ピンと GPIO 番号の対応

ピン番号	ピン名	GPIO 番号
1	GPIO1_IO22	22
2	GPIO1_IO23	23
3	GPIO1_IO17	17
4	GPIO1_IO31	31
5	GPIO1_IO16	16
6	GPIO1_IO30	30
13	GPIO3_IO23	87
14	GPIO3_IO24	88
15	GPIO3_IO25	89
16	GPIO3_IO26	90
17	GPIO3_IO27	91
18	GPIO3_IO28	92
25	GPIO4_IO06	102
26	GPIO4_IO07	103
27	GPIO4_IO08	104
28	GPIO4_IO09	105

「表 6.9. CON9 ピンと GPIO 番号の対応」の CON9 1, 3~6 ピンは初期出荷状態では GPIO として利用することができません。これらのピンを GPIO として利用する場合は、 at-dtweb を用います。

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。

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(CON9 27 ピン)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 22 > /sys/class/gpio/export
[armadillo ~]# ls /sys/class/gpio/gpio22/
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 22 > /sys/class/gpio/unexport
[armadillo ~]# ls /sys/class/gpio/gpio22/
ls: cannot access '/sys/class/gpio/gpio22/': 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 6.10 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。

設定	説明
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

7. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-640 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-640 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-640 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。

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-640 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 (CON9)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-640 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-640

Armadillo-640 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-640.dts
- 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-640 のシリアルは、i.MX6ULL の UART (Universal Asynchronous Receiver/Transmitter) を利用しています。Armadillo-640 の標準状態では、UART1 (CON9) をコンソールとして利用しています。

フォーマット
- データビット長: 7 or 8 ビット
- ストップビット長: 1 or 2 ビット
- パリティ: 偶数 or 奇数 or なし
- フロー制御: CTS/RTS or XON/XOFF or なし
- 最大ボーレート: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
UART3	/dev/ttymxc2

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-640 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

Armadillo-600 シリーズ WLAN オプションモジュールには、アットマークテクノ製 Armadillo-WLAN モジュール(AWL13)が搭載されています。AWL13 は、CON9 を通して「7.3.6. USB ホスト」に示す USB OTG2 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>
```

[1]STA=ステーション
[2]AP=アクセスポイント

7.3.5. SD ホスト

Armadillo-640 の SD ホストは、i.MX6ULL の uSDHC (Ultra Secured Digital Host Controller) を利用しています。Armadillo-640 では、オンボード microSD コネクタ (CON1) が uSDHC2 を利用しています。

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>
    [*]   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-640 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-640 では、USB ホストインターフェース (CON5) が OTG1 (下段) と OTG2 (上段) を利用しています。OTG2 は CON5 と CON9 と排他利用になっており、外部からの信号で切り替えることができるようになっています。詳しくは「16.5. CON5(USB ホストインターフェース)」を参照してください。

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. リアルタイムクロック

Armadillo-640 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、「18.5. Armadillo-600 シリーズ RTC オプションモジュール」および「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」には、日本電波工業(NDK)製 NR3225SA が搭載されています。NR3225SA は、「7.3.10. I2C」に示す I2C-GPIO1 (I2C ノード: 5-0032) に接続されています。

機能

- アラーム割り込みサポート

デバイスファイル

- /dev/rtc
- /dev/rtc0
- /dev/rtc1

NR3225SA が /dev/rtc0 となるよう、Device Tree でエイリアスを設定しています。そのた

め、「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」および「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」の接続時、i.MX6ULL の RTC 機能は /dev/rtc1 となります。

エイリアスの設定は、linux-4.14-at[VERSION]/arch/arm/boot/dts/armadillo-640_con9_awl13_rtc.dts で行っています。

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。

7.3.8. LED

Armadillo-640 に搭載されているソフトウェア制御可能な LED には、GPIO が接続されています。Linux では、GPIO 接続用 LED ドライバ (leds-gpio) で制御することができます。

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

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

7.3.9. ユーザースイッチ

Armadillo-640 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>
```

7.3.10. I2C

Armadillo-640 の I2C インターフェースは、i.MX6ULL の I2C(I2C Controller) および GPIO を利用した I2C バスドライバ(i2c-gpio)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。

Armadillo-640 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
4(I2C5)	0x51	PF3000
5(I2C-GPIO1)	0x32	NR3225SA [a]

[a]CON9 に Armadillo-600 シリーズ WLAN オプションモジュールまたは Armadillo-600 シリーズ RTC オプションモジュールを接続した場合。

Armadillo-640 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-4 (I2C5)
- /dev/i2c-5 (I2C-GPIO1)

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

[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

- 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>
```

7.3.11. パワーマネジメント

Armadillo-640 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-640 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機能を利用するには、Linux カーネルのバージョンが v4.14-at10 以降、U-Boot のバージョンが v2018.03-at4 以降である必要があります。

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-640 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

パワーマネジメント状態	文字列	説明
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（CON9）　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 OTG1（下段）　起床要因　USB デバイスの挿抜

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/devices/
2184000.usb/power/wakeup
[armadillo ~]# echo enabled > /sys/bus/platform/drivers/
ci_hdrc/ci_hdrc.0/power/wakeup
[armadillo ~]# echo enabled > /sys/bus/platform/drivers/
ci_hdrc/ci_hdrc.0/usb1/power/wakeup
```

USB OTG2（上段）　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/devices/soc0/i2c-gpio1/
i2c-5/5-0032/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-640 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-640 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-640 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。

2018/04/12 時点で、Armadillo-640 のソフトウェアは `reboot` コマンドに対応していません。順次ソフトウェアアップデートにて対応予定です。

`halt` コマンドを実行して電源を再接続してください。

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

[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 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

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-640 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-640 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-640 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-640 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-640 用の U-Boot に独自実装されている機能です。

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	スライドスイッチ	説明
保守モード	外側	各種設定が可能な 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	内側	電源投入後、自動的に Linux カーネルを起動させます。

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-at1 (Apr 04 2018 - 12:12:16 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
```

```
Reset cause: POR
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
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-640 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
bootargs=root=/dev/mmcblk0p2 rootwait
bootcmd=ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load mmc 0:2 0x83000000 /
boot/a640.dtb; bootm ${loadaddr} - 0x83000000;
bootdelay=0
ethact=FEC
ethaddr=00:11:0c:00:07:90
loadaddr=0x82000000
stderr=serial
stdin=serial
stdout=serial
tftpboot=tftpboot uImage; tftpboot 0x83000000 a640.dtb; bootm ${loadaddr} - 0x83
```

```
000000;

Environment size: 402/524284 bytes
=>
```

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 a640
=> env print hello
hello=a640
=>
```

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 a640
=> env save
Saving Environment to MMC... Writing to MMC(0)... OK
=>

....電源入れなおし....

U-Boot 2018.03-at1 (Apr 04 2018 - 12:12:16 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
Net:   FEC
=> env print hello
hello=a640
=>
```

表示された文字からも分るように、Armadillo-640 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 3.4. 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=ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load mmc 0:2 0x83000000 /boot/a640.dtb;
bootm ${loadaddr} - 0x83000000;
=> run bootcmd
3345800 bytes read in 127 ms (25.1 MiB/s)
23185 bytes read in 54 ms (418.9 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
```

```
   Image Name:   Linux-4.14-at1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3345736 Bytes = 3.2 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 9eeff000, end 9ef07a90 ... 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/a640.dtb;
bootm ${loadaddr} - 0x83000000;
```

最初のコマンドは ext4load です。このコマンドは、指定された **mmc** タイプの **0** 番目のデバイスにある、 **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
3345800 bytes read in 128 ms (24.9 MiB/s)
=>
```

Armadillo-640 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、 a640.dtb 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。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/a640.dtb
23185 bytes read in 54 ms (418.9 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) 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-640 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-at1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3345736 Bytes = 3.2 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 9eeff000, end 9ef07a90 ... 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-640 では、オンボード eMMC が 0 番目の mmc タイプのデバイスです。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 タイプのデバイス情報は 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: Q2J55
Bus Speed: 52000000
Mode : MMC High Speed (52MHz)
Rd Block Len: 512
MMC version 5.0
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: 2 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。

```
=> 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
=>
```

microSD カードが装着されていれば、上記のように表示されます。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-640 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-640 では オンボード eMMC が 0 番目の MMC デバイスです。Armadillo-640 の初期出荷時に、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 MMM 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。

bootargs と setup_mmcargs に関する仕様は v2018.03-at3 から変更しています。at2 を利用する場合は、以前の製品マニュアルをご覧ください。

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-640 ドキュメント・ダウンロード
>
> http://armadillo.atmark-techno.com/armadillo-640/downloads

開発作業では、基本ライブラリ・アプリケーションやシステム設定ファイルの作成・配置を行います。各ファイルは作業ディレクトリ配下で作成・配置作業を行いますが、作業ミスにより誤って作業用 PC 自体の OS を破壊しないために、すべての作業は root ユーザーではなく一般ユーザーで行ってください。

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-640 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
   ```

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
   ```

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-640.dtb
arch/arm/boot/dts/armadillo-640.dtb
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-640_con9_awl13_rtc.dtb
arch/arm/boot/dts/armadillo-640_con9_awl13_rtc.dtb
```

armadillo-640_con9_awl13_rtc.dtb は、「18.5. Armadillo-600 シリーズ RTC オプションモジュール」または「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」を動作させる際に使用する DTB です。linux-v4.14-at7 以降で作成されます。

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 を一度起動した後に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。

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-640 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-640 ドキュメント・ダウンロード
>
> http://armadillo.atmark-techno.com/armadillo-640/downloads

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-a640-**[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-a640-[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 4.15. 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 と JP2 を共に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-at1 installer+ (Apr 04 2018 - 21:17:22 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
In:    serial
Out:   serial
Err:   serial
Net:   FEC
Warning: FEC (eth0) using random MAC address - 5a:fd:55:1c:bb:91

=>
```

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
3345800 bytes read in 204 ms (15.6 MiB/s)
23185 bytes read in 53 ms (426.8 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3345736 Bytes = 3.2 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 9eeff000, end 9ef07a90 ... 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 赤	ユーザー LED 緑	ユーザー LED 黄
実行中	消灯	点滅	消灯
正常終了	点灯	点灯	点灯
異常終了	点滅	消灯	消灯

11.2. 特定のイメージファイルだけを書き換える

Armadillo-640 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	armadillo-640-v4.14-at[version].dtb		/dev/mmcblk0p2
WLAN/RTC オプションモジュール用 Device Tree Blob	armadillo-640_con9_awl13_rtc-v4.14-at[version].dtb		/dev/mmcblk0p2

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-640-v4.14-at[version].dtb /mnt/boot/a640.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 a640.dtb にリネームして/mnt/boot/ディレクトリにコピーします。

WLAN/RTC オプションモジュールを利用する場合は、 `armadillo-640-v4.14-at[version].dtb` の代わりに `armadillo-640_con9_awl13_rtc-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 ~]# ls | grep -v -E 'boot|lost\+found' | xargs rm -rf ❷
[armadillo ~]# 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-640 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-640 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-640 にインストールします。そのため、あらかじめ ATDE7 および Armadillo-640 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-640 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-640 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-640 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-640 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-640 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-640 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-640 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-640 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-640 へのファイルの転送

ATDE7 上で作成したソースコードを Armadillo-640 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-640 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-640 への client.rb の転送

12.1.5. クライアントの実行

Armadillo-640 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、「15.5. 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-640 CON9 11 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

i.MX6ULL 自身による制御を行うには、Linux カーネルのバージョンが v4.14-at3 以降である必要があります。

13.2. WLAN/RTC オプションモジュールによる制御

WLAN/RTC オプションモジュールに搭載されている RTC のアラーム割り込みによって、i.MX6ULL の電源を ON にすることができます。

このためには、WLAN/RTC オプションモジュール上の JP1(はんだジャンパ)をショートしておく必要があります。JP1 については「18.5.3.4. JP1(ONOFF ピン接続ジャンパ)」を参照してください。また、RTC の割り込み信号はアラームを解除するまで出力され続けるため、i.MX6ULL の電源が ON になってから 5 秒以内に、ブートローダーもしくは Linux カーネルがアラームを解除する必要があります。

ブートローダーの起動時に RTC のアラームを解除するには次のようにコマンドを実行します。

```
=> setenv stop_nr3225sa_alarm yes
=> saveenv
```

ブートローダーの起動時に RTC のアラームを解除するにはブートローダーのバージョンが 2018.03-at4 以降である必要があります。

i.MX6ULL の電源を OFF にし、1 分後に電源を ON にするには、次のようにコマンドを実行します。

```
[armadillo ~]# echo +60 > /sys/class/rtc/rtc0/wakealarm ❶
[armadillo ~]# poweroff ❷
: (省略)
[   32.428051] reboot: Power down


U-Boot 2018.03-at4 (Dec 18 2018 - 18:34:38 +0900) ❸

CPU:   Freescale i.MX6ULL rev1.0 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
Net:   FEC
=>
```

図 13.2 i.MX6ULL の電源を OFF にし、1 分後に電源を ON にする手順

❶ 1 分後にアラームを設定します。

❷ i.MX6ULL の電源を OFF にします。

❸ 1 分経過後、自動的に i.MX6ULL の電源が ON になります。

WLAN/RTC オプションモジュールに搭載されている RTC NR3225SA は、毎分 0 秒にしかアラーム割り込みを発生させることができません。

0 時 0 分 30 秒の時に、1 秒後にアラームが鳴るように設定したとしても、実際にアラーム割り込みが発生するのは 0 時 1 分 0 秒となります。

sysfs RTC クラスディレクトリ以下の wakealarm ファイルからアラーム割り込みを発生させるには、Linux カーネルのバージョンが v4.14-at10 以降である必要があります。

linux-v4.14-at10 以前の Linux カーネルでも、ioctl RTC_ALM_SET, ioctl RTC_AIE_ON を使用してアラーム割り込みを発生させることができます。

詳しくは、RTC の man ページを参照してください。

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-640 ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-640/downloads

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_a640__[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-640-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	a640.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-640-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-640-v4.14-at[version].dtb sd/boot/a640.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 CON1(microSD スロット)に挿入します。また、JP1 と JP2 を共に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-at1 (Apr 04 2018 - 12:12:16 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
Net:   FEC
Warning: FEC (eth0) using random MAC address - c2:37:f4:1b:c3:e4

=>
```

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/a640.dtb\; 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
```

bootcmd と bootargs, setup_mmcargs に関する仕様は v2018.03-at3 から変更しています。at2 を利用する場合は、以前の製品マニュアルをご覧ください。

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	VCC_5V	-0.3	5.25	V	-
入出力電圧(USB 信号以外)	VI,VO	-0.3	OVDD +0.3	V	OVDD=VCC_3.3V
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
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

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	VCC_5V	4.75	5	5.25	V	-
RTC バックアップ電源電圧	RTC_BAT	2.75 [a]	-	3.3	V	Topr=+25℃
使用温度範囲	Ta	-20	25	70	V	結露なきこと

[a]S/N: 009C00010001~009C00060102 の Armadillo-640 では、下限電圧 2.95V です。

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	VCC_5V USB_OTG1_VBUS USB_OTG2_VBUS	4.75	5	5.25	V	-
3.3V 電源電圧	VCC_3.3V	3.102	3.3	3.498	V	-

表 15.4 入出力インターフェースの電気的仕様(OVDD = VCC_3.3V)

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	-

項目	記号	Min.	Max.	単位	備考
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

電源回路の構成は次のとおりです。電源入力インターフェース(CON12 または CON13)からの入力電圧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。

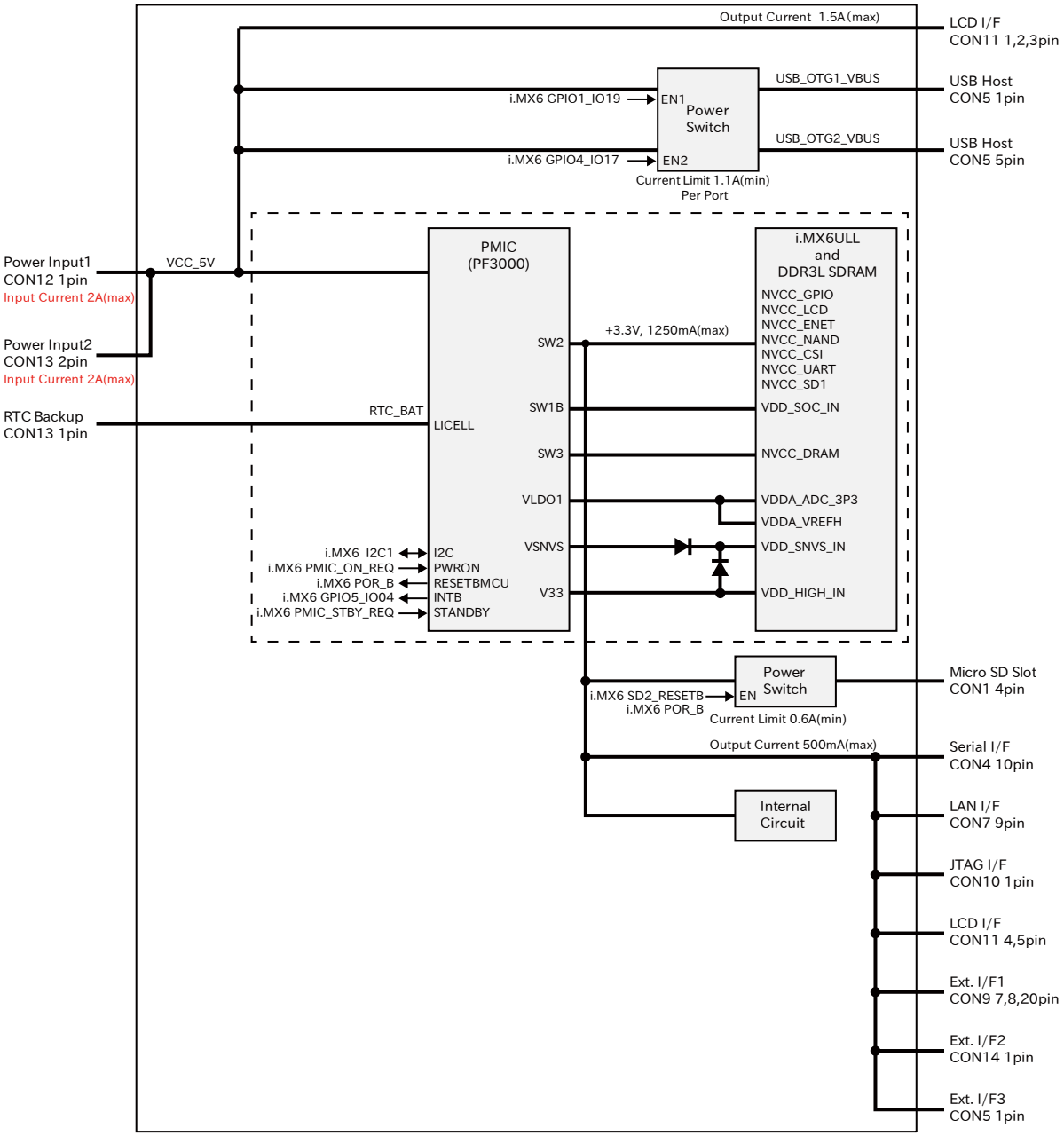


図 15.1 電源回路の構成

電源シーケンスは次のとおりです。

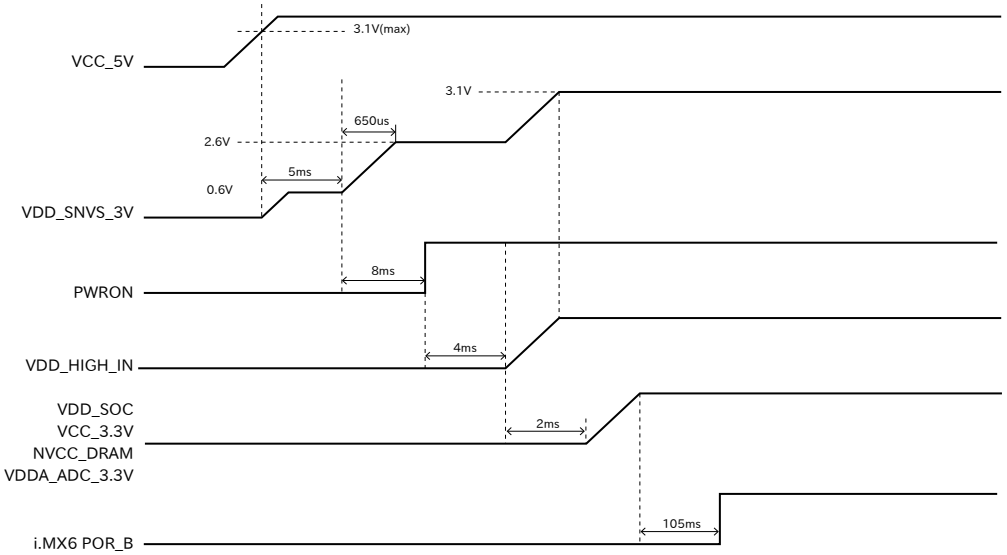


図 15.2 電源シーケンス

15.5. 外部からの電源制御

Armadillo-640 は、拡張インターフェースのピンを制御することにより電源をオンまたはオフに切り替えることができます。

ここでは、外部からの電源制御に必要な以下の項目を説明します。

- ONOFF ピンの制御
- PWRON ピンの制御
- RTC_BAT ピン

15.5.1. ONOFF ピンの制御について

ONOFF ピンは、一定時間以上 GND とショートすることで、Armadillo-640 の電源をオフまたはオンすることができます。外部から ONOFF ピンを制御する場合、電圧の印加はできませんのでオープンドレインなどの出力を接続し GND とショートする回路を接続してください。

15.5.1.1. 電源オンから電源オフに切り替える方法

ONOFF ピンを 5 秒以上 GND とショートすることで、電源がオフになります。

15.5.1.2. 電源オフから電源オンに切り替える方法

ONOFF ピンを 500 ミリ秒以上 GND とショートすることで、電源がオンになります。

連続して電源を切り替える場合、確実に動作させるため 5 秒以上空けてから ONOFF ピンを GND とショートしてください。

電源のオンまたはオフの状況は i.MX6ULL の低消費電力ドメイン(SNVS_LP)で保持されているため、電源オフの状態で 5V 電源入力を切ってもしばらくは電源オフであることを保持しています。そのため、すぐに電源を再入力した場合電源が入らない状態になる可能性があります。電源オフの状態で 5V 電源を再入力する場合は、確実に電源を入れるため、5 秒以上間隔を空けてから 5V 電源を入れてください。

電源のオンまたはオフの状況は RTC_BAT ピンからのバックアップ電源により保持されるため、RTC_BAT ピンにバックアップ電源を入力した状態で 5V 電源を切ったのち 5V 電源を再入力しても、5V 電源切断前の電源のオンまたはオフの状況が継続されます。

15.5.2. PWRON ピンの制御について

PWRON ピンは、GND とショートすることで、Armadillo-640 の電源を即座にオフすることができます。外部から ONOFF ピンを制御する場合、電圧の印加はできませんのでオープンドレインなどの出力を接続し GND とショートする回路を接続してください。

15.5.2.1. 電源オンから電源オフに切り替える方法

PWRON ピンを GND とショートすることで、即座に電源がオフになります。

15.5.2.2. 電源オフから電源オンに切り替える方法

PWRON ピンをオープンにすることで、電源がオンになります。

15.5.3. RTC_BAT ピンについて

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す。長時間電源が切断されても時刻データを保持させたい場合にご使用ください。

16. インターフェース仕様

Armadillo-640 のインターフェース仕様について説明します。

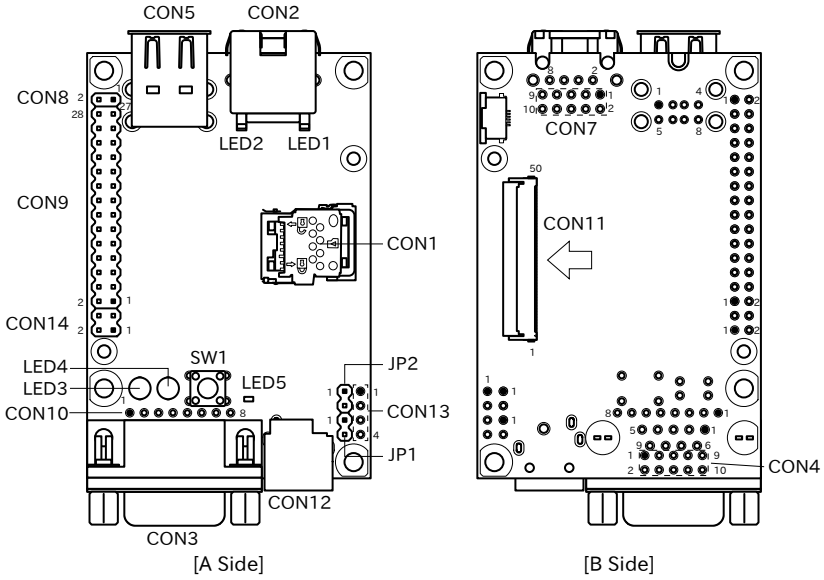


図 16.1 Armadillo-640 のインターフェース

表 16.1 Armadillo-640 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	LAN インターフェース	TM11R-5M2-88-LP	HIROSE ELECTRIC
CON7		A1-10PA-2.54DSA(71)	HIROSE ELECTRIC
CON3	シリアルインターフェース	XM2C-0942-132L	OMRON
CON4		A1-10PA-2.54DSA(71)	HIROSE ELECTRIC
CON5	USB インターフェース	UBA-4RS-D14T-4D(LF)(SN)	J.S.T.Mfg.
CON8	拡張インターフェース	A1-34PA-2.54DSA(71)	HIROSE ELECTRIC
CON9			
CON14			
CON10	JTAG インターフェース	A2-8PA-2.54DSA(71)	HIROSE ELECTRIC
CON11	LCD 拡張インターフェース	XF2M-5015-1A	OMRON
CON12	電源入力インターフェース	HEC3690-015210	HOSIDEN
CON13		A2-4PA-2.54DSA(71)	HIROSE ELECTRIC
JP1	起動デバイス設定ジャンパ	A2-4PA-2.54DSA(71)	HIROSE ELECTRIC
JP2			
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(赤)	SLR-342VC3F/LK-12	ROHM/MAC8
LED4	ユーザー LED(緑)	SLR-342MC3F/LK-12	ROHM/MAC8
LED5	ユーザー LED(黄)	SML-310YTT86	ROHM
SW1	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC

[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。

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 CON1、CON9、CON11 で利用可能ですが、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(VCC_3.3V)
5	CLK	Out	i.MX6ULL の NAND_RE_B ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	i.MX6ULL の NAND_DATA00 ピンに接続
8	DAT1	In/Out	i.MX6ULL の NAND_DATA01 ピンに接続

16.2. CON2、CON7(LAN インターフェース)

CON2、CON7 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 Ethenet 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。

表 16.3 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(+)、CON7 の 1 ピンと共通
2	TX-	In/Out	送信データ(-)、CON7 の 4 ピンと共通
3	RX+	In/Out	受信データ(+)、CON7 の 3 ピンと共通
4	-	-	CON2 の 5 ピンと接続後に 75Ω 終端、CON7 の 5 ピンと共通
5	-	-	CON2 の 4 ピンと接続後に 75Ω 終端、CON7 の 5 ピンと共通
6	RX-	In/Out	受信データ(-)、CON7 の 6 ピンと共通
7	-	-	CON2 の 8 ピンと接続後に 75Ω 終端、CON7 の 7 ピンと共通
8	-	-	CON2 の 7 ピンと接続後に 75Ω 終端、CON7 の 7 ピンと共通

表 16.4 CON7 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	LINK_ACTIVITY_LED	Out	LINK/ACTIVITY 表示 (High: リンクが確立されている、Low: リンクが確立されていない、Pulse：リンクが確立されており、データを送受信している)
3	RX+	In/Out	受信データ(+)
4	TX-	In/Out	送信データ(-)
5	-	-	
6	RX-	In/Out	受信データ(-)
7	-	-	75Ω 終端、CON2 の 7、8 ピンと共通
8	SPEED_LED	Out	SPEED 表示 (High: 10Mbps または Ethernet ケーブル未接続、Low: 100Mbps)
9	VCC_3.3V	Power	電源(VCC_3.3V)
10	GND	Power	電源(GND)

CON2 と CON7 は、共通の信号が接続されており、同時に使用することはできません。どちらか一方のコネクタでのみ、ご使用ください。

16.3. LED1、LED2(LAN LED)

LED1、LED2 は LAN インターフェースのステータス LED です。CON2 の上部に表示されます。信号線は Ethenet PHY(LAN8720AI-CP/Microchip Technology)の LED ピンに接続されています。

表 16.5 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

16.4. CON3、CON4(シリアルインターフェース)

CON3、CON4 は非同期(調歩同期)シリアルインターフェースです。信号線は RS232C レベル変換 IC を経由して i.MX6ULL の UART コントローラ(UART3)に接続されています。

- 信号入出力レベル: RS232C レベル
- 最大データ転送レート: 230.4kbps
- フロー制御: CTS、RTS、DTR、DSR、DCD、RI

表 16.6 CON3 信号配列

ピン番号	ピン名	I/O	説明
1	DCD	In	キャリア検出、i.MX6ULL の SNVS_TAMPER2 ピンに接続、CON4 の 1 ピンと共通

ピン番号	ピン名	I/O	説明
2	RXD	In	受信データ、i.MX6ULL の UART3_RX_DATA ピンに接続、CON4 の 3 ピンと共通
3	TXD	Out	送信データ、i.MX6ULL の UART3_TX_DATA ピンに接続、CON4 の 5 ピンと共通
4	DTR	Out	データ端末レディ、i.MX6ULL の GPIO1_IO00 ピンに接続、CON4 の 7 ピンと共通
5	GND	Power	電源(GND)
6	DSR	In	データセットレディ、i.MX6ULL の SNVS_TAMPER0 ピンに接続、CON4 の 2 ピンと共通
7	RTS	Out	送信要求、i.MX6ULL の UART3_CTS_B ピンに接続、CON4 の 4 ピンと共通
8	CTS	In	送信可能、i.MX6ULL の UART3_RTS_B ピンに接続、CON4 の 6 ピンと共通
9	RI	In	被呼表示、i.MX6ULL の SNVS_TAMPER1 ピンに接続、CON4 の 8 ピンと共通

表 16.7 CON4 信号配列

ピン番号	ピン名	I/O	説明
1	DCD	In	キャリア検出、i.MX6ULL の SNVS_TAMPER2 ピンに接続、CON3 の 1 ピンと共通
2	DSR	In	データセットレディ、i.MX6ULL の SNVS_TAMPER0 ピンに接続、CON3 の 6 ピンと共通
3	RXD	In	受信データ、i.MX6ULL の UART3_RX_DATA ピンに接続、CON3 の 2 ピンと共通
4	RTS	Out	送信要求、i.MX6ULL の UART3_CTS_B ピンに接続、CON3 の 7 ピンと共通
5	TXD	Out	送信データ、i.MX6ULL の UART3_TX_DATA ピンに接続、CON3 の 3 ピンと共通
6	CTS	In	送信可能、i.MX6ULL の UART3_RTS_B ピンに接続、CON3 の 8 ピンと共通
7	DTR	Out	データ端末レディ、i.MX6ULL の GPIO1_IO00 ピンに接続、CON3 の 4 ピンと共通
8	RI	In	被呼表示、i.MX6ULL の SNVS_TAMPER1 ピンに接続、CON3 の 9 ピンと共通
9	GND	Power	電源(GND)
10	VCC_3.3V	Power	電源出力(VCC_3.3V)

CON3 と CON4 は、共通の信号が接続されており、同時に使用することはできません。どちらか一方のコネクタでのみ、ご使用ください。

16.5. 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、下段の信号線は i.MX6ULL の USB コントローラ(USB OTG1)接続されています。上段の信号線はマルチプレクサを経由して、i.MX6ULL の USB コントローラ(USB OTG2)に接続されています。

マルチプレクサのセレクトピンは CON9 の 24 ピンで制御することが可能で、オープンもしくは High レベルを入力することで CON5 の上段、Low レベルを入力することで CON9 に USB OTG2 の接続先が変更されます。

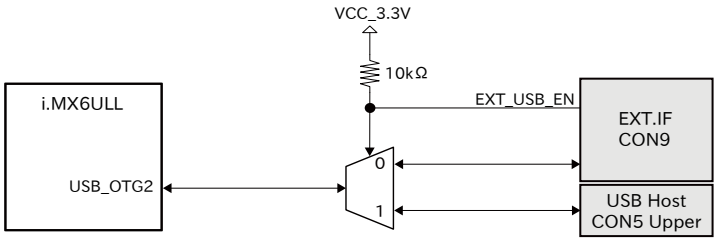


図 16.2 USB OTG2 の接続先の変更

下段に供給される電源(USB_OTG1_VBUS)は i.MX6ULL の UART1_RTS_B ピン(GPIO1_IO19)、上段に供給される電源(USB_OTG2_VBUS)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。

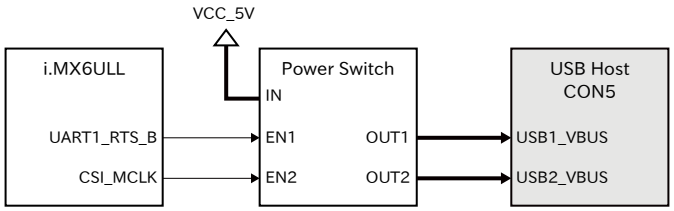


図 16.3 USB ホストインターフェースの電源制御

- データ転送モード
  - High Speed(480Mbps)
  - Full Speed(12Mbps)
  - Low Speed(1.5Mbps)

表 16.8 CON5 信号配列

ピン番号	ピン名	I/O	説明
1	USB1_VBUS	Power	電源(USB_OTG1_VBUS)、i.MX6ULL の USB_OTG1_VBUS ピンに接続
2	USB1_DN	In/Out	USB1 のマイナス側信号、i.MX6ULL の USB_OTG1_DN ピンに接続
3	USB1_DP	In/Out	USB1 のプラス側信号、i.MX6ULL の USB_OTG1_DP ピンに接続
4	GND	Power	電源(GND)
5	USB2_VBUS	Power	電源(USB2_OTG2_VBUS)、i.MX6ULL の USB_OTG2_VBUS ピンに接続
6	USB2_DN	In/Out	USB2 のマイナス側信号、i.MX6ULL の USB_OTG2_DN ピンに接続
7	USB2_DP	In/Out	USB2 のプラス側信号、i.MX6ULL の USB_OTG2_DP ピンに接続
8	GND	Power	電源(GND)

16.6. CON8、CON9、CON14(拡張インターフェース)

CON8、CON9、CON14 は機能拡張用のインターフェースです。複数の機能(マルチプレクス)をもった i.MX6ULL の信号線、パワーマネジメント IC の ON/OFF 信号、i.MX6ULL の PWRON 信号等が接続されています。

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 ユーザーズサイト」 [https://users.atmark-techno.com/]からダウンロードできる『Armadillo-640 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。

表 16.9 CON8 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	GND	Power	電源(GND)

表 16.10 CON9 信号配列

ピン番号	ピン名	I/O	説明
1	GPIO1_IO22	In/Out	拡張入出力、i.MX6ULL の UART2_CTS_B ピンに接続(CTS/RTS 信号線を利用する際の注意点)
2	GPIO1_IO23	In/Out	拡張入出力、i.MX6ULL の UART2_RTS_B ピンに接続(CTS/RTS 信号線を利用する際の注意点)
3	GPIO1_IO17	In/Out	拡張入出力、i.MX6ULL の UART1_RX_DATA ピンに接続
4	GPIO1_IO31	In/Out	拡張入出力、i.MX6ULL の UART5_RX_DATA ピンに接続
5	GPIO1_IO16	In/Out	拡張入出力、i.MX6ULL の UART1_TX_DATA ピンに接続
6	GPIO1_IO30	In/Out	拡張入出力、i.MX6ULL の UART5_TX_DATA ピンに接続
7	VCC_3.3V	Power	電源(VCC_3.3V)
8	VCC_3.3V	Power	電源(VCC_3.3V)
9	GND	Power	電源(GND)
10	GND	Power	電源(GND)
11	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_3V)されています。
12	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_3V)されています。
13	GPIO3_IO23	In/Out	拡張入出力、i.MX6ULL の LCD_DATA18 ピンに接続、基板上で 10kΩ プルダウンされています。
14	GPIO3_IO24	In/Out	拡張入出力、i.MX6ULL の LCD_DATA19 ピンに接続、基板上で 10kΩ プルダウンされています。
15	GPIO3_IO25	In/Out	拡張入出力、i.MX6ULL の LCD_DATA20 ピンに接続、基板上で 10kΩ プルダウンされています。
16	GPIO3_IO26	In/Out	拡張入出力、i.MX6ULL の LCD_DATA21 ピンに接続、基板上で 10kΩ プルダウンされています。
17	GPIO3_IO27	In/Out	拡張入出力、i.MX6ULL の LCD_DATA22 ピンに接続、基板上で 10kΩ プルダウンされています。
18	GPIO3_IO28	In/Out	拡張入出力、i.MX6ULL の LCD_DATA23 ピンに接続、基板上で 10kΩ プルダウンされています。
19	GND	Power	電源(GND)
20	VCC_3.3V	Power	電源(VCC_3.3V)
21	USB2_DN	In/Out	USB マイナス側信号、i.MX6ULL の USB_OTG2_DN ピンに接続
22	USB2_DP	In/Out	USB プラス側信号、i.MX6ULL の USB_OTG2_DP ピンに接続
23	USB2_VBUS	Power	電源(USB_OTG2_VBUS)、i.MX6ULL の USB_OTG2_VBUS ピンに接続
24	USB2_EN	In	USB OTG2 の切り替え信号、(Low: USB OTG2 を CON9 で使用する、High: USB OTG2 を CON5 で使用する)、基板上で 10kΩ プルアップ(VCC_3.3V)されています。
25	GPIO4_IO06	In/Out	拡張入出力、i.MX6ULL の NAND_DATA04 ピンに接続
26	GPIO4_IO07	In/Out	拡張入出力、i.MX6ULL の NAND_DATA05 ピンに接続
27	GPIO4_IO08	In/Out	拡張入出力、i.MX6ULL の NAND_DATA06 ピンに接続
28	GPIO4_IO09	In/Out	拡張入出力、i.MX6ULL の NAND_DATA07 ピンに接続

CTS/RTS 信号線を利用する際の注意点

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。誤接続に注意してください。

CON9 のブートモード設定ピンについて

CON9 の 17 ピン(GPIO3_IO27)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います。電源投入時、ブードモード設定のために、基板上のプルダウン抵抗で Low レベルの状態を保持しています。High レベルの状態で電源投入した場合、SPI EEPROM ブートを有効/無効にするためのピン(BOOT_CFG4[6])に割り当てられているため、EEPROM recovery モードが有効になり、意図しない動作を引き起こす原因となります。電源投入時から U-Boot が動作するまでは、Low レベルを保持した状態でご使用ください。ブートモード設定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.MX6ULL Applications Processor Reference Manual』をご参照ください。

表 16.11 CON14 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	GND	Power	電源(GND)
3	GPIO1_IO20	In/Out	拡張入出力、i.MX6ULL の UART2_TX_DATA ピンに接続
4	GPIO1_IO21	In/Out	拡張入出力、i.MX6ULL の UART2_RX_DATA ピンに接続

16.7. 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。

EXT_RESET_B ピンからシステムリセットを行うことが可能です。システムリセットを行う際は、「図 16.4. リセットシーケンス」のとおり、20ms 以上の Low 期間を設定してください。

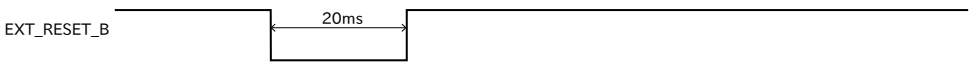


図 16.4 リセットシーケンス

CON10 に接続されている信号線は、JTAG 以外の機能でも使用可能です。詳細につきましては、「アットマークテクノ ユーザーズサイト」 [https://

users.atmark-techno.com/]からダウンロードできる『Armadillo-640 マルチプレクス表』をご参照ください。

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 SW1 に接続されています。

表 16.12 CON10 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_3.3V	Power	電源(VCC_3.3V)
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(VCC_3.3V)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(VCC_3.3V)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(VCC_3.3V)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(VCC_3.3V)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)

16.8. CON11(LCD 拡張インターフェース)

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 拡張インターフェースです。信号線は i.MX6ULL の LCD コントローラ等に接続されています。

CON11 に接続されている信号線は、LCD 以外の機能でも使用可能です。詳細につきましては、「アットマークテクノ ユーザーズサイト」 [https://users.atmark-techno.com/]からダウンロードできる『Armadillo-640 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。

表 16.13 CON11 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_5V	Power	電源出力(VCC_5V)
2	VCC_5V	Power	電源出力(VCC_5V)
3	VCC_5V	Power	電源出力(VCC_5V)

ピン番号	ピン名	I/O	説明
4	VCC_3.3V	Power	電源出力(VCC_3.3V)
5	VCC_3.3V	Power	電源出力(VCC_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、基板上で 10kΩ プルダウンされています。
14	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(VCC_3.3V)されています。
15	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。
16	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。
17	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。
18	LCD_DATA05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続、JP1 がオープン時、10kΩ プルアップ(VCC_3.3V)、ショート時、10kΩ プルダウンされます。
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、基板上で 10kΩ プルアップ(VCC_3.3V)されています。
21	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。
22	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。
23	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。
24	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。
25	LCD_DATA11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続、JP1 がオープン時、10kΩ プルダウン、ショート時、10kΩ プルアップ(VCC_3.3V)されます。
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、基板上で 10kΩ プルダウンされています。
28	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。
29	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。
30	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。
31	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。
32	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。
33	GND	Power	電源(GND)
34	XPUL	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	XNUR	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	YPLL	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	YNLR	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

ピン番号	ピン名	I/O	説明
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	GPIO4_IO22	In/Out	拡張入出力、i.MX6ULL の CSI_DATA01 ピンに接続
43	GPIO4_IO23	In/Out	拡張入出力、i.MX6ULL の CSI_DATA02 ピンに接続
44	GPIO4_IO28	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
45	GPIO4_IO27	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
46	GPIO4_IO26	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
47	GPIO4_IO25	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
48	GPIO4_IO20	In/Out	拡張入出力、i.MX6ULL の CSI_HSYNC ピンに接続
49	GPIO4_IO19	In/Out	拡張入出力、i.MX6ULL の CSI_VSYNC ピンに接続
50	GND	Power	電源(GND)

CON11 のブートモード設定ピンについて

CON11 の 13～18、20～25、27～32 ピン(LCD_DATA00～17)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います。電源投入時、ブードモード設定のために、基板上のプルアップ/ダウン抵抗で、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。ブートモード設定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.MX6ULL Applications Processor Reference Manual』をご参照ください。

16.9. CON12、CON13(電源インターフェース)

CON12、CON13 は電源供給用インターフェースです。

CON12 と CON13 の電源(VCC_5V)供給ラインは接続されていますので、同時に電源を供給することはできません。どちらか一方からのみ電源を供給してください。

CON12 には DC ジャックが実装されており、「図 16.5. 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。AC アダプタのジャック形状は JEITA RC-5320A 準拠(電圧区分 2)です。

図 16.5 AC アダプタの極性マーク

CON13 からは電源(VCC_5V)供給の他、バックアップ電源(RTC_BAT)供給、i.MX6ULL の ON/OFF 制御を行うことができます。バックアップ電源供給は、長時間電源を切断しても、i.MX6ULL の一部データ(時刻データ等)を保持したい場合にご使用ください。

表 16.14 CON13 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	電源(RTC_BAT)、パワーマネジメント IC の LICELL ピンに接続
2	VCC_5V	Power	電源(VCC_5V)
3	GND	Power	電源(GND)
4	ONOFF	In	i.MX6ULL の ON/OFF 用信号、i.MX6ULL の ONOFF ピンに接続、オープンドレイン入力

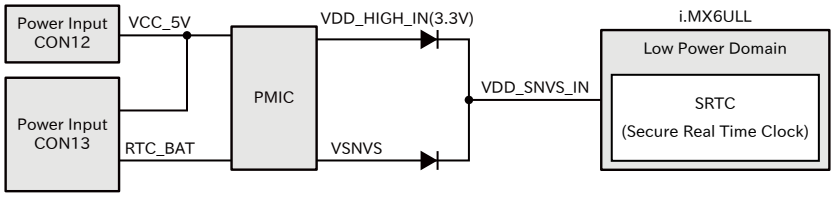


図 16.6 バックアップ電源供給

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、電源(VCC_5V)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 2.95V～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。

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早くなります。

バッテリ持続例: CR2032 の場合、約 4 か月

バッテリの消耗が製品の運用に支障をきたす場合には、消費電力が少ないリアルタイムクロック IC を外付けすることを推奨します。CON9(拡張インターフェース)に接続可能な Armadillo-600 シリーズ RTC オプションモジュールもありますので、ご検討ください。

16.10. LED3、LED4、LED5(ユーザー LED)

LED3、LED4、LED5 は、ユーザー側で自由に利用できる LED です。

表 16.15 LED3、LED4、LED5

部品番号	名称(色)	説明
LED3	ユーザー LED(赤)	i.MX6ULL の GPIO1_IO05 ピンに接続、(Low: 消灯、High: 点灯)
LED4	ユーザー LED(緑)	i.MX6ULL の GPIO1_IO08 ピンに接続、(Low: 消灯、High: 点灯)
LED5	ユーザー LED(黄)	i.MX6ULL の UART1_CTS_B ピンに接続、(Low: 消灯、High: 点灯)

16.11. 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。

表 16.16 SW1 信号配列

部品番号	名称	説明
SW1	ユーザースイッチ	i.MX6ULL の JTAG_MOD ピンに接続、(Low: 押されていない状態、High: 押された状態)

16.12. JP1、JP2(起動デバイス設定ジャンパ)

JP1、JP2 は起動デバイス設定ジャンパです。JP1、JP2 の状態で、起動デバイスを設定することができます。

表 16.17 ジャンパの設定と起動デバイス

JP1	JP2	起動デバイス
-	オープン	i.MX6ULL の eFUSE 設定[a]に基づいたデバイス eFUSE が未設定の場合は eMMC
オープン	ショート	eMMC
ショート	ショート	microSD

[a]eFUSE 設定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。

出荷時、i.MX6ULL の起動デバイスに関する eFUSE は未設定です。未設定のまま JP2 をオープン状態で使用すると、eMMC から起動します。

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-640 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。

JP2 をオープン状態で使用する場合、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。

表 16.18 JP1 信号配列

ピン番号	ピン名	I/O	説明
1	JP1	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(Low: LCD_DATA05 ピンはプルアップ、LCD_DATA11 ピンはプルダウンされます。High: LCD_DATA05 ピンはプルダウン、LCD_DATA11 ピンはプルアップされます。)、基板上で 47kΩ プルダウンされています。
2	JP1_PU	Out	VCC_3.3V で 1kΩ プルアップ

表 16.19 JP2 信号配列

ピン番号	ピン名	I/O	説明
1	JP2_PU	Out	VDD_SNVS_3V で 1kΩ プルアップ
2	JP2	In	起動デバイス設定用信号、i.MX6ULL の BOOT_MODE1 ピンに接続、i.MX6ULL 内部で 100kΩ プルダウンされています。

17. 基板形状図

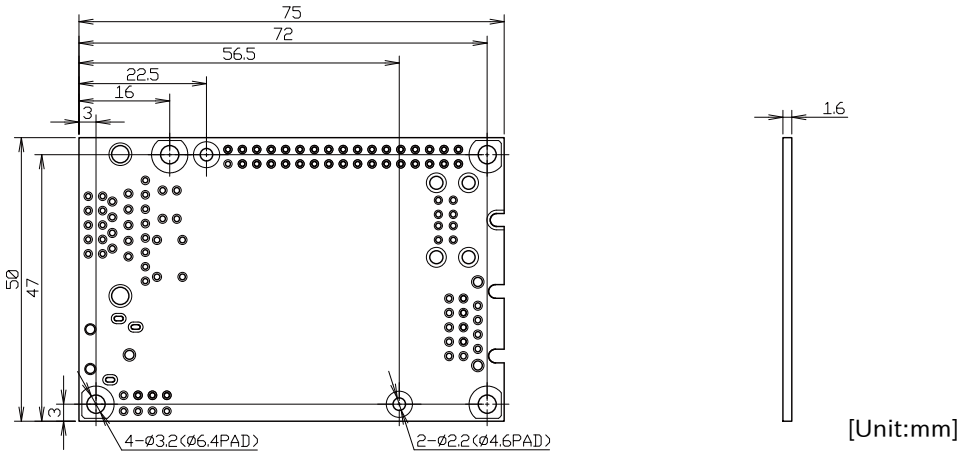


図 17.1 基板形状および固定穴寸法

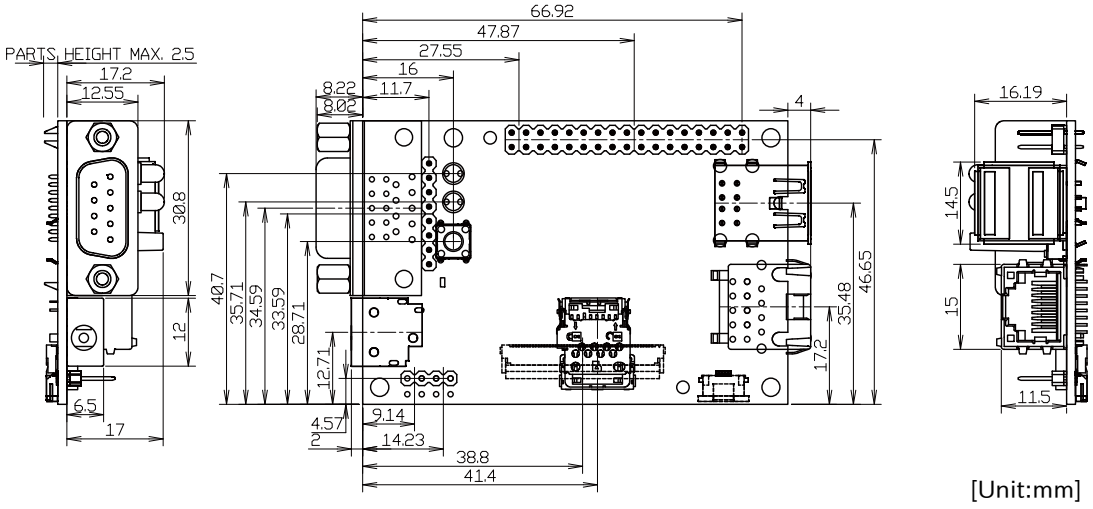
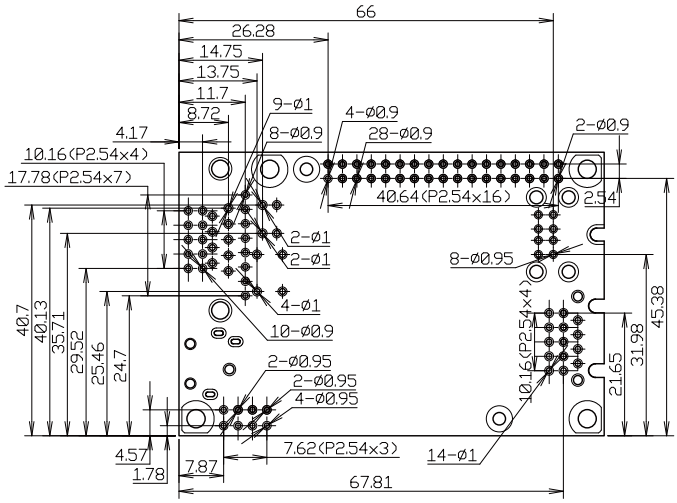


図 17.2 コネクタ中心寸法



[Unit:mm]

図 17.3 コネクタ穴寸法

DXF 形式の基板形状図を「アットマークテクノ ユーザーズサイト」から「購入者向けの限定公開データ」としてダウンロード可能です。

18. オプション品

本章では、Armadillo-640 のオプション品について説明します。

表 18.1 Armadillo-640 関連のオプション品

名称	型番	備考
USB シリアル変換アダプタ(Armadillo-640 用)	SA-SCUSB-10	Armadillo-640 ベーシックモデル開発セットに付属
Armadillo-600 シリーズ オプションケース(樹脂製)	OP-CASE600-PLA-00	Armadillo-640 ベーシックモデル開発セットに付属
Armadillo-600 シリーズ オプションケース(金属製)	OP-CASE600-MET-00	
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	
Armadillo-600 シリーズ RTC オプションモジュール	OP-A600-RTCMOD-00	
Armadillo-600 シリーズ WLAN オプションモジュール	OP-A600-AWLMOD-00	
Armadillo-600 シリーズ Thread オプションモジュール	OP-A600-THRMOD-00	Degu ゲートウェイ A6 に搭載
無線 LAN 用外付けアンテナセット 01	OP-ANT-WLAN-01W	

18.1. USB シリアル変換アダプタ(Armadillo-640 用)

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。

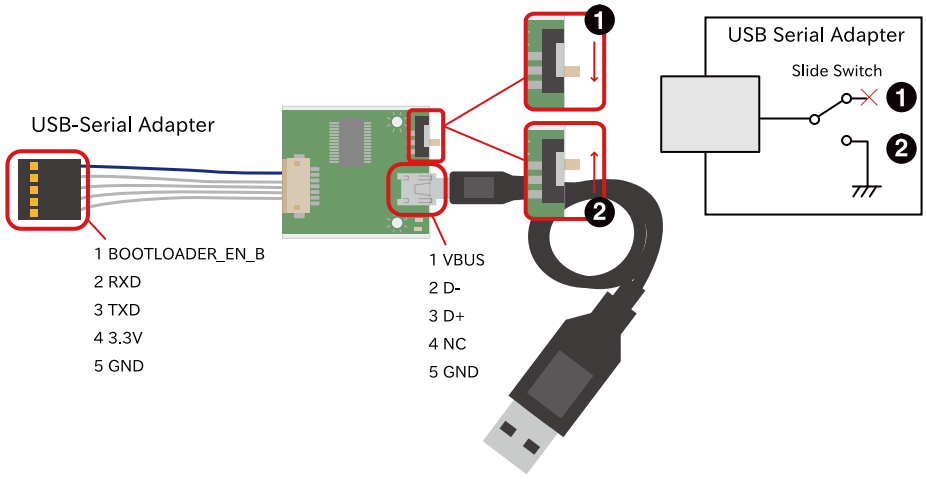


図 18.1 USB シリアル変換アダプタの配線

❶ オープン

❷ GND ショート

Armadillo-640 の CON9 の「1、3、5、7、9」または「2、4、6、8、10」ピンに接続して使用することが可能です。

各ピンに対応する UART コントローラは以下のとおりです。

表 18.2 各ピンに対応する UART コントローラ

CON9 ピン番号	UART 名
1、3、5、7、9	UART1
2、4、6、8、10	UART5

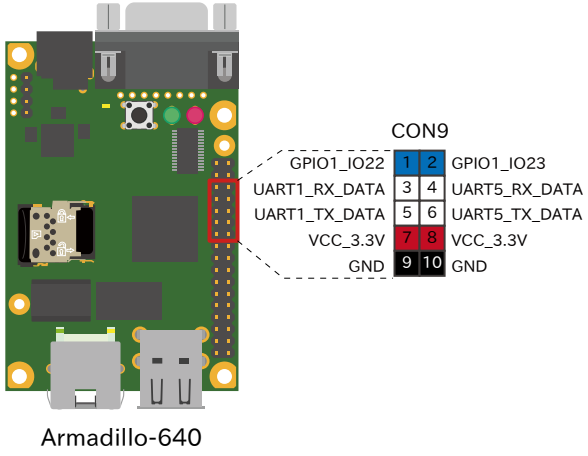


図 18.2 Armadillo-640 のシリアル信号線

ご使用の際は、USB シリアル変換アダプタの 5 ピンコネクタ(青いケーブル側)を Armadillo-640 CON9 の 1 ピンもしくは 2 ピンに合わせて接続してください。UART1 側で使用した場合、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 18.3 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-640 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-600 シリーズ オプションケース(樹脂製)

18.2.1. 概要

Armadillo-640 用のプラスチック製小型ケースです。Armadillo-640 の基板を収めた状態で、DC ジャック、シリアルインターフェース(D-Sub9 ピン)、USB インターフェース、LAN インターフェースにアクセス可能となっています。

取り外しが可能なパーツにより、CON9(拡張インターフェース)等の機能を外部に取り出すための開口部も用意しています。

Armadillo-600 シリーズ オプションケース(樹脂製)は Armadillo-640 ベーシックモデル開発セットに付属しています。樹脂ケースのみ必要なお客様のためにオプション品として別売りもしています。

表 18.4 Armadillo-600 シリーズ オプションケース(樹脂製)について

商品名	Armadillo-600 シリーズ オプションケース(樹脂製)
型番	OP-CASE600-PLA-00
内容	樹脂ケース、ネジ、ゴム足

表 18.5 樹脂ケース材料仕様

型番	VA55
グレード	難燃(標準一般)
ハロゲン	ハロゲン品種
UL94	V-0.5V
温度インデックス(最高使用温度)	50℃

最高使用温度よりも高い温度で保管または使用した場合、樹脂ケースが変形する可能性があります。

18.2.2. 組み立て

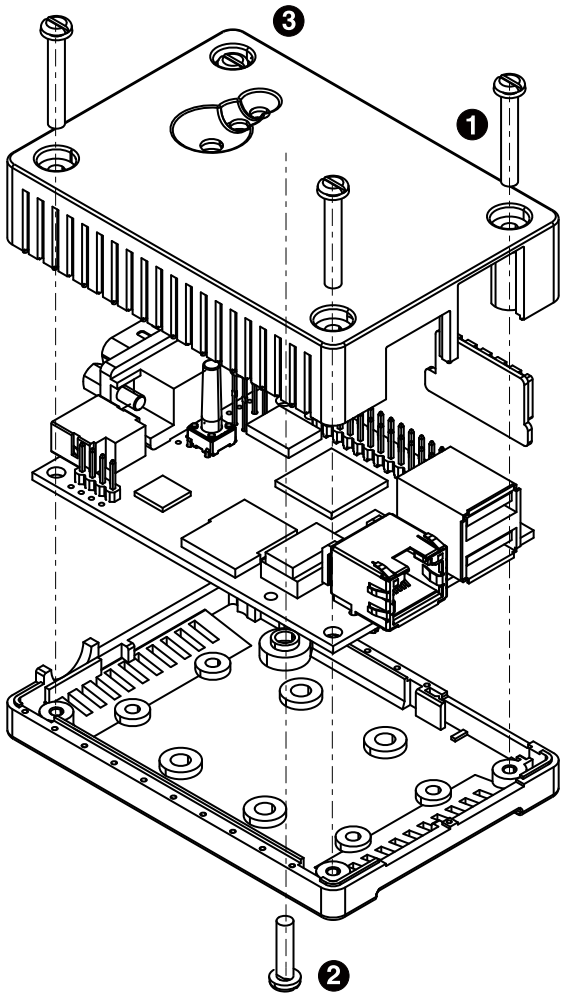


図 18.3 Armadillo-600 シリーズ オプションケース(樹脂製)の組み立て

❶ タッピングねじ(M2.6、L=20mm) x 3

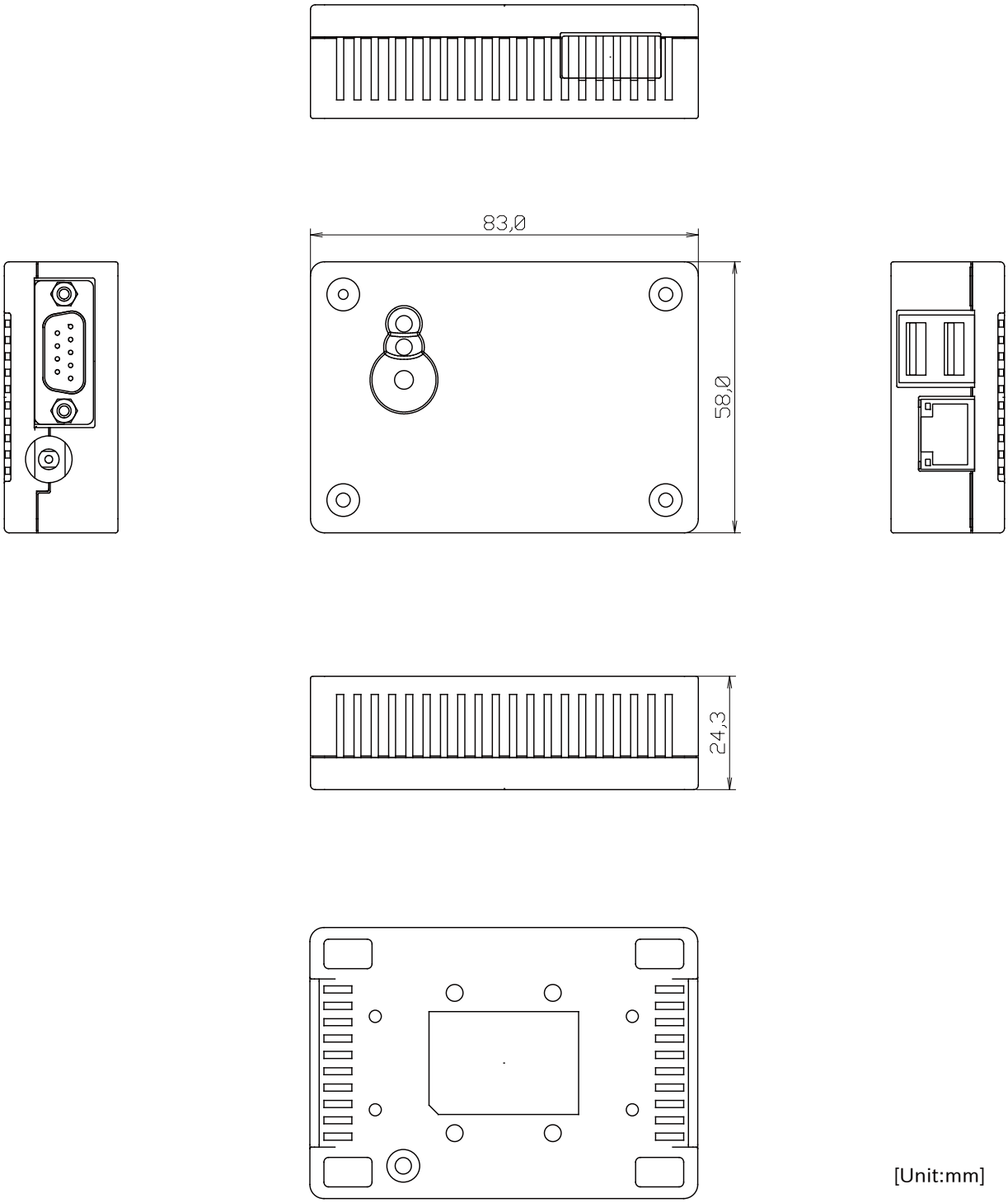
❷ タッピングねじ(M3、L=12mm) x 1

❸ 飾りねじ x 1

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

飾りネジはボンド止めされておりますので、無理に取り外さないで下さい。

18.2.3. 形状図



[Unit:mm]

図 18.4 樹脂ケース形状図

18.3. Armadillo-600 シリーズ オプションケース(金属製)

18.3.1. 概要

Armadillo-640 用のアルミ製小型ケースです。Armadillo-640 の基板を収めた状態で、DC ジャック、シリアルインターフェース(D-Sub9 ピン)、USB インターフェース、LAN インターフェースにアクセス可能となっています。ケース固定ネジを利用して、AC アダプタ固定用パーツおよびアース線を接続することが可能です。

金属ケースはオプション品として別売りしています。

表 18.6 Armadillo-600 シリーズ オプションケース(金属製)について

商品名	Armadillo-600 シリーズ オプションケース(金属製)
型番	OP-CASE600-MET-00
内容	アルミケース、ネジ、ゴム足、AC アダプタケーブル固定用パーツ

18.3.2. 組み立て

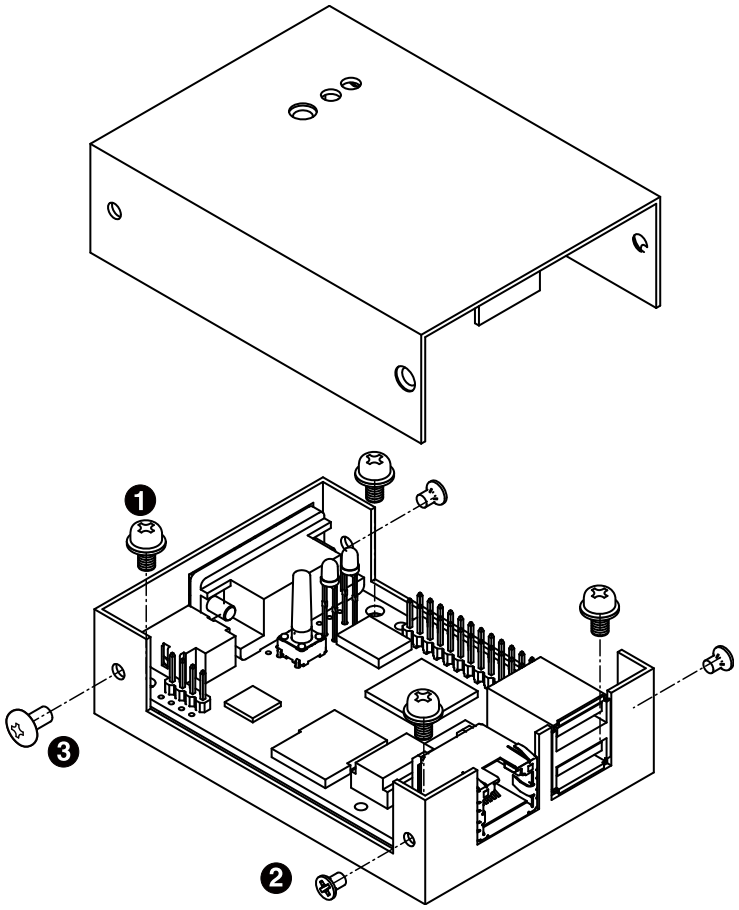


図 18.5 Armadillo-600 シリーズ オプションケース(金属製)の組み立て

❶ なべ小ネジ、スプリングワッシャ付き(M3、L=5mm) x 4

❷ 皿ネジ(M2.6、L=4mm) x 3

❸ トラス小ネジ(M3、L=5mm) x 1

18.3.3. 形状図

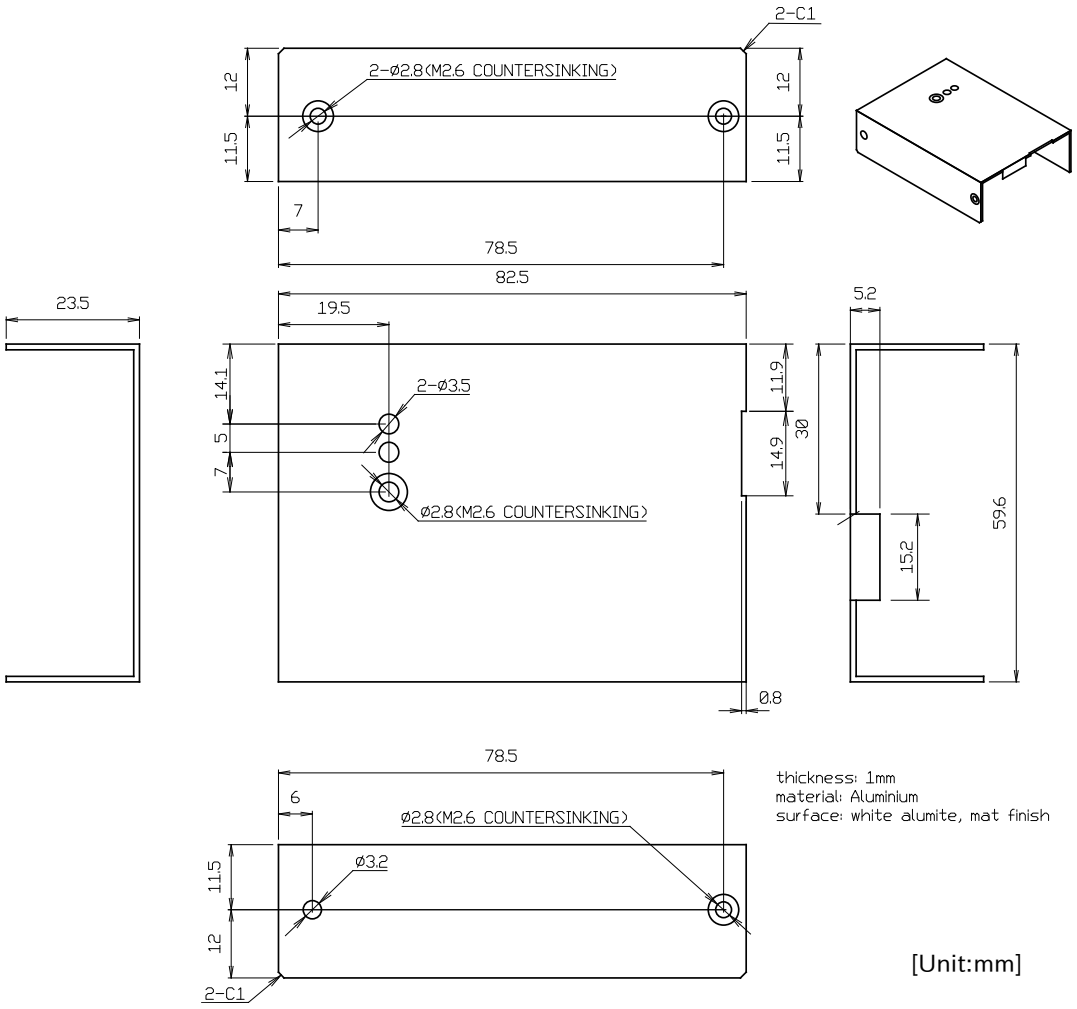


図 18.6 金属ケース(上板)寸法図

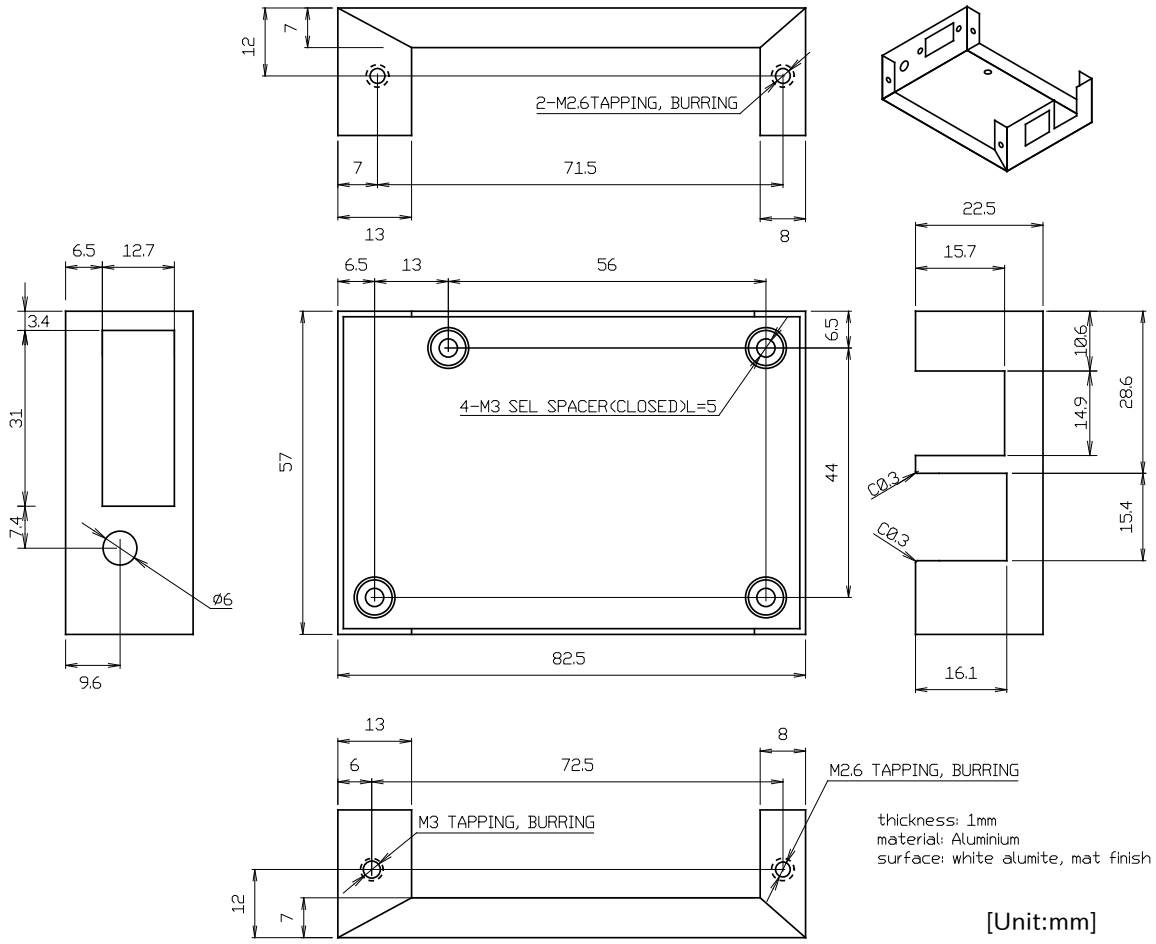


図 18.7 金属ケース(下板)寸法図

18.4. LCD オプションセット(7 インチタッチパネル WVGA 液晶)

18.4.1. 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。Armadillo-640 の CON11(LCD 拡張インターフェース)に接続して使用することが可能です。

表 18.7 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 18.8 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]Armadillo-640 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 ユーザーズサイト」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。

18.4.2. 組み立て

「図 18.8. 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-640 の CON11 の 1 ピンが対応するように、FFC を接続します。FFC の向きは、タッチパネル LCD 側は電極が下、Armadillo-640 側は電極が上になるようにします。

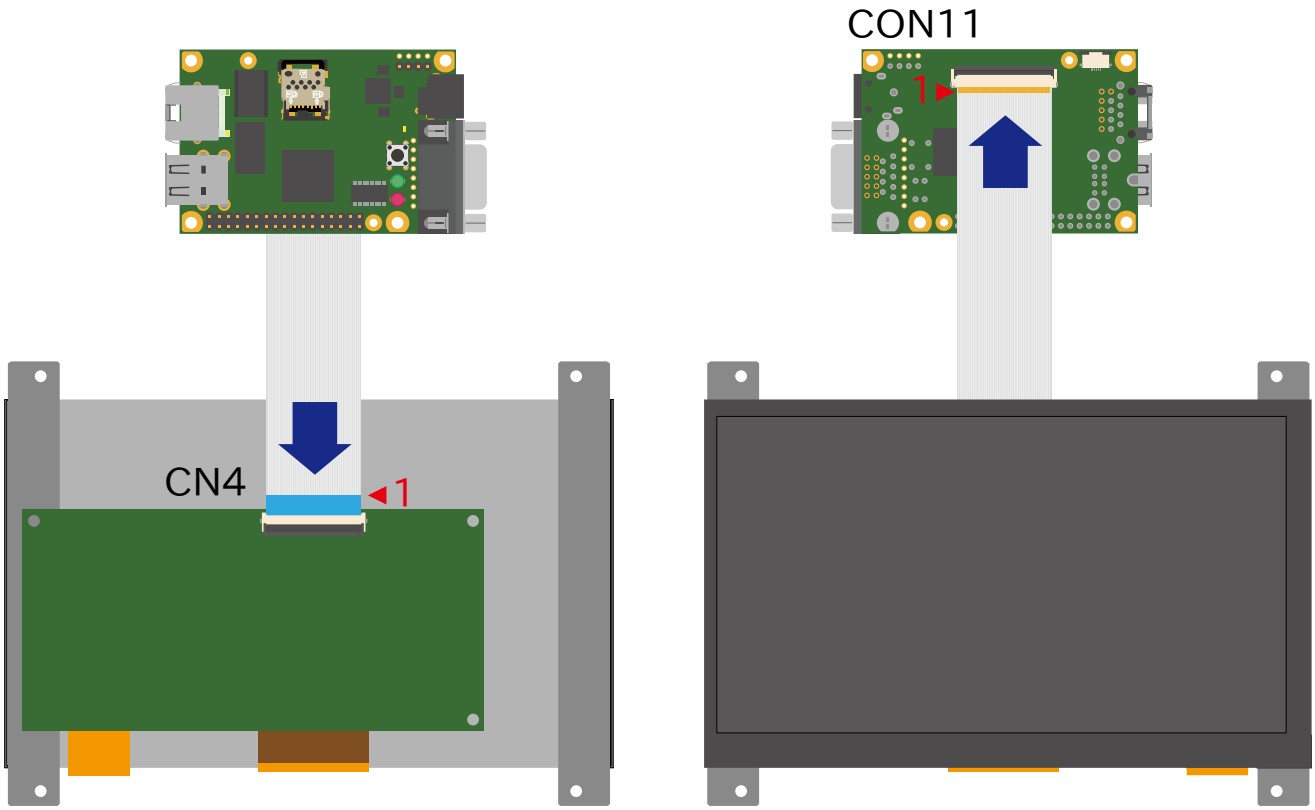


図 18.8 LCD の接続方法

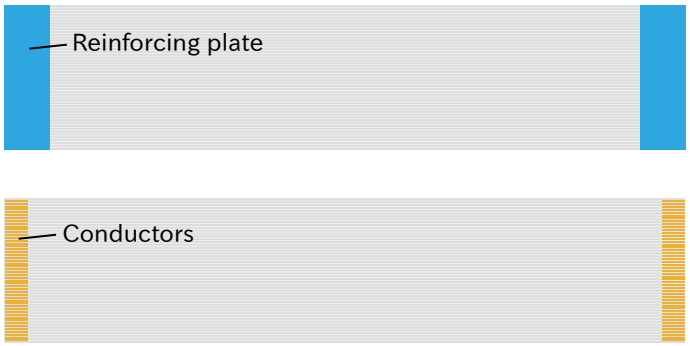


図 18.9 フレキシブルフラットケーブルの形状

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。

FFC の電極の上下を逆に接続した場合、Armadillo-640 の実装部品と電極が接触し、故障する可能性があります。

18.5. Armadillo-600 シリーズ RTC オプションモジュール

18.5.1. 概要

温度補償高精度リアルタイムクロック(RTC)と USB コネクタを搭載したオプションモジュールです。時刻データを保持するための電池ホルダも搭載しています。Armadillo-640 の CON9(拡張インターフェース)に接続して使用することが可能です。

Armadillo-600 シリーズ RTC オプションモジュールの回路図、部品表は「アットマークテクノ ユーザーズサイト」からダウンロード可能です。

表 18.9 Armadillo-600 シリーズ RTC オプションモジュールについて

商品名	Armadillo-600 シリーズ RTC オプションモジュール
型番	OP-A600-RTCMOD-00
内容	RTC オプションモジュール、ネジ、スペーサ

表 18.10 RTC オプションモジュールの仕様

リアルタイムクロック	型番	NR3225SA
	メーカー	日本電波工業
バックアップ	電池ホルダ、外部バッテリ接続コネクタ搭載(対応電池: CR2016、CR2032 WK11 等)	

平均月差(参考値)	約 21 秒@-20℃、約 8 秒@25℃、約 21 秒@70℃
USB	Type A コネクタ搭載
電源電圧	DC 3.3V±6%(RTC)、DC 2.0～3.5V(RTC バックアップ)、DC 5V±5%(USB)
使用温度範囲	-20～+70℃(結露なきこと) [a]
外形サイズ	42 x 50 mm

[a]Armadillo-600 シリーズ オプションケースセット(樹脂製)を組み合わせた場合、USB デバイスの発熱量によっては、使用温度範囲が狭くなる可能性があります。

RTC の時間精度は周囲温度に大きく影響を受けますので、ご使用の際には十分に特性の確認をお願いします。

18.5.2. ブロック図

RTC オプションモジュールのブロック図は次のとおりです。

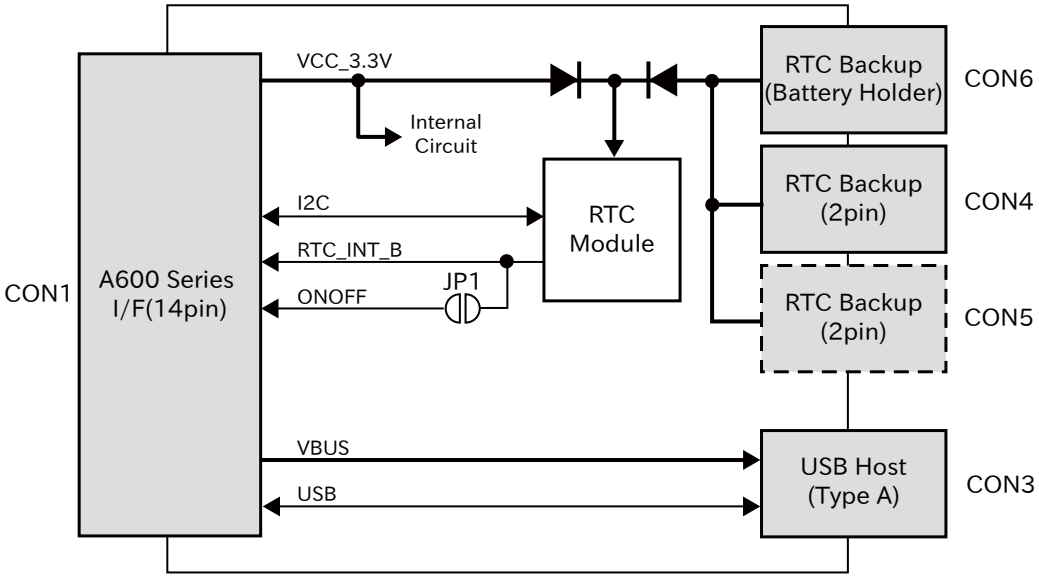


図 18.10 RTC オプションモジュールのブロック図

18.5.3. インターフェース仕様

RTC オプションモジュールのインターフェース仕様について説明します。

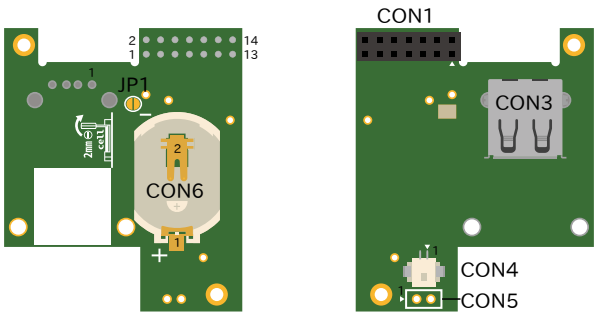


図 18.11 RTC オプションモジュールのインターフェース

表 18.11 RTC オプションモジュール 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
CON1	Armadillo-600 シリーズ接続インターフェース	PPPC072LFBN-RC	Sullins Connector Solutions
CON3	USB ホストインターフェース	SS-52100-001	Stewart Connector
CON4	RTC バックアップインターフェース	DF13A-2P-1.25H(21)	HIROSE ELECTRIC
CON5		B2B-EH(LF)(SN)	J.S.T.Mfg.
CON6		BLP2016SM-G	Memory Protection Devices

[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。

18.5.3.1. CON1(Armadillo-600 シリーズ接続インターフェース)

CON1 は Armadillo-600 シリーズの基板と接続するためのインターフェースです。Armadillo-640 の CON9(拡張インターフェース)の 11 ピンから 24 ピンに接続して使用します。

表 18.12 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	ONOFF	Out	JP1 を経由して RTC の割り込みピンに接続、オープンドレイン出力 [a]
2	NC	-	未接続
3	I2C_SCL	In	RTC の I2C クロックに接続、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
4	I2C_SDA	In/Out	RTC の I2C データピンに接続、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
5	Reserved	-	-
6	RTC_INT_B	Out	RTC の割り込みピンに接続、オープンドレイン出力、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
7	NC	-	未接続
8	USB2_PORT_EN	In	CON1 の 14 ピン(USB2_EN_B)に接続されているバッファのイネーブルピンに接続 (High: USB2_EN_B が Hi-Z、Low: USB2_EN_B が Low)
9	GND	Power	電源(GND)
10	VCC_3.3V	Power	電源(VCC_3.3V)
11	USB2_DN	In/Out	USB のマイナス側信号、CON3 の 2 ピン(D-)に接続
12	USB2_DP	In/Out	USB のプラス側信号、CON3 の 3 ピン(D+)に接続
13	USB2_VBUS	Power	電源(VBUS)、CON3 の 1 ピン(VBUS)に接続
14	USB2_EN_B	Out	バッファの出力ピンに接続、CON1 の 8 ピン(USB_PORT_EN)で設定した値が出力されます

[a]出荷時 JP1 はオープンですので、使用する場合は JP1 をショートする必要があります。

18.5.3.2. CON3(USB ホストインターフェース)

CON3 は USB ホストインターフェースです。

CON1 の 8 ピン(USB2_PORT_EN)から USB コントローラ(USB OTG2)の接続先を変更して使用します。Low レベルを入力すると RTC オプションモジュールの CON3、High レベルを入力すると Armadillo-640 の CON5 の上段に USB OTG2 が接続されます。詳細については、「16.5. CON5(USB ホストインターフェース)」および回路図をご確認ください。

RTC オプションモジュール CON3 と Armadillo-640 CON5 上段は同じ USB コントローラ(USB_OTG2)に接続されており、同時に使用できません。

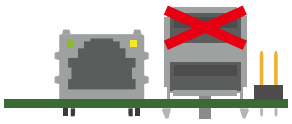


図 18.12 Armadillo-640 CON5 上段と RTC オプションモジュール CON3 は排他利用

CON3 は活線挿抜非対応となっております。電源を切断した状態でデバイスを挿抜してください。

表 18.13 CON3 信号配列

ピン番号	ピン名	I/O	説明
1	VBUS	Power	電源(VBUS)
2	D-	In/Out	USB のマイナス側信号、CON1 の 11 ピンに接続
3	D+	In/Out	USB のプラス側信号、CON1 の 12 ピンに接続
4	GND	Power	電源(GND)

18.5.3.3. CON4、CON5、CON6(RTC バックアップインターフェース)

CON4、CON5、CON6 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。別途バックアップ用のバッテリを接続することで、電源(VCC_3.3V)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。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。

表 18.14 対応バッテリ例

部品番号	対応電池	バックアップ時間(参考値)
CON4	CR2032 WK11	6.2 年
CON5 [a]	-	-
CON6	CR2016	2.5 年

[a]CON5 には部品が実装されていません。2.5mm ピッチのコネクタを実装してご使用ください。

表 18.15 CON4、CON5、CON6 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)

CON4、CON5、CON6 は共通の端子に接続されており、同時に使用できません。

CON4、CON5、CON6 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。

CON6 に搭載している電池ホルダは、大変破損しやすい部品となっております。電池の取り付け、取り外しの手順について、「18.5.4.2. 電池の取り付け、取り外し」をご確認ください。

18.5.3.4. JP1(ONOFF ピン接続ジャンパ)

JP1 は RTC のアラーム割り込み端子と Armadillo-640 の ONOFF ピン(Armadillo-640 CON9 11 ピン)を接続するジャンパです。JP1 をショートすることで、RTC のアラーム割り込みによる i.MX6ULL の電源制御が可能になります。

JP1 は、はんだジャンパになります。半分に割れたパッドになっておりますので、はんだごてでパッドを熱し、はんだを盛ってショートしてください。

18.5.4. 組み立て

18.5.4.1. Armadillo-640 と RTC オプションモジュールの組み立て

RTC オプションモジュールは Armadillo-640 の CON9 の 11 ピンから 24 ピンに接続します。「図 18.13. RTC オプションモジュールの組み立て」のようにコネクタ接続後、金属スペーサで固定してください。

Armadillo-640 CON1(microSD スロット)に microSD カードを挿抜する際には、RTC オプションモジュールを取り外してください。組み立て後は、RTC オプションモジュールと Armadillo-640 の間隔が狭いため、無理に挿抜した場合、microSD カードが正常に挿入されないなどの原因で、動作不良を起こす場合があります。

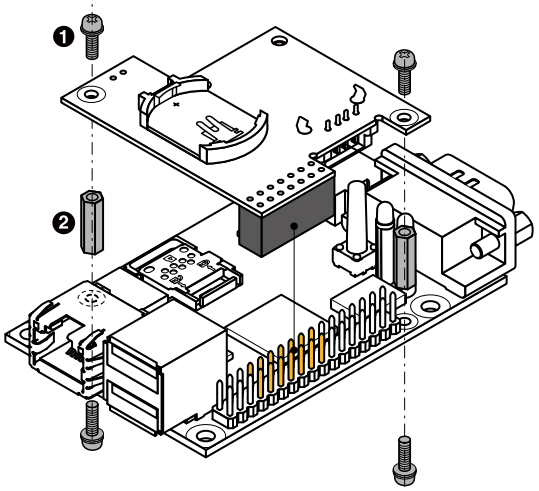


図 18.13 RTC オプションモジュールの組み立て

❶ なべ小ネジ、スプリングワッシャー、小径平ワッシャー付(M2、L=6mm) x 4

❷ 金属スペーサ(M2、L=11mm) x 2

18.5.4.2. 電池の取り付け、取り外し

RTC オプションモジュールの CON6(RTC バックアップインターフェース)には CR2016 等のリチウムコイン電池を搭載可能です。電池を取り付ける手順は以下のとおりです。

1. プラス端子側に電池を入れる
2. 電池ホルダのツメの下に電池を押し込む

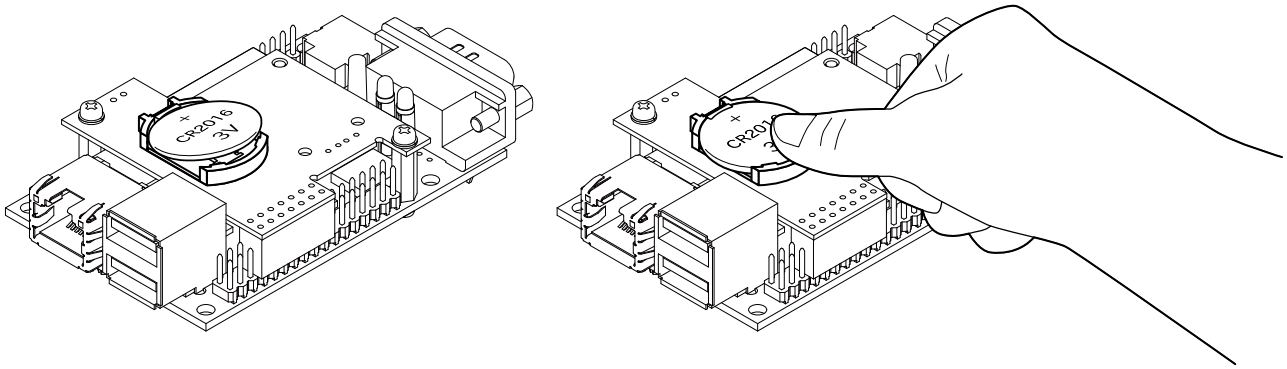


図 18.14 電池ホルダに電池を取り付ける

電池を取り外す手順は以下のとおりです。

1. プラスチック製もしくは絶縁テープを巻き付けたマイナスドライバー(2mm)を用意する
2. 電池を軽く押さえる
3. 電池ホルダの縁の中央部分と電池の間にマイナスドライバーを挿入する
4. マイナスドライバーで電池を持ち上げる

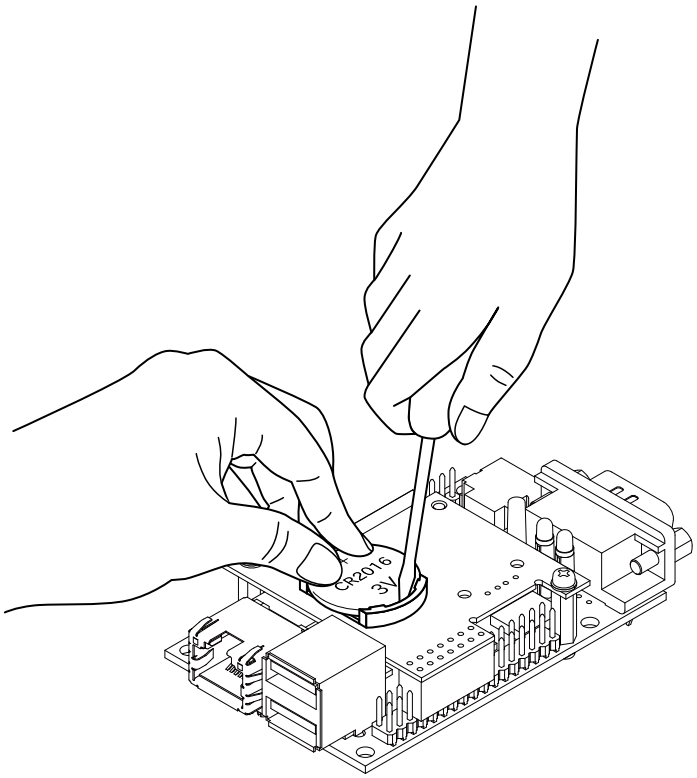


図 18.15 電池ホルダから電池を取り外す

18.5.5. 形状図

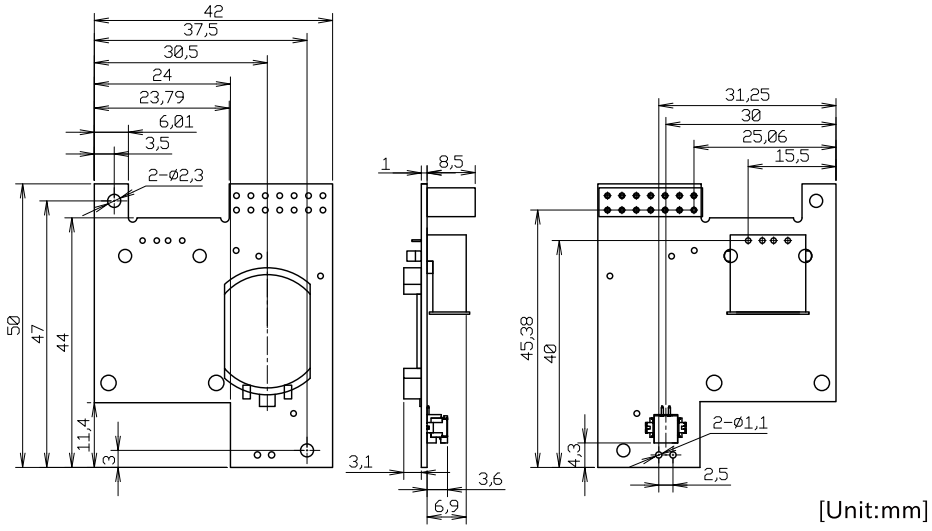


図 18.16 RTC オプションモジュール形状

18.5.6. 動作確認の前に

Armadillo-600 シリーズ RTC オプションモジュールには、次のバージョンのソフトウェアが対応しています。

表 18.16 Armadillo-600 シリーズ RTC オプションモジュール対応ソフトウェアバージョン

名称	対応バージョン	イメージファイル名
ブートローダー	at1 以降	u-boot-a600-v2018.03-at[version].imx
Linux カーネル	at7 以降	uImage-a600-v4.14-at[version]
WLAN/RTC オプションモジュール用 Device Tree Blob	at7 以降	armadillo-640_con9_awl13_rtc-v4.14-at[version].dtb
Debian GNU/Linux ルートファイルシステム	20180411 以降	debian-stretch-armhf-a600-[version].tar.gz

対応前のソフトウェアをご利用の場合は、イメージファイルの書き換えを行ってください。イメージファイルの書き換え方法については「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。

18.5.7. 動作確認

ここでは Armadillo-600 シリーズ RTC オプションモジュールの動作確認を行います。

18.5.7.1. 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
```

RTC オプションモジュール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、RTC オプションモジュール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。

18.5.7.2. RTC に時刻を設定する

「6.6. RTC」と同様の手順で RTC に時刻を設定することができます。システムクロックの設定方法については「6.6. RTC」をご参照ください。

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
2018-11-19 15:41:00.213911+0900
```

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。

18.5.7.3. RTC のアラーム割り込みを使用する

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

wakealarm ファイルに UNIX エポックからの経過秒数、または先頭に+を付けて現在時刻からの経過秒数を書き込むと、アラーム割り込み発生時刻を指定できます。

3600 秒後、アラーム割り込みを発生させるには、次のようにコマンドを実行します。

```
[armadillo ~]# echo +3600 > /sys/class/rtc/rtc0/wakealarm
```

図 18.17 アラーム割り込みの設定

WLAN/RTC オプションモジュールに搭載されている RTC NR3225SA は、毎分 0 秒にしかアラーム割り込みを発生させることができません。

0 時 0 分 30 秒の時に、1 秒後にアラームが鳴るように設定しても、実際にアラーム割り込みが発生するのは 0 時 1 分 0 秒となります。

sysfs RTC クラスディレクトリ以下の wakealarm ファイルからアラーム割り込みを発生させるには、Linux カーネル linux-v4.14-at10 以降である必要があります。

linux-v4.14-at10 以前の Linux カーネルでも、ioctl RTC_ALM_SET, ioctl RTC_AIE_ON を使用してアラーム割り込みを発生させることができます。

詳しくは、RTC の man ページを参照してください。

18.5.7.4. RTC オプションモジュールの USB ポートを使用する

RTC オプションモジュールの利用時には自動的に Armadillo-640 の USB_OTG2 の接続先が RTC オプションモジュール CON3 に切り替わります。

そのため、RTC オプションモジュール CON3 の USB ポートに USB メモリなどを接続し、使用することができます。ただし、Armadillo-640 CON5 上段の USB ポートは利用できなくなる点には注意してください。

```
[armadillo ~]# ls /dev/sd*
/dev/sda  /dev/sda1
```

図 18.18 USB メモリの接続確認

RTC オプションモジュール CON3 と Armadillo-640 CON5 上段は同じ USB コントローラ(USB_OTG2)に接続されており、同時に使用できません。

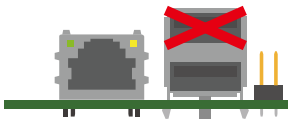


図 18.19 Armadillo-640 CON5 上段と RTC オプションモジュール CON3 は排他利用

Armadillo-640 CON5 USB ポートを使用したい場合は、Linux カーネルをカスタマイズする必要があります。

次のように DTS を修正することで、RTC オプションモジュール利用時に Armadillo-640 CON5 USB ポートを使用することができます。

```
[ATDE ~/linux-v4.14-at[version]]$ vi arch/arm/boot/dts/
armadillo-640_con9_awl13_rtc.dts
:（省略）
&gpio3 {
        usb2_port_en {
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_usb2_port_en>;

                gpio-hog;
                gpios = <28 GPIO_ACTIVE_HIGH>;
                line-name = "USB2_PORT_EN";
                output-high;     /* low: USB2 via CON9, high: USB2 via
CON5 */ ❶
        };
};
:（省略）
```

❶　output-low から output-high に変更

18.6. Armadillo-600 シリーズ WLAN オプションモジュール

18.6.1. 概要

Armadillo-WLAN モジュール(AWL13)と温度補償高精度リアルタイムクロック(RTC)を搭載したオプションモジュールです。チップアンテナと外付けアンテナ接続用の同軸コネクタ、時刻データを保持するための電池ホルダも搭載しています。

Armadillo-640 の CON9(拡張インターフェース)に接続して使用することが可能です。

Armadillo-600 シリーズ WLAN オプションモジュールの回路図、部品表は「アットマークテクノ ユーザーズサイト」からダウンロード可能です。

表 18.17 Armadillo-600 シリーズ WLAN オプションモジュールについて

商品名	Armadillo-600 シリーズ WLAN オプションモジュール
型番	OP-A600-AWLMOD-00
内容	WLAN オプションモジュール、ネジ、スペーサ

表 18.18 WLAN オプションモジュールの仕様

無線 LAN モジュール	型番	AWL13-U00Z
	メーカー	アットマークテクノ
無線 LAN 規格	IEEE 802.11b/g/n 対応(最大通信速度 72.2Mbps/理論値)	
リアルタイムクロック	型番	NR3225SA
	メーカー	日本電波工業
バックアップ	電池ホルダ、外部バッテリ接続コネクタ搭載(対応電池: CR2016、CR2032 WK11 等)	
平均月差(参考値)	約 21 秒@-20℃、約 8 秒@25℃、約 21 秒@70℃	
電源電圧	DC 3.3V±6%(メイン電源)、DC 2.0～3.5V(RTC バックアップ)	
使用温度範囲	-20～+70℃(結露なきこと) [a]	
外形サイズ	42 x 50 mm	

[a]Armadillo-600 シリーズ オプションケースセット(樹脂製)を組み合わせた場合の使用温度範囲は-10℃～+50℃です。(無線 LAN 送信出力を 6dBm に設定した場合)

Armadillo-WLAN モジュール(AWL13)の詳細な仕様については、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参照ください。

RTC の時間精度は周囲温度に大きく影響を受けますので、ご使用の際には十分に特性の確認をお願いします。

18.6.2. ブロック図

WLAN オプションモジュールのブロック図は次のとおりです。

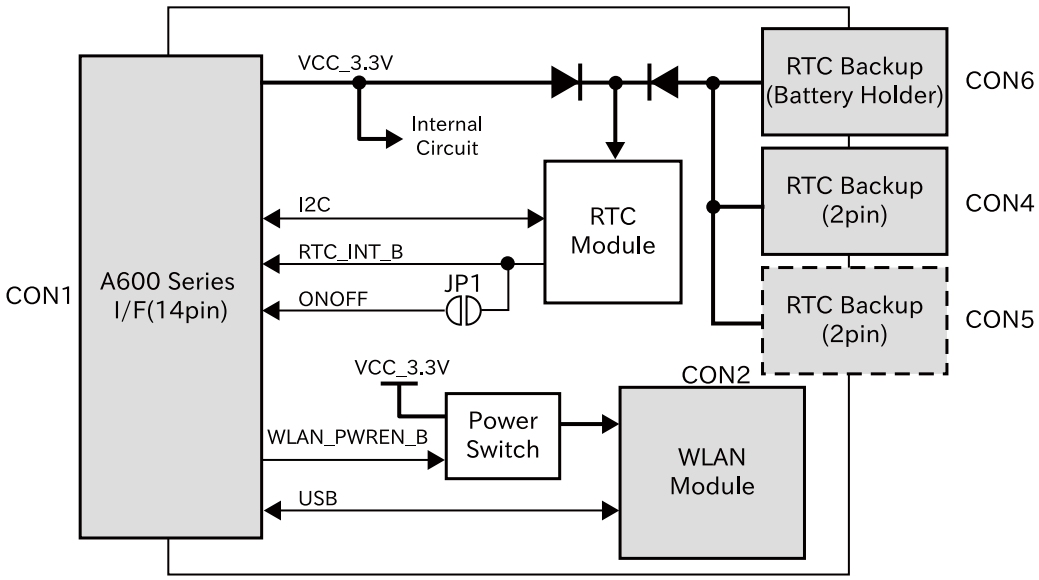


図 18.20 WLAN オプションモジュールのブロック図

18.6.3. インターフェース仕様

WLAN オプションモジュールのインターフェース仕様について説明します。

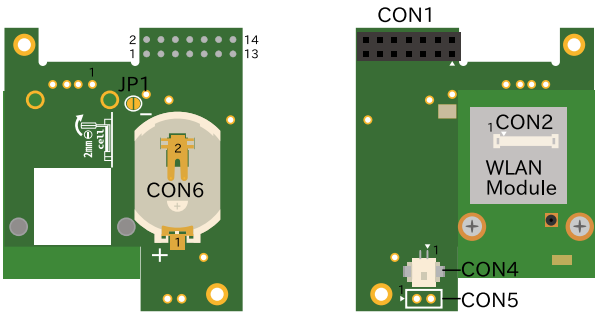


図 18.21 WLAN オプションモジュールのインターフェース

表 18.19 WLAN オプションモジュール 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-600 シリーズ接続インターフェース	PPPC072LFBN-RC	Sullins Connector Solutions
CON2 [b]	WLAN インターフェース	AXK6F34347YG	Panasonic
CON4	RTC バックアップインターフェース	DF13A-2P-1.25H(21)	HIROSE ELECTRIC
CON5		B2B-EH(LF)(SN)	J.S.T.Mfg.
CON6		BLP2016SM-G	Memory Protection Devices

[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。
[b]CON2 には Armadillo-WLAN モジュール(AWL13)が接続されています。

18.6.3.1. CON1(Armadillo-600 シリーズ接続インターフェース)

CON1 は Armadillo-600 シリーズの基板と接続するためのインターフェースです。Armadillo-640 の CON9(拡張インターフェース)の 11 ピンから 24 ピンに接続して使用します。

表 18.20 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	ONOFF	Out	JP1 を経由して RTC の割り込みピンに接続、オープンドレイン出力 [a]
2	NC	-	未接続
3	I2C_SCL	In	RTC の I2C クロックに接続、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
4	I2C_SDA	In/Out	RTC の I2C データピンに接続、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
5	WLAN_PWREN_B	In	WLAN モジュールのパワースイッチのイネーブルピンに接続、基板上で 2kΩ プルダウンされています。 (High: WLAN モジュールへの電源切断、Low: WLAN モジュールに電源供給)
6	RTC_INT_B	Out	RTC の割り込みピンに接続、オープンドレイン出力、基板上で 2kΩ プルアップ(VCC_3.3V)されています。
7	NC	-	未接続
8	USB2_PORT_EN	In	CON1 の 14 ピン(USB2_EN_B)に接続されているバッファのイネーブルピンに接続 (High: USB2_EN_B が Hi-Z、Low: USB2_EN_B が Low)
9	GND	Power	電源(GND)
10	VCC_3.3V	Power	電源(VCC_3.3V)
11	USB2_DN	In/Out	USB のマイナス側信号、CON3 の 2 ピン(D-)に接続
12	USB2_DP	In/Out	USB のプラス側信号、CON3 の 3 ピン(D+)に接続
13	USB2_VBUS	Power	電源(VBUS)、CON3 の 1 ピン(VBUS)に接続
14	USB2_EN_B	Out	バッファの出力ピンに接続、CON1 の 8 ピン(USB_PORT_EN)で設定した値が出力されます

[a]出荷時 JP1 はオープンですので、使用する場合は JP1 をショートする必要があります。

18.6.3.2. CON2(WLAN インターフェース)

CON2 には Armadillo-WLAN モジュール(AWL13)が接続されています。Armadillo-WLAN モジュール(AWL13)は、USB 起動モードに設定されています。

CON1 の 8 ピン(USB2_PORT_EN)から USB コントローラ(USB OTG2)の接続先を変更して使用します。Low レベルを入力すると WLAN オプションモジュールの CON2、High レベルを入力すると Armadillo-640 の CON5 の上段に USB OTG2 が接続されます。

CON1 の 5 ピン(WLAN_PWREN_B)から Armadillo-WLAN モジュール(AWL13)への電源の ON/OFF 制御をすることが可能です。Low レベルを入力すると電源が供給され、High レベルを入力すると電源が切断されます。

詳細については、「16.5. CON5(USB ホストインターフェース)」および回路図をご確認ください。

WLAN オプションモジュールの CON2 と Armadillo-640 CON5 上段は同じ USB コントローラ(USB_OTG2)に接続されているので、同時に使用できません。

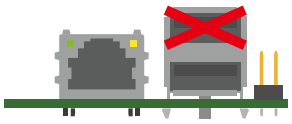


図 18.22 Armadillo-640 CON5 上段と WLAN オプションモジュール CON2 は排他利用

18.6.3.3. CON4、CON5、CON6(RTC バックアップインターフェース)

CON4、CON5、CON6 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。別途バックアップ用のバッテリを接続することで、電源(VCC_3.3V)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。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。

表 18.21 対応バッテリ例

部品番号	対応電池	バックアップ時間(参考値)
CON4	CR2032 WK11	6.2 年
CON5 [a]	-	-
CON6	CR2016	2.5 年

[a]CON5 には部品が実装されていません。2.5mm ピッチのコネクタを実装してご使用ください。

表 18.22 CON4、CON5、CON6 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)

CON4、CON5、CON6 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。

CON4、CON5、CON6 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。

CON6 に搭載している電池ホルダは、大変破損しやすい部品となっております。電池の取り付け、取り外しの手順について、「18.6.4.2. 電池の取り付け、取り外し」をご確認ください。

18.6.3.4. JP1(ONOFF ピン接続ジャンパ)

JP1 は RTC のアラーム割り込み端子と Armadillo-640 の ONOFF ピン(Armadillo-640 CON9 11 ピン)を接続するジャンパです。JP1 をショートすることで、RTC のアラーム割り込みによる i.MX6ULL の電源制御が可能になります。

JP1 は、はんだジャンパになります。半分に割れたパッドになっておりますので、はんだごてでパッドを熱し、はんだを盛ってショートしてください。

18.6.4. 組み立て

18.6.4.1. Armadillo-640 と WLAN オプションモジュールの組み立て

WLAN オプションモジュールは Armadillo-640 の CON9 の 11 ピンから 24 ピンに接続します。「図 18.23. WLAN オプションモジュールの組み立て」のように、コネクタ接続後、金属スペーサで固定してください。

Armadillo-640 CON1(microSD スロット)に microSD カードを挿抜する際には、WLAN オプションモジュールを取り外してください。組み立て後は、WLAN オプションモジュールと Armadillo-640 の間隔が狭いため、無理に挿抜した場合、microSD カードが正常に挿入されないなどの原因で、動作不良を起こす場合があります。

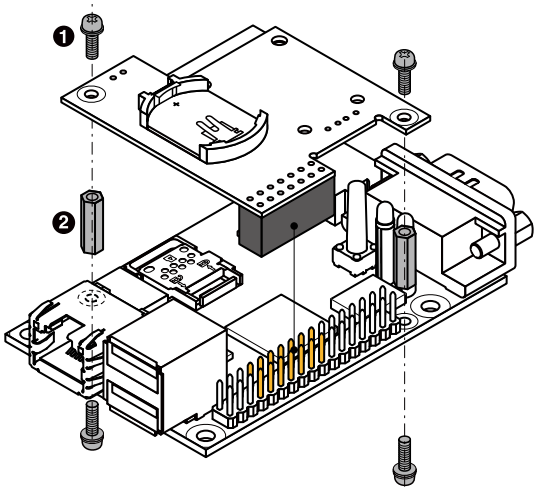


図 18.23 WLAN オプションモジュールの組み立て

❶ なべ小ネジ、スプリングワッシャー、小径平ワッシャー付(M2、L=6mm) x 4

❷ 金属スペーサ(M2、L=11mm) x 2

18.6.4.2. 電池の取り付け、取り外し

WLAN オプションモジュールの CON6(RTC バックアップインターフェース)には CR2016 等のリチウムコイン電池を搭載可能です。電池を取り付ける手順は以下のとおりです。

1. プラス端子側に電池を入れる
2. 電池ホルダのツメの下に電池を押し込む

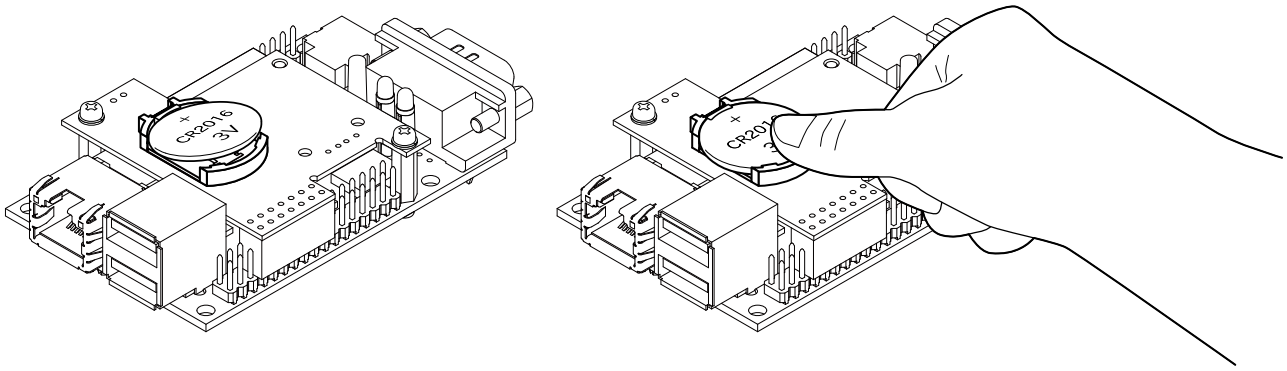


図 18.24 電池ホルダに電池を取り付ける

電池を取り外す手順は以下のとおりです。

1. プラスチック製もしくは絶縁テープを巻き付けたマイナスドライバー(2mm)を用意する
2. 電池を軽く押さえる
3. 電池ホルダの縁の中央部分と電池の間にマイナスドライバーを挿入する
4. マイナスドライバーで電池を持ち上げる

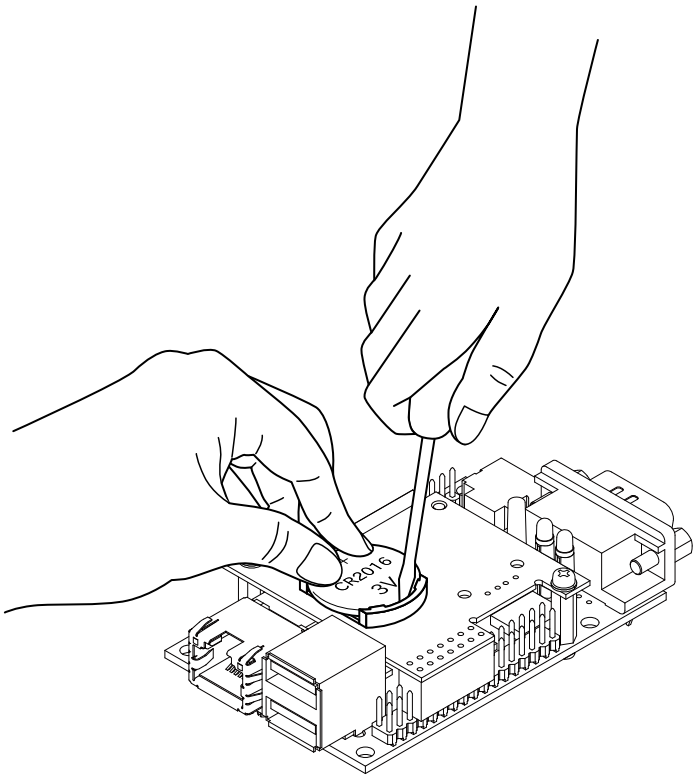


図 18.25 電池ホルダから電池を取り外す

18.6.5. 形状図

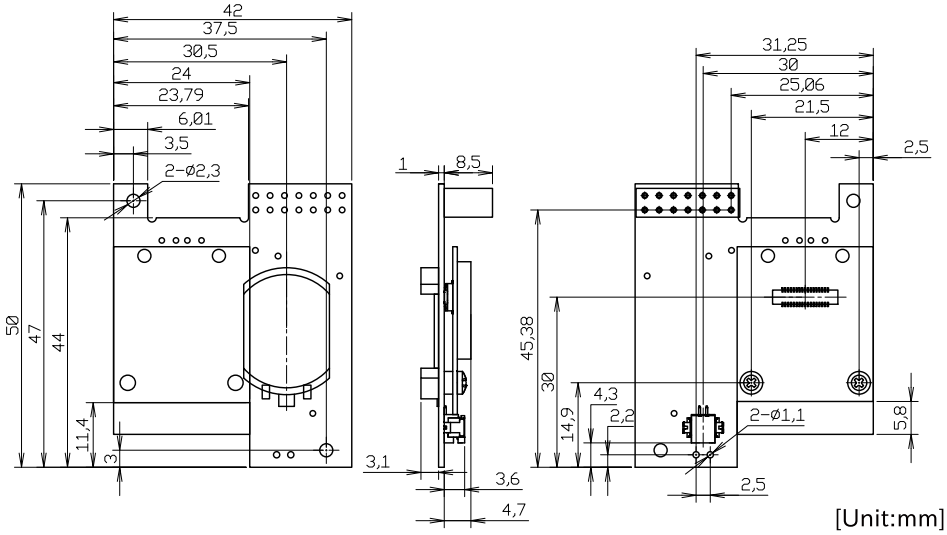


図 18.26 WLAN オプションモジュール形状

18.6.6. 動作確認の前に

Armadillo-600 シリーズ WLAN オプションモジュールには、次のバージョンのソフトウェアが対応しています。

表 18.23 Armadillo-600 シリーズ WLAN オプションモジュール対応ソフトウェアバージョン

名称	対応バージョン	イメージファイル名
ブートローダー	at1 以降	u-boot-a600-v2018.03-at[version].imx
Linux カーネル	at7 以降	uImage-a600-v4.14-at[version]
WLAN/RTC オプションモジュール用 Device Tree Blob	at7 以降	armadillo-640_con9_awl13_rtc-v4.14-at[version].dtb
Debian GNU/Linux ルートファイルシステム	20180411 以降	debian-stretch-armhf-a600-[version].tar.gz

対応前のソフトウェアをご利用の場合は、イメージファイルの書き換えを行ってください。イメージファイルの書き換え方法については「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。

18.6.7. 動作確認

ここでは Armadillo-600 シリーズ WLAN オプションモジュールの動作確認を行います。

18.6.7.1. AWL13 を使用するための準備

AWL13 を使用するためには、AWL13 の起動後にファームウェアをロードし、その後で通信設定を行う必要があります。

そのため、次のパッケージを Armadillo-640 にインストールします。

表 18.24 AWL13 に使用する際に必要なパッケージ

パッケージ名	説明
awl13-usb-firmwares	AWL13 にファームウェアや AWL13 の認識時にファームウェアを自動的にロードするスクリプトを含んだパッケージ
wireless-tools	AWL13 に無線 LAN の設定を行うツールを含んだパッケージ

```
[armadillo ~]# apt-get install awl13-usb-firmwares
[armadillo ~]# apt-get install wireless-tools
```

図 18.27 awl13-usb-firmwares と wireless-tools のインストール

18.6.7.2. AWL13 を使用して無線 LAN アクセスポイントに接続する

AWL13 を使用して無線 LAN アクセスポイントに接続するには、AWL13 に"STA"(ステーション用ファームウェア)がロードされている必要があります。

AWL13 に"STA"をロードするには、次のように /etc/awlan/awl13.conf を修正し、Armadillo を再起動します。

1. AWL13 に"STA"がロードされるよう設定を変更する

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

2. Armadillo を再起動する

```
[armadillo ~]# reboot
```

AWL13 に"STA"をロードしたらネットワーク設定を行います。

AWL13 のネットワークデバイスは awlan0 となります。設定方法は「6.2. ネットワーク」と変わりません。

「6.2.3.3. 固定 IP アドレスに設定する」と同じ設定を AWL13 で行う場合、 /etc/network/interfaces の末尾に次のように設定を追加します。

```
[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 dhcp

iface awlan0 inet static  ❶
        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
        pre-up iwpriv awlan0 set_psk [passphrase]  ❷
```

```
        pre-up iwpriv awlan0 set_cryptmode WPA2-AES ❸
        pre-up iwconfig awlan0 essid [essid] ❹
        wireless-mode managed ❺
```

図 18.28 無線 LAN 固定 IP 設定

❶　iface に awlan0 を指定します。

❷　PSK を設定します。

❸　暗号化方式 WPA2-AES に設定します。

❹　ESSID を設定します。

❺　無線をアクティブにします。

ネットワークの設定が完了後、インターフェースを有効化することで、無線 LAN アクセスポイントに接続できます。

```
[armadillo ~]# ifup awlan0
[armadillo ~]# ping -I awlan0 -c 3 192.0.2.30
PING 192.0.2.30 (192.0.2.30) 56(84) bytes of data.
64 bytes from 192.0.2.30: icmp_seq=1 ttl=63 time=1.39 ms
64 bytes from 192.0.2.30: icmp_seq=2 ttl=63 time=1.35 ms
64 bytes from 192.0.2.30: icmp_seq=3 ttl=63 time=1.34 ms

--- 192.0.2.30 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 18.29 インターフェースの有効化と無線 LAN アクセスポイントの PING 確認

AWL13 には、通信レートや出力調整の変更、WPS プッシュボタン(PBC)方式による無線設定機能などもあります。

詳しくは Armadillo-WLAN(AWL13) ソフトウェアマニュアルをご参照ください。

Armadillo-WLAN(AWL13) ソフトウェアマニュアルを参照する際には、AWL13 の SYSFS インターフェースのパスを読み替えてご利用ください。

- Armadillo-WLAN(AWL13) ソフトウェアマニュアルに記載の SYSFS インターフェースのパス

  /sys/module/awl13_[mode]/awlan0/

- Armadillo-640 で AWL13 利用時の SYSFS インターフェースのパス

  /sys/class/net/awlan0/awl13/firmware

Armadillo-WLAN(AWL13) ソフトウェアマニュアルは、ユーザーズサイト - Armadillo-WLAN 製品マニュアル・ドキュメントからご利用いただけます。

ユーザーズサイト - Armadillo-WLAN 製品マニュアル・ドキュメント

https://users.atmark-techno.com/armadillo-wlan/awl13/manual

18.6.7.3. AWL13 を使用して無線 LAN アクセスポイントを作成する

AWL13 を使用して無線 LAN アクセスポイントを作成するには、AWL13 に"AP"(アクセスポイント用ファームウェア)がロードされている必要があります。

AWL13 に"AP"をロードするには、次のように /etc/awlan/awl13.conf を修正し、Armadillo を再起動します。

1. AWL13 に"AP"がロードされるよう設定を変更する

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

2. Armadillo を再起動する

```
[armadillo ~]# reboot
```

AWL13 に"AP"をロードしたらネットワーク設定を行います。

AWL13 のネットワークデバイスは awlan0 となります。設定方法は「6.2. ネットワーク」と変わりません。

無線 LAN アクセスポイントを「表 18.25. 無線 LAN アクセスポイント設定例」の内容に設定する例を、以下に示します。

表 18.25 無線 LAN アクセスポイント設定例

項目	設定
ESSID	myap
チャンネル	1
PSK パスフレーズ	mypresharedkey
暗号化方式	WPA2-PSK(AES)
IP アドレス	192.0.2.30
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

```
[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 dhcp

iface awlan0 inet static ❶
    address 192.0.2.30
    netmask 255.255.255.0
    network 192.0.2.0
    broadcast 192.0.2.255
    pre-up iwpriv awlan0 set_psk mypresharedkey ❷
    pre-up iwpriv awlan0 set_cryptmode WPA2-AES ❸
    pre-up iwconfig awlan0 essid myap ❹
    pre-up iwconfig awlan0 channel 1 ❺
    wireless-mode master ❻
```

図 18.30 無線 LAN アクセスポイント設定

❶　iface に awlan0 を指定します。

❷　PSK を設定します。

❸　暗号化方式を設定します。

❹　ESSID を設定します。

❺　チャンネルを設定します。

❻　無線をアクティブにします。

ネットワークの設定が完了後、インターフェースを有効化することで、無線 LAN アクセスポイントを作成できます。

```
[armadillo ~]# ifup awlan0
```

図 18.31 インターフェースの有効化

AWL13 には、通信レートや出力調整の変更、WPS プッシュボタン(PBC)方式による無線設定機能などもあります。

詳しくは Armadillo-WLAN(AWL13) ソフトウェアマニュアルをご参照ください。

Armadillo-WLAN(AWL13) ソフトウェアマニュアルを参照する際には、AWL13 の SYSFS インターフェースのパスを読み替えてご利用ください。

- Armadillo-WLAN(AWL13) ソフトウェアマニュアルに記載の SYSFS インターフェースのパス

  /sys/module/awl13_[mode]/awlan0/

- Armadillo-640 で AWL13 利用時の SYSFS インターフェースのパス

/sys/class/net/awlan0/awl13/firmware

Armadillo-WLAN(AWL13) ソフトウェアマニュアルは、ユーザーズサイト - Armadillo-WLAN 製品マニュアル・ドキュメントからご利用いただけます。

ユーザーズサイト - Armadillo-WLAN 製品マニュアル・ドキュメント

https://users.atmark-techno.com/armadillo-wlan/awl13/manual

18.6.7.4. 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
```

RTC オプションモジュール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、RTC オプションモジュール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。

18.6.7.5. RTC に時刻を設定する

「6.6. RTC」と同様の手順で RTC に時刻を設定することができます。システムクロックの設定方法については「6.6. RTC」をご参照ください。

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
2018-11-19 15:41:00.213911+0900
```

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。

18.6.7.6. RTC のアラーム割り込みを使用する

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

wakealarm ファイルに UNIX エポックからの経過秒数、または先頭に+を付けて現在時刻からの経過秒数を書き込むと、アラーム割り込み発生時刻を指定できます。

3600 秒後、アラーム割り込みを発生させるには、次のようにコマンドを実行します。

```
[armadillo ~]# echo +3600 > /sys/class/rtc/rtc0/wakealarm
```

図 18.32 アラーム割り込みの設定

WLAN/RTC オプションモジュールに搭載されている RTC NR3225SA は、毎分 0 秒にしかアラーム割り込みを発生させることができません。

0 時 0 分 30 秒の時に、1 秒後にアラームが鳴るように設定しても、実際にアラーム割り込みが発生するのは 0 時 1 分 0 秒となります。

sysfs RTC クラスディレクトリ以下の wakealarm ファイルからアラーム割り込みを発生させるには、Linux カーネル linux-v4.14-at10 以降である必要があります。

linux-v4.14-at10 以前の Linux カーネルでも、ioctl RTC_ALM_SET, ioctl RTC_AIE_ON を使用してアラーム割り込みを発生させることができます。

詳しくは、RTC の man ページを参照してください。

18.7. Armadillo-600 シリーズ Thread オプションモジュール

18.7.1. 概要

Thread 通信モジュールと Armadillo-WLAN モジュール(AWL13)、セキュアエレメントを搭載したオプションモジュールです。Armadillo-640 の CON14(拡張インターフェース)、CON9(拡張インターフェース)に接続して使用します。

Thread オプションモジュールは Degu ゲートウェイ A6 に搭載されています。

Thread 通信モジュールと Armadillo-WLAN モジュール(AWL13)が Armadillo-640 の USB コントローラ(USB OTG2)を使用するため、Armadillo-640 の CON5 の上段は使用できなくなります。詳細については、「16.5. CON5(USB ホストインターフェース)」および回路図をご確認ください。

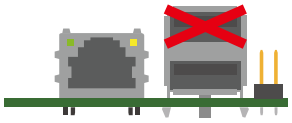


図 18.33 Thread オプションモジュール接続時 Armadillo-640 CON5 上段は使用不可

Armadillo-600 シリーズ Thread オプションモジュールの回路図、部品表は 「アットマークテクノ ユーザーズサイト」 [https://users.atmark-techno.com/]からダウンロード可能です。

表 18.26 Armadillo-600 シリーズ Thread オプションモジュールについて

商品名	Armadillo-600 シリーズ Thread オプションモジュール
型番	OP-A600-THRMOD-00
内容	Thread オプションモジュール、ネジ、スペーサ

表 18.27 Thread オプションモジュールの仕様

Thread 通信モジュール	型番	EYSKBNZWB
	メーカー	太陽誘電
	無線規格	Bluetooth 5.0/IEEE 802.15.4(Thread)
無線 LAN モジュール	型番	AWL13-U00Z
	メーカー	アットマークテクノ
	無線規格	IEEE 802.11b/g/n 対応(アクセスポイントモード)
セキュアエレメント	型番	A71CH
	メーカー	NXP Semiconductors
電源電圧	DC 3.3V±6%、DC 5V±5%	
使用温度範囲	-20～+70℃(結露なきこと) [a]	
外形サイズ	42 x 50 mm	

[a]Armadillo-600 シリーズ オプションケースセット(樹脂製)を組み合わせた場合の使用温度範囲は-10℃～+40℃です。

Thread 通信モジュール、セキュアエレメントの詳細な仕様については各メーカーサイトにてご確認ください。

Armadillo-WLAN モジュール (AWL13) の詳細な仕様については、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。

18.7.2. ブロック図

Thread オプションモジュールのブロック図は次のとおりです。

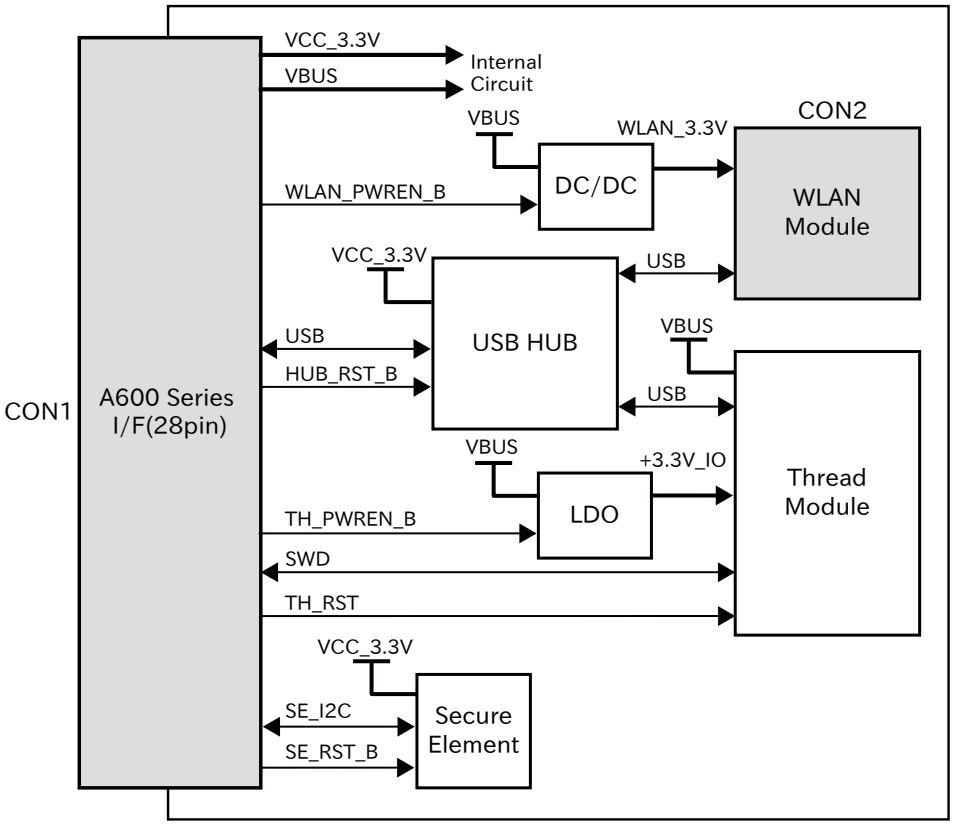


図 18.34 Thread オプションモジュールのブロック図

18.7.3. インターフェース仕様

Thread オプションモジュールのインターフェース仕様について説明します。

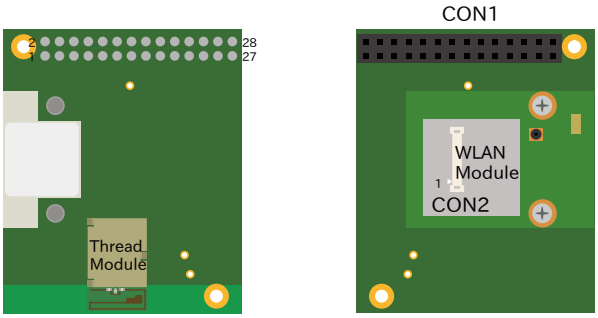


図 18.35 Thread オプションモジュールのインターフェース

表 18.28 Thread オプションモジュール 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-600 シリーズ接続インターフェース	PPPC142LFBN-RC	Sullins Connector Solutions
CON2 [b]	WLAN インターフェース	AXK6F34347YG	Panasonic

[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。
[b]CON2 には Armadillo-WLAN モジュール(AWL13)が接続されています。

18.7.3.1. CON1 (Armadillo-600 シリーズ接続インターフェース)

CON1 は Armadillo-600 シリーズの基板と接続するためのインターフェースです。

表 18.29 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	GND	Power	電源(GND)
3	SE_I2C_SCL	In	セキュアエレメントの I2C クロックに接続されています。
4	SE_I2C_SDA	In/Out	セキュアエレメントの I2C データに接続されています。
5	NC	-	未接続
6	SWDCLK	In	Thread 通信モジュールの SWDCLK に接続されています。
7	NC	-	未接続
8	SWDIO	In/Out	Thread 通信モジュールの SWDIO に接続されています。
9	NC	-	未接続
10	HUB_RESET_B	In	USB HUB をリセットするための信号です。 (High: リセット解除、Low: リセット)
11	VCC_3.3V	Power	電源(VCC_3.3V)
12	VCC_3.3V	Power	電源(VCC_3.3V)
13	GND	Power	電源(GND)
14	GND	Power	電源(GND)
15	NC	-	未接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	WLAN_PWREN_B	In	WLAN モジュールへの電源を ON/OFF 制御するための信号です。 (High: 電源切断、Low: 電源供給)
20	TH_RESET	In	Thread 通信モジュールをリセットするための信号です。 (High: リセット、Low: リセット解除)
21	TH_PWREN_B	In	Thread 通信モジュールの電源を ON/OFF 制御するための信号です。 (High: 電源切断、Low: 電源供給)
22	SE_RESET_B	In	セキュアエレメントをリセットするための信号です。 (High: リセット解除、Low: リセット)
23	GND	Power	電源(GND)

ピン番号	ピン名	I/O	説明
24	VCC_3.3V	Power	電源(VCC_3.3V)
25	USB2_DN	In/Out	USB のマイナス側信号です。
26	USB2_DP	In/Out	USB のプラス側信号です。
27	USB2_VBUS	Power	電源(VBUS)
28	USB2_EN_B	Out	Armadillo-640 の USB OTG2 の接続先を切り替えるための信号です。GND に接続されています。

18.7.3.2. CON2(WLAN インターフェース)

CON2 には Armadillo-WLAN モジュール(AWL13)が接続されています。Armadillo-WLAN モジュール(AWL13)は、USB 起動モードに設定されています。

18.7.4. 組み立て

Thread オプションモジュールは Armadillo-640 の CON14(拡張インターフェース)の 1 ピンから 4 ピン、CON9(拡張インターフェース)の 1 ピンから 24 ピンに接続します。電波強度等に影響がでますので、Armadillo-640 と Thread オプションモジュールは、金属スペーサで固定してください。

組み立て後でも Armadillo-600 シリーズ オプションケース(樹脂製) に収めることが可能です。

Armadillo-640 CON1(microSD スロット)に microSD カードを挿抜する際には、Thread オプションモジュールを取り外してください。組み立て後は、Thread オプションモジュールと Armadillo-640 の間隔が狭いため、無理に挿抜した場合、microSD カードが正常に挿入されないなどの原因で、動作不良を起こす場合があります。

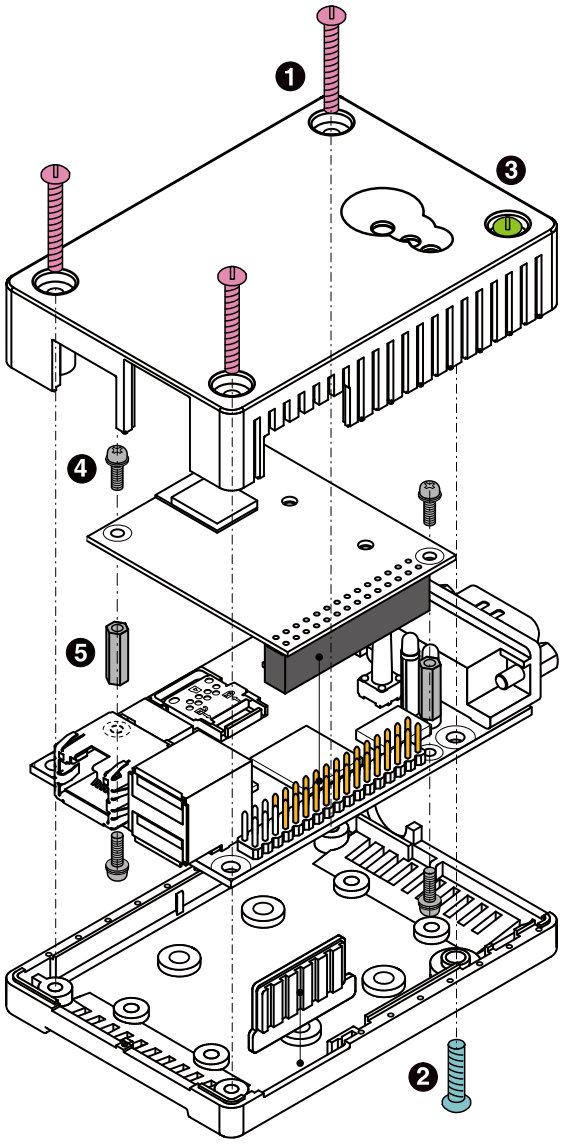


図 18.36 Thread オプションモジュールの組み立て

❶ タッピングねじ(M2.6、L=20mm) x 3

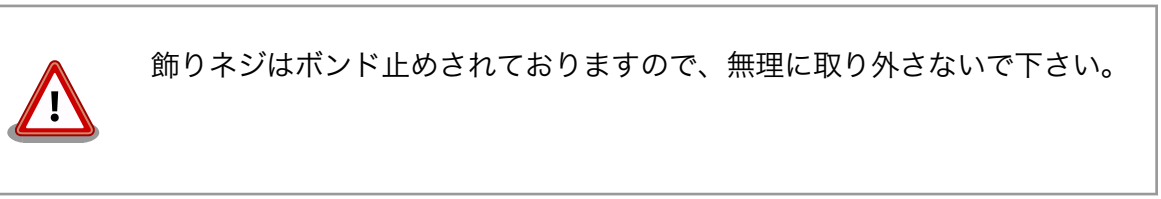
❷ タッピングねじ(M3、L=12mm) x 1

❸ 飾りねじ x 1

❹ なべ小ネジ、スプリングワッシャー、小径平ワッシャー付(M2、L=6mm) x 4

❺ 金属スペーサ(M2、L=11mm) x 2

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



飾りネジはボンド止めされておりますので、無理に取り外さないで下さい。

18.7.5. 形状図

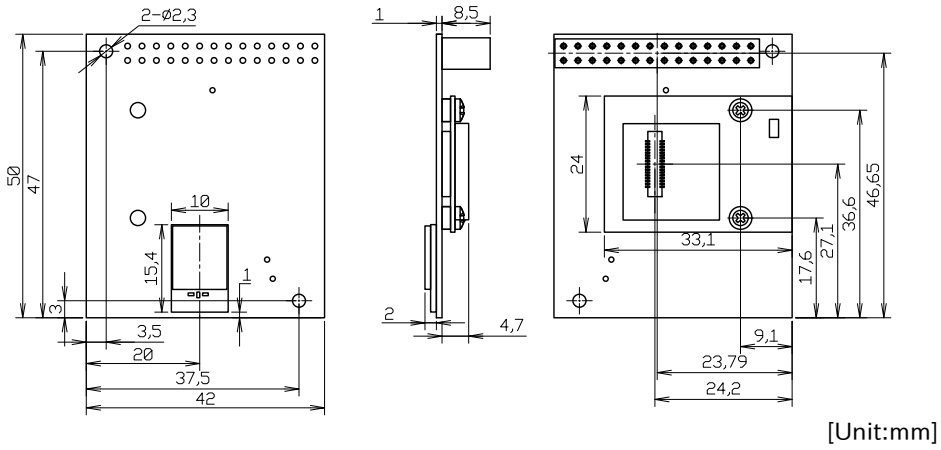


図 18.37 Thread オプションモジュール形状

18.8. 無線 LAN 用外付けアンテナセット 01

18.8.1. 概要

Armadillo-600 シリーズ WLAN オプションモジュール(OP-A600-AWLMOD-00)に接続可能な外付けアンテナセットです。

表 18.30 無線 LAN 用外付けアンテナセット 01 について

商品名	無線 LAN 用外付けアンテナセット 01
型番	OP-ANT-WLAN-01W
内容	外付けアンテナ、アンテナケーブル

表 18.31 無線 LAN 用外付けアンテナセット 01 の仕様

名称	指向性	入力インピーダンス	VSWR	利得	コネクタタイプ
アンテナ	水平面内無指向性	50Ω	<2.0dBi	2.0dBi	SMA-P リバース
アンテナケーブル		50Ω		0dBi	SMA-J リバース、MS-156C-LP-068/HIROSE ELECTRIC

18.8.2. 組み立て

18.8.2.1. 外付けアンテナの取り付け

Armadillo-600 シリーズ WLAN オプションモジュール(OP-A600-AWLMOD-00)に取り付けることが可能です。アンテナ端子にアンテナケーブルを接続してください。

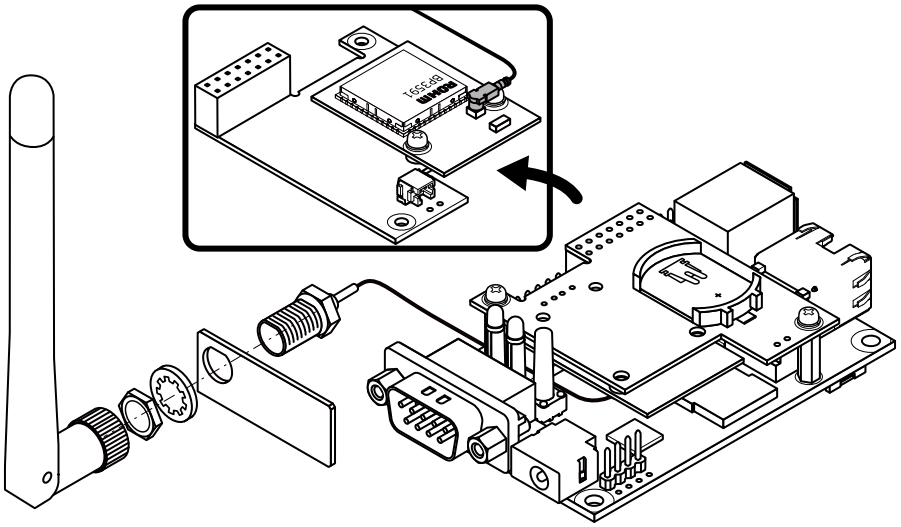


図 18.38 外付けアンテナの取り付け

アンテナ端子に外付けアンテナケーブルを接続する際、無理な力を加えないでください。破損の原因となります。

外付けアンテナケーブルは、専用の引き抜き工具(U.FL-LP-N-2/HIROSE ELECTRIC)で引き抜くことを推奨します。無理な引き抜きはコネクタの変形や断線の原因となります。

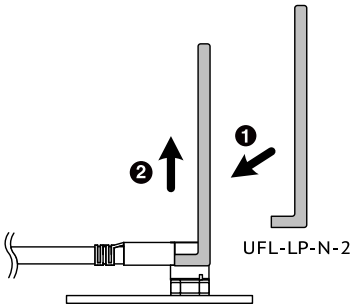
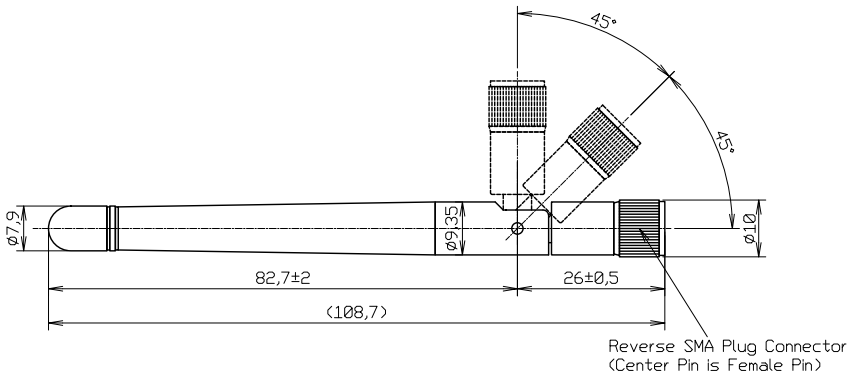


図 18.39 外付けアンテナケーブルの引き抜き方法

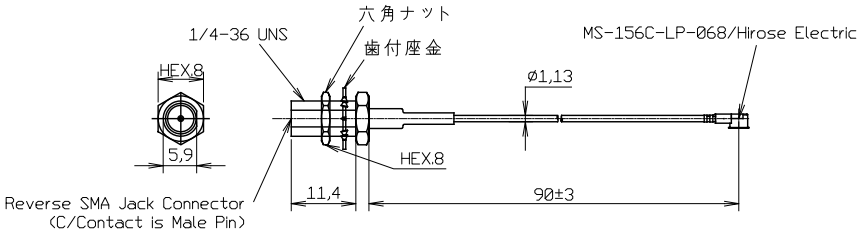
アンテナ端子に外付けアンテナケーブルを長期間接続した場合、同軸コネクタのスイッチ機能が復帰しない場合があります。復帰しない場合は、チップアンテナが使用できなくなりますのでご注意ください。

18.8.3. 形状図



[Unit:mm]

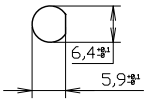
図 18.40 アンテナの形状図



[Unit:mm]

図 18.41 アンテナケーブルの形状図

取り付け穴寸法図



[Unit:mm]

図 18.42 アンテナの取り付け穴寸法図

19. 設計情報

本章では、Armadillo-640 の信頼性向上のための設計情報について説明します。

19.1. 放射ノイズ

CON11(LCD 拡張インターフェース)を使用して、Armadillo-640 と拡張基板を接続すると、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拡張基板に搭載する場合、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
  - シールドは拡張基板の GND に接続する
- Armadillo-640 の GND(固定穴等)と拡張基板の GND を太い導線や金属スペーサ等で接続する
- 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. ESD/雷サージ

Armadillo-640 の ESD 耐性を向上させるために、以下の対策が効果的です。

- Armadillo-640 を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

また、Armadillo-640 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。

- Armadillo-640 と通信対向機の 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-640 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 1(CON9)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-640.dts` • `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-640.dtb
arch/arm/boot/dts/armadillo-640.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。カスタマイズの対象は拡張インターフェース(CON8、CON9、CON14)と LCD 拡張インターフェース(CON11)です。

at-dtweb を利用した Device Tree のカスタマイズは linux-4.14-at11 から対応しています。

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
```

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。

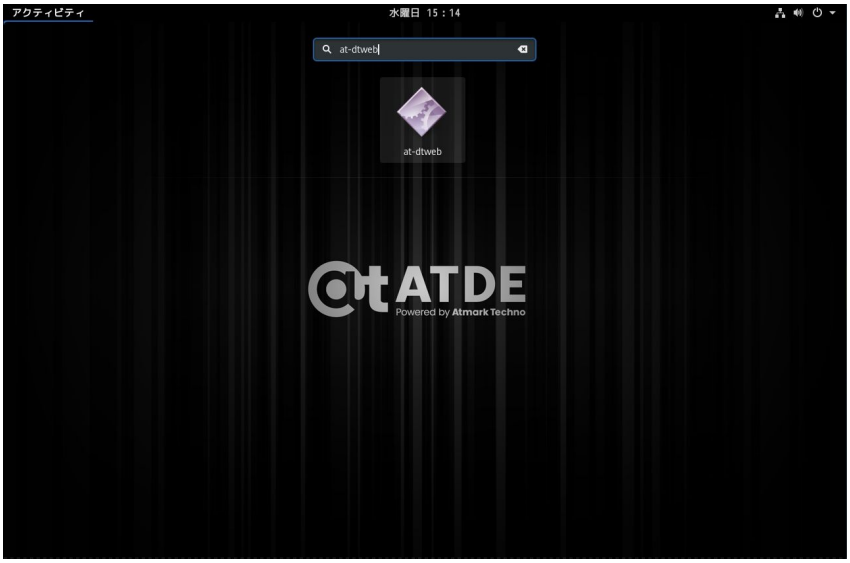


図 20.1 at-dtweb の起動開始

2. ボードの選択

   ボードを選択します。Armadillo-640 を選択して、「OK」をクリックします。

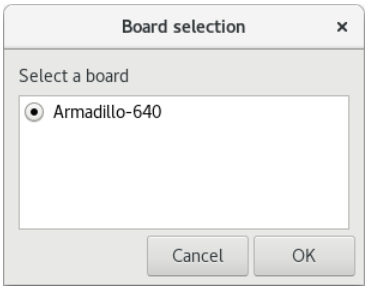


図 20.2 ボード選択画面

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。

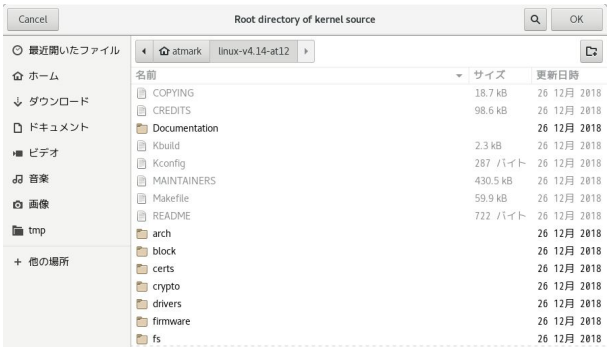


図 20.3 Linux カーネルディレクトリ選択画面

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。

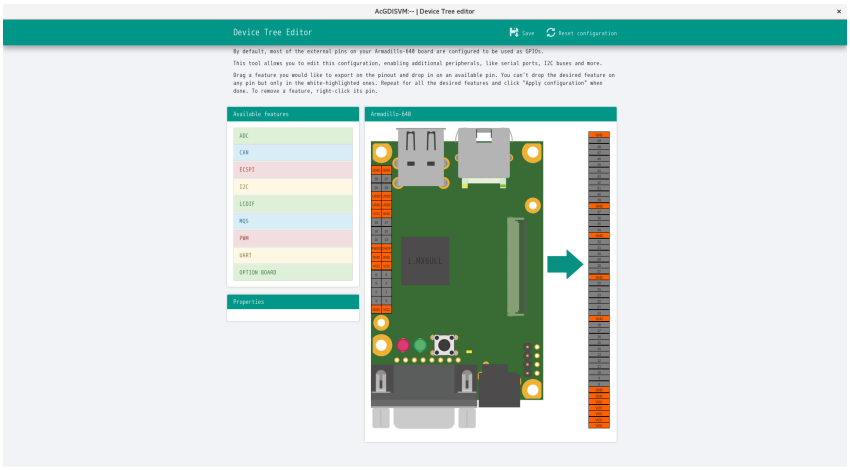


図 20.4 at-dtweb 起動画面

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-640」の白色に変化したピンにドロップします。例として CON9 3/5 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。

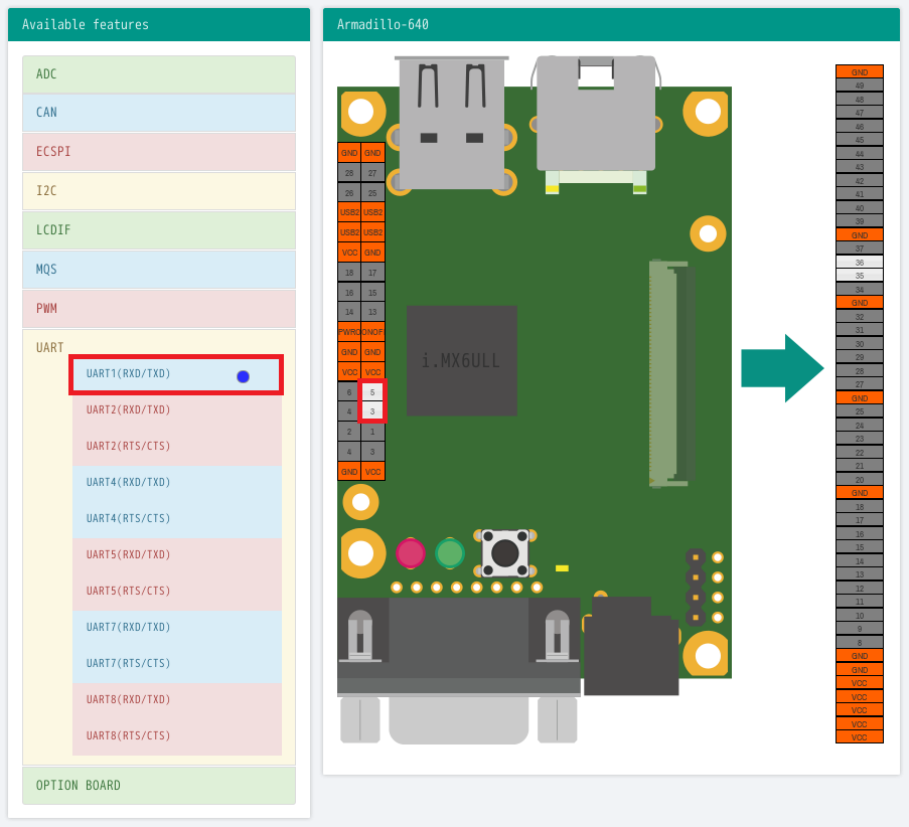


図 20.5 UART1(RXD/TXD)のドラッグ

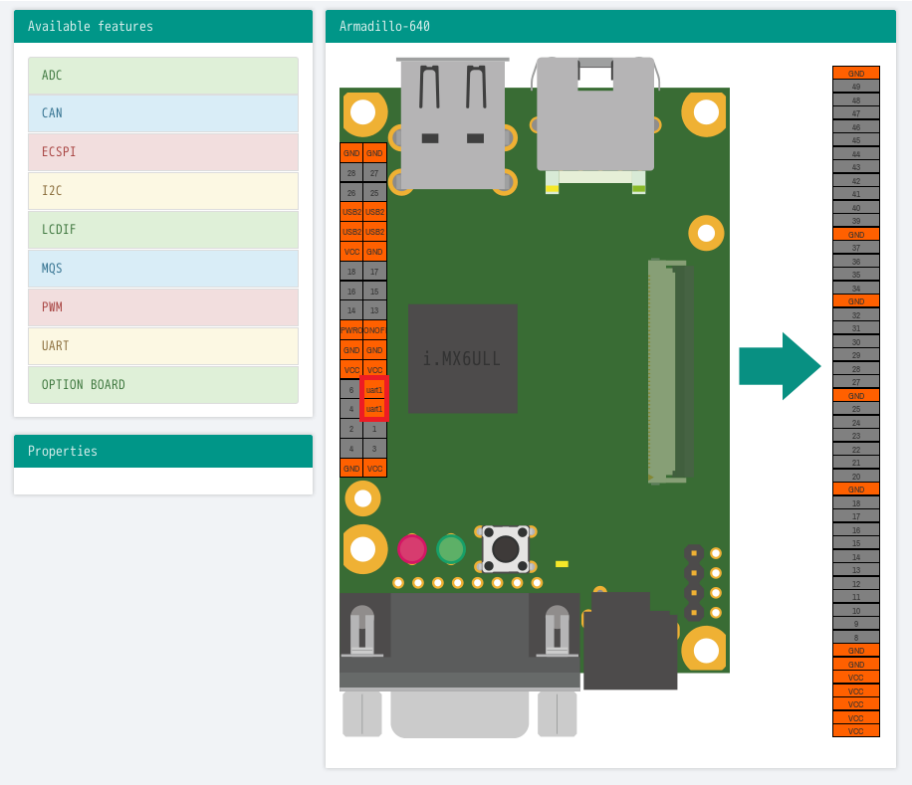


図 20.6 CON9 3/5 ピンへのドロップ

20.3.3.2. 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-640」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。

> プロパティの設定を行うには、at-dtweb のバージョンが v2.1.0 以降である必要があります。

例として CON9 4/6 ピンの I2C2(SCL/SDA)の clock_frequency プロパティを設定します。

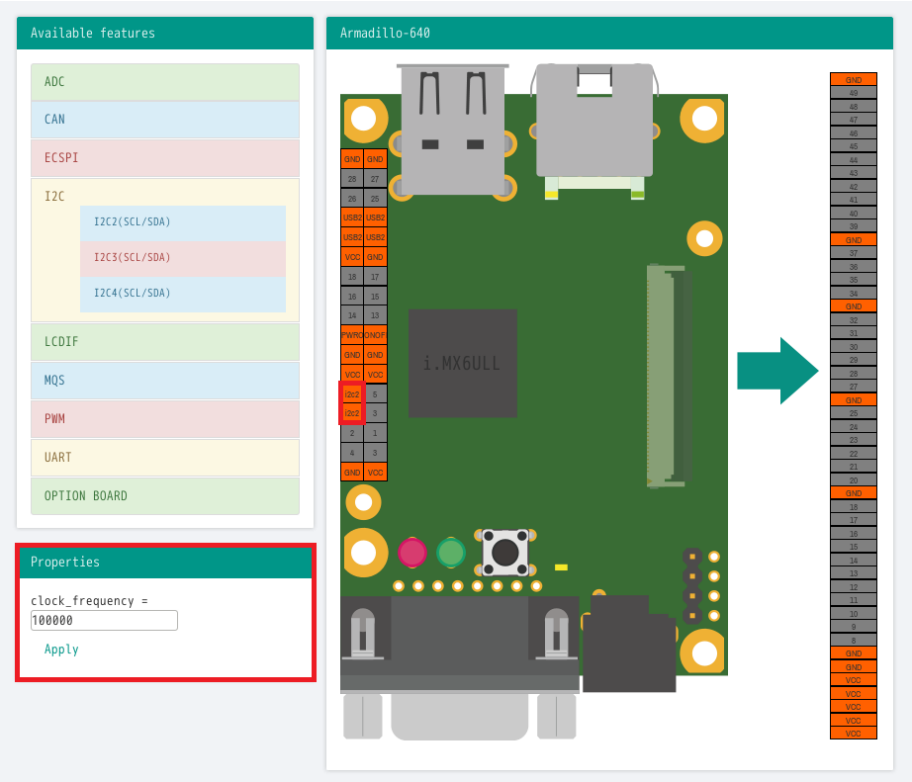


図 20.7 プロパティの設定

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。

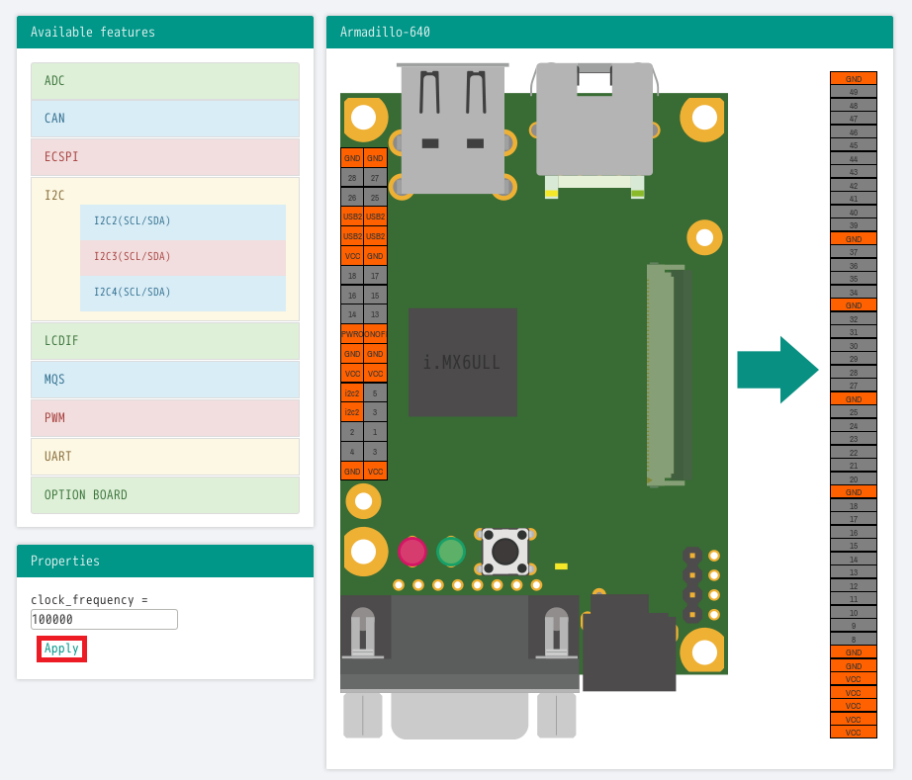


図 20.8 プロパティの保存

20.3.3.3. 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-640」のピンを右クリックして「Remove」をクリックします。

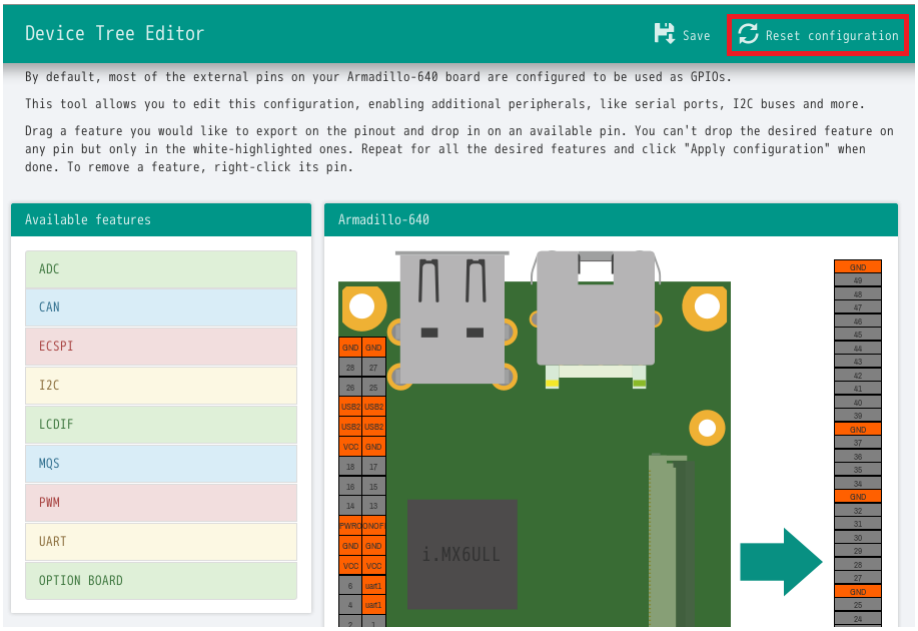


図 20.9 全ての機能の削除

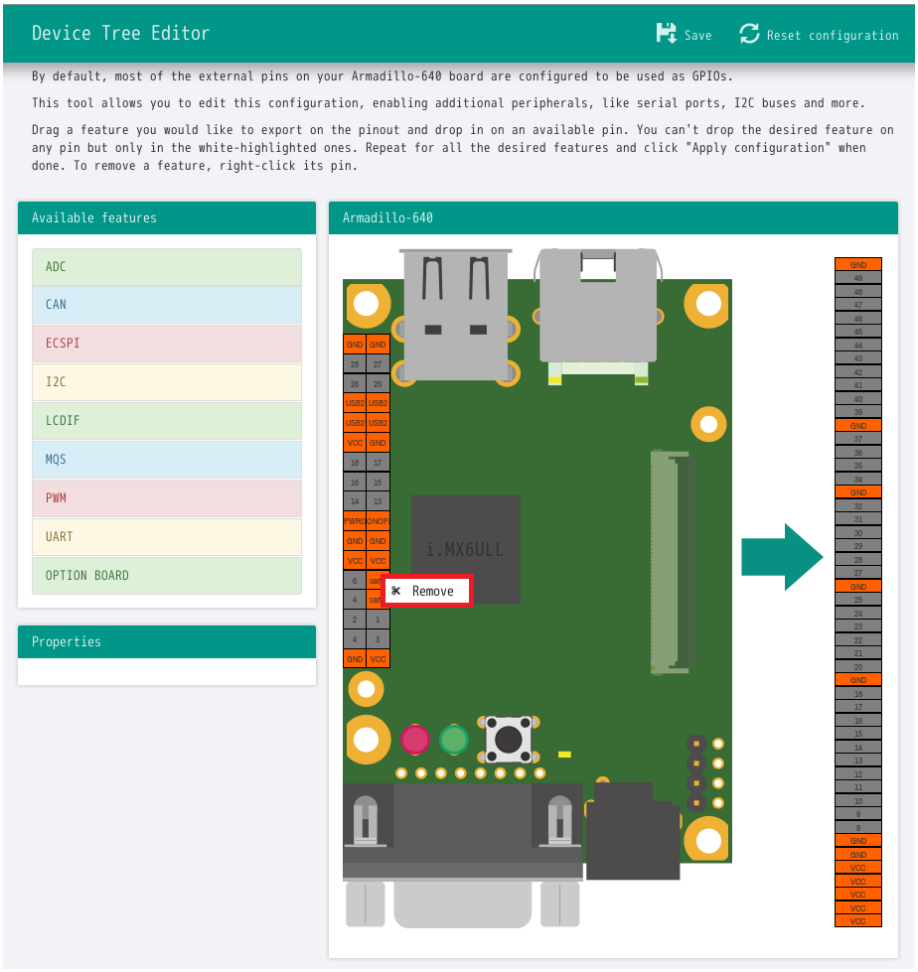


図 20.10 UART1(RXD/TXD)の削除

20.3.3.4. 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。

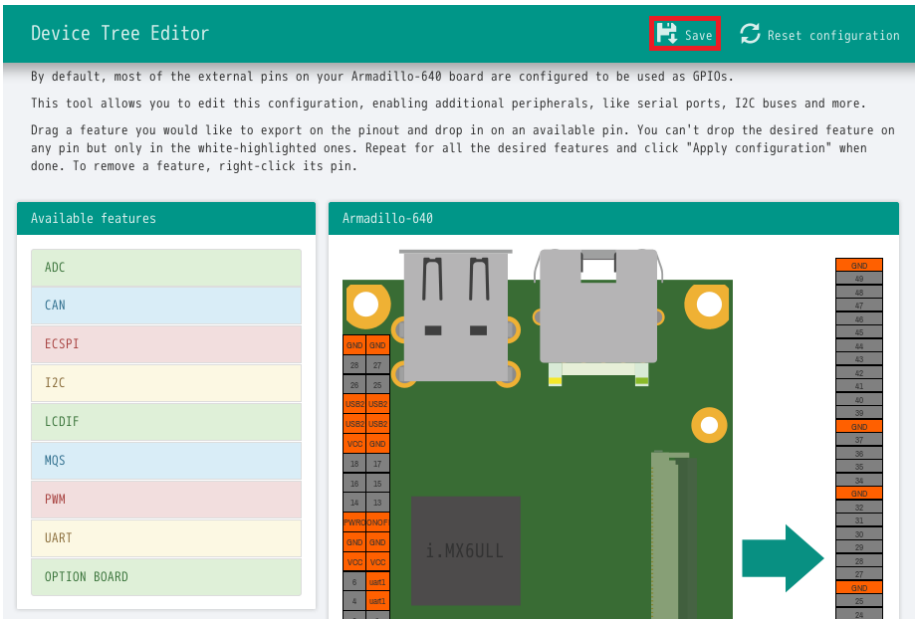


図 20.11 DTS/DTB の生成

「Device tree built!」と表示されると、DTS および DTB の生成は完了です。

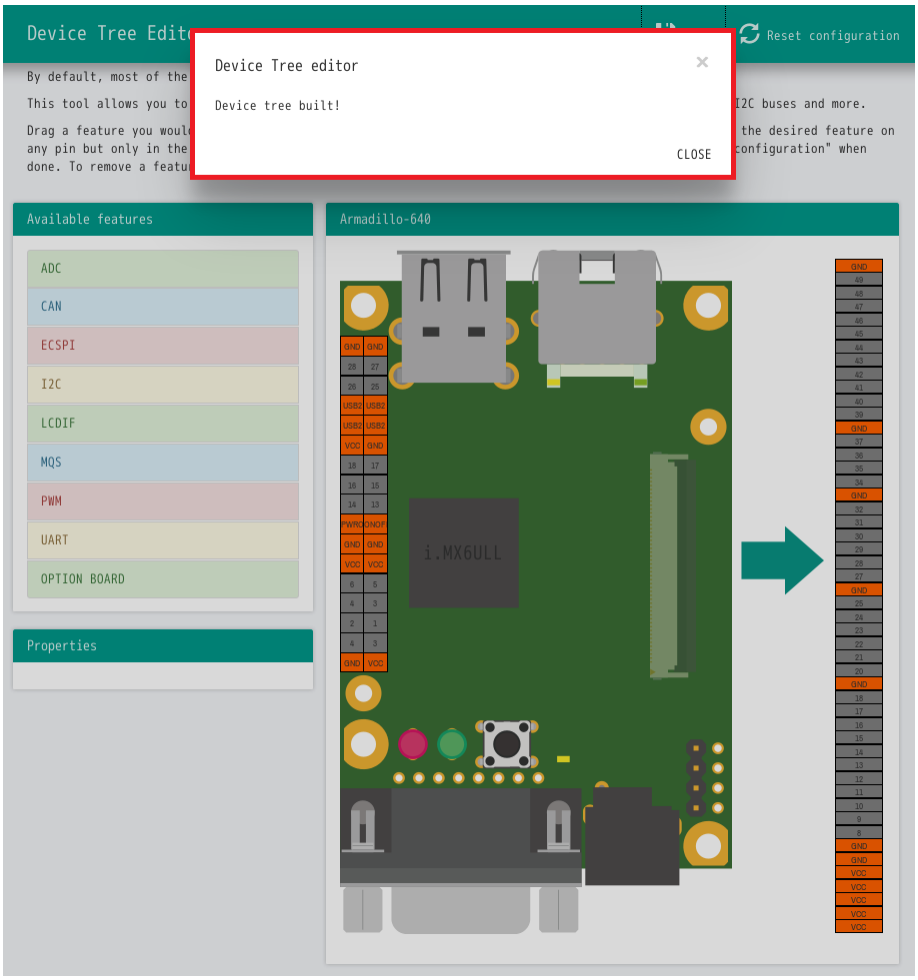


図 20.12 DTS/DTB の生成完了

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-640-expansion-interface.dtsi
armadillo-640-expansion-interface.dtsi
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-640-at-dtweb.dtb
armadillo-640-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-640 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-640 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-640 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
```

overlayfs による、eMMC への書き込み保護は linux-4.14-at5, u-boot-a600-v2018.03-at3 から対応しています。

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-640 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 20.13 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-640 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 20.14 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 20.15 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. 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。

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。

20.6.1. 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。

20.6.1.1. 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
```

20.6.1.2. 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/a640-wxwidgets-howto/
wxwidgets_led.tar.gz
[armadillo ~]# tar zxf wxwidgets_led.tar.gz
```

20.6.1.3. 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 RED、YELLOW ボタンを押すと、Armadillo-640 上の LED3、LED5 が点灯・消灯します。Armadillo-640 上の SW1 を押すと、サンプルアプリケーション上の表示が変化します。

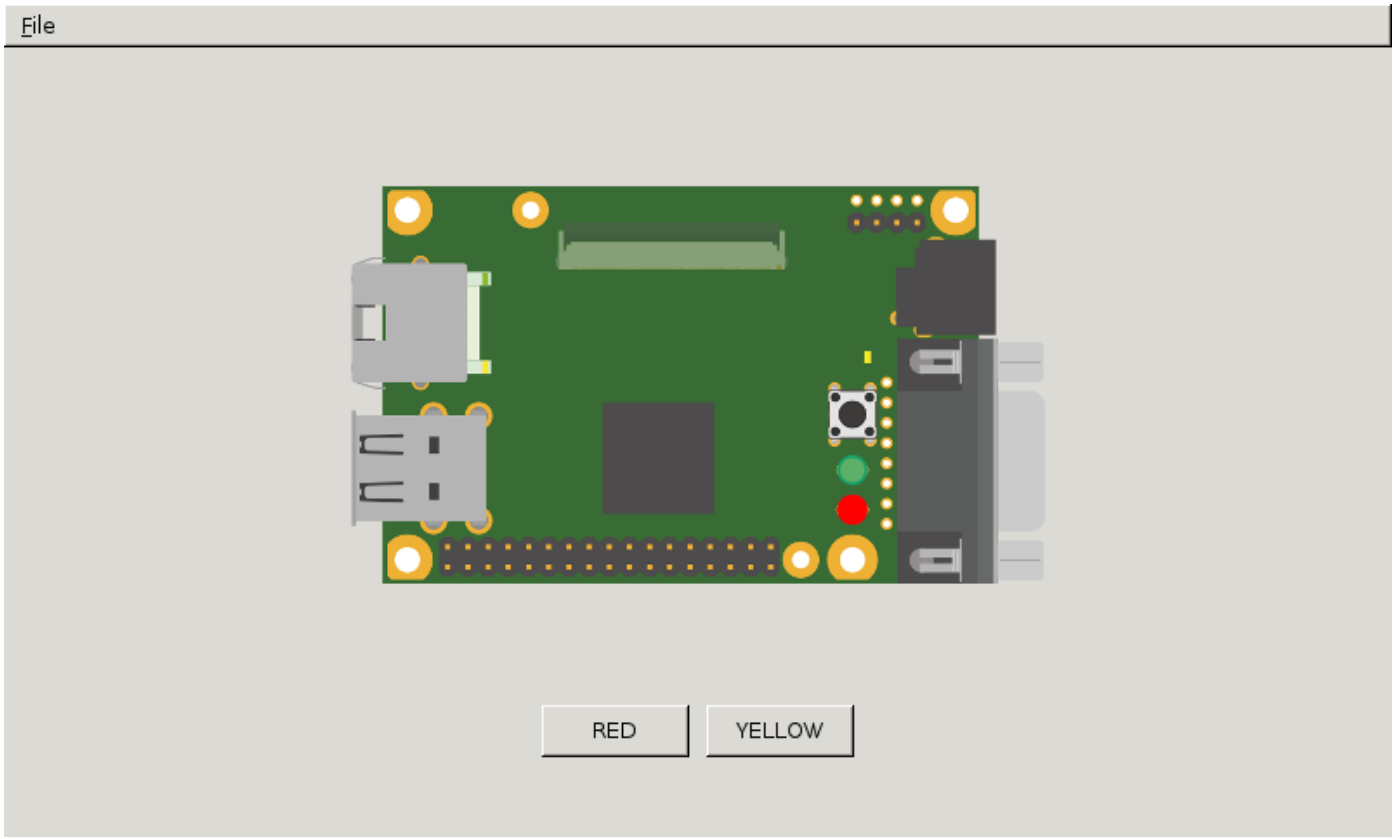


図 20.16 wxWidgets サンプルアプリケーション

20.6.1.4. 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-640 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。

20.6.2. 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「20.6.1.3. 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。)

20.6.2.1. 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
```

20.6.2.2. 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/a640-wxwidgets-howto/
wxpython_led.tar.gz
[armadillo ~]# tar zxf wxpython_led.tar.gz
```

20.6.2.3. 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 RED、YELLOW ボタンを押すと、Armadillo-640 上の LED3、LED5 が点灯・消灯します。Armadillo-640 上の SW1 を押すと、サンプルアプリケーション上の表示が変化します。

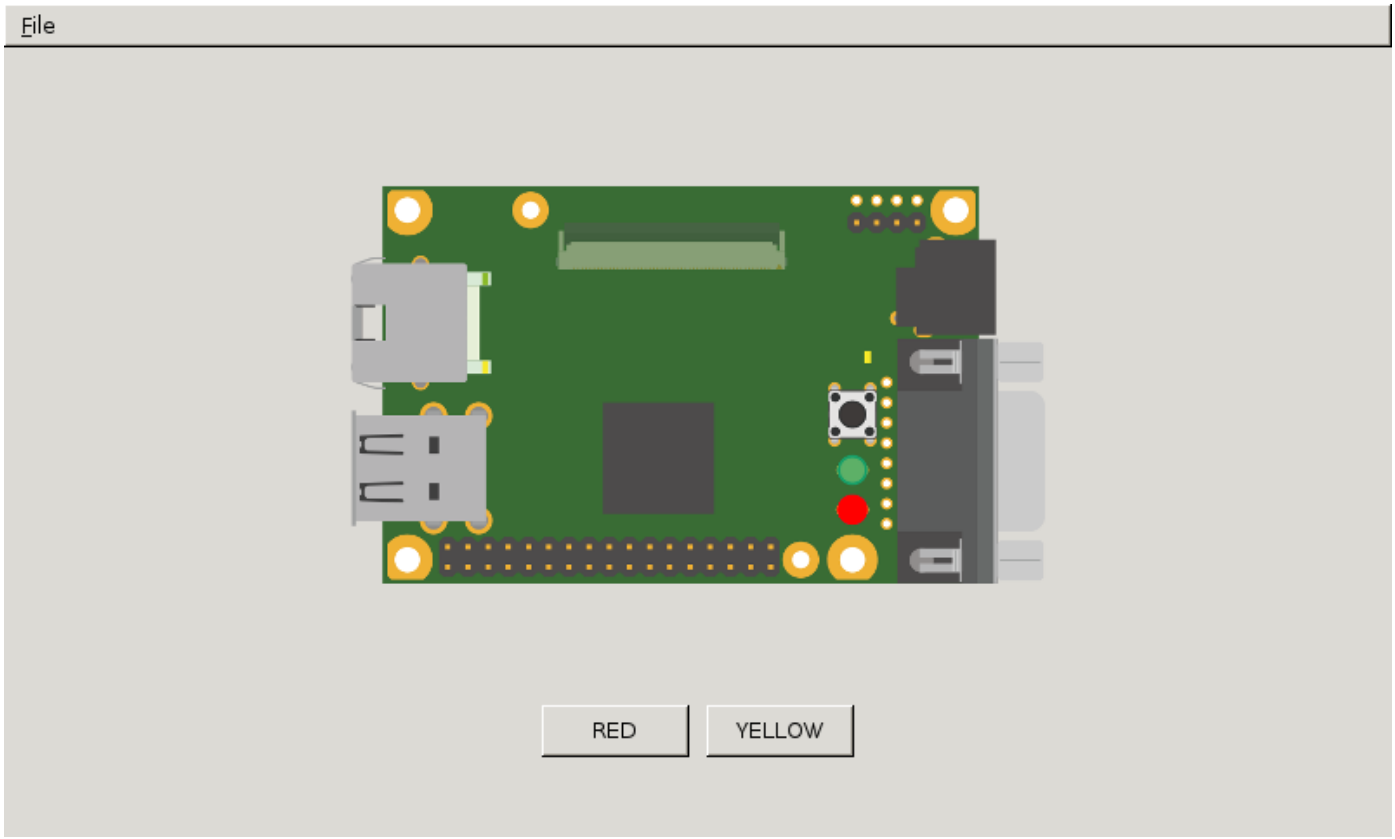


図 20.17 wxPython サンプルアプリケーション

20.6.2.4. ソースファイル構成

ハードウェア(LED3、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-640 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。

20.7. LCD インターフェースの動作確認

LCD オプションセット(7 インチタッチパネル WVGA 液晶)には Linux カーネル v4.14-at6 から対応しています。

CON11 に接続された LCD オプションセット(7 インチタッチパネル WVGA 液晶) の動作確認のため、Qt5 のサンプルアプリケーションを利用します。次に示すコマンドでサンプルアプリケーションをインストールし起動してください。

```
[armadillo ~]# apt-get update
[armadillo ~]# apt-get install qtbase5-examples qt5-default
[armadillo ~]# export QT_QPA_PLATFORM=linuxfb
[armadillo ~]# export QT_QPA_GENERIC_PLUGINS=evdevtouch:/dev/input/event0
[armadillo ~]# cd /usr/lib/arm-linux-gnueabihf/qt5/examples/touch/fingerpaint
[armadillo /usr/lib/arm-linux-gnueabihf/qt5/examples/touch/fingerpaint]# ./fingerpaint
```

サンプルアプリケーションを起動すると、LCD 上に指で絵を描画することができます。 LCD オプションセット(7 インチタッチパネル WVGA 液晶) のタッチパネルは最大 10 点のマルチタッチに対応しているので、10 本の指で同時に描画できることを確認することができます。

オープンソース版の Qt は GPLv3/LGPLv3 ライセンスの下で配布されています。利用する際には必ず各ライセンスに従ってください。Qt には商用ライセンスも用意されています。ライセンスに関する詳細は、Qt 公式サイト上の情報 (https://doc.qt.io/qt-5/licensing.html) を参照してください。

図 20.18 Qt5 サンプルアプリケーション

LCD の LED バックライト機能は、バックライトクラスとして実装されています。LED バックライトの輝度変更には、/sys/class/backlight/backlight-display/ ディレクトリ以下の、次に示すファイルを使用します。

表 20.1 輝度設定に使用するファイル

brightness	0(消灯) から max_brightness(最高輝度) までの数値を書き込むことで輝度を変更します。
max_brightness	brightness に書きこむ数値の最大値(最高輝度)が読み出せます。

例として、LED バックライトの輝度の最大値を確認した上で、最大値に設定する手順を示します。

```
[armadillo ~]# cat /sys/class/backlight/backlight-display/max_brightness
128
[armadillo ~]# echo 128 > /sys/class/backlight/backlight-display/brightness
```

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 ユーザーズサイト」にユーザー登録をする必要があります。

ユーザー登録すると次のようなサービスを受けることができます。

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

詳しくは、「アットマークテクノ ユーザーズサイト」をご覧ください。

アットマークテクノ ユーザーズサイト

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

付録 A eFuse

Armadillo-640 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-640 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-640 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-640 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. ブートモードとジャンパーピン

A.1.1. ブートモードと JP2

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。 BOOT_MODE0 は GND になっているので必ず 0 になります。BOOT_MODE1 は JP2 に接続されています。 JP2 がショートされていると BOOT_MODE1 は 1 になり Internal Boot モードになります。開放されていると 0 となり、Boot From Fuses というモードになります。

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-640 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。つまり eFuse の設定がどうなっていようと、GPIO のピンでブートデバイスを決めることができます。

Boot From Fuses モードでは、単純に言えば GPIO による override が禁止され eFuse に書き込まれた状態でしかブートしません。この機能を有効にすることで、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.1.2. ブートデバイスと JP1

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、JP1 はまさにこの機能を使っています。 JP1 は 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-640 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-640 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-640 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] が 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.2. eFuse の書き換え

Armadillo-640 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.3. Boot From Fuses モード

A.3.1. BT_FUSE_SEL

Boot From Fuses を有効にするには、 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。BOOT_MODE が 00 、つまり JP2 がオープンで、且つこのビットが 1 であれば Boot From Fuses モードになります。 BOOT_MODE が 00 でもこのビットが 0 であれば Boot From Fuses モードにはならず、 SD/MMC マニュファクチャリングモードやシリアルダウンロードモードになってしまいます。SD/MMC マニュファクチャリングモードについては 「8.12 SD/MMC manufacture mode」に、シリアルダウンロードモードについては「8.9 Serial Downloader」に記載されています。

Armadillo-640 では BOOT_MODE が 00 で、且つ BT_FUSE_SEL が 0 の場合は SD/MMC マニュファクチャリングモードで eMMC から起動します。Internal Boot モードで起動する場合は、JP2 をショートしてください。

A.3.2. 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-640 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-640 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.3.3. 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	2018/04/13	・初版発行
1.1.0	2018/07/30	・A6400-B00Z に対応 ・「2.4. 電波障害について」の VCCI についての記載を修正 ・「11.1.2. インストールの実行」にジャンパを設定する手順を追加 ・「14.3.1. 手順：Linux カーネルイメージおよび DTB の配置」の Linux カーネルイメージおよび DTB の配置先を SD カードの第 2 パーティション上の/boot に修正 ・「20.2. イメージをカスタマイズする」の make コマンドの実行順序を「10.2.1. 手順：Linux カーネルをビルド」と同じ順序に修正 ・CTS/RTS 信号線を利用する際の注意点を追加 ・誤記修正 ・「5. 起動と終了」から reboot コマンド非対応の注記を削除(linux-v4.14-at3 で対応) ・「10.3. Debian GNU/Linux ルートファイルシステムをビルドする」を追加
1.2.0	2018/09/28	・「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」を追加 ・Linux カーネルのビルド方法を initramfs アーカイブを使った手順に変更 ・ブートローダーの環境変数を変更 ・「12. 開発の基本的な流れ」を追加 ・誤記修正
1.3.0	2018/10/22	・「図 17.1. 基板形状および固定穴寸法」の CON9 寸法線引き出し箇所を修正 ・「表 15.4. 入出力インターフェースの電気的仕様(OVDD = VCC_3.3V)」の備考を修正 ・「表 18.1. Armadillo-640 関連のオプション品」に USB シリアル変換アダプタを追加 ・「18.4. LCD オプションセット(7 インチタッチパネル WVGA 液晶)」を追加 ・「19. 設計情報」を追加 ・「20.7. LCD インターフェースの動作確認」を追加 ・CON9 のブートモード設定ピンについて、CON11 のブートモード設定ピンについてを追加 ・誤記修正
1.4.0	2018/10/26	・「7. Linux カーネル仕様」に「7.3.4. WLAN」、「7.3.7. リアルタイムクロック」、「7.3.10. I2C」を追加 ・「18.5. Armadillo-600 シリーズ RTC オプションモジュール」を追加 ・「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」を追加 ・「18.8. 無線 LAN 用外付けアンテナセット 01」を追加 ・「2. 注意事項」に無線に関する内容を追加 ・「16.9. CON12、CON13(電源インターフェース)」に内蔵リアルタイムクロックを使用する際の注意事項を追加 ・誤記修正
1.5.0	2018/11/21	・製品ラインアップに A6400-B00Z、A6400-N00Z を追加 ・「16.6. CON8、CON9、CON14(拡張インターフェース)」、「16.8. CON11(LCD 拡張インターフェース)」に、同じ信号は複数のピンで同時利用できないことを記載

		・i.MX6ULL の RTC への時刻設定に対応 「7. Linux カーネル仕様」に i.MX6ULL の RTC 機能の情報を追加 「6.6. RTC」にハードウェアクロックの設定手順、systemd-timesyncd.service を停止する手順を追加 ・「6.6. RTC」にタイムゾーンの変更手順を追加 ・「11. イメージファイルの書き換え方法」に WLAN/RTC オプションの dtb の利用方法を記載 ・「5.4. 終了方法」に poweroff での Armadillo-640 の終了方法に関する情報を追加 ・誤記修正
1.6.0	2018/12/26	・「7. Linux カーネル仕様」に「7.3.11. パワーマネジメント」を追加 ・「13. i.MX6ULL の電源制御方法」を追加 ・「18.6. Armadillo-600 シリーズ WLAN オプションモジュール」に「18.6.7.6. RTC のアラーム割り込みを使用する」を追加 ・「18.5. Armadillo-600 シリーズ RTC オプションモジュール」に「18.5.7.3. RTC のアラーム割り込みを使用する」を追加 ・「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」に setenv した値を消去する方法を追記 ・「図 18.26. WLAN オプションモジュール形状」が WLAN オプションモジュールのものではなく、RTC オプションモジュールのものになっていた問題を修正 ・誤記修正
1.7.0	2019/01/25	・GPP(General Purpose Partition)についてを追加 ・「20. Howto」に「20.5. eMMC の GPP(General Purpose Partition) を利用する」を追加 ・「20. Howto」に「20.3. Device Tree をカスタマイズする」を追加 ・誤記修正
1.8.0	2019/02/27	・「6.2. ネットワーク」に「6.2.4. ファイアーウォール」を追加 ・誤記修正
1.8.1	2019/03/26	・「表 3.5. eMMC(GPP)メモリマップ」を追加 ・iptables のポリシーの設定で受信と転送を許可するを追加 ・iptables-persistent をインストールする前に、iptables のポリシー設定をもとに戻す手順を追加 ・「11.2.4. ルートファイルシステムの書き換え」の手順をより詳細に記載 ・bootcmd と setup_mmcargs をデフォルトの設定に戻す方法を追加 ・RTC バックアップ電源電圧の下限電圧(「表 15.2. 推奨動作条件」)を変更 ・表記ゆれ修正 ・誤記修正
1.8.2	2019/04/25	・コマンド実行やログの表示などが図になるように修正
1.9.0	2019/05/28	・「4. Armadillo の電源を入れる前に」の minicom 設定方法を詳しく記載 ・「20.7. LCD インターフェースの動作確認」に Qt による GUI アプリケーションの開発は推奨していない旨の表記を追加 ・「20. Howto」に「20.6. wxWidgets を利用して GUI アプリケーションを開発する」を追加 ・at-dtweb v2.1.0 に対応

Armadillo-640 製品マニュアル

1.10.0	2019/06/26	・「18.5.7. 動作確認」(RTC オプションモジュール)および「18.6.7. 動作確認」(WLAN オプションモジュール)に「RTC から時刻を取得する」を追加 ・「RTC から時刻を取得する」に電池残量が少ない状態では RTC から時刻を取得しない機能について記載 ・目次のリンクが正しく機能しないことがあった問題を修正
1.11.0	2019/07/29	・HTML 版のヘッダおよびフッタを変更 ・「18.7. Armadillo-600 シリーズ Thread オプションモジュール」を追加 ・「2.5. 無線モジュールの安全規制について」から各国電波法規制への対応情報を削除(同じ内容が「2.7. 輸出について」にも記載されているため)
1.11.1	2019/07/30	・Qt の利用に関する注記を修正
1.11.2	2019/08/29	・JP2 についての記載見直し ・誤記修正
1.12.0	2019/10/25	・「表 3.2. 仕様」の LAN 通信時の消費電力を修正、待機時の消費電力を追加
1.12.1	2019/11/07	・誤記修正
1.12.2	2020/01/21	・誤記修正

Armadillo-640 製品マニュアル
Version 1.12.2
2020/01/29