参考資料

Technical White Paper

適切なプロセッサとグラフィカル ユーザー インターフェイス (GUI) の選定

Shriya Surti, Soobin Kim, Tobias Morell (KDAB)

概要

適切な組込みユーザー インターフェイス (UI) フレームワークの選定は、技術的な判断であると同時に、戦略的な判断でもあります。このホワイトペーパーでは、テキサス・インスツルメンツの Sitara™ プロセッサ ファミリ全体を対象に、主要な UI フレームワークを対応付けた、プラットフォームの観点からの選定ガイドを提供します。このガイドでは、Sitara デバイスを複数の層に分類し、各フレームワークを評価するとともに、実装におけるベスト プラクティスについて解説します。ハードウェア制約と UI フレームワークがどのように相互に作用するかを理解することで、開発者は各製品の要件に応じて、コスト、性能、市場投入までの時間を最適化するための適切な選定を行うことができます。さらに詳しく見ていく前に、留意すべき点があります。それは、「最適な」プロセッサや GUI フレームワークは一切存在しないということです。適切な選定は、各プロジェクトの要件、チームの専門知識、ライセンス要件によって異なります。また、ハードウェアの性能やエコシステムの進化に応じて、選定を見直す必要があります。

目次

1 概要....2
2 ハードウェアの全体像と UI への影響を理解する....2
2.1 ハードウェアプラットフォームの分類：基本構成....2
2.2 主なプラットフォームの仕様とアーキテクチャに関する背景....3
3 技術の詳細解説：フレームワーク....4
3.1 Qt: クロスプラットフォーム アプリケーション フレームワーク....4
3.2 Slint: 最新の軽量フレームワーク....5
3.3 LVGL: マイコン向けに特化したフレームワーク....6
3.4 Flutter: モバイルファーストのアプローチ....7
3.5 その他の特筆すべきフレームワーク....8
4 KDAB の選定フレームワークと実装に関するベスト プラクティス....8
4.1 シリコン以外の選定基準....8
4.2 実装のベスト プラクティス：パフォーマンスの最適化....10
4.3 起動時間に関する考慮事項 (「コールド ブート」の問題)....10
4.4 電力効率 (バッテリ駆動デバイス)....10
4.5 今後の動向と検討事項....10
5 まとめ：TI と KDAB の見解....11
5.1 KDAB Group について....11
5.2 テキサス・インスツルメンツの Sitara MPU グループについて....11
6 参考資料....11

商標

Sitara™ is a trademark of Texas Instruments.
Arm® and Cortex-A® are registered trademarks of Arm Limited.
すべての商標は、それぞれの所有者に帰属します。

Copyright © 2026 Texas Instruments Incorporated

1 概要

組込みシステムに適したユーザー インターフェイス (UI) テクノロジーを選択することは、もはや単なるソフトウェア エンジニアリングに関する決定にとどまらず、ビジネス戦略を左右する重要な意思決定となっています。エンド ユーザーは、エンド アプリケーションに関係なく、60fps のアニメーション、高解像度ディスプレイ、直感的なタッチ操作を期待しています。

このレベルのパフォーマンスを実現するには、ハードウェア機能、パフォーマンス目標、開発リソース、および長期メンテナンスを慎重に検討する必要があります。ハードウェアを過剰にエンジニアリングして、大量の UI フレームワークをサポートすることで、部品表 (BOM) の予算が増加します。逆に、高性能なハードウェアに対して初歩的な UI フレームワークを選択すると、GPU リソースが十分に活用されず、開発者向けツールの貧弱さに起因して市場投入までの時間が長期化してしまいます。

このガイドでは、先進的な組込み UI テクノロジーについてプラットフォームの観点から概説するとともに、TI Sitara ファミリのハードウェア性能の幅に応じて、これらの選択肢を位置付けます。

2 ハードウェアの全体像と UI への影響を理解する

ソフトウェア フレームワークを評価する前に、ハードウェアの基盤を理解することが重要です。組込みハードウェアの性能範囲は非常に幅広く、マイコン (MCU) とマイクロプロセッサ (MPU) の間の境界はますます曖昧になってきています。それでも、ハードウェア プラットフォームは依然として個別のカテゴリに分類され、それぞれが UI テクノロジーの選定に大きく影響する制約をもたらします。

2.1 ハードウェアプラットフォームの分類：基本構成

表 2-1. TI Sitara プロセッサの HMI の階層

TI Sitara HMI の階層	デバイス	CPU	3D GPU / ビデオ	NPU	ディスプレイ	UI の制約
上位クラス	AM67A	最大でクワッド Cortex-A53	3D GPU 最大 50GFLOPS およびビデオ コーデック	最大 4 つの TOPS	トリプル ディスプレイのサポート	現実的な帯域幅の想定の下で、高性能なマルチディスプレイ UI を実現可能。デバイスによっては、AI の統合や、より大きな電力予算の確保が可能。
	AM67			該当なし		
	AM62P			該当なし		
中位クラス	AM62	最大でクワッド Cortex-A53	3D GPU 最大 8GFLOPS	該当なし	デュアル ディスプレイのサポート	エントリレベルの 3D ハードウェア アクセラレーション機能を採用、CPU と RAM に十分なヘッドルームあり。デバイスによっては、AI との統合が可能。
	AM62A		該当なし	最大 2 つの TOPS	シングル ディスプレイのサポート	
下位クラス	AM62L	最大でデュアル Cortex-A53	該当なし	該当なし	シングル ディスプレイのサポート	BOM を低く維持するためにソフトウェア効率が不可欠な、コストを重視する製品に導入。
	AM335	シングル Cortex-A8	3D GPU 最大 1.6GFLOPS	該当なし	シングル ディスプレイのサポート	

注

適合性は、実際のメモリ構成、グラフィックス ドライバ、Linux/RTOS スタック、ディスプレイ トポロジ、アプリケーションの複雑さ、およびターゲット ハードウェア上で測定された挙動によって異なります。この表は選定のためのガイドであり、サポートや性能を保証するものではありません。

Copyright © 2026 Texas Instruments Incorporated

2.2 主なプラットフォームの仕様とアーキテクチャに関する背景

UI フレームワークが実際の環境でどのように動作するかを理解するためには、現在広く普及している組込み向けシリコンの具体的なアーキテクチャを確認することが有効です。

注

これらの SoC の多くは、初めからカスタムのカスタムチップダウン設計として製品に採用されることはほとんどありません。むしろ、業界をリードする企業が提供する、充実したシステム オン モジュール (SoM) のエコシステムを通じて製品に採用されます。SoM パートナーは、産業用製品の市場投入までの期間を短縮するとともに、最新のモノのインターネット (IoT) デバイスにおけるハードウェア認証を簡素化します。

2.2.1 テキサス・インスツルメンツ AM335 プラットフォーム

- **アーキテクチャ:** NEON SIMD を搭載した 1GHz で動作するシングルコア Cortex-A8。すべてのアプリケーション コードが同一のコア上で実行されるため、UI 処理とリアルタイム処理が同じリソースを共有します。そのため、リアルタイム処理専用のコアは搭載されていません。
- **グラフィックおよびディスプレイ:** Imagination SGX530 3D グラフィックスと 24 ビット LCD コントローラを搭載。最大 1920 × 1080 の解像度、30fps でのシングル RGB ディスプレイをサポート。

2.2.2 テキサス・インスツルメンツ AM62L プラットフォーム

- **アーキテクチャ:** 最大 1.25GHz で動作するデュアルコア 64 ビット Cortex-A53 を搭載。低コストかつ低消費電力という特長を維持しながら、Linux ベースの UI を動作させるのに十分な処理能力を備えています。
- **グラフィックおよびディスプレイ:** DSI または DPI 経由で、最大 1920 × 1080 の解像度、60fps に対応するシングル ディスプレイ出力をサポートしています。

2.2.3 テキサス・インスツルメンツ AM62A プラットフォーム

- **アーキテクチャ:** 最大 1.4GHz で動作するクワッドコア Cortex-A53、マイコンチャネルとデバイス管理作業用の 800MHz で動作する 2 個の Cortex-R5F コア、および最大 2TOPS の C7x ディープラーニング アクセラレータ。R5F コアは決定論的で低レイテンシのタスクを処理し、A53 クラスタは Linux とグラフィカル UI を実行します。
- **グラフィックスおよびディスプレイ:** シングル ディスプレイをサポートし、DPI を介して 60fps で最大 2048 × 1080 まで対応。

2.2.4 テキサス・インスツルメンツ AM62 プラットフォーム

- **アーキテクチャ:** 最大 1.4GHz で動作するクアッド Cortex-A53 と、最大 400MHz で動作するシングル Cortex-M4F を搭載。これにより、顧客は、汎用アプリケーション全般において、応答性に優れた多用途な GUI を実現できます。
- **グラフィックスおよびディスプレイ:** 2D/3D アクセラレーションに対応した Imagination AXE-1-16M GPU を搭載。最大 1920 × 1080 の解像度、60fps でのデュアル LVDS/DPI ディスプレイ出力をサポートしています。

2.2.5 テキサス・インスツルメンツ AM62P および AM67 プラットフォーム

- **アーキテクチャ:** 最大 1.4GHz で動作するクワッド Cortex-A53 に加え、800MHz で動作する 2 基の独立した Cortex-R5F コアを搭載。R5F コアは、Linux 駆動の UI から独立して、リアルタイム制御または安全関連タスクを実行できます。
- **グラフィックスおよびディスプレイ:** 2D/3D アクセラレーションに対応した Imagination BXS-4-64 GPU を搭載。LVDS、DSI、または DPI を介して、最大 3840 × 1080 の解像度、60fps でのトリプル ディスプレイをサポート。これらのデバイスは、H.264/H.265 4K UHD ビデオ機能にも対応しています。

2.2.6 テキサス・インスツルメンツ AM67A プラットフォーム

- **アーキテクチャ:** 最大 1.4GHz で動作するクワッド Cortex-A53 に加え、800MHz で動作する 2 基の独立した Cortex-R5F コアを搭載。このデバイスには、最大 4TOPS を備えた 2 基のディープラーニング アクセラレータが搭載されており、AI 機能をサポートしています。
- **グラフィックスおよびディスプレイ:** 2D/3D アクセラレーションに対応した Imagination BXS-4-64 GPU を搭載。LVDS、DSI、または DPI を介して、最大 3840 × 1080 の解像度、60fps でのトリプル ディスプレイをサポート。このデバイスは、H.264/H.265 4K UHD ビデオ機能にも対応しています。

English Document: SDAA554
Copyright © 2026 Texas Instruments Incorporated

注

KDAB の見解：総所有コストの方程式

アーキテクチャ レビューでは、チームが開発者の習熟度だけを基準に UI フレームワーク (たとえば、Web 技術や Flutter) を選定し、その結果、フレームワークを円滑に動作させるためには、想定を大幅に上回るメモリが必要であることが、開発の後期段階になって初めて判明するケースを頻繁に目にします。ソフトウェアのオーバーヘッドを吸収するために、低コストのプロセッサ構成から上位のプロセッサ構成にアップグレードすると、製品のマージンが損なわれる可能性があります。ハードウェアと UI ソフトウェアは、相互依存的に選定する必要があります。

3 技術の詳細解説：フレームワーク

3.1 Qt: クロスプラットフォーム アプリケーション フレームワーク

Qt は、単なる UI レンダラーではなく、C++ をベースとした完全なアプリケーション プラットフォームであり、25 年以上にわたり継続的に開発されています。Qt は、ツール、クロスプラットフォームでの展開、および長期商用サポート (LTS) を提供しており、製品ライフサイクルが 10 ～ 15 年に及ぶ製品にも適しています。

3.1.1 テクノロジー アーキテクチャ

- **Qt Quick (QML):** ハードウェア アクセラレーションに対応したシーン グラフ上で動作する宣言型 UI。
- **Qt Widgets:** 従来の命令型 C++ UI ツールキット。データ量の多いインターフェイスや、CPU で描画するインターフェイスに適しています。
- **Qt Quick Compiler:** QML を事前コンパイルすることで、起動時間とメモリのフットプリントを削減します。
- **Qt Quick 3D:** 2D インターフェイス内の 3D コンテンツをネイティブに統合できます。
- **Qt for MCUs (Qt Quick Ultralite):** マイコンを対象とした、QML のカスタム サブセットと決定論的なランタイムを提供します。

3.1.2 プラットフォームの適合性と実環境におけるスケーリング

- **下位クラスの Sitara デバイス (AM335、AM62L):** 徹底した最適化と、適切な範囲に絞った UX を活用して実現可能。QML のネストの深さを最小限に抑え、オーバードローを削減し、グラフィックス処理経路を早期に検証します。Qt Widgets は、データ指向の限定的なインターフェイスに適している場合がありますが、最新のタッチ操作に対応した UX を実現するには、製品ごとに相当な設計投資が必要となります。
- **中位クラスの Sitara デバイス (AM62、AM62A):** Qt Quick の有力なターゲット。これらのプラットフォームは、中程度の複雑さのアプリケーションに対応するために、CPU、RAM、およびグラフィックス性能のバランスを備えています。ターゲット ハードウェア上で、起動時間、メモリ使用量、および継続動作時のフレーム描画の挙動を測定してください。Qt Quick Compiler を使用することで、起動時の処理負荷を軽減できます。
- **上位クラスの Sitara デバイス (AM62P、AM67、AM67A)**: Qt は、ハードウェア アクセラレーションによるグラフィックス処理経路を利用でき、高性能なマルチディスプレイ UI 設計にも対応できるため、非常に優れた選択肢です。製品で使用するグラフィックス スタックを確認し、想定するディスプレイ構成におけるコンポジションと帯域幅を検証してください。

3.1.3 ビジネス ケース：利点と課題

- **長所：**成熟したエコシステム。Qt Creator および Qt Design Studio (Figma ブリッジ付き)。Linux、RTOS、Windows、iOS、Android の各プラットフォームでのシングルコードベースの導入。Qt Safe Renderer による IEC 62304 および ISO 26262 の認証サポート (別個の製品。これらの認証は Qt フレームワーク全体を対象としていません)。テスト自動化ツール Squish。
- **課題：**最小限のフレームワークと比べてフットプリントが大きい。CMake/qmake の習得に一定の学習時間が必要。クローズドソース製品や静的リンクを使用する場合には、ライセンス体系が複雑。商用 Qt ライセンスは、導入モデル、製品量、サポートのニーズ、静的リンクの方式、および追加の Qt 製品の利用状況によっては、総所有コストを左右する重要な検討事項となり得ます。チームは、最新の商用ライセンス条件を確認し、エンジニアリング コストおよび BOM (部品表) コストと併せて、ライセンス費用を評価する必要があります。

Copyright © 2026 Texas Instruments Incorporated

注

KDAB の見解：ライセンスと開発速度の比較

商用 Qt ライセンスは、直接的かつ継続的に発生する費用です。実際に比較すべきなのは、ツールのエコシステムです。Qt Creator と Qt Design Studio、Slint のライブプレビュー機能、Flutter のホットリロード機能は、それぞれ異なる開発ワークフローを効率化します。チームは、特定のフレームワークだけが特別に生産性に優れていると考えるのではなく、自社製品に対する各ツールの適合性を評価する必要があります。

3.2 Slint: 最新の軽量フレームワーク

Slint は、極めて高い効率性、メモリの安全性、および開発者の生産性を実現するために、一から設計された Rust ベースの宣言型 UI フレームワークです。Slint によると、アニメーションやタッチ入力、ハードウェア アクセラレーションによるレンダリングをサポートしながら、ランタイムの RAM フットプリントを 300KB 未満に抑えています。

Copyright © 2026 Texas Instruments Incorporated

3.2.1 テクノロジー アーキテクチャ

- **宣言型 UI 言語：**カスタムの、可読性に優れたマークアップ言語を採用しており、充実したツール サポートを備えています。
- **複数のレンダリング バックエンド：**マイコン向けのソフトウェア レンダラーに加え、高い処理能力を備えた MPU 向けに、OpenGL、Metal、Vulkan、Direct3D、および Skia に対応したアクセラレーション対応レンダラーを提供しています。
- **言語バインディング：**Rust および C++ との統合を標準的にサポートしており、さらに Python や JavaScript/TypeScript 向けのバインディングも提供しています。

3.2.2 プラットフォームの適合性と実環境におけるスケーリング

- **下位クラスの Sitara デバイス (AM335、AM62L):** Slint は、フットプリントの小さ、予測可能なメモリ挙動、宣言的なワークフローが重視される場合、有力な選択肢となります。ソフトウェア レンダリングやアクセラレーションされたレンダリング バックエンドの選定にあたっては、本番環境のグラフィックス スタックおよび測定されたワークロードを考慮する必要があります。
- **中位クラスの Sitara デバイス (AM62、AM62A):** このクラスは、Slint の強みとよく合致しています。サポートされている環境では、GPU アクセラレーションを利用したバックエンドを使用できます。レンダラーを設計する際は、実際のグラフィックス API やドライバの利用可能性を考慮してください。
- **上位クラスの Sitara デバイス (AM62P、AM67、AM67A):** Slint は、マルチディスプレイ設計を含め、高機能で応答性に優れた、アクセラレーションを活用した UI をサポートできます。リソースがより豊富になるにつれ、重要な利点は、Rust のメモリ安全性、ライセンス モデル、そして言語間バインディングへと移行していきます。

3.2.3 ビジネス ケース：利点と課題

- **長所：**約 300KB のランタイム、Rust レベルのメモリ安全性、ライブ プレビューおよびホット リロード機能。既存の Figma デザインがある場合に便利な Figma インポート機能も提供 (ただし、これが主要な制作ワークフローというわけではありません)。GPLv3 と商用ライセンスを組み合わせたデュアル ライセンスは、分かりやすいです。CPU のウェーク イベントが少ないことも、Slint がバッテリ駆動デバイスに適している理由の一つです。
- **課題：**Slint は比較的新しいフレームワークです。サードパーティ製ウィジェットの充実度やエコシステムの成熟度では、Qt に後れを取っています。中核となる設計ツールは優れていますが、IDE との統合や、複雑な特殊ケースに対応する包括的なドキュメントは、依然として発展途上の段階にあり、また、認証対象分野における実績もまだ初期の段階にあります。

注

KDAB の見解：組込み分野における Rust 革命

Slint のサービス パートナーとして、KDAB はその採用動向を綿密に監視しています。当社では、EV 充電器や産業用 IoT 機器の開発チームを中心に、Slint への関心が高まっていることを確認しています。これらのプロジェクトには、メモリおよび CPU の予算制限、長期にわたる製品ライフサイクル、ソフトウェア セキュリティに対する重要なニーズという共通点があり、これらは、Rust が強みを発揮する領域です。Slint を使用することで、これらのチームは、従来は基本的な低解像度 UI しかサポートしていなかったハードウェア上でも、滑らかでアニメーションを伴うインターフェイスを実現できます。

3.3 LVGL: マイコン向けに特化したフレームワーク

Light and Versatile Graphics Library (LVGL) は、マイコン用に特別に最適化された業界をリードするオープンソース UI フレームワークです。LVGL は完全に C 言語で書かれており、外部依存関係が最小限に抑えられています。これにより、従来はセグメント表示やモノクロのビットマップ表示しか対応できなかったハードウェア上でも、高度なグラフィカル インターフェイスを実現できます。

3.3.1 テクノロジー アーキテクチャ

- **Pure C 言語による実装：**コンパイラやプラットフォームを問わず、高い移植性を実現します。
- **ハードウェア抽象化：**ディスプレイ、入力、ファイルシステムの各ドライバをプラグイン形式で組込み可能。
- **レンダリングの俊敏性：**プラットフォーム固有の GPU アクセラレーションを実現するために、フックを使用して高度に最適化された CPU ソフトウェアレンダリング。

Copyright © 2026 Texas Instruments Incorporated

- **RTOS の統合：**FreeRTOS、Zephyr、RT-Thread、あるいはベアメタル環境と完全に互換性があります。

3.3.2 プラットフォームの適合性と実環境におけるスケーリング

- **下位クラスの Sitara デバイス (AM335、AM62L):** LVGL は、コスト重視のアプリケーションに非常に適した選択肢です。LVGL はフットプリントが小さいため、アプリケーションのロジックにより多くのリソースを割り当てることができ、機能や操作が限定された家電製品のようなインターフェイスに非常に適した選択肢です。
- **中位クラスの Sitara デバイス (AM62、AM62A):** LVGL は、移植性、フットプリントの小さ、または既存の資産を重視する場合にも、引き続き適した選択肢です。グラフィックが豊富な HMI 設計においては、GPU 指向の宣言型フレームワークを採用することで、プラットフォームをより効果的に活用し、イテレーション速度を向上させられるかどうかを検討してください。
- **上位クラスの Sitara デバイス (AM62P、AM67、AM67A):** LVGL は技術的には実現可能ですが、製品がグラフィックスやマルチディスプレイ機能を幅広く使用することを目的としている場合、その価値は低下する可能性があります。その選択は、自動的なものではなく、意図的なものである必要があります。

3.3.3 ビジネス ケース：利点と課題

- **長所：**マイコンに最適化されていること、商用利用に制限のない寛容な MIT ライセンスを採用していること、活発なコミュニティが存在すること、およびビジュアル設計ツールである LVGL Pro が利用できること。
- **課題：**C 言語による UI コードを手動で記述する方法は、宣言型のアプローチと比較して開発に時間がかかり、エラーも発生しやすくなります。ピュアな C 言語には Rust のようなメモリの安全性の保証がないため、メモリの正しいアロケーションやポインタの適切な取り扱いは、開発者の責任となります。命令型 API は、状態の複雑さが増すにつれて保守が難しくなります。

3.4 Flutter: モバイルファーストのアプローチ

Flutter は、Google が掲げるクロスプラットフォーム開発のビジョンを体現するものです。Flutter は、モバイル開発者の間で広く採用されています (開発者向け調査では、常にクロスプラットフォーム モバイル フレームワークの上位にランクインしています)。Flutter の主な対象は、引き続きモバイルおよび Web です。組込み Linux への対応も可能ですが、完全にサポートされた Google 公式の取り組みというよりも、コミュニティ主導で進められています。

3.4.1 テクノロジー アーキテクチャ

- **Dart 言語：**ガベージ コレクション機能を備えており、JIT (開発時およびホット リロード用) と AOT (本番環境) のコンパイル方式に対応しています。
- **カスタム レンダリング：**すべてのピクセルは、ハードウェア アクセラレーションを利用する Skia エンジン (Impeller へと移行中) によって描画され、ピクセル単位で完全に一貫した表示を保証します。
- **ウィジェットの構成：**「すべてをウィジェットとして扱う」という設計思想に基づいており、高い応答性を備えた更新を実現します。

3.4.2 プラットフォームの適合性と実環境におけるスケーリング

- **下位クラスの Sitara デバイス (AM335、AM62L):** Flutter はデフォルトの推奨選択肢ではありません。生産決定は、起動、メモリ使用量、持続的な応答性、グラフィックス ドライバの動作、および電力予算を網羅したターゲットボードの概念実証に基づいて行われる必要があります。
- **中位クラスの Sitara デバイス (AM62、AM62A):** 製品に十分な RAM が搭載されており、成熟したグラフィックス アクセラレーション経路が確立されている場合には、Flutter を評価対象とすることができます。Flutter は、モバイルコードの共有や、Flutter スキルの活用によってランタイム上の追加コストが相殺されるのであれば、採用する正当な理由があります。ただし、その結果は、ワークロードとハードウェア構成によって大きく異なります。
- **上位クラスの Sitara デバイス (AM62P、AM67、AM67A):** システムが十分な RAM と堅牢な GPU アクセラレーションを備えている場合、Flutter は視覚的に高度なアプリケーションに適した信頼できる選択肢となります。適切なワークロードであれば 60fps の達成は可能ですが、最終的なアプリケーション、グラフィックス スタック、解像度、およびディスプレ トポロジについて測定を行う必要があります。

3.4.3 ビジネス ケース：利点と課題

- **長所：**比類のない開発者体験：ホット リロード、IDE サポート、豊富な組込みウィジェット (Material/Cupertino)、および大規模なパッケージ エコシステム (pub.dev)。モバイル コンパニオン アプリと組込みデバイスの間でコードベースを共有できるため、採用上のボトルネックを軽減できます。

Copyright © 2026 Texas Instruments Incorporated

- **課題：**深刻なリソースのオーバーヘッドにより、ハードウェアの BOM (部品表) のアップグレードを余儀なくされる場合があります。組込み Linux のサポートは主にコミュニティ主導で行われています。主な組み込み事例としては、AGL ベースの車載インフォテイメントで使用されているトヨタの ivi-homescreen (https://github.com/toyota-connected/ivi-homescreen) や、Raspberry Pi および同様のボード向けの flutter-pi (https://github.com/ardera/flutter-pi) があります。Dart のガベージ コレクションによって予測不能なレイテンシが発生するため、Flutter は厳密なリアルタイム性が求められる要件には適していません。専用のリアルタイム プロセッサまたはソフトウェア コンポーネントを備えたシステムでは、これらのタスクをアーキテクチャ上、Flutter で駆動する UI から分離することができます。

注

KDAB の見解：Flutter に潜む「隠れたコスト」の落とし穴

Flutter は BSD ライセンスで提供されているため、企業は Flutter が最も安価な選択肢であると考えがちです。しかし、C/C++ チームが Dart を学習しなければならず、ハードウェア チームが Flutter のオーバーヘッドに対応するために RAM を 2 倍にして、GPU をアップグレードしなければならない場合、総所有コスト (TCO) は Qt や Slint などの高度に最適化されたフレームワークの商用ライセンス コストを上回ることが少なくありません。

3.5 その他の特筆すべきフレームワーク

- **Altia**: テキサス・インスツルメンツによるデモを含むパートナー エコシステムを有する、組込み機器および車載 HMI 開発向けの商用プラットフォーム。モデルベースの UI 開発ワークフローや、組込み向けターゲットへのベンダーサポート付きの展開を求めるチームにとって、評価する価値があります。一方で、ライセンス、生成コードを用いる開発ワークフロー、ランタイムのメモリ使用量、グラフィックス スタックとの統合、および既存のアプリケーション アーキテクチャとの適合性については、検討が必要です。
- **Avalonia UI**: クロスプラットフォーム .NET フレームワーク (C#/XAML)。DRM/KMS を介して Linux 上で実行できます。デメリット：.NET ランタイムによるオーバーヘッド。マイコンには適していません。
- **Crank Software:** MPU およびマイコンに対応する商用 GUI 開発ツールチェーン。幅広いシリコンへの対応と、迅速な UI プロトタイピングが特長で、ベンダに依存しない設計 / ランタイムの開発パイプラインを求めるチームにとって、評価する価値があります。
- **Embedded Wizard**: 車載機器や白物家電の分野に強みを持つ、マイコン / MPU 向けの商用フレームワーク。デメリット：ライセンス コストの増加。
- **Kanzi (Rightware)**: 車載インフォテインメント分野で広く採用されている高価格帯フレームワーク。デメリット：ライセンス費用が高額。
- **.NET MAUI**: Microsoft の Xamarin の後継。モバイル / デスクトップ向けに強みあり。組込み Linux への対応は、実質的にコミュニティによるサポートに限定。デメリット：組込みデバイス自体よりもコンパニオン アプリに適しています。
- **Unreal Engine UMG**: ハイエンド サイネージや高級 EV のダッシュボードに使用される AAA クラスのゲーム エンジン。デメリット：要求されるハードウェア性能が高く、習得までに時間を要し、バイナリ サイズが大きい。

4 KDAB の選定フレームワークと実装に関するベスト プラクティス

本版では、セクション 2.1 に記載されているテキサス・インスツルメンツ Sitara HMI 階層表を、プラットフォームを優先した主要な選定ガイドとして位置付けています。表 2-1 では、最初に検討すべきフレームワークと、評価対象となる代替フレームワークを示しています。

4.1 シリコン以外の選定基準

4.1.1 ライセンスとビジネス モデル

- **制約の緩やかなオープンソース：**LVGL (MIT)、Flutter (BSD)。
- **コピーレフト型のオープンソース：**Qt (GPL/LGPL)、Slint (GPL)。静的リンクおよびティボイゼーション (tivoization) がもたらす影響を確認してください。
- **商用ライセンスが必要：**Qt (クローズドソース / 静的)、Slint (商用)、Qt for MCUs。、Embedded Wizard、Kanzi。

4.1.2 チームの専門知識と開発速度

- **C/C++ 組込みチーム：**Qt、Slint (C++ バインディング)、または LVGL。
- **Rust チーム：**Slint。

Copyright © 2026 Texas Instruments Incorporated

- **C#/.NET チーム:** Avalonia。
- **モバイル開発者:** Flutter (ハードウェアが対応している場合)。

注

開発速度に関する補足:宣言型フレームワーク (QML、Slint、Flutter、XAML) は一般的に、必須の手動コーディング (LVGL、Qt Widgets) よりも大幅に高速なイテレーションを実現します。

4.1.3 長期メンテナンスおよび認定

- **実績のあるフレームワーク:** Qt (25 年以上)、LVGL (10 年以上)、Embedded Wizard。
- **新興のフレームワーク:** Slint (比較的新しく、商業的な支援を受けているもの)、Flutter の組込み対応 (コミュニティ主導)。
- **セーフティクリティカルな用途 (IEC 62304/ISO 26262):** Qt (商用版) と Embedded Wizard が、リスクを抑えた堅実な選択肢です。Slint は検証可能ですが、そのためには相応の投資が必要です。Flutter はガベージ コレクションを採用しているため、ハード リアルタイム要件に対応した認証の取得は極めて困難です。

Copyright © 2026 Texas Instruments Incorporated

4.2 実装のベスト プラクティス：パフォーマンスの最適化

4.2.1 メモリおよびグラフィックスの管理

- **決定論的なメモリ：**可能な限り静的アロケーションを使用してください。最悪時の RAM 使用量を早い段階でプロファイリングし、頻繁に生成・破棄されるオブジェクトにはメモリ プールを使用してください。
- **ターゲット ハードウェアで早期にテストする：**一般的でコストの高い失敗例として、ワークステーション上で UI 全体を開発し、リリース直前になって初めてターゲット上で検証することが挙げられます。最初のスプリントからターゲットのシリコン上で動作させ、温度特性やブート時間の測定も含めて検証してください。
- **アセットの最適化：**アセットを圧縮してストリーミングしてください。Qt プロジェクトでは、常に Qt Quick Compiler を使用してください。
- **レンダリング効率：**効率的なレイヤー構成によってオーバードローを最小限に抑えてください。利用可能な場合は、OpenGL、OpenGL ES、Vulkan、またはプラットフォーム固有の API を使用して、ハードウェア アクセラレーションを活用してください。

4.2.2 電力効率

- 使用していないときに停止できるものは、すべて停止する必要があります。これには、GPU、ディスプレイ、コントローラ、センサ、および周辺装置が含まれます。UI のステート マシンをシステムのパワー マネージャと共同設計します。

4.3 起動時間に関する考慮事項 (「コールド ブート」の問題)

アプリケーションの起動時間は、組込みデバイスの体感品質に決定的な影響を及ぼします。

- **Qt:** Qt Quick Compiler は、AOT コンパイルによって、コールド ブート時の起動時間を数百ミリ秒短縮します。
- **Slint:** バイナリ サイズが小さく、起動時のオーバーヘッドが最小限に抑えられています。
- **LVGL:** 最小限のランタイム セットアップで迅速に初期化できます。
- **Flutter:** Dart VM とフレームワークのロードにより、初期化が長くなります。

共通の最適化手法：依存関係を必要最小限に絞り込み、事前コンパイルを行うことで、ブート時間を短縮できます。静的リンクも有効な場合がありますが、万能な手段ではありません。デフォルトの性能改善手段として一律に適用するのではなく、選択的に適用してください。

4.4 電力効率 (バッテリ駆動デバイス)

- UI のアクティビティに基づいて適応型フレームレートを実装します (たとえば、アクティブ時は 60fps、アイドル時は 30fps、静止時は 1fps)。
- 使用していないときに停止できるものは、すべて停止する必要があります。これには、GPU、ディスプレイ、コントローラ、センサ、および周辺装置が含まれます。UI のステート マシンをシステムのパワー マネージャと共同設計します。
- メモリ帯域幅を削減し、CPU のウェーク イベントを最小限に抑えるため、ディスプレイの更新をまとめて実行します。

4.5 今後の動向と検討事項

組込み UI を取り巻く環境は、現在も急速に進化し続けています。定期的な再評価により、長期にわたる製品ライフサイクル全体を通じて、テクノロジーの選定を最適な状態に保つことができます。

- **WebAssembly および Web ベースの UI:** ダッシュボードおよびリモート監視向けに、WebAssembly を介して UI を展開することへの関心が高まっています。Qt for WebAssembly と Slint for WebAssembly は、どちらも実用的な選択肢です。依然として、移植性とランタイムのオーバーヘッドとの間でトレードオフが生じます。
- **機械学習の統合：**最新のプラットフォームには AI アクセラレータが搭載されることが増えており、ジェスチャの認識や予測型インターフェイスを実現するため、UI フレームワークと ML (機械学習) パイプラインとの統合が進んでいます。Qt はすでに ML フレームワークとのバインディングを提供しています。
- **Rust エコシステムの成長：**ガベージ コレクションを使用せずにメモリの安全性を実現できることは、組込みシステムにとって非常に魅力的です。Slint は Rust ネイティブの UI フレームワークとして先行しており、Qt も実験的な Rust バインディングの開発を積極的に進めています。
- **ハードウェアの進化：**新しいマイコンは、エントリレベルの MPU の性能に近づいており、マイコンと MPU の境界が曖昧になっています。これにより、従来のマイコン向け予算の範囲内で、より高機能な UI を実現することが可能となっています。Vulkan の採用は増加しており、ベンダ間でより一貫性のあるグラフィックス API の実現が期待されています。

Copyright © 2026 Texas Instruments Incorporated

5 まとめ:TI と KDAB の見解

組込み UI テクノロジーの選定は、妥協点を見いだす作業です。「最適な」フレームワークは一切存在しません。ハードウェアの性能、チームの専門知識、ライセンスに関する方針、および製品のライフサイクルを組み合わせた条件に対して、最適なフレームワークが存在するだけです。同様に、「最適な」プロセッサも一切存在しません。TI Sitara MPU 製品ラインアップは、コストを最適化したエントリーレベルの設計から、多機能なマルチディスプレイ対応システム、さらには AI アクセラレーションを備えたシステムまで、組込み HMI のあらゆる性能要件に対応できるよう、意図的に幅広い製品構成で設計されています。

TI は Sitara MPU 製品ラインアップ全体にわたり、Linux SDK のサポート、ディスプレイドライバ、ハードウェアアクセラレーションに対応したグラフィックス スタックなど、一貫性のあるソフトウェア基盤を提供しています。これにより、開発チームは、製品要件の変化に応じて、ソフトウェアへの投資を無駄にすることなく異なる製品階層へ移行できます。現時点でのプロジェクトの制約に合わないフレームワークであっても、ハードウェアの性能の向上やエコシステムの成熟に伴い、採用可能になる場合があります。そのため、フレームワークの選定は一度限りの決定とせず、状況の変化に応じて見直すべきものとして捉えてください。

5.1 KDAB Group について

KDAB Group は、デスクトップ、組込み、モバイルの各プラットフォームにおける Qt、C++、および OpenGL アプリケーションのアーキテクチャ、開発、設計を専門とする、世界有数のソフトウェア コンサルティング企業です。KDAB は、Qt への独立系コントリビューターとして最大規模を誇り、世界で初めて ISO 9001 認証を取得した Qt 専門のコンサルティングおよび開発企業です。KDAB の専門家は、ランタイムの構築、ネイティブ技術と Web 技術の組み合わせ、ハードウェア スタックにおけるパフォーマンスの問題や移植に関する問題の解決に取り組んでおり、その顧客は数百社に及び、その多くは Fortune 500 に名を連ねる企業です。KDAB のツールと、複雑なアプリケーションの作成、デバッグ、プロファイリング、移植に関する豊富な経験は、世界中の開発者がプロジェクトを成功に導くうえで役立っています。KDAB の講師は全員が専任の開発者であり、Qt、OpenGL、およびモダン C++ に関する、業界最高水準の実践的なトレーニングを複数の言語で提供しています。

5.2 テキサス・インスツルメンツの Sitara MPU グループについて

テキサス・インスツルメンツの Sitara 製品ラインアップは、産業用オートメーション、HMI、ロボット、モーター制御、エッジ AI アプリケーションなどを対象として設計されたプロセッサ ファミリです。Arm® Cortex-A® アーキテクチャをベースとした Sitara MPU ファミリは、エントリレベルのコスト最適化済みデバイスから、マルチコアデバイスや AI アクセラレーション デバイスに至るまで、多様な性能レベルを網羅しています。TI は、市場投入までの時間を短縮するため、Linux 向けプロセッサ SDK、リアルタイム ソフトウェア、リファレンス デザイン、評価基板に加え、システム オン モジュール (SoM) およびキャリアボードの幅広いパートナーで構成されるエコシステムなど、包括的な開発リソースを提供しています。組込みプロセッサ分野における数十年にわたる経験を活かし、TI は 10 年以上にわたる製品ライフ サイクルをサポートしています。これにより、産業分野および商業分野のお客様は、長期間にわたる過酷な使用が求められる用途においても、安心して Sitara を採用した製品を開発できます。

6 参考資料

- KDAB Group、www.kdab.com
- テキサス・インスツルメンツ、www.ti.com

Copyright © 2026 Texas Instruments Incorporated

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

Copyright © 2026, Texas Instruments Incorporated

最終更新日：2025 年 10 月