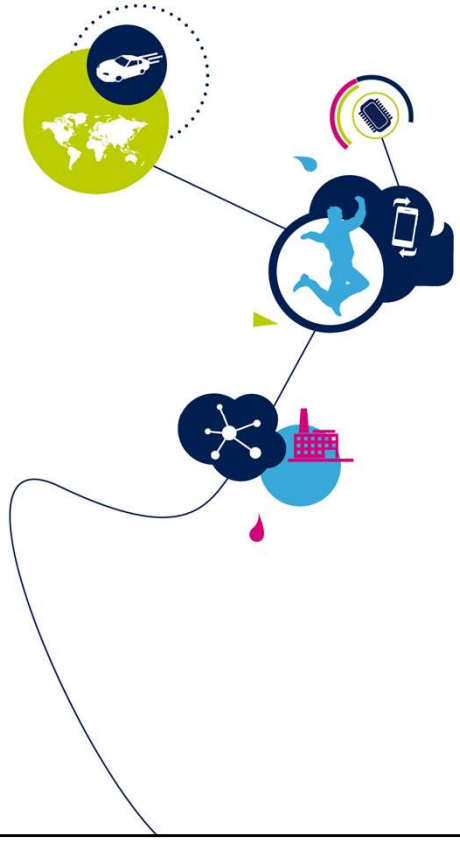
STM32MP1 SWARCH

組込みソフトウェアのアーキテクチャ

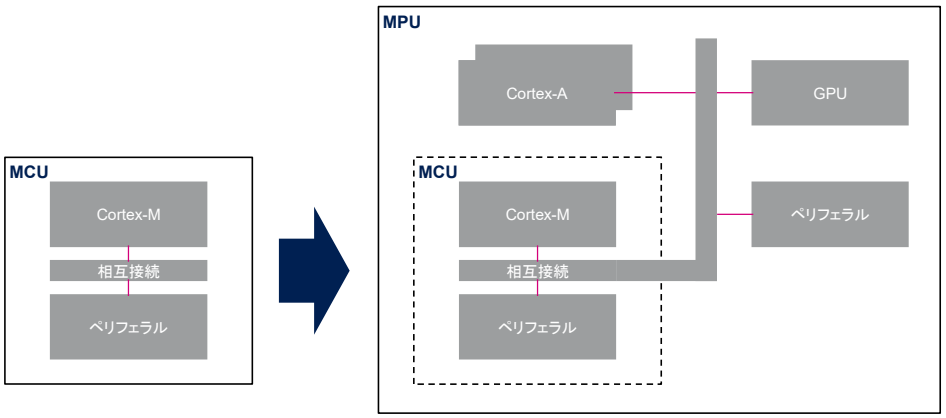
1.0 版



life.augmented

ようこそ。このトレーニングでは、主に STM32MP1 マイクロプロセッサシリーズを対象とした組込みソフトウェアのアーキテクチャとソフトウェア配布について説明します。

MCU から MPU へ



life.augmented

出典：**ST Wiki article** [Getting started with STM32 MPU devices]

マイクロコントローラユニット、いわゆる MCU は、Arm Cortex-M などの MMU のないコアを中心に構築されており、ベアメタルまたはリアルタイムオペレーティングシステムの環境での決定的動作に非常に効率的です。STM32 MCU は、多くのアプリケーションに十分なスタティック RAM と Flash メモリを組み込んでおり、さらに外部メモリで補えます。

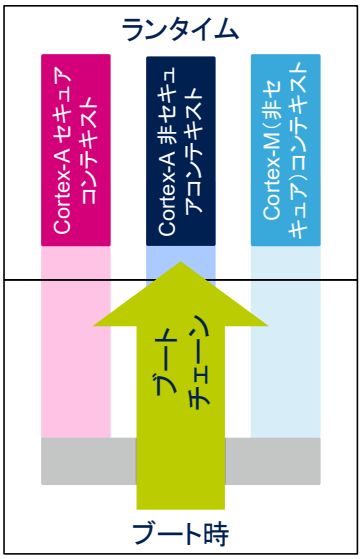
マイクロプロセッサユニット、いわゆる MPU は、Arm Cortex-A などのコアに依存しており、仮想メモリ空間を管理するメモリ管理ユニットを備えているため、Linux などの豊富なオペレーティングシステムを効率的にサポートできます。高速相互接続により、処理装置、広バンド幅のペリフェラル、外部メモリ、およびオプションのグラフィック処理装置（GPU）間のブリッジが形成されます。

STM32MP1 と呼ばれるこの最初の MPU プラットフォームは、デュアルコア Arm Cortex-A7 と Arm Cortex-M4 コアを中心に構築されています。このプラットフォームは、産業、コンスーマ、ヘルスケア、ホームおよびビルディングオートメーションなどの複数の市場セグメントに対応することを目的としています。

MPU の世界に円滑に移行するための ST マイクロエレクトロニクスのアプローチは、MCU を MPU のサブシステムと見なして、両方の世界を 1 つのデバイスに配置することで構成されています。

マルチコアアーキテクチャの概念

- ハードウェア実行環境
  - « 1 つのコアと 1 つのセキュリティモード »
- 実行されるファームウェアのランタイムコンテキスト
  - Arm® Cortex®-A セキュア（Trustzone）は OP-TEE を実行
  - Arm Cortex-A 非セキュアは Linux を実行
  - Arm Cortex-M（非セキュア）は STM32Cube を実行
- ペリフェラルのランタイムコンテキストへの割当て
  - 割当てまたは共有



life.augmented

出典：**ST Wiki article** [Getting started with STM32 MPU devices]

STM32MP1 などのマルチコアアーキテクチャをよく理解するには、いくつかの新しい概念を導入する必要があります。

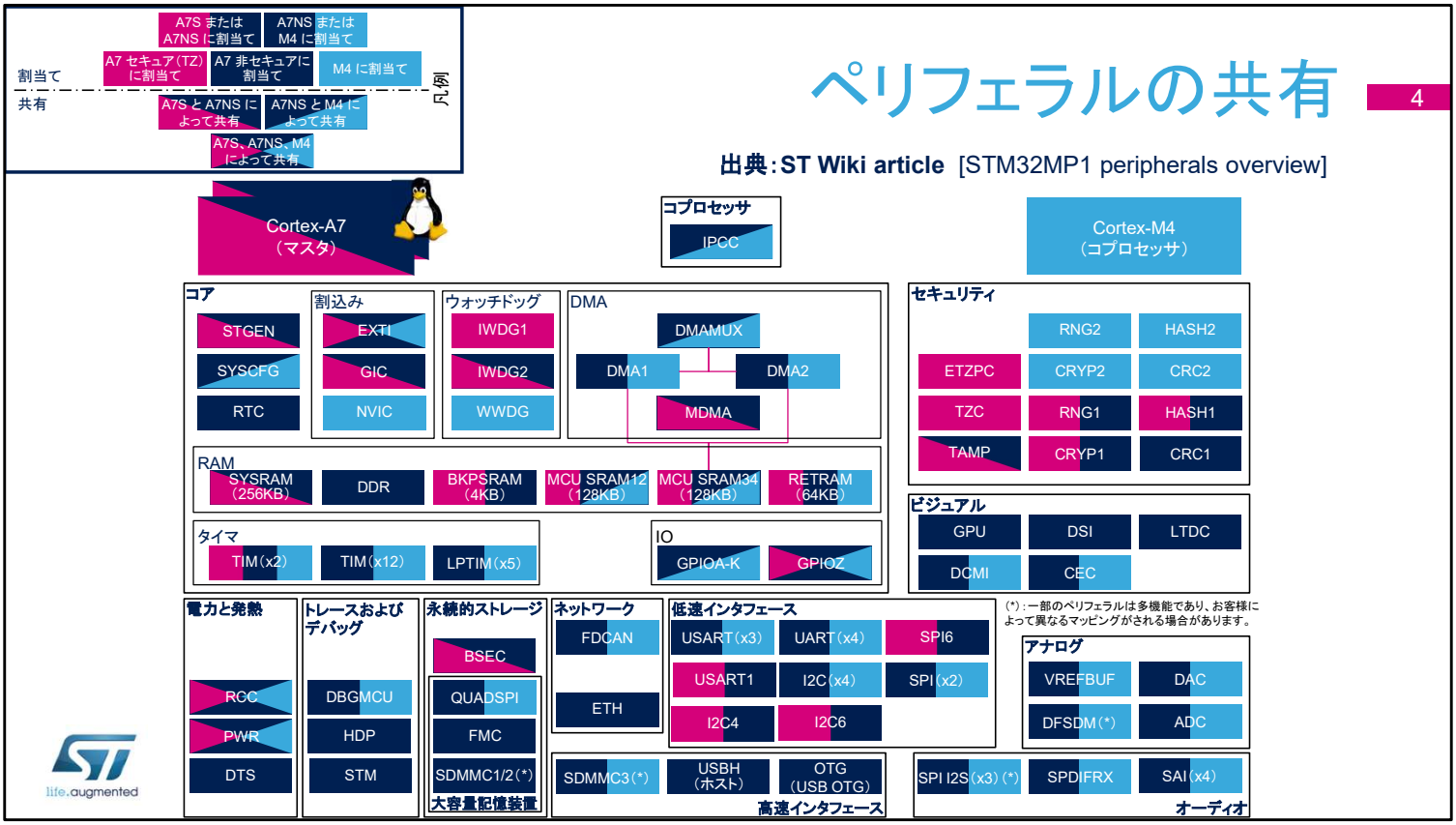
まず、1 つのハードウェア実行環境は、1 つのコアと 1 つのセキュリティモードによって定義されます。
STM32MP1 マイクロプロセッサには、以下の 3 つのハードウェア実行環境があります。

- Arm Cortex-A セキュア（Trustzone とも呼ばれます）
- Arm Cortex-A 非セキュア
- Arm Cortex-M（非セキュアモードのみをサポート）

各ハードウェア実行環境は、ブート時のブートチェーン実行の一部をサポートし、実行時のファームウェアの実行をサポートします。また、STM32MP1 マイクロプロセッサには、以下の 3 つのランタイムコンテキストがあります。

- Arm Cortex-A セキュアは OP-TEE セキュア OS を実行します。
- Arm Cortex-A 非セキュアは Linux OS を実行します。
- Arm Cortex-M は STM32Cube を実行します。

「ペリフェラルの割当て」という用語は、1 セットのペリフェラルをランタイムコンテキストに割り当てる動作を識別するために使われます。
各ペリフェラルは、1 つのランタイムコンテキストに割り当てるか、複数のコンテキストで共有できます。
共有ペリフェラルは通常、リセットとクロック制御用の RCC のようなシステムリソースです。



前のスライドで紹介した概念に沿って、この図の凡例は、各ペリフェラルの以下の 2 つの可能な状態を区別して示しています。

- 「専有」とは、1 つのハードウェア実行コンテキストのみが実行時にペリフェラルを使用していることを意味します。単一色のボックスは割当てが静的であることを意味しますが、垂直セパレータ付きの 2 色または 3 色のカラーボックスは、アプリケーションのニーズに応じて、特定のコンテキストにペリフェラルを割り当てるためにユーザの選択が必要であることを意味します。この割当ては、STM32CubeMX ツールで、または手動で行うことができ、Cortex-A7 セキュアおよび Cortex-M4 のハードウェア分離によって補強されています。
- 「共有」とは、ペリフェラルが同時に 2 つまたは 3 つでさえも異なる実行コンテキストによって使用できることを意味します。このモードは、共通リソースにアクセスするときに、指定されたコンテキスト間で競合が発生しないことを保証するレジスタバンキングや他のメカニズムを使うことを意味します。

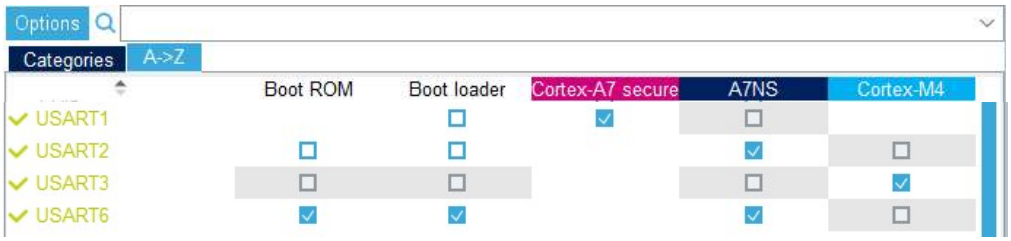
例：

- 左側にある TIM インスタンスは 1 つのランタイムコンテキストに割り当てることができ、このコンテキストのみによって使用されます。
- すぐ下の RCC は、3 つのランタイムコンテキストから同時にアクセスできるシステムペリフェラルです。

割当てに関係するすべてのメカニズムは、ST Wiki [STM32MP15 peripherals overview]の記事で説明されています。

この図は、STM32 MPU 組込みソフトウェアディストリビューションにおける割当ての ST マイクロエレクトロニクスの推奨または選択を示していることに注意してください。その他の可能性については、STM32MP15 のリファレンスマニュアルで説明されている可能性があり、後に弊社のディストリビューションで検討される場合があります。

STM32 CubeMX による ペリフェラルの割当て



このスクリーンショットは、STM32CubeMX ツールで実行できる割当ての例を示しています。

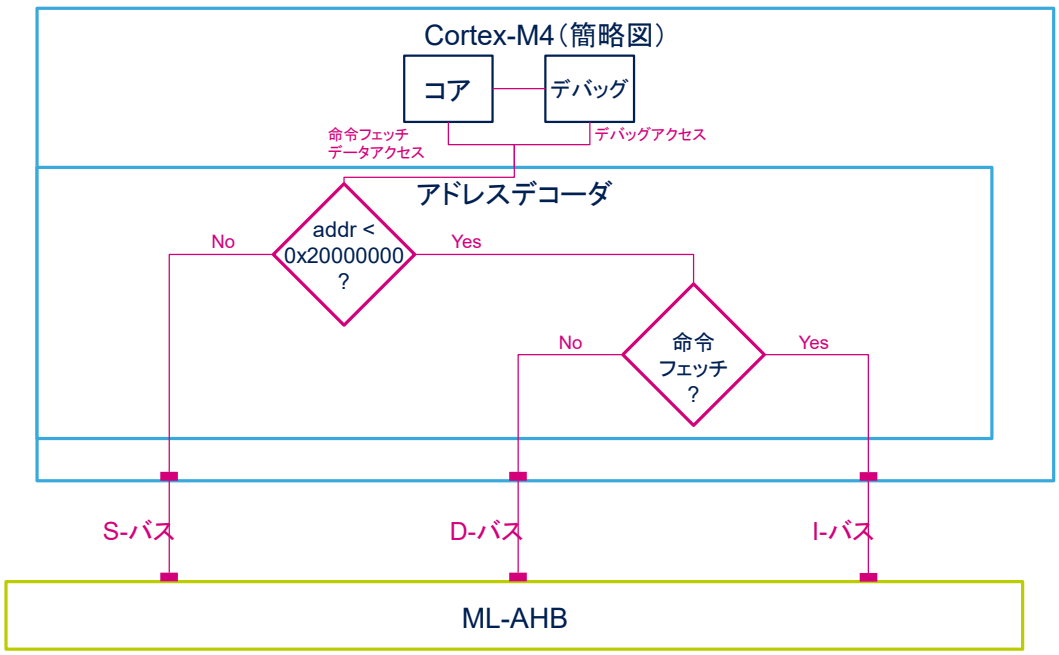
- USART1 は、OP-TEE 用に Cortex-A7 セキュアに割り当てられています。
- USART2 は、Linux 用に CortexA7 非セキュアに割り当てられています。
- USART3 は、STM32Cube 用に Cortex-M4 に割り当てられています。

USART6 に使用されている、これまで言及されていなかった左側の 2 つの列に注意してください。

- « ブート ROM »列は、ROM コードがこのインスタンスから起動できるようにしています。この選択により、ピンの可能性が ROM コードでサポートされるピンのみに制限されます。
- « ブートローダ »列は、選択されたペリフェラルがブートローダから見えるようにします。この選択の主な目的は、ブートローダ用に生成されるデバイスツリーのサイズを制限することです。

ROM コードとブートローダは、プラットフォームのブートのトレーニングで紹介されています。

Cortex-M4 ポート

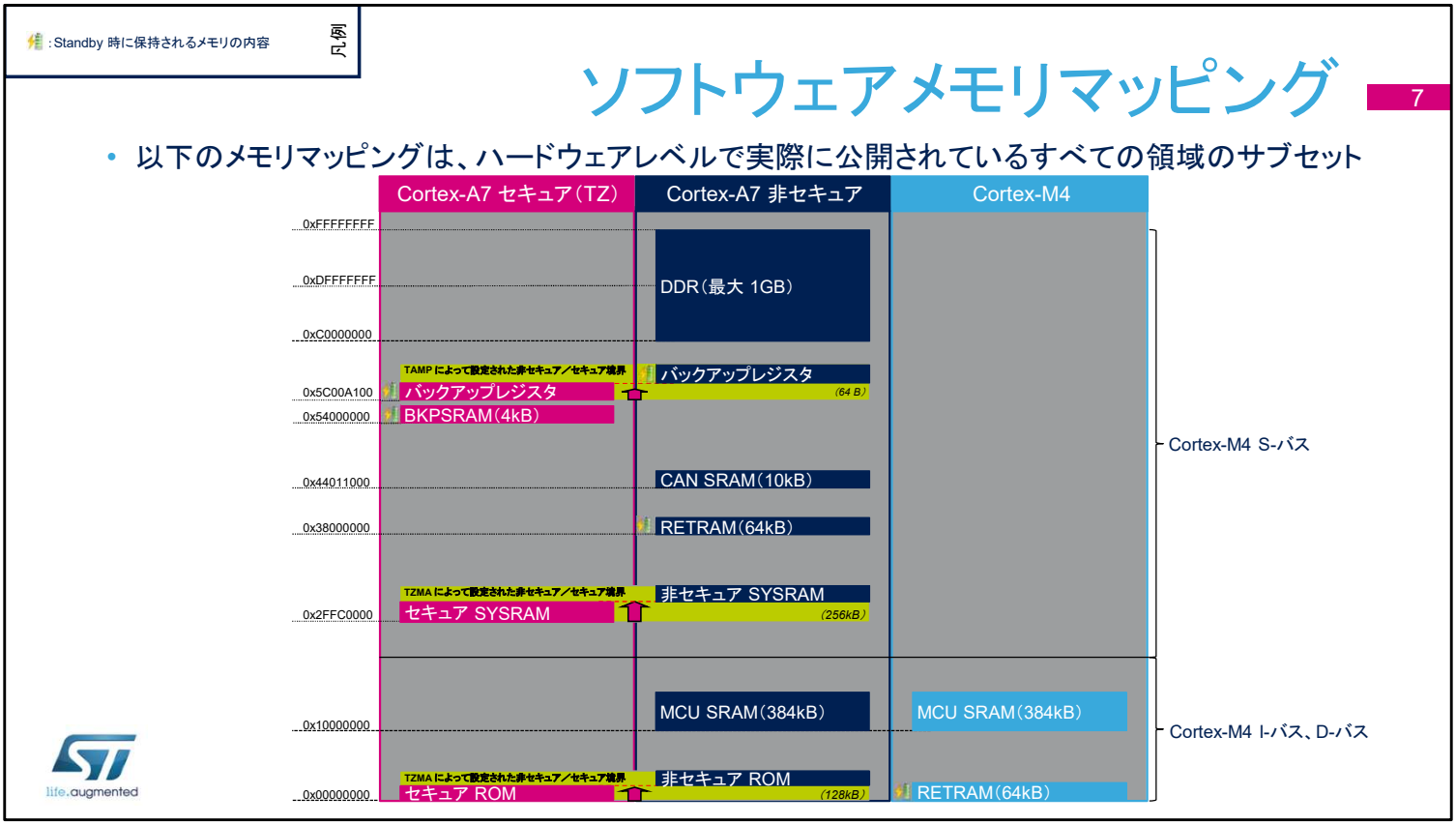


出典：**ST Wiki article** [STM32MP1 RAM mapping]

このスライドは、図に示した 3 つの異なるポートを介して MLAHB 相互接続に接続されているという、Cortex-M4 の特異性に焦点を当てています。

- I-バスは、0x00000000--0x1FFFFFFF のアドレス範囲のコードの命令をフェッチするために使用されます。
- D-バスは、0x00000000--0x1FFFFFFF のアドレス範囲のデータの読取り／書込みに使用されます。
- S-バスは、0x20000000--0xFFFFFFFF のアドレス範囲のすべてのアクセスに使用されます。すべての STM32MP15 内部ペリフェラルのレジスタは、この範囲にマッピングされています。

これらのポート間で Cortex-M4 ファームウェアのアクセスのバランスをとることにより、システム性能のチューニングが可能になります。これが、MCU SRAM が 0x10000000 から始まる最初のアドレス範囲で定義されている理由ですが、0x30000000 から始まる 2 番目の範囲においても参照可能です。便宜上、この 2 番目の範囲は次のスライドでは示していませんが、存在することに注意してください。



この図は、STM32 MPU 組込みディストリビューションでメモリ領域を使用するさまざまな実行コンテキストに関するさまざまな RAM および ROM 領域を図示しています。

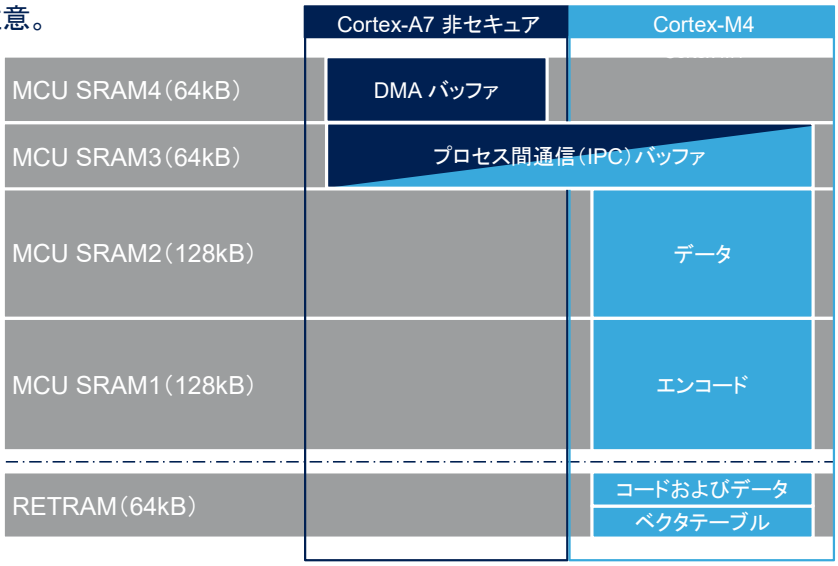
ここで RETRAM が 2 回現れていることに注意してください。

- コプロセッサのファームウェアのロード用に Cortex-A7 非セキュア側で 1 回
- コプロセッサのファームウェア実行のために Cortex-M4 側で 1 回（0x00000000 のアドレスからブートを開始します）

この図は、前の図と同じ ST Wiki 記事 [STM32MP1 RAM mapping]で入手できます。
Wiki では、これらの各領域の使用方法に関する情報も提供しています。

共有 RAM メモリのマッピング

- 前のスライドですでに説明したように、それぞれのコアは同じアドレスで同じ領域を参照しているわけではないことがあることに注意。



- もちろん、各お客様は、製品のニーズに合わせてこのマッピング（領域の場所とサイズ）を調整可能。

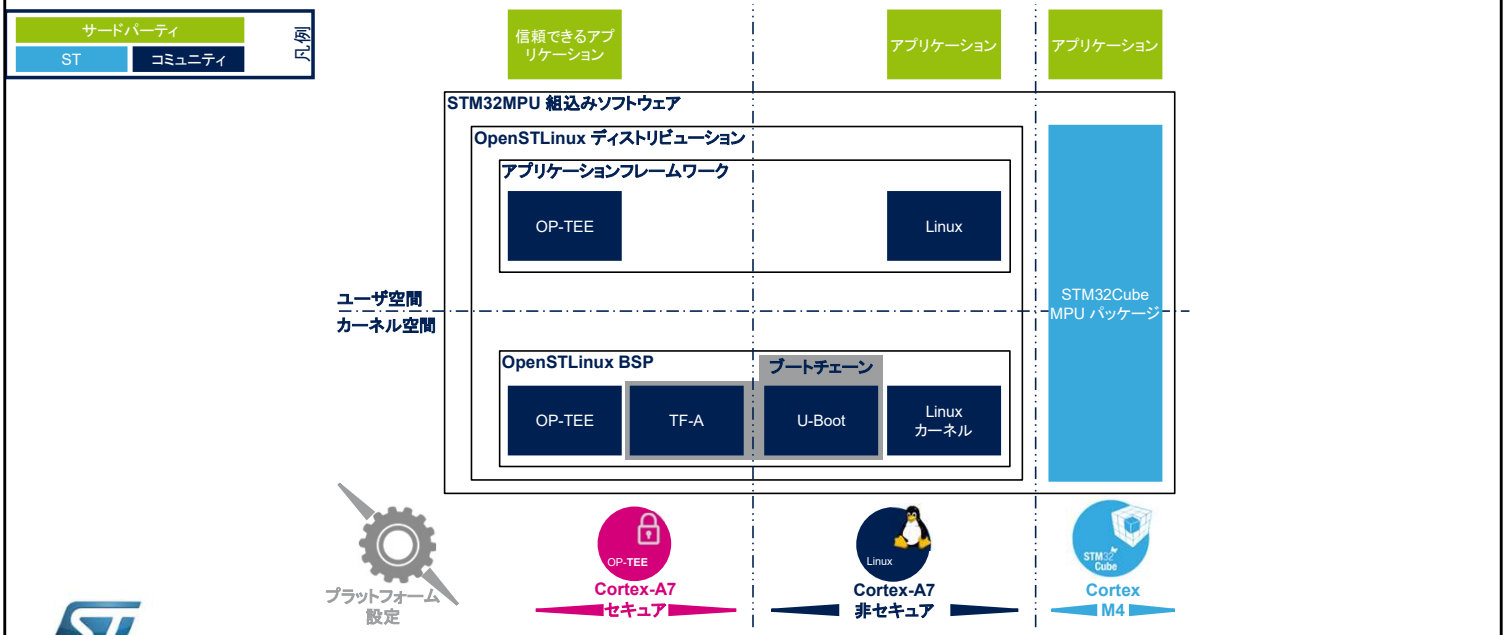
life.augmented

このスライドは、STM32MP1 マイクロプロセッサで利用可能な RAM バンクと、STM32 MPU の組込みソフトウェアディストリビューションに適用される一般的なマッピングを示しています。

- RETRAM は、Cortex-M4 がベクタテーブルとコードおよびデータを配置するために使用します。
- MCU SRAM1 および SRAM2 を使用して、Cortex-M4 ファームウェアの残りのコードとデータを配置できます。
- MCU SRAM3 は通常、プロセス間通信バッファのマッピングに使用されます。これについては、コプロセッサ管理のトレーニングで詳しく説明しています。
- MCU SRAM4 は、DMA1 または DMA2 インスタンスを使用するときに広バンド幅が必要な場合に、Cortex-A7 コア用の DMA バッファを配置するために使用できます。

このマッピングを領域の境界に合わせことは必須ではありませんが、Cortex-M4 メモリのハードウェア分離がバンク単位の粒度でサポートされているため、これは大きな関心事になる可能性があります。

STM32MPU 組込みソフトウェア



出典：**ST Wiki article** [STM32MPU Embedded Software architecture overview]

それでは、STM32MPU 組込みソフトウェアディストリビューションを紹介しましょう。
この図は、STM32MPU 組込みソフトウェアの主要コンポーネントを定義するために使用される用語を説明しています。
Arm Cortex-A で実行される OpenSTLinux ディストリビューションには、次のものが含まれます。

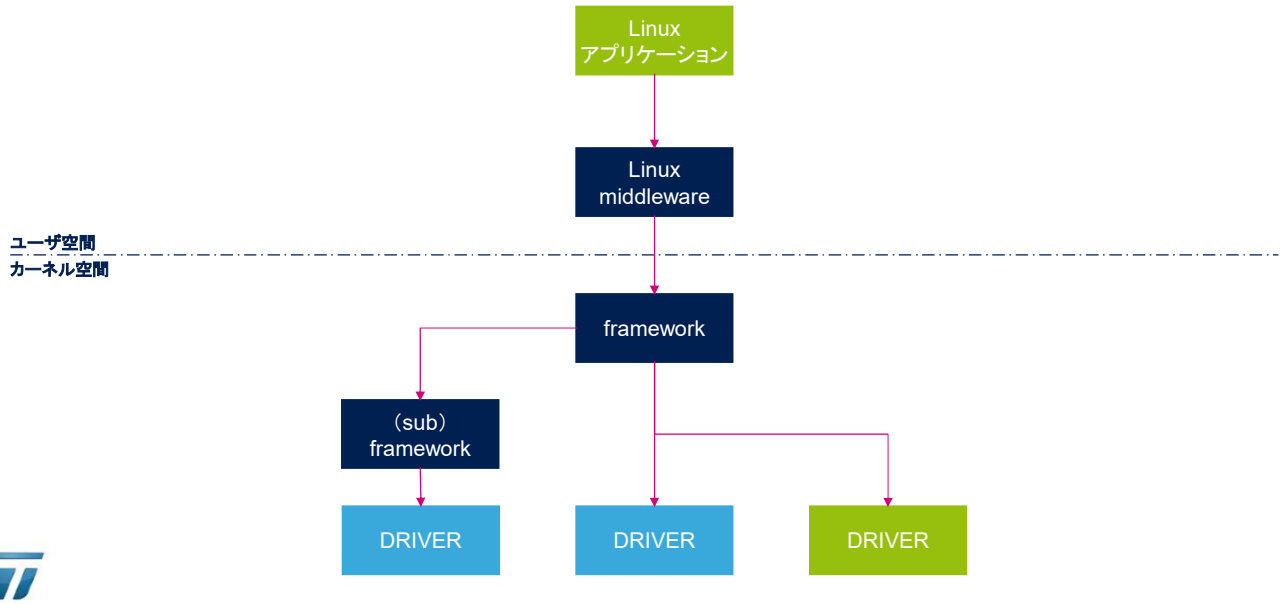
- OpenSTLinux BSPは次の3つを含みます。
  - TF-A および U-Boot に基づくブートチェーン
  - ArmCortex-A のセキュアモードで実行される OP-TEE セキュア OS
  - Arm Cortex-A の非セキュアモードで実行される Linux® カーネル
- アプリケーションフレームワークは BSP に依存するミドルウェアであり、次の API を提供します。
  - OP-TEE 側で、機密操作を許可する信頼できるアプリケーションを実行するためのAPI（これは Linux および STM32Cube MPU パッケージからは見えません）
  - Linux 側で、通常、ディスプレイやタッチスクリーンなどを介してユーザと対話するアプリケーションを実行するためのAPI

STM32Cube MPU パッケージは、ArmCortex-M で実行されます。これは、他の STM32 マイクロコントローラと同様に、コプロセッサ管理を備えた HAL ドライバとミドルウェアに基づいています。

この図は、ST Wiki でクリック可能な画像として利用できるので、探索したい領域に簡単にジャンプして、その詳細を得ることができます。

凡例
サードパーティ
ST
コミュニティ
- 小文字 = コミュニティのフレームワーク
- 大文字 = ペリフェラルドライバ

Linux フレームワークおよびドライバ



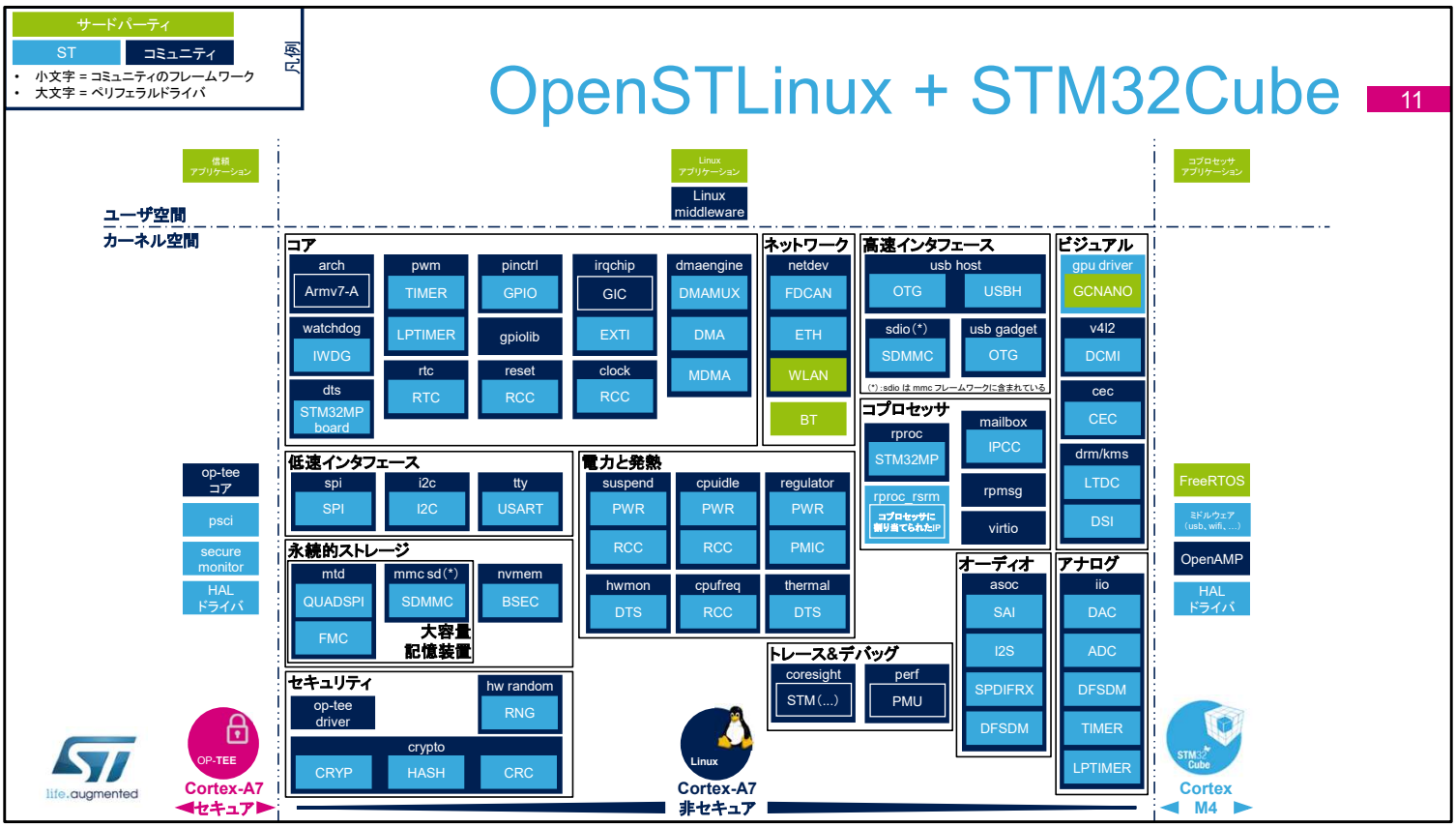
life.augmented

このスライドの目的は、次のスライドの内容を理解しやすくするために、Linux システムがどのように構成されているかについて非常に抽象度の高いレベルの概要を説明することです。

ユーザ側には、Linux カーネルを呼び出すために通常ミドルウェアに依存するアプリケーションがあります。

Linux ドライバは、その API を直接ユーザ空間には公開していません。その代わりに、ドライバは、それらが統合された API をユーザ空間に公開しているフレームワークに登録されます。フレームワークのコンセプトによって、安定したフレームワークの API で動作するアプリケーションが Linux カーネルの将来のバージョンでも動作し、他のドライバを実装する他のプラットフォームでも動作できることが保証されるため、このコンセプトは非常に重要です。

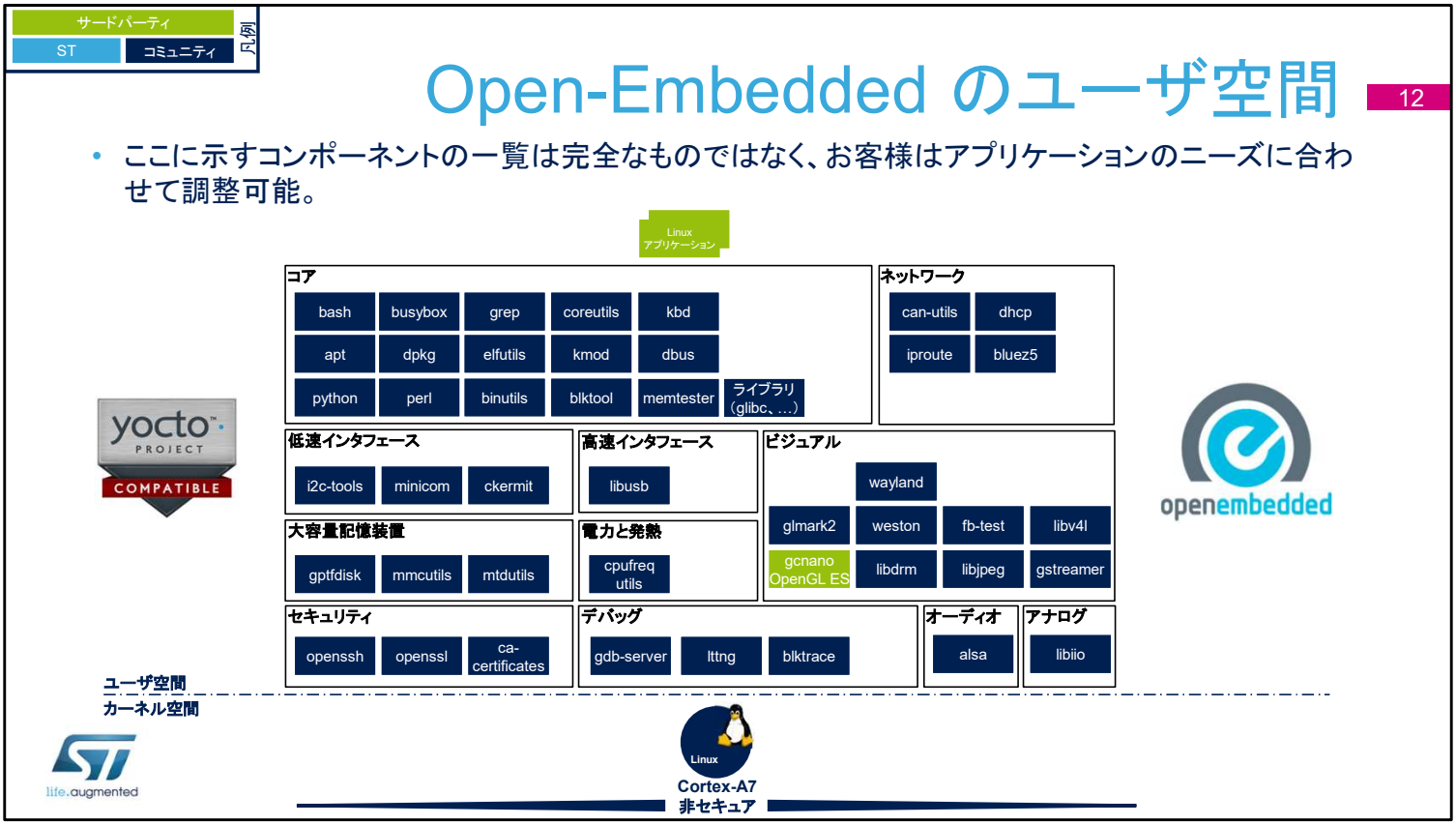
左側では、フレームワークによっては、いくつかのサブフレームワークが含まれることがあるのが分かります。これは、たとえば MMC フレームワークのサブパートである SD カードサポートの場合にあてはまります。



このスライドは、下 STM32MP1 ペリフェラルドライバとそれぞれの Linux フレームワークを示しています。

- フレームワーク名は小文字で書かれています。
- ペリフェラルのドライバ名は大文字で書かれています。
- 色分けによって、各コンポーネントのソースコードの提供元：すなわち、3rdパーティ、コミュニティ、ST マイクロエレクトロニクスを示します。

この図は ST Wiki でも見られます。



STM32MPU 組込みソフトウェアディストリビューションは、組込み Linux プラットフォーム用の Open-Embedded ビルドフレームワークによってビルドされています。

この図は、弊社のディストリビューションに組み込まれている通常のコンポーネントを示しています。ST Wiki と Open-Embedded オンラインディストリビューションを自由に探索して、このセットアップをお客様のニーズに合わせてカスタマイズしてください。