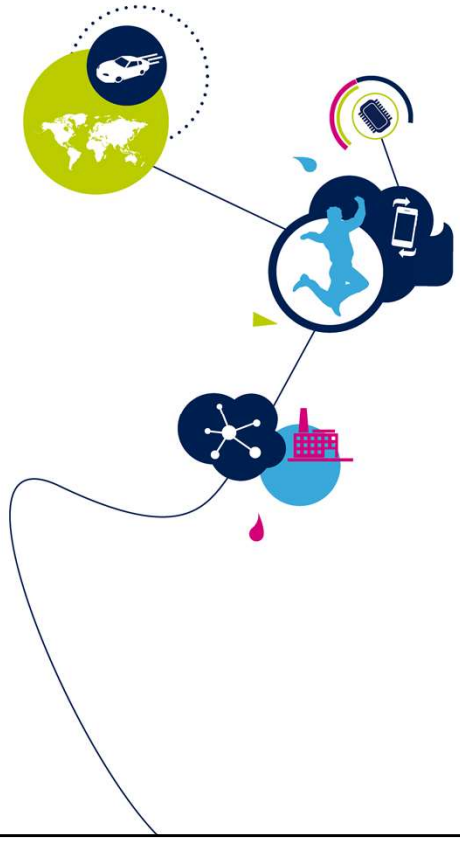
STM32MP1 – 電源管理

電源管理

1.0 版



こんにちは、STM32MP1 デバイスの電源管理のプレゼンテーションへようこそ。ここでは、低電力モードへの移行と終了のシーケンスを説明します。

STM32MP1 の電源メカニズム（1/3）

2

- 電源管理レベルでは、次の 2 つの CPU クラスタが見えます。
  - CPU1 または MPU とも呼ばれるデュアル Cortex®-A7 コア
  - CPU2 または MCU とも呼ばれる Cortex-M4 コア
- クラスタごとに複数の電力モードが利用できます – 「C」のプレフィックスが付きます（例：CRUN）
- プラットフォームのモードは、両方のクラスタモードの組み合わせです。
- どちらのクラスタも、共通のレギュレータ（Vddcore）によって電力が供給されます。
- PLL、マルチプレクサ、分周器、ゲートを備えたクロックツリーにより、クロックの制御ができます。

life.augmented

STM32MP1 の低電力メカニズムのハードウェアの説明から始めましょう。
まず、注意することは、STM32MP1 には 2 つのドメイン（クラスタとも呼ばれる）があり、1 つは Cortex-A7 コアに関連し、もう 1 つは Cortex-M4 コアに関連していることです。どちらのドメインも、Vddcore という名前の単一のレギュレータから電力が供給されています。
各ドメインは、それぞれの側で個別に選択できるさまざまな低電力モードをサポートしています。
全体的なチップの状態は、各ドメインの電力モードの組み合わせによって決まります。
柔軟なクロックツリーにより、さまざまな分周器、マルチプレクサ、ゲートを利用し、クロックを微調整できます。

STM32MP1 の電源メカニズム(2/3)

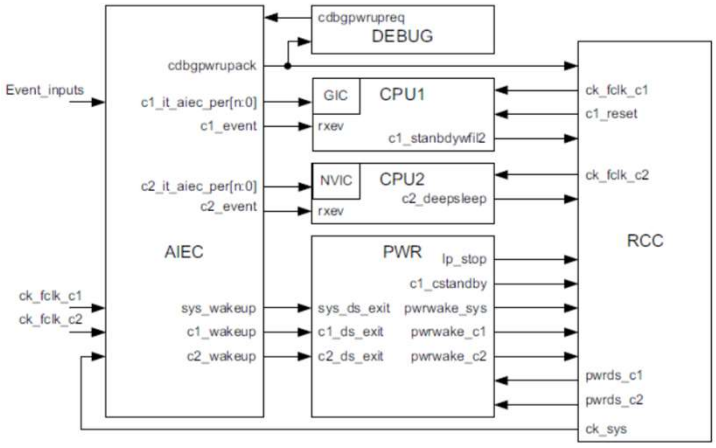
- 電源管理機能は、STM32MP1 の RCC(リセットおよびクロック制御)と PWR ブロックの間に分散されています。
  - RCC ブロックにより、クロックツリーの処理(PLL、マルチプレクサ、分周器、ゲート)とリセット(ペリフェラルローカルリセット、Cortex-A7 リセット、Cortex-M4 リセット、プラットフォームリセット)が確保されます。
  - RCC ブロックにより、電力モードを選択できます。
  - PWR ブロックは、低電力への移行/終了を担当します。

life.augmented

次に、電源管理機能を取り扱う STM32MP1 内部ペリフェラルに焦点を当てましょう。
1 つ目は「RCC」です。これは「リセットおよびクロック制御」を表します。
その役割はクロックツリーを制御することです。これにより、次のクロックツリーの状態を定義するレジスタが提供されます。PLL の設定、マルチプレクサによるクロック経路、および分周器。また、クロックのゲーティングを有効にします。
RCC はシステムのさまざまなリセットも制御します。それには個別のリセット(ペリフェラルのリセット)とプラットフォームのリセットがあります。

2 つ目のペリフェラルは「PWR」です。これは、低電力への移行と終了を担当します。
RCC と PWR は密接に結合されています。これらは協調して動作し、電源管理に関連するすべてのリソース(クロック、電源、ウェイクアップ信号)を処理します。

STM32MP1 の電源メカニズム（3/3）



- 電源管理は RCC ブロックと PWR ブロックによって行われます（クロック、リセット、電源制御）。

AIEC ブロックはウェイクアップソースを処理します。

life.augmented

AIEC のブロックの物理的な表現を次に示します。
ここでは、いくつかの信号を通じてそれらの間の強い相互作用を示しています。
2 つのドメインが表示されており、対応する信号「_c1」と「_c2」も見られます。

ペリフェラルからのウェイクアップ信号を収集する「AIEC」ブロック（「EXTI」とも呼ばれます）について追加の説明をすることができます。AIEC は、正しいドメイン向けに 1 つの単一ウェイクアップ信号を生成するように設定できます。

STM32MP1 の電力モード(1/2)

- **CRUN**:相互接続/ペリフェラルのクロックがオンで、CPU は動作中または WFI 状態。
- **CSLEEP**:CPU が WFI 状態のとき、ペリフェラルのクロックはハードウェアによってスイッチオフ。
- **CSTOP**:クロックはオフで、電源はオンのまま。
- **CSTANDBY**:クロックはオフで、電源はオンのまま。Cortex-A7 側でのみ利用可能。評価ボードでは SD カードの制約のため非サポート。
- **STOP**:両方のクラスタ(Cortex-A7 + Cortex-M4)が CSTOP 状態。電源はオンのまま。
- **LPLV-STOP**:両方のクラスタ(Cortex-A7 + Cortex-M4)が CSTOP 状態。電源は標準の 0.9V まで削減。
- **STANDBY**:Cortex-A7 は CSTANDBY、Cortex-M4 は CSTOP 状態。電源はオフ。DDR の内容は、お客様の選択に応じて保持または喪失。

このスライドでは、STM32MP1 マイクロプロセッサでサポートされている電力モードの定義に焦点を当てています。
前述のように、ドメインレベルの電力モード(「C」のプレフィックス付き)とプラットフォームレベルの電力モード(プレフィックスなし)があります。
プラットフォームのモードを説明するためにも必要なので、ドメインレベルで使用可能なモードから始めましょう。この一覧は、最も軽い節電ゲインから最も深い(大きい)順に並べられています。

最初のモードは CRUN と名付けられています。これは Cortex-A7 側および Cortex-M4 側で使用できます。CPU は実行中または WFI(割込み待機)状態で動作しており、相互接続とペリフェラルのクロックは使用可能です。
CSLEEP モードは、CRUN モードへのアドオンと見なすことができます。このモードでは、CPU が WFI 状態のとき、ペリフェラルのクロックを自動的に停止する可能性があります。この機能は、ペリフェラルごとに個別に設定できます。
CSLEEP は両方のドメインで利用できます。
CSTOP モードでは、CPU、相互接続、およびペリフェラルのクロックはオフです。Vddcore は変わりません。このモードは両方のドメインで利用できます。
CSTANDBY は、ハードウェアリソースの状態に関して CSTOP とよく似ています。二つのモードの違いは、ウェイクアップする方法です。
CSTANDBY は Cortex-A7 側でのみ利用できることに注意ください。
プラットフォームのモードは、上述のクラスタモードの組み合わせです。
両方のクラスタが CSTOP モードのとき、プラットフォームは STOP モードになります。PLL は無効になります。Vddcore は変わりません。
LPLV-STOP モードは、Vddcore が標準の 0.9V に削減されることを除いて、STOP モードに似ています。これは保持モードと見なすことができます。
STANDBY モードは最も深い低電力モードです。Cortex-A7 コアは CSTOP モードまたは CSTANDBY モードで、Cortex-M4 コアは CSTOP モードになります。
このモードでは、PLL を含め、すべてのクロックがオフになり、Vddcore もオフになります。

STM32MP1 の電力モード(2/2)

- ウェイクアップソースとチップの電力モード

電力モード	ウェイクアップソース	ウェイクアップ動作	遅延
WFI	有効になっている IRQ(割込み)すべて	コード実行の再開	なし
CSLEEP	有効になっている IRQ(割込み)すべて	ペリフェラルクロックの再開。コード実行の再開	数µs
CSTOP/STOP	ペリフェラルの IRQ(Uart、I²C、CEC、ETH、SPI、LPTIM …)	システムクロックの再開。コード実行の再開	
LPLV-STOP	RTC、TAMP、ウェイクアップピン、AVD、PVD	電圧の昇圧、クロック再開。コード実行の再開	
CSTANDBY	ペリフェラルの IRQ(Uart、I²C、CEC、ETH、SPI、LPTIM …)	リセットの発生。ROM コードおよびブートローダの再実行	
DDR が SR 状態での STANDBY	RTC、TAMP、ウェイクアップピン	リセットの発生。ROM コードおよびブートローダの再実行	~1s (カーネルのリロードなし)

life.augmented

前述の各モードのウェイクアップ動作を見てみましょう。
ここでの主な注意点は、各モードで利用可能なウェイクアップソースを強調し、モード終了時の遅延を把握することです。
WFI/CSLEEP モードでは CPU クロックはオフになっています。このモードは、有効になっている割込みのいずれかが発生すると直ちに終了します。終了時の遅れは無視できます。
CSTOP では、相互接続、CPU、およびペリフェラルのクロックは再開する必要があります。ウェイクアップ機能を備えたペリフェラルは、ウェイクアップ状態を生成できます。
STOP モードでは、PLL ロック時間のため、CSTOP と比べて追加の遅延があります。
LPLV-STOP では、クロックを再開する前に Vddcore が公称値に戻る必要があるため、STOP モードと比べて追加の遅延があります。Vddcore が低下していると、ウェイクアップ機能を備えたペリフェラルは動作しなくなります。
CSTANDBY は CSTOP と同じウェイクアップペリフェラル IRQ を受けますが、CSTANDBY は Cortex-A7 側のリセットによって終了します。したがって、CPU はブート ROM、ブートローダ、そして Linux カーネルを実行します。
STANDBY モードでは、Vddcore がオフになっていたため、CPU がブート ROM、ブートローダ、および場合によっては Linux カーネルのリロードと再起動を実行する前に、Vddcore を再起動する必要があります。

DDR サブシステムに関する一言:STANDBY モードでは、提供ソフトウェアにより DDR をセルフリフレッシュにしてコンテンツを保持するか、または DDR をオフにすることができます(これはソフトウェアのビルド時に選択されます)。

システムにおける DDR の影響

- DDR は Cortex-A7 側で使われています。Cortex-M4 では使われていません。
- DDR の特性は、以下の理由によりプラットフォーム上で非常に重要です。
  - DDR の種類（DDR または LpDDR）は消費電力に影響します。
  - ターゲット製品（ポータブルデバイス、産業用アプリケーションなど）に応じて、メモリコスト／消費電力のトレンドによって、DDR または LpDDR を選択することになります。
  - セルフリフレッシュを使用する場合（低電力中にメモリを保持する場合）、DDR サブシステムは一部の低電力モードに対して専用の処理を必要とします。

DDR メモリは、電源管理プロセスに関して重要です。
STM32MP1 マイクロプロセッサは、DDR3、LpDDR2、および LpDDR3 のメモリ種類をサポートしています。
STOP/CSTOP/LPLV-STOP モードでは、クロックがオフになっているので、DDR コントローラはリフレッシュ命令を発行できないため、DDR メモリはセルフリフレッシュ状態になります。
STANDBY モードでは、Vddcore がオフになっているので、DDR コントローラには電源が供給されなくなります。目標とする節電量に応じて、DDR をオフにするか、セルフリフレッシュ状態に保つことができます。

配電（1/3）

- 次の 2 つのソリューションが利用できます。
  - PMIC 配電
    - PMIC は、すべての電源を内蔵する専用の ST 回路です。
    - 占有面積を削減し、PCB の集積度を改善します。
    - STM32MP1 との通信用の I²C リンクを備えています。
    - 短絡検出、レギュレータ電圧設定、レギュレータの電源投入／切断シーケンスのカスタマイズなどの高度な機能を提供します。
  - ディスクリート配電
    - これは、PCB で個別のレギュレータを使用することで構成されます。
    - 電圧は通常固定されており、制御はレギュレータをオンまたはオフにすることです。

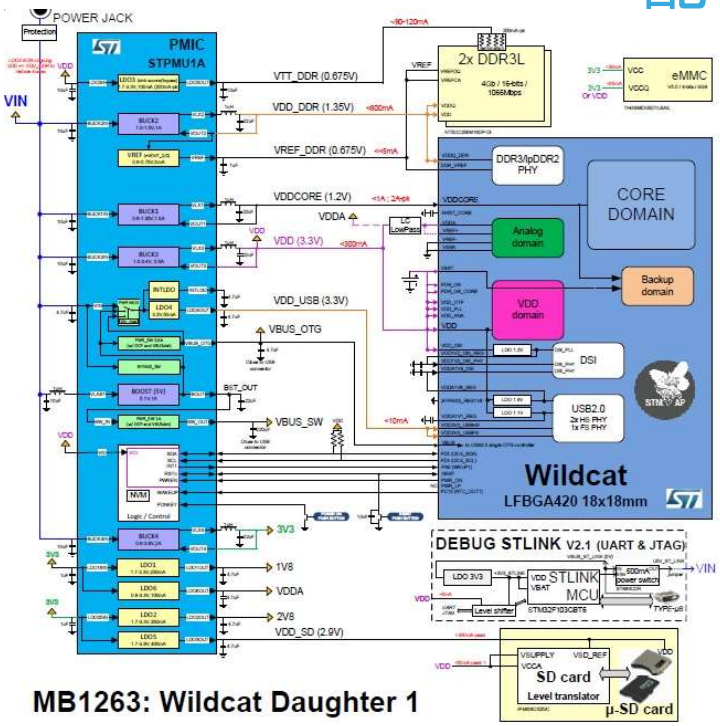
life.augmented

STM32MP1 デバイスは、カメラやディスプレイなどの外部コンポーネントを組み込むこともできる評価ボードにマウントされています。
したがって、ボード配電ツリーは納品の一部です。
次の 2 種類の配電が利用可能です。

- PMIC ベースのソリューション。これは、プラットフォームに必要なすべてのレギュレータを統合する専用回路です。コンパクトで、さらに、過電流検出、熱検出、低電力モード用のバンクレジスタなどの機能を提供しています。
- ボードにはんだ付けされた物理的に独立したレギュレータで構成されるディスクリートソリューション。 電圧は固定です。したがって、たとえば LPLV-STOP のようなモードはサポートされません。

配電（2/3）

- PMIC ソリューション
  - 集積化された電源
  - 制御ロジック

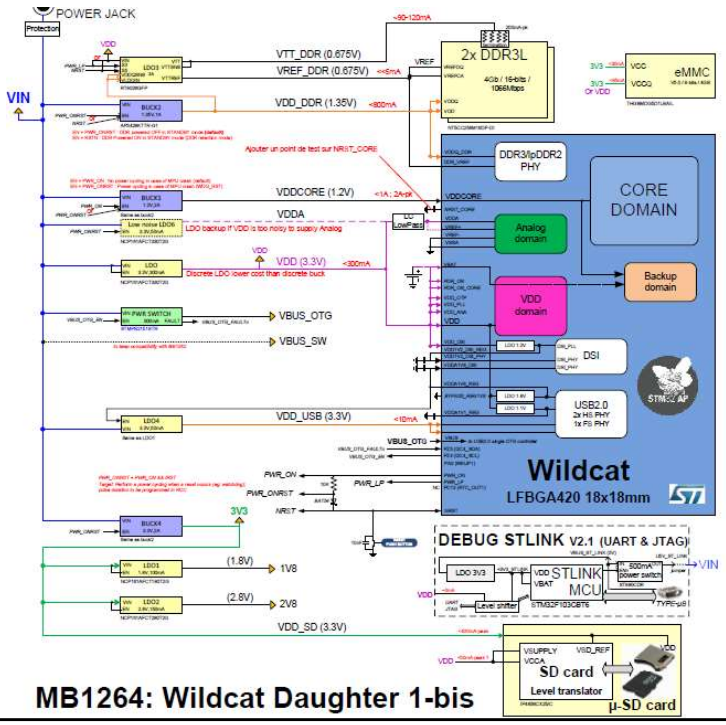


MB1263: Wildcat Daughter 1

これは、PMIC 配電方式のグラフィック表示です。
I²C リンクにより、STM32MP1 は PMIC レジスタにアクセスできます。
2 つのチップ間の一部のハードウェアラインでは、STM32MP1 の PWRCTRL ラインを介して、割込み、リセット、PMIC レジスタバンク選択を生成できます。

配電（3/3）

- ディスクリート ソリューション：
  - レギュレータの組み合わせ
  - 固定電圧



MB1264: Wildcat Daughter 1-bis

このスライドは、ディスクリート電源方式の構築方法を示しています。
レギュレータの制御は、電源のオン／オフで構成されます。
これには、STM32MP1 の PWRCTRL ラインが使用されます。
STM32MP1 の PWRON_LP ラインは、DDR3 の終端をオフにするために使用されます。

Linux の電源管理(1/3)

- Linux/Documentation/power/interface.txt:
  - /sys/power/state は、お客様のアプリケーションによって駆動されるシステムスリープ状態制御ファイルです。
    - 「**freeze**」(アイドル状態へサスペンド):カーネルは一時停止されていますが、電源管理は行われていません。
    - 「**STANDBY**」(電源オンのサスペンド):このモードは STM32MP1 では使用していません。
    - 「**mem**」(RAM へサスペンド):カーネルが一時停止され、アクティブ化されたウェイクアップソースと適合性のある最も深い低電力モードに移行します。
    - 「**disk**」(ディスクへサスペンド):DDR のイメージがストレージデバイス(USB キー、ハードドライブ)に保存されます。**このモードは STM32MP1 ではサポートされていません。**
  - 例:
    - echo mem > /sys/power/state
  - Weston 環境では、このコマンドは「systemctl」コマンドにカプセル化されているのでご注意ください。

life.augmented

ここまで、ハードウェアの状況を説明したので、ソフトウェアの実装について詳しく見る必要があります。
ソフトウェアプラットフォームのブートのトレーニングがあるため、このパートでは bootrom とブートローダの側面については説明しません。
Cortex-A7 側から開始して、アプリケーションは Linux カーネル上で実行されます。
API は、ユーザアプリケーションから電力モードを呼び出すために使用されます。
「mem」は、STM32MP1 低電力モードに到達するために使用されるコマンドです。

Weston 環境では、「mem」コマンドが「systemctl」コマンドに置き換えられていることに注意してください。

Linux の電源管理(2/3)

- Linux 共通クロックフレームワーク(CCF)は、クロックツリーを処理し、下記の標準化された API でデバイスドライバに公開するために使用されます。
  - clk_prepare_enable、clk_disable_unprepare
  - clk_get_rate
  - clk_set_parent、clk_get_parent
  - CCF は ST クロックドライバに依存しています。
- Linux サスペンドフレームワークは、低電力モードに移行するのに使用されます。
  - これは genPD(汎用電源ドメイン)フレームワークと PSCI 1.0 ライブラリに依存しています。
- ウェイクアップソースと低電力モードは genPD により整合しています。

life.augmented

Linux コミュニティの共通クロックフレームワークとリセットフレームワークが使用されています。
これらによって、RCC ペリフェラル内でクロックツリーとリセットを完全に構成し、制御することができます。
Linux サスペンドフレームワークは、ユーザアプリケーションから低電力モードを呼び出すために使用されます。これは、genPD(汎用電源ドメイン)フレームワークおよび PSCI 1.0 ライブラリとインタフェースを取ります。

Linux の電源管理(3/3)

- STANDBY モードの特殊ケース
- 電源(Vddcore)はオフで、モードはチップリセットによって終了します。これは、CPU(Cortex-A7 および Cortex-M4)およびペリフェラルのレジスタ内容が失われることを意味します。DDR の内容は保持されるか、または失われる可能性があります(お客様の選択による)。
- 以下のように、いくつかの STANDBY 終了シーケンスが可能です。
  - Cortex-A7 クラスタ単独で再起動できる(Cortex-M4 は CSTOP にとどまる)
  - 保持 RAM によって、Cortex-M4 クラスタ単独で再起動できる(Cortex-A7 は CSTANDBY にとどまる)
  - Cortex-A7 と Cortex-M4 が一緒に再起動する。Cortex-M4 は保持 RAM によって動作する
- 各シナリオは、デバイスツリーのプロパティにより、HOLD_BOOT 機能を使用して RCC をプログラミングすることで選択できます。

life.augmented

いくつかの設定が可能であるため、STANDBY の終了出口に焦点を合わせることが重要です。
STANDBY モードはチップリセットによって終了されます。つまり、ブート ROM が最初に実行されます。
どの CPU をウェイクアップする必要があるかをブート ROM に指定できます。
Cortex-M4 が単独で、または Cortex-A7 と同時にウェイクアップする必要がある場合、Vddcore が 0 に低下するために SYSRAM が失われるので、そのファームウェア(スリムタイプ)は保持 RAM に配置する必要があります。

ブート ROM は、ハッシュメカニズムを通じてオプションの Cortex-M4 ファームウェア認証サービスを提供していることに注意してください。Linux は、ハッシュを専用のバックアップレジスタに記入することができるため、ブート ROM は STANDBY 終了出口で、ハッシュが Cortex-M4 常駐ファームウェアのものと同じであることを確認できます。目的は、変更された Cortex-M4 ファームウェアの起動を回避することです。

ウェイクアップソースの定義

- ウェイクアップソースは 3 つのグループに分けられます。
  - グループ 1:STOP、LP-STOP モードで使用可能
    - USB、CEC、ETH、USART、I²C、SPI、LPTIM
  - グループ 2:STOP、LP-STOP、LPLV-STOP モードで使用可能
    - PVD、AVD、IWDG、GPIO
  - グループ 3:STANDBY、STOP、LP-STOP、LPLV-STOP モードで使用可能
    - BOR、Vbat mon、Temp mon、LSE CSS、RTC、TAMP、ウェイクアップピン

life.augmented

このスライドは、有効になっているウェイクアップソースに応じて
到達できる低電力モードを示しています。
有効なソースと低電力モードの間の整合性を確保するのは
genPD フレームワークの責任です。

低電力モードのマッピング（1/3）

- 「mem」コマンドは、すべてのアクティビティを停止するよう要求します。
- 一時停止手順中に、有効なウェイクアップソースがチェックされ、最も深い低電力モードに移行します。
- この戦略は、DDR の種類（LpDDR または DDR）に関係なく機能します。
- セキュアモニタのデバイスツリーには、STOP、LP-STOP の中でどの STOP モードをサポートするかを指定するための追加の柔軟性が備わっています。

life.augmented

Linux カーネルは、STM32MP1 低電力モードを呼び出すコマンドを 1 つ提供しています。
プラットフォームは複数の低電力モードをサポートしているため、提供ソフトウェアでは、使用可能なモードに到達するメカニズムが実装されています。

前のスライドでは、各低電力モードですべてのウェイクアップソースを使用できるわけではないと述べています。
genPD ソフトウェアフレームワークを使用して、有効になっているウェイクアップソースを分析し、これらのソースと適合性のある最も深い低電力モードを呼び出します。

いくつかの例を見てみましょう。
1）UART4 がウェイクアップソースとして有効にされています。すると、genPD は STOP モードより深くには行きません。
2）RTC がウェイクアップソースとして有効にされています。すると、genPD は STANDBY モードへの到達を許します。
3）UART4 と RTC がウェイクアップソースとして有効にされています。すると、genPD は STOP モードより深くには行きません。

STOP モードの複数のバージョンが使用できます。お客様は、ソフトウェアのビルド時にどれをサポートするかを選択できます。これを指定するために、セキュアモニタのデバイスツリーに専用のセクションがあります。

低電力モードのマッピング（2/3）

- STOP/LPLV-STOP モードの例
  - これらの低電力モードでは、DDR コントローラクロックがオフになるので、（Lp）DDR メモリがセルフリフレッシュモードになります。
  - セルフリフレッシュモードでは、DDR メモリの消費がシステム全体の電力消費の主な要因です。MPU については、STOP と LPLV-STOP の消費電力の差は非常に小さくなります。この場合、STOP モードを使用することをお勧めします。これにより、PMIC 電圧の立ち上がり時間のペナルティが回避され、ウェイクアップソースの全範囲が受け入れられます。
  - LpDDR2 メモリデバイスは、セルフリフレッシュモードで少ししか電力を消費しません（設計上の DDR の約 6 分の 1）、よって STOP モードではなく LPLV-STOP モードを使用することをお勧めします。

life.augmented

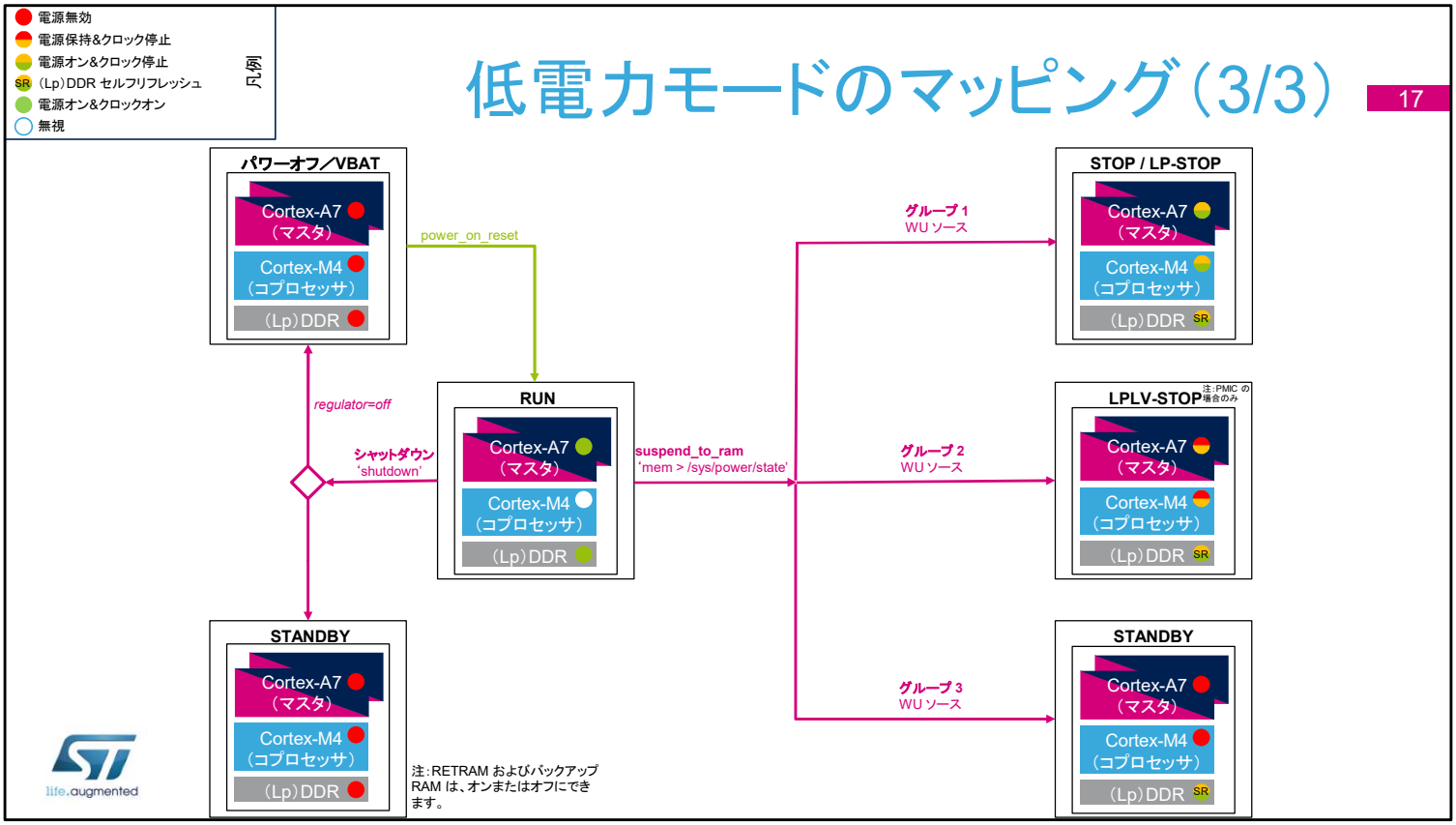
このスライドは、サポートすべき低電力モードを選択できる考えられる分析例を示しています。
ボードに搭載されている DDR の種類を中心にして分析しています。
低電力モードでは、DDR メモリはセルフリフレッシュモードになります。

「STOP モード + セルフリフレッシュモードの DDR3」の全体的な消費電力と、「STANDBY モード + セルフリフレッシュモードの DDR3」の全体的な消費電力を比較すると、DDR3 メモリの消費が主な要因であるため、両方の構成は同じ消費範囲内にあります。したがって、「STOP モード + セルフリフレッシュモードの DDR3」と、「STANDBY + DDR3 オフ」を選択することが理にかなっています。

LpDDR2 で同じ検討を行うと、セルフリフレッシュモードでの LpDDR2 メモリデバイスの消費が少ないため、結論は異なります。よって、LpDDR をセルフリフレッシュモードにして LPLV-STOP または STANDBY モードを選択することが興味深いものになります。

最適な選択を行うには、次のことを考慮する必要があることに注意してください。

- ウェイクアップ遅延時間
- 節電ゲインの目標値
- 必要なウェイクアップソースの種類



ST マイクロエレクトロニクスは、有効なウェイクアップソースの種類を考慮し、低電力モードとの整合性を確保するための全体的な提案を提供しています。
繰り返しになりますが、最終的な選択は、前のスライドで述べた 3 つの基準間のトレードオフであり、お客様は最適な決定のためにそれらを考慮する必要があります。

電源管理の一般的な動的シーケンス

- ユーザアプリケーションは、ウェイクアップソースを選択してプログラムします。
- A7 クラスタまたは M4 クラスタ、あるいはその両方で低電力モードへの移行が要求されます。
- 電力マネージャは、要求されたモードに入るように RCC と PWR をセットアップします。
- DDR は要求された場合、セルフリフレッシュになります。
- ウェイクアップイベントで、RCC/PWR は、到達していた低電力モードに応じて、クロックを再度有効にするか、リセットを発生させます。
- DDR は要求された場合、セルフリフレッシュを終了し、コードの実行が再開されます。

life.augmented

これは、低電力モードへの移行／終了シーケンスの概要です。

低電力モードへの移行は、ユーザアプリケーションによって開始されます。
Linux 側では、すべての活動が一時停止され、RCC と PWR がプログラムされます。
ユーザが少なくとも 1 つのウェイクアップソースを有効にしていることを前提としています。
最後のステップは DDR メモリをセルフリフレッシュ状態にすることです（このオプションが選択されている場合）。
この瞬間から、プラットフォームは低電力モードになり、ウェイクアップイベントの発生を待ちます。

ウェイクアップイベントが発生すると、RCC と PWR はソフトウェアの実行を可能にする前にクロックと電力を復元します。

デバッグのヒント

- Linux のクロックの要約
  - cat /sys/debug/kernel/clk/clk_summary
  - これにより、クロックツリーが周波数と状態で階層的に表示されます。
- Devregs を使用すれば、電源ステータスフラグの監視ができます。
  - PWR_MPUCR:MPU の STOP フラグが格納されています。
  - PWR_MCUCR:MCU の STOP フラグが格納されています。
  - RCC_MP_RSTSR: STANDBY や CSTANDBY などのリセット理由が格納されています。
- HDP(ハードウェアデバッグポート)を使用すると、ウェイクアップ信号や CPU の WFI ステータスなど、電源関連の信号をオシロスコープでチェックできます。

電源管理のデバッグを簡単にするためのヒントをいくつか紹介します。

debugfs は、Linux カーネルのクロック専用のツールです。クロックツリーの設定がコンソールに表示されます。これは、欠落しているクロック、または未使用の場合でもオンのままであるクロックを検出するのに役立ちます。また、間違った周波数の検出にも役立ちます。これは、Linux コミュニティの共通クロックフレームワークからの標準ツールです。

RCC および PWR ペリフェラルから一部のレジスタをダンプすることも重要です。
PWR ペリフェラルでは、一部のフラグが STOP モードに関連しています。STOP モードに到達したかどうかを確認するのに役立ちます。
RCC ペリフェラルでは、興味深い情報がリセットフラグに関連しています。これは、CSTANDBY または STANDBY モードに到達したかどうかを検出するのに役立ちます。
最後に、専用のデバッグペリフェラル(HDP という名前)があります。これは、デバイスツリーを介して、低電力モードへの移行/終了に関わる内部ハードウェア信号を観察できるように構成できます。

代表的な問題(1/2)

- システムがウェイクアップしません

  (すべてのモードで、少なくとも 1 つのウェイクアップソースが有効になっていることを確認してください)

  - (C)STOP/CSTANDBY から:ウェイクアップペリフェラル(EXTI を含む)が適切にプログラムされていることを確認します。
  - (C)STOP/CSTANDBY から:ウェイクアップペリフェラルが HSI_ker または HSE_ker からクロック供給されていることを確認します。
  - STANDBY から:ウェイクアップピン、RTC、または TAMP がウェイクアップソースとして選択されていることを確認します。ウェイクアップのために、ピンの極性とプルアップ/プルダウンをセットアップする必要があります。

life.augmented

このセクションでは、電源管理に関して頻繁に発生する問題について説明し、分析を開始する最初の手順を示します。

システムがウェイクアップしない場合、最初に確認するポイントは、少なくとも 1 つのウェイクアップソースが有効になっていることです。
ウェイクアップする必要があるペリフェラルのクロックソースもチェックする必要があります。動作していなければなりません。

ウェイクアップピンによるウェイクアップの場合、極性やプルアップ/プルダウン構成が正しくないと、ピンは動作しません。

代表的な問題(2/2)

- システムが低電力モードに移行しません
  - モードを呼び出すときに、ウェイクアップイベントがすでに保留されていないかを確認します。
  - cat /proc/interrupts によって、そのようなイベントを示すことができます。
  - EXTI のセットアップを確認します。
- 全体的な消費電力が高すぎます
  - ペリフェラルを使用しない場合は、クロックが解放されていることを確認してください。これは clk_summary コマンドで確認できます。

life.augmented

別の一般的な問題は、システムが低電力モードへの切り替えを拒否することです。
これは通常、CPU の割込みコントローラに存在するすでに保留中のイベントが原因です。これは、入力ラインのプルアップ／プルダウンが正しく構成されていない場合に発生する可能性があります。

実行時のプラットフォームの消費電力が高すぎる場合、確認できる点がいくつかあります。

- クロックを必要とするものがないのに、オンのままのクロックがないか確認してください。
- レギュレータを必要とするものがないのに、オンのままのレギュレータがないか確認してください。
- 共通クロックフレームワークとレギュレータフレームワークの debugfs は、このような状況を特定するのに役立ちます。

参照

- STM32MP1 リファレンスマニュアル: RM0436_STM32MP157xx.pdf
  - 電力モードの RCC および PWR の章を参照してください。
  - EXTI の章では、ウェイクアップソースに関する情報を提供しています。
- STPMIC1 データシート

life.augmented

STM32MP1 ボードに組み込まれたハードウェアの詳細については、STM32MP1 リファレンスマニュアルと STPMIC1 データシートを参照してください。