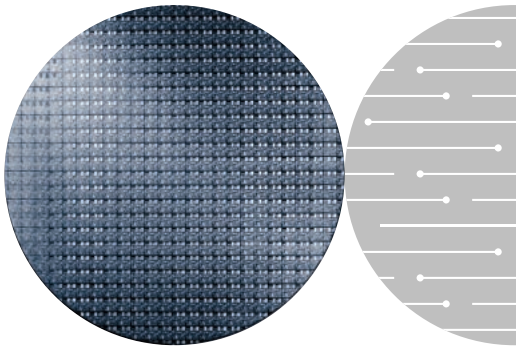
ホワイトペーパー

ワイヤレス Intel SpeedStep® パワー・マネージャ

インテル® PXA27x プロセッサ・ファミリの消費電力を最適化



目 次

1.0 はじめに

インテル® PXA27x プロセッサ・ファミリで初めて採用されたワイヤレス Intel SpeedStep® テクノロジは、CPU の負荷に応じてプロセッサの電力とパフォーマンスを動的に調整する機能を提供します。これにより、モバイル機器の消費電力を大幅に削減し、スタンバイ時間と通話時間を延長できます。ワイヤレス Intel SpeedStep® テクノロジは、Intel XScale® マイクロアーキテクチャに組み込まれていた低消費電力機能を強化したものであり、3 つの低電力ステートを新たに追加し、先進的なワイヤレス Intel SpeedStep® パワー・マネージャ（以下パワー・マネージャまたは PM）ソフトウェアの利用によって、電力とパフォーマンスに対するエンドユーザのニーズに高度に対応します。

OEM（Original Equipment Manufacturer）や ODM（Original Design Manufacturer）は、パワー・マネージャ・ソフトウェアの利用により、インテル® PXA27x プロセッサを搭載したスマートフォンや PDA 上で消費電力を管理したり、システム・スタンバイ時間と通話時間を最適化することができます。インテル® PXA27x プロセッサには、複数の動作モードと低電力モードのほか、インテル® Dynamic Voltage Management（インテル® DVM）機能やインテル® Dynamic Frequency Management（インテル® DFM）機能が搭載されています。パワー・マネージャ・ソフトウェアは、こうしたモードや機能の利用を最適化し、さまざまなワークロードやアイドル状態の下で電力（電圧）とパフォーマンス（周波数）の動的なスケーリングを行います。このソフトウェアは、インテル® PXA27x プロセッサのボード・サポート・パッケージ（BSP）に不可欠な要素です。

パワー・マネージャ・ソフトウェアは、オペレーティング・システム（OS）によって提供されるネイティブのパワー・マネジメント機能/サービスを使用します。補完的な API（Application Programming Interface）、DPI（Device Programming Interface）、サービス、ポリシー、インフラストラクチャを提供することでこれらの機能やサービスを拡張し、高度な電力管理によって省電力の向上を図ります。OS ベンダ（OSV）は、パワー・マネージャ・ソフトウェアを使用するに当たって、自社の OS を修正する必要はありません。

パワー・マネージャ・ソフトウェアは、デバイスドライバに DPI を、アプリケーションには API を提供します。パワー・マネージャ・ソフトウェアを特定のプラットフォームに適応させるには、このソフトウェアのプラットフォーム固有のレイヤが使用されます。

プラットフォームを設計する際は、パワー・マネージャ・ソフトウェアの要件を満たす必要があります。デバイスが DPI 経由で電力ポリシーの変更や電力ステートに関する通知を受け取れるように、各デバイスドライバをパワー・マネージャ・ソフトウェアのクライアントとして設定しなければなりません。オプションとして、アプリケーション上で API を使用すると、省電力のさらなる向上が可能です。

2.0 利用モード

パワー・マネージャ・ソフトウェアには、以下の利用モードがあります。

- スタンバイ・モード（スタンバイ時間）
- 音声通信（通話時間）
- データ通信
- マルチメディア（オーディオ、ビデオ、カメラ）
- マルチメディアおよびデータ通信（ビデオ会議）

パワー・マネージャ・ソフトウェアは各利用モードにおいて以下を提供します。

- 電力とパフォーマンスの動的なスケーリングに関する最適な電力ポリシー
- 最適な動作周波数および電圧
- すべてのデバイスを含むシステム全体での低電力モードの利用
- デバイスにおけるステートの移行とパワー・マネジメント

一般的なシステムの動作プロファイルは、図 1 に示すように、実行/アイドルのデューティ・サイクルによって表されます。

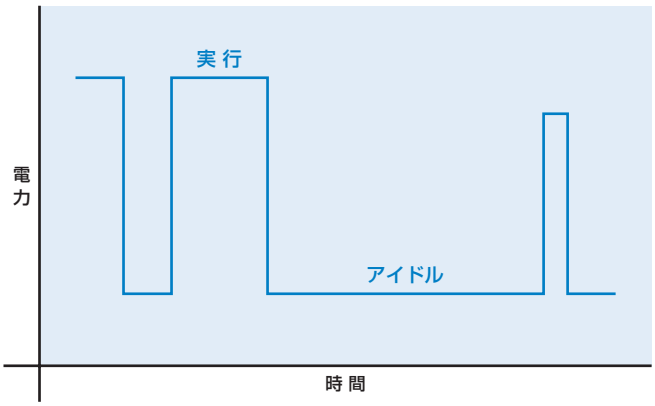


図 1. パワー・マネージャ・ソフトウェアを使用しない場合のシステム・プロファイル

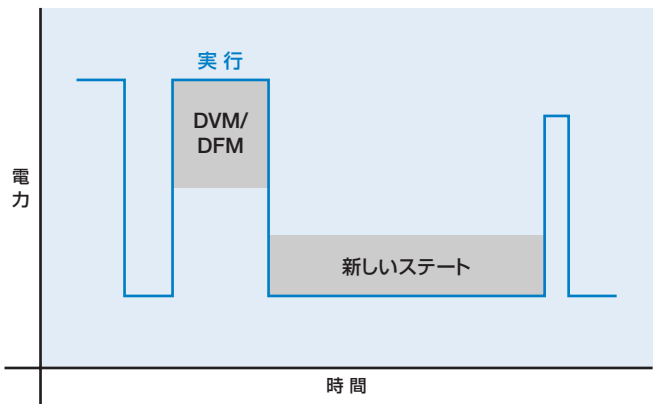


図 2. パワー・マネージャ・ソフトウェアを使用した場合のシステム・プロファイル

パワー・マネージャ・ソフトウェアは、前ページの図 2 に示すように、一般的なシステムの動作プロファイルを変更します。インテル® DVM とインテル® DFM は、「実行」時の周波数と電圧の動的なスケーリングに使用され、最低限の消費電力で直接的なパフォーマンス要件を満たします。新しい電力ステートは、「アイドル」時の消費電力を最小限に抑えるために使用されます。

3.0 アーキテクチャ

パワー・マネージャ・ソフトウェアのアーキテクチャは、図 3 に示すように、5 つのソフトウェア・コンポーネントで構成されています。

- **ポリシー・マネージャ：**ポリシー・マネージャは、システムの電力ポリシーを決定します。周波数および電圧の動的なスケーリングを利用し、あらゆるワークロードにおいて消費電力を最小限に抑えます。ワークロードは、CPU バウンド、メモリバウンド、CPU バウンドとメモリバウンドの両方のいずれでも問題ありません。ポリシー・マネージャは、アイドル・プロファイラ、パフォーマンス・プロファイラ、ユーザ設定、DPI、インテル® API（オプション）から提供される情報を評価します。また、最低限の消費電力に適した周波数と電圧でインテル® PXA27x プロセッサを実行するための電力ステートを定義します。電力ステートが OS にとってネイティブである場合、ポリシー・マネージャはネイティブの電力ステートと OS のインターフェイスを DPI に利用して、システムを指定の電力ステートに移行させます。電力ステートが OS にとってネイティブでない場合は、電力ステートを生成し、DPI を利用して、システムを指定の電力ステートに移行させます。
- **アイドル・プロファイラ：**アイドル・プロファイラは、ワークロードに基づいて、OS のアイドルスレッドで利用可能なパラメータをモニタします。このステータス情報は、ポリシー・マネージャに提供されます。
- **パフォーマンス・プロファイラ：**パフォーマンス・プロファイラは、インテル® PXA27x プロセッサ内の性能モニタリング・ユニット（PMU）を利用して、ワークロードが CPU バウンド、メモリバウンド、または CPU バウンドとメモリバウンドの両方のいずれであるかを判断します。このステータス情報は、ポリシー・マネージャに提供されます。
- **ユーザ設定：**ユーザ設定を利用すると、ユーザは、ポリシー・マネージャによって使用されるパラメータを指定して、システムの電力ポリシーを決定することができます。
- **OS マッピング：**OS マッピングを利用することにより、パワー・マネージャ・ソフトウェアを複数の OS に移植できます。

3.1 DPI

各デバイスドライバは、それぞれの DPI を介して、パワー・マネージャ・ソフトウェアへの登録を行います。ポリシー・マネージャは、最低限の消費電力でシステムのパフォーマンス要件を満たす、最適な電力ステートを決定します。電力ステートが OS にとってネイティブである場合、パワー・マネージャ・ソフトウェアは OS のインターフェイスを介して、デバイス内で必要な電力ステートの移行をデバイスドライバに通知します。電力ステートが OS にとってネイティブでない場合は、DPI を介して、デバイス内で必要な電力ステートの移行をデバイスドライバに通知します。デバイスドライバへの電力ステートの通知は、次のようにシンプルです。まず、デバイス #1 へのクロックを有効化します。次に、デバイス #1 の有効化とデバイス #2 の無効化のいずれか、または両方を行ってから、デバイス #2 へのクロックを無効化します。デバイスドライバは、デバイスを指定の電力ステートに移行させた後、次のイベントに備えさせる必要があります。

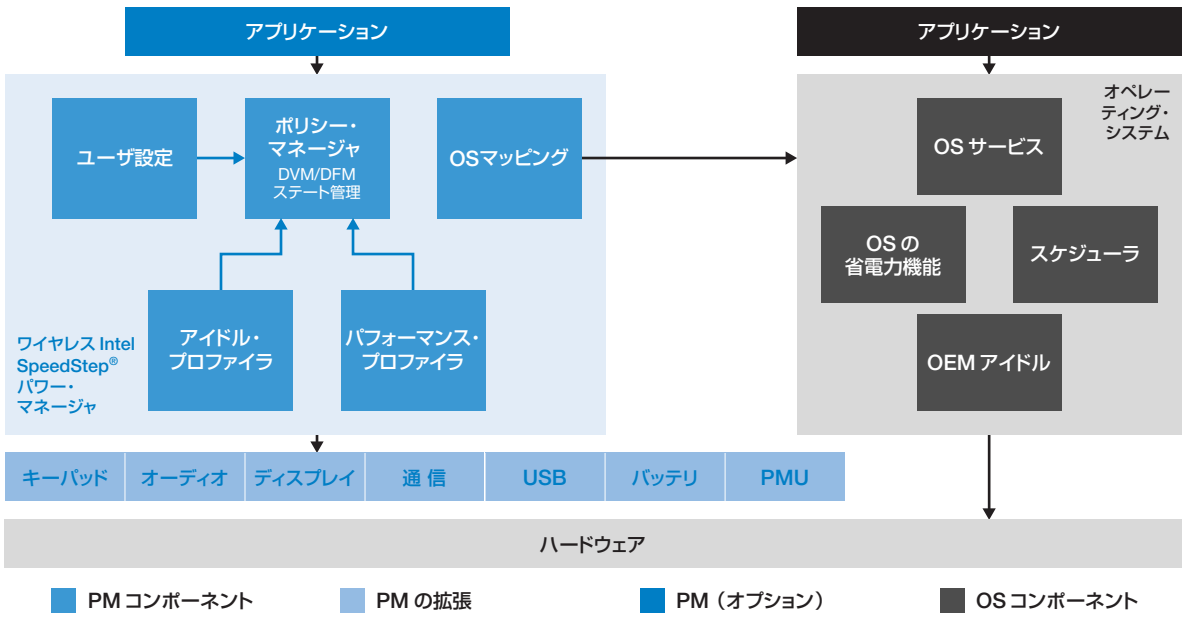


図 3. パワー・マネージャ・ソフトウェアのアーキテクチャ

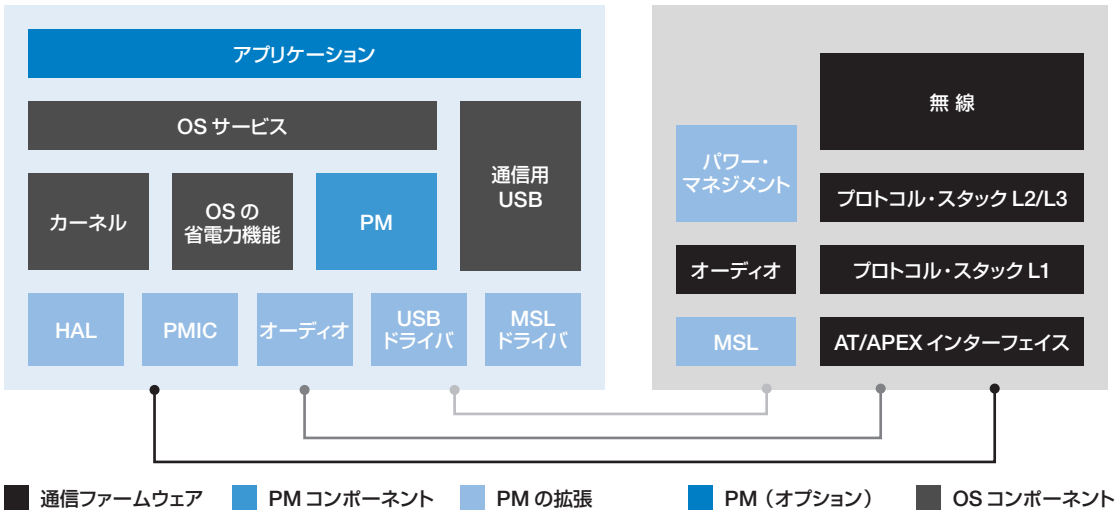


図 4. 通信サブシステム

またポリシー・マネージャは、DPI を介して新しい周波数と新しい電圧（最低限の消費電力でシステムのパフォーマンス要件を満たすもの）をインテル® PXA27x プロセッサの Power I²C（PWR_I²C）デバイスドライバに通知します。インテル® PXA27x プロセッサ内の PWR_I²C ユニットは、インテル® DVM を利用して、指定の電圧への物理的な変更を制御します。周波数の制御は、インテル® DFM をインテル® DVM と組み合わせることによって可能になります。各デバイスドライバは、ポリシー・マネージャによる電力ステートの変更を柔軟に要求できます。例えば、バッテリドライバは、バッテリのしきい値をモニタしてから、DPI を介して、必要な電力ステートの変更をポリシー・マネージャに通知することができます。各デバイスドライバは、パワー・マネージャ・ソフトウェアへ互換性を確保して設計しなければなりません。

3.2 API

パフォーマンスへの依存性があるアプリケーションは、最低限の消費電力で直接的なパフォーマンス要件を満たす周波数と電圧を迅速に決定するよう、API を介してポリシー・マネージャに要求することができます。ポリシー・マネージャは、DPI を介して、必要な周波数と電圧を PWR_I²C デバイスドライバに通知します。インテル® PXA27x プロセッサ内の PWR_I²C ユニットは、インテル® DVM を利用して、必要な電圧への物理的な変更を制御します。インテル® DFM をインテル® DVM と組み合わせることにより、周波数の制御が可能になります。パワー・マネージャ・ソフトウェアの使用により、オプションで拡張されたアプリケーションによる省電力は、アイドル・プロファイラ、パフォーマンス・プロファイラ、DPI による省電力を増強します。

3.3 セルラー・プロセッサ・インターフェイス

図 4 の右側に示された通信ソフトウェアは、インテルのセルラー・プロセッサ上で動作します。通信ソフトウェアのパワー・マネジメント・コンポーネントは、通信サブシステムの電力を管理して、独自のステートマシンを維持します。このパワー・マネジメント・コンポーネントは、通信プロトコル・スタックの L1/L2/L3 レイヤへのインターフェイスとして機能し、個々の実行/低電力モードのデューティ・サイクルを GSM および GPRS における各ステートに関連付けています。

アプリケーション・サブシステムと通信サブシステムは、物理的なリンクによって接続されています。インテル® モバイル・スケーラブル・リンク（インテル® MSL）は、アプリケーション・サブシステム内のインテル® PXA27x プロセッサと通信サブシステム内のインテル通信プロセッサを接続することができます。その他の物理的なリンクは、UART またはシンクロナス・シリアル・ポートを使用する場合があります。

アプリケーション・サブシステム上で動作する通信デバイスドライバは、パワー・マネージャ・ソフトウェアのクライアントであると同時に、OS のネイティブ・パワー・マネジメント・コンポーネントのクライアントでもあります。この通信デバイスドライバは、パワー・マネージャ・ソフトウェアと、OS のパワー・マネジメント・コンポーネントから、必要な電力ステートの移行に関する通知を受け取ります。例えば、OS がスタンバイ電力ステートに移行する際は、以下のステップが実行されます。

1. パワー・マネージャ・ソフトウェアが、通信サブシステムをスタンバイ電力ステートに移行させる必要があることを通信ドライバに通知します。
2. 通信ドライバは、インテル® MSL を介して、「スタンバイ電力ステートへの変更」というメッセージを通信サブシステムの通信パワー・マネジメント・コンポーネントに送信します。
3. 通信サブシステムは、スタンバイ電力ステートに移行します。さらに、アプリケーション・サブシステムでの処理を要する新しいステートに移行した場合にアプリケーション・サブシステムをウェイクアップできるよう、準備しておきます。

電力とパフォーマンスの動的なスケーリングの場合も同様です。通信デバイスドライバは、周波数と電圧の変更に関する通知を受けると、インテル® MSL を介して、その内容を通信ソフトウェアに通知します。

4.0 インテル® PXA27x—低消費電力機能の拡張

インテル® PXA27x プロセッサは、以下の機能を提供することによってワイヤレス Intel SpeedStep® テクノロジをハードウェアに実装します。

4.1 リセットリソース

- パワーオン・リセット
- ハードウェア・リセット
- ウォッチドッグ・タイマ・リセット
- GPIO リセット
- スリープモード終了時のリセット
- ディープ・スリープ・モード終了時のリセット

4.2 周辺回路に対するクロック・ゲーティング

インテル® PXA27x プロセッサでは、それぞれの周辺回路を独立して有効化/無効化できるだけでなく、各周辺回路へのクロックについてもゲートのオン/オフを独立して切り替えられます。

4.3 電力モード

- **ノーマルモード**：すべての内部電力領域と外部電源が、十分な電力供給を受けて機能している状態にあります。プロセッサ・クロックが動作中です。
- **アイドルモード**：CPU へのクロックが無効であり、リカバリは割り込みアサーションによって行われます。
- **ディープ・アイドル・モード**：コア周波数が 13MHz に変更された後のみ、このモードに移行することができます。CPU へのクロックが無効であり、リカバリは割り込みアサーションによって行われます。
- **スタンバイ・モード**：VCC_RTC と VCC_OSC を除くすべての内部電力領域は、低電力モードに移行します。ステートは維持されますが、アクティビティは許可されません。クロックソースは無効になる可能性があります。内部電力領域の一部をオフにすることが可能で、両方の PLL が無効になります。リカバリは、外部および選択された内部のウェイクアップ・イベントによって行われます。
- **スリープモード**：VCC_RTC と VCC_OSC（いずれも内部供給）を除くすべての内部電力領域をオフにできます。クロックソースは、リアルタイム・クロック（RTC）とパワー・マネージャ・ユニットによって使用されるものを除いて、すべて無効です。インテル® PXA27x プロセッサの PWR_EN 出力ピンは、アサートを解除してインテル® PXA27x プロセッサの低電圧領域への外部低電圧電源をオプションで無効にします。残りの電力領域は、低電力ステートに移行します。ステートは維持されますが、アクティビティは許可されません。リカバリは、外部および選択された内部のウェイクアップ・イベントによって行われます。プログラム・カウンタが無効であるため、リカバリにはシステムの再起動が必要です（プログラム・カウンタが 0x0 から再開するため、コアはリセットベクトルで始まる実行を開始します）。
- **ディープ・スリープ・モード**：VCC_RTC と VCC_OSC を除くすべての内部電力領域をオフにできます。クロックソースは、RTC とパワー・マネージャ・ユニットによって使用されるものを除いて、すべて無効です。インテル® PXA27x プロセッサの PWR_EN 出力ピンは Low になり、インテル® PXA27x プロセッサの低電圧領域への外部低電圧電源をオプションで無効にします。また、インテル® PXA27x プロセッサの SYS_EN 出力ピンも同じく Low になり、インテル® PXA27x プロセッサの高電圧領域への外部高電圧電源をオプションで無効にします。すべての電力領域は、バックアップ・バッテリ・ピン VCC_BATT から直接、電力を供給されます。残りの電力領域は、低電力ステートに移行します。ステートは維持されますが、アクティビティは許可されません。リカバリは、外部および選択された内部のウェイクアップ・イベントによって行われます。プログラム・カウンタが無効であるため、リカバリにはシステムの再起動が必要です（プログラム・カウンタが 0x0 から再開するため、コアはリセットベクトルからの実行を開始します）。

インテル® PXA27x プロセッサは、37 の GPIO 入力を備えています。これをウェイクアップ・イベントとして設定した場合、インテル® PXA27x プロセッサの低電力モードを終了させることができます（上記のモードは、ノーマルモードを除いてすべて低電力モードです）。例えば、インテル® PXA27x プロセッサには、キーパッド上でキーが押されたときにウェイクアップ・イベントとして機能する GPIO が複数存在します。このウェイクアップ・イベントは、低電力のスタンバイ・モードまたはスリープモードからインテル® PXA27x プロセッサをウェイクアップすることができます。ウェイクアップ・イベントは、以下によって発生します。

- キープレス
- 音声通話受信
- GPRS データ受信
- SMS メッセージ受信
- ヘッドセット取り付け
- USB ケーブル取り付け/取り外し
- MMC 取り付け/取り外し
- RTC ウェイクアップ
- タッチスクリーン操作
- 折りたたみ式携帯電話のシェルオープン

4.4 プログラム可能な周波数変更管理（インテル® DFM）とプログラム可能な電圧変更管理（インテル® DVM）

4.4.1 DFM

インテル® PXA27x プロセッサのコアクロックと周辺クロックは、PLL から得られます。インテル® PXA27x プロセッサは、ソフトウェアによるコアクロックの動的な設定を可能にすることにより、インテル® DFM を実装しています。コアクロック周波数は、以下のようにいくつかの方法で変更できます。

- 13MHz クロックソースの選択
- コア PLL 周波数の変更
- ターボモードまたはハーフ・ターボ・モードの有効化/無効化

ソフトウェアは、インテル® PXA27x プロセッサの CCCR（Core Clock Configuration Register）をプログラムし、以下について選択します。

- **実行モードとオシレータとの比率、CCCR [L]**
  L ビットは、外部水晶オシレータの入力値に L を掛けることにより実行周波数を決定します。
- **ターボモードと実行モードとの比率、CCCR [2N]**
  N ビットは、実行周波数に 2N を掛けることによってターボ周波数を決定します。
- **メモリ・コントローラ・クロックの選択、CCCR [A]**
  A ビットをセットすると、メモリ・コントローラのクロック周波数はシステムバスのクロック周波数と同じになります。A ビットをクリアすると、メモリ・コントローラのクロック周波数は表 1 のようになります。

ソフトウェアは次に、コプロセッサ 14、レジスタ C6（CLKCFG）をプログラムし、以下について選択します。

- **CLKCFG [B] —高速バスモード。** B ビットをセットすると、システムバス周波数は CCCR で指定された実行モード周波数と同じになります。B ビットをクリアすると、システムバス周波数は CCCR で指定された実行モード周波数の半分になります。
- **CLKCFG [F] —コア周波数の変更。** F ビットをセットすると、コア PLL が中止され、新しい CCCR 設定で再開されます。
- **CLKCFG [T] —ターボモード。** T ビットをセットすると、CPU はターボ周波数で動作します。T ビットをクリアすると、CPU は実行モード周波数で動作します。
- **CLKCFG [HT] —ハーフ・ターボ・モード。** HT ビットをセットすると、T ビットがセットされていてもクリアされていても、CPU はターボ周波数の半分で動作します。HT ビットをクリアし、T ビットもクリアすると、CPU は実行モード周波数で動作します。HT ビットをクリアして、T ビットをセットすると、CPU はターボ周波数で動作します。

ソフトウェアが CLKCFG [F] をセットすると、自動周波数変更シーケンスが開始されます。このシーケンスは、CCCR 値と、CLKCFG で選択されたモードを確定します。

4.4.2 DVM

インテル® PXA27x プロセッサは、ボルテージ・マネージャを通じてインテル® DVM を実装しています。ボルテージ・マネージャは、外部 PMIC レギュレータとの通信専用の I^2C ユニット（PWR_I^2C）と、電圧変更シーケンサを利用して電圧管理を行います。

ソフトウェアが電圧変更モードを開始すると、電圧変更シーケンサは、コマンドを PWR_I^2C ユニット経由で外部 PMIC レギュレータに自動送信します。電圧変更シーケンサは、最大で 32 個のコマンドを送信できます。コマンドは、動的コマンドと静的コマンドに分類されます。

- **動的コマンド**は、コアの動作中に実行されます。
- **静的コマンド**は、プロセッサへのクロックが無効化された後に実行されます。

インテル® PXA27x プロセッサは、Power Manager General Configuration Register（PCFR）、Power Manager Voltage change Control Register（PVCR）、Power Manager I^2C Command Register File（PCMDx）を利用して、電圧変更シーケンスを定義および制御します。

- **周波数/電圧の変更、PCFR [FVC] —**FVC ビットをセットすると、周波数変更シーケンス（DFM）は電圧変更シーケンス（DVM）もトリガします。
- **読み出しポインタ、PVCR [RP] —**PVCR 中の読み出しポインタ・フィールドは、PWR_I²C 経由で外部レギュレータに送信されるコマンド（コマンドが複数の場合は最初のコマンド）が含まれた PCMD レジスタ・ロケーションを指しています。コマンド・シーケンスは、32 個の PCMD レジスタのいずれからでも開始できます。コマンドが送信されると、読み出しポインタはインクリメントし、次の PCMD レジスタ・ロケーションを指します。現在のコマンドが PCMD [LC] のセットによって指定された最終コマンドである場合、読み出しポインタはインクリメントしません。
- **遅延コマンド実行、PCMD [DCE] —**現在の PCMD 中の DCE ビットをセットすると、PVCR 中のコマンド遅延ビットによってセットされたカウンタは、13MHz プロセッサ・オシレータ・サイクル単位（サイクル数はプログラム可能）で待機してから、コマンドの実行を継続します。これは、コマンドの間隔を長くする必要がある場合に有効です。
- **マルチバイト・コマンド、PCMD [MBC] —**MBC ビットをセットすると、電圧変更シーケンサは、MBC ビットがクリアされたコマンドが実行されるまで、遅延やパワー・マネージャ・ユニットとのハンドシェイクなしでバイトを送信し続けます。
- **最終コマンド、PCMD [LC] —**LC ビットをクリアすると、電圧変更シーケンサは、次の大きなアドレスの PCMD レジスタに追加のコマンドが格納されていると予期します。PCMD31 中の LC ビットをクリアすると、PVCR 読み出しポインタは PCMD31 内のコマンドを実行してから、PCMD0 に戻ります。LC ビットをセットすると、電圧変更シーケンサは、現在のコマンドが最終コマンドであるとみなして、実行完了後に終了します。各電圧変更コマンド・シーケンスは、シーケンス中の最終コマンドの LC ビットをセットすることによって終了する必要があります。LC ビットがセットされている場合、PVCR 読み出しポインタはインクリメントしません。

4.4.3 電圧変更と周波数変更の組み合わせ

周波数の変更（クロックソースの変更またはコア PLL 周波数の変更）を利用すると、コア、システムバス、メモリ・コントローラ、LCD コントローラの周波数を、ターボモードまたは高速バスモードで利用不可能な値に変更できます。周波数の変更は、PCFR [FVC] をセットすることによって、電圧の変更と組み合わせられます。同様に、電圧の変更は、高速バスモードへの変更や高速バスモードからの変更と組み合わせられます。

4.5 プログラム可能な動作周波数

表 1 に、インテル® PXA27x プロセッサのプログラム可能な動作周波数を示します。

コア実行周波数	CLKCFG [T]	コアターボ周波数	CLKCFG [T]	CLKCFG [HT]	CCCR [L]	CCCR [2N]	システムバス	CLKCFG [B]	CLK_MEM（メモリ・コントローラ）	CCCR [A]	SDCLK<2:1> SDRAMクロック	MDREFR [KxDB2]††	シンクロナス・フラッシュ	MDREFR [K0DB4]	MDREFR [K0DB2]	LCD
91.0†	0	—	—	0	7	2	45.0	0	91.0	0	45.0	1	22.5	1	1	91.0
104.0	0	104.0	1	0	8	2	104.0	1	104.0	1	104.0	0	52.0	0	1	52.0
156.0	0	156.0	1	1	8	6	104.0	1	104.0	1	104.0	0	52.0	0	1	52.0
104.0	0	312.0	1	0	8	6	104.0	1	104.0	1	104.0	0	52.0	0	1	52.0
208.0	0	208.0	1	0	16	2	208.0	1	208.0	1	104.0	1	52.0	1	X	104.0
208.0	0	312.0	1	0	16	3	208.0	1	208.0	1	104.0	1	52.0	1	X	104.0
208.0	0	416.0	1	0	16	4	208.0	1	208.0	1	104.0	1	52.0	1	X	104.0
208.0	0	520.0	1	0	16	5	208.0	1	208.0	1	104.0	1	52.0	1	X	104.0
208.0	0	624.0	1	0	16	6	208.0	1	208.0	1	104.0	1	52.0	1	X	104.0

† L=7（コア = 91.0MHz）は必ず起動周波数のみに使用し、使用後はすぐにその他の周波数ポイントのいずれかに再設定する必要があります。
†† KxDB2 は、K1DB2 および K2DB2 を表しています。

表 1. クロック周波数

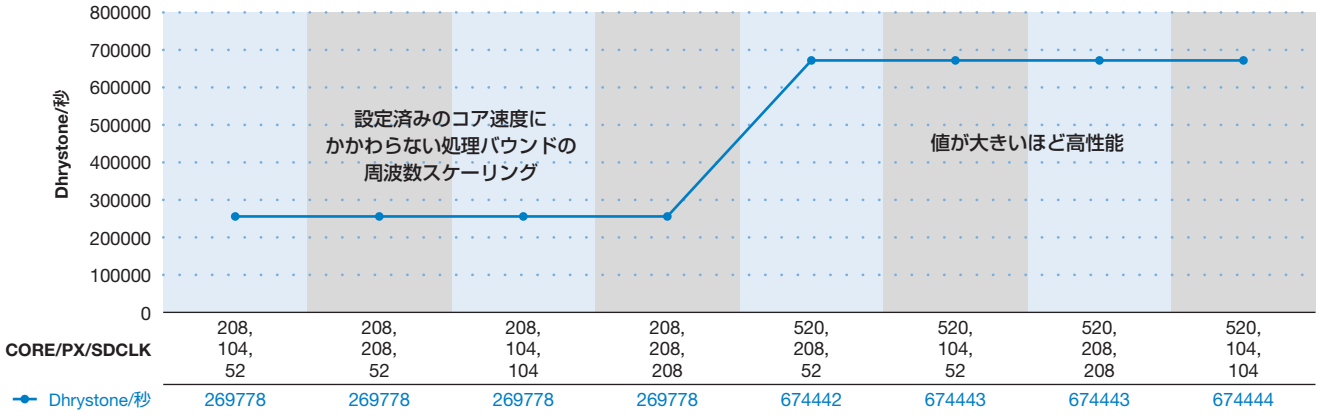


図 5. Dhrystone/秒とコア/PX/SDCLK の関係

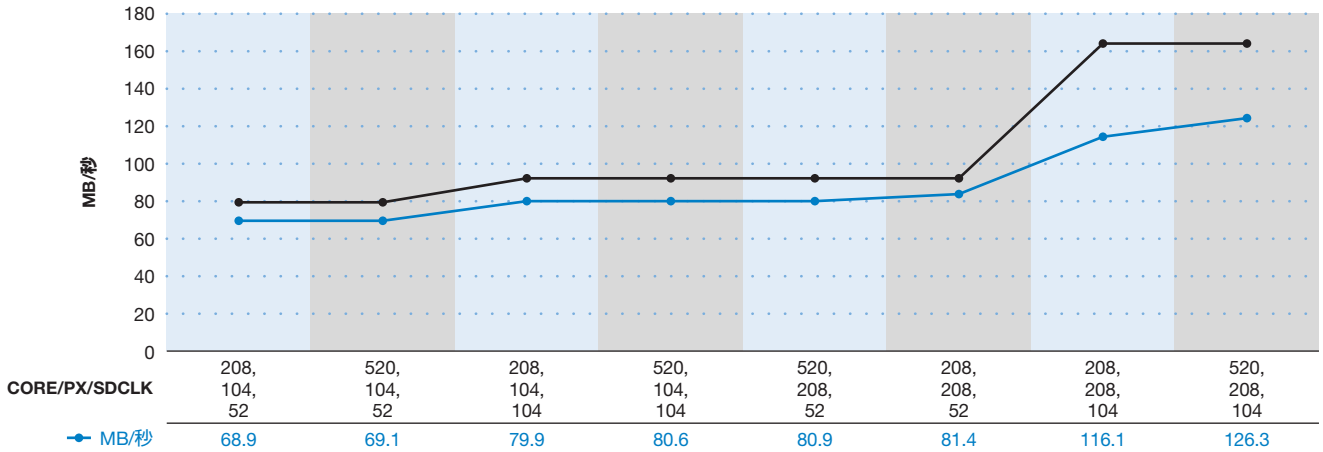


図 6. MB/秒とコア/PX/SDCLK の関係

5.0 インテル® DFM およびインテル® DVM のワークロード特性評価

ほとんどのソフトウェア・アプリケーションやワークロードは、3 つの主要カテゴリに分類できます。

- CPU（処理）バウンド・アプリケーション
- メモリ・バウンド・アプリケーション
- I/O バウンド・アプリケーション
- CPU/メモリ・バウンド・アプリケーション

5.1 CPU バウンド・アプリケーション

D キャッシュおよび I キャッシュにロードされたデータと命令を使って実行時間の大半を処理に費やすアプリケーションは、一般に、CPU バウンドであるとみなされます。基本的なプロセッサ・アーキテクチャに適した命令をスケジューリングすると（ストールの削減）、CPU バウンド・アプリケーションのパフォーマンスが向上する可能性があります。こうしたアプリケーションは、CPU を常時ビジーにする傾向があり、プロセッサのアイドル時間はわずかです。プロセッサ周波数（および電圧）が増加すれば、アプリケーションのパフォーマンスが向上します。例えば、Dhrystone は純粋に CPU バウンドなワークロードであり、図 5 に示すように、パフォーマンスは CPU（コア）周波数に比例しています。パフォーマンス要件を満たすには、アプリケーションを最大限の周波数で動作させることが欠かせません。この情報は、パフォーマンスを最適化するポリシー・マネージャにとって極めて重要です。

5.2 メモリ・バウンド・アプリケーション

キャッシュ・サイズよりも大きなデータブロックを扱うアプリケーションは通常、キャッシュ外にあるデータにアクセスしなければならず、メモリやシステムバスの速度に依存します。メモリコピーをはじめとするこうしたアプリケーションでは、大きなデータブロックが移動するため、大量のメモリ・トラフィックが発生し、CPU サイクルの大半がデータ待ちに費やされる傾向があります。このような場合、コアの速度が増しても、パフォーマンスはメモリ速度に比例しているため、パフォーマンスは向上しません（図 6 を参照）。この情報は、パフォーマンスを最適化するポリシー・マネージャにとって極めて重要です。

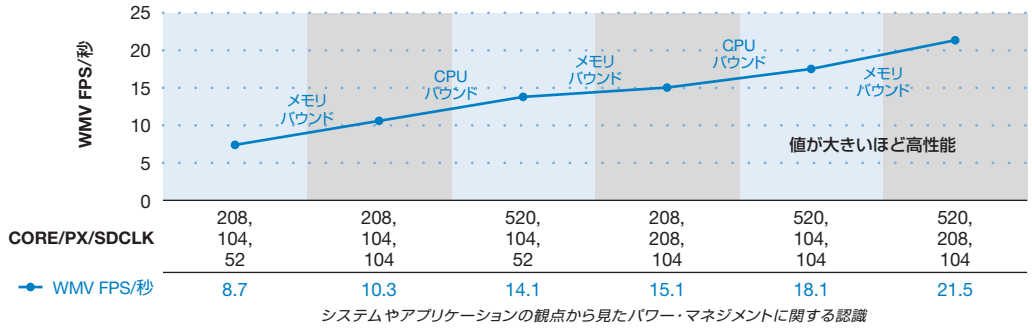


図 7. WMV FPS とコア/PX/SDCLK の関係

5.3 I/O バウンド・アプリケーション

何らかの I/O デバイス（周辺機器）上でデータを待機するアプリケーションは、I/O バウンドであるとみなされます。例としては、ネットワークからのデータを待機するイーサネット・ドライバがあります。

5.4 CPU/メモリ・バウンド・アプリケーション

多くのアプリケーションでは、要求されるパフォーマンスが時間とともに変化します。ある時点において、CPU（処理）バウンドのこともあれば、メモリバウンドのこともあります。大量の処理を行い、大きなデータブロックを扱うマルチメディア・アプリケーションは、このカテゴリに属します。こうしたマルチメディア・アプリケーションの特性は、パフォーマンスがメモリと CPU の速度のどちらにも依存していることを示しています。このようなワークロードの場合、特性を正確に予測・判断すると、電力ポリシーの向上につながります。例えば、ビデオプレーヤは、CPU/メモリバウンドのアプリケーション・タイプです。図 7 に示すように、パフォーマンスはコアおよびメモリの周波数に比例しています。

6.0 アイドル・プロファイラ

アイドル・プロファイラは、CPU の使用情報と OS のアイドル情報をパワー・マネージャ・ソフトウェアに提供します。図 8 に、アイドル・プロファイラへの入力を行う OS のアイドルスレッドを示します。アイドルスレッドは、OS がビジーでないとき（コードを実行していないとき）に実行されるため、CPU の使用情報を提供するのに適した選択肢の 1 つです。ただし、アイドルスレッドは実行準備の整ったタスクがない場合のみ実行されるため、情報が提供されるのは CPU の使用時間が 100% 未満の場合に限られます。CPU の使用時間が 100% 未満であるが極めて長い場合は、ISR を利用して CPU の使用情報をアイドル・プロファイラに提供できます。

7.0 パフォーマンス・プロファイラ

システム・ワークロードは、どのようなときでも静的な状態で維持されることはありません。そのため、最低限の消費電力でシステム・パフォーマンスを最適化するには、動的なワークロード特性評価が欠かせません。パフォーマンス・プロファイラは、システム情報をモニタし、システムステートを管理します。ポリシー・マネージャはいつでも、現在のシステムステートを返すように、パフォーマンス・プロファイラに指示することができます。パフォーマンス・プロファイラはイベント駆動型であるため、システムステートに変更があると、ポリシー・マネージャに自動的に通知されます。

CPU バウンド、メモリバウンド、CPU およびメモリバウンドのワークロードの動的な特性評価に基づいて電力とパフォーマンスの動的なスケーリングを行うに当たり、パフォーマンス・プロファイラは、次ページの図 9 に示すように、インテル® PXA27x プロセッサの PMU をモニタします。

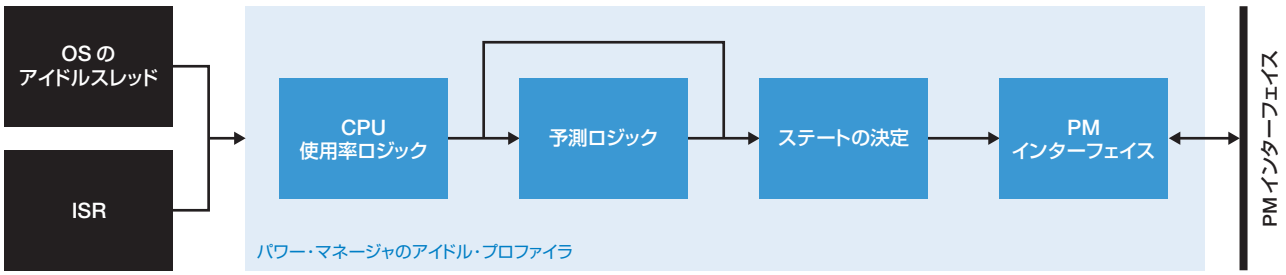


図 8. パワー・マネージャ・ソフトウェアのアイドル・プロファイラ

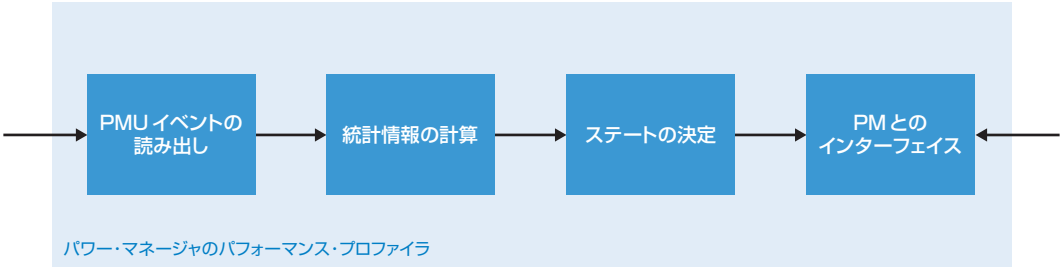


図 9. パフォーマンス・プロファイラ

PMU は、表 2 に示されたプロセッサ・イベントを追跡します。パフォーマンス・プロファイラは、インテル® PXA27x プロセッサの LCD コントローラをはじめとする周辺機器のステータス・レジスタもモニタします。このモニタリングによって、パフォーマンス・プロファイラはシステムステートを動的に特性評価できるようになります。

8.0 ポリシー・マネージャ

ポリシー・マネージャは、電力ステートを定義した上で、インテル® PXA27x プロセッサやワイヤレス・インテル® 通信プロセッサの電力モードに関連付けます。電力ステートが OS にとってネイティブである場合、ポリシー・マネージャはネイティブの電力ステートと OS のインターフェイスを DPI に利用して、システムを指定の電力ステートに移行させます。電力ステートが OS にとってネイティブでない場合は、電力ステートを生成し、DPI を利用して、システムを指定の電力ステートに移行させます。ポリシー・マネージャは、デバイスドライバの入力値、アプリケーション・ワークロードの入力値（オプション）、アイドル・プロファイラの入力値、パフォーマンス・プロファイラ（システム・ワークロード）の入力値に基づいて、システムの電力ポリシーを定義します。また、OEM は、これらの入力値に基づいて独自の電力ポリシーを定義することもできます。

パワー・マネージャ・ソフトウェアは、ステートの変更、周波数の変更、電圧の変更のための基盤をデバイスドライバに提供します。パワー・マネージャ・ソフトウェアはこの基盤を利用し、直接または OS のパワー・マネジメント・サービスを介して、変更をクライアント・ドライバに通知します。

パワー・マネージャ・ソフトウェアによって有効化されたシステムは常に、グローバル変数の一部として事前定義された動作ポイント、つまり IPMOperatingPoint =[State, Voltage, Frequency, Frequency2, Frequency3] を維持します。

情報源	イベントの説明	特性評価の用途
PMU レジスタ	命令キャッシュ・ミス 命令キャッシュの命令伝達が不能	命令のトラフィックおよび CPI に関する予測と、キャッシュの局所性
	データ依存性のストール	メモリ使用率
	命令 TLB ミス データ TLB ミス	ページの局所性
	命令の実行	CPU 使用率
	ストール数: D キャッシュ・バッファが満杯のためにストールが発生した回数（すべてのサイクル条件が存在）	データ・トラフィック/データアクセスの輻輳/輻輳率および輻輳の長さ
	データ・キャッシュ・アクセス	データ・アクセス・レート、メモリバウンド、メモリ使用率
	データ・キャッシュ・ミス	データ・トラフィックおよびメモリ使用率
	データ・キャッシュ・ライトバック	データ・トラフィック
	ソフトウェアが PC を変更	関数呼び出しの回数

表 2. インテル® PXA27x プロセッサの PMU

各項目の意味は以下のとおりです。

- State ＝プロセッサのステート
- Voltage ＝プロセッサのコア電圧
- Frequency1 ＝プロセッサのコア周波数
- Frequency2 ＝システムバスの周波数
- Frequency3 ＝ SDCLK の周波数

例えば、インテル® PXA27x プロセッサのステートには、実行、スリープ、スタンバイ、アイドル、ディープ・アイドルがあります。

同様に、電圧と周波数には、インテル® PXA27x プロセッサによってサポートされた値をいずれも適用できます。ポリシー・マネージャの出力はいつでも、IPMOperatingPoint を最適な値に設定し、必要なパフォーマンスを最低限の消費電力で達成します。

9.0 デバイス・ドライバ・インターフェイス

9.1 概要

パワー・マネージャ・ソフトウェアのデバイス・インターフェイス・レイヤは、図 10 に示すように、デバイスドライバや、パワー・マネージャ・ソフトウェアのポリシー・マネージャとやりとりします。

多くの OS はある程度のパワー・マネジメント・インフラストラクチャをネイティブで備えているため、パワー・マネージャ・ソフトウェアは、可能であるときはいつでも、このネイティブのインフラストラクチャへのインターフェイスとして機能します。OS がネイティブのパワー・マネジメント・インフラストラクチャを備えていない場合は、パワー・マネージャ・ソフトウェアがマネジメント・インフラストラクチャを追加します。パワー・マネージャ・ソフトウェアは、各 OS のドライバ・インターフェイス・レイヤを移植して、ポリシー・マネージャをドライバとの直接的な相互作用から切り離せる抽象レベルを提供します。

以下のセクションでは、ドライバの相互作用に関する一般的なアーキテクチャについて説明します。実装の詳細は OS によって異なるため、各 OS のドキュメントを参照してください。

9.2 登録

パワー・マネージャ・ソフトウェアは、電力パラメータに変更があったときにドライバに通知する機能を備えています。この通知により、ドライバはペンディングとなっている移行の準備をしたり、必要に応じて移行を拒否することができます。ドライバは登録中、コールバック関数を既存のインフラストラクチャに提供する必要があります（そのようなインフラストラクチャが存在しない場合はパワー・マネージャ・ソフトウェアに直接提供します）。また、パワー・マネージャ・ソフトウェアが該当する場合のみペンディング中のステート移行をドライバに通知するように、ドライバは、周波数とステートに対するターゲット・デバイスの依存性をパワー・マネージャ・ソフトウェアに通知しておく必要があります。

9.3 デバイス・ドライバ・インターフェイス・レイヤ

フローの暗黙の方向に基づくと、ドライバ・インターフェイス・タイプには 2 つのカテゴリがあります。

- **パワー・マネージャ・ソフトウェアからドライバへのインターフェイス：**このインターフェイスでは、入力値を登録済みドライバに提供する一連のルーチンを定義します。パワー・マネージャ・ソフトウェアは、これらのルーチンを利用して何らかの方法（スタンバイ・モードやアイドルモードへの移行など）でシステム電力ステートを変更する意思を登録済みドライバに通知できます。電力パラメータの移行というパワー・マネージャ・ソフトウェアの要求が不適切な結果につながる場合、ドライバは、その要求を拒否できる必要があります。
- **ドライバからパワー・マネージャ・ソフトウェアへのインターフェイス：**このインターフェイスでは、ターゲット・デバイスのロードおよび電力ステートに関する個別の情報をパワー・マネージャ・ソフトウェアに提供するために登録済みドライバが利用できる一連のルーチンを定義します。パワー・マネージャ・ソフトウェアは、この情報を利用して十分な情報に基づいた最適なシステム電力ステートを決定できます。登録済みドライバは、スタブをこれらのルーチンに提供するだけの場合もあります。

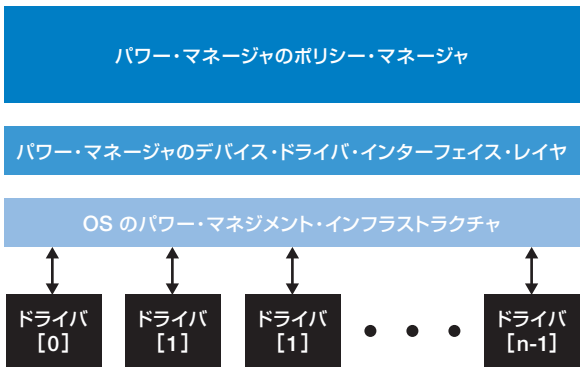


図 10. パワー・マネージャ・ソフトウェアのデバイス・ドライバ・インターフェイス

すべてのドライバが独自のデバイス電力ステート管理を行うことが重要です。ドライバは少なくとも、各自のアクティビティをモニタし、各自のステートを自己管理できる必要があります。例えば、必要時に周辺機器をオフにしたり、移行をパワー・マネージャ・ソフトウェアに通知できることが必須です。パワー・マネージャ・ソフトウェアは、各ドライバのステートを追跡しますが、デバイスのステート管理を強制することはありません。

9.3.1 デバイスの電力ステート

現在の OS の多くはある程度のデバイス・パワー・マネジメントを備えているため、デバイス・ドライバ・インターフェイス・レイヤは、OS 固有のデバイスステートをパワー・マネージャ・ソフトウェア固有の適切なステートに変換します。OS がパワー・マネジメント・デバイス・ステートを備えていない場合、デバイスドライバはパワー・マネージャ・ソフトウェアによって生成された電力ステートを利用します。登録済みドライバは、デバイスの電力ステートを移行する際、移行をパワー・マネージャ・ソフトウェアに通知する必要があります。表 3 に、パワー・マネージャ・ソフトウェアのデバイスステートを示します。

ステート	説 明
オン	デバイスは、十分な電力供給を受けて、完全に機能できる状態にあります。
オフ	デバイスはオフになっています。
低電力	デバイスはオンですが、低電力ステートにあります。デバイスによっては、こうした低電力ステートが複数存在します。

表 3. デバイスステート

9.3.2 デバイスステートの依存性

ドライバの登録の一環として、ドライバは、サポートされている全プロセッサ・モードに対してのデバイスの依存性をパワー・マネージャ・ソフトウェアに伝達する必要があります。特定のステートに依存していることをドライバが伝達すると、パワー・マネージャ・ソフトウェアは、ステート移行をドライバに予告してから、ステート移行を実行します。

9.3.3 デバイス周波数の依存性

ドライバ登録の一環として、ドライバは、ポリシー・マネージャによるデバイス周波数やデバイス電圧をスケーリングする際に通知が必要かどうかを、パワー・マネージャ・ソフトウェアに伝える必要があります。例えば、表 1 では、ポリシー・マネージャがインテル® PXA27x プロセッサのコア実行周波数を 208MHz から 104MHz にスケーリングする際、インテル® PXA27x プロセッサのその他の周波数（システムバス、メモリ・コントローラ、LCD コントローラなど）もスケーリングが必要であることを示しています。

10.0 パワー・マネージャ・ソフトウェア認識アプリケーション・インターフェイス（オプション）

10.1 概要

アプリケーションを拡張して、パフォーマンスや処理の要件（サイクル、期限など）をパワー・マネージャ・ソフトウェアに容易に提供することができます。その結果、周波数の変更と電圧の変更のスケジューリングが、よりシンプルで効果的なものになります。例えば、インテル® PXA27x プロセッサを利用して小型のディスプレイを制御するモバイル・プラットフォームについて検討します。MPEG4 ビデオ・アプリケーションが 30fps をデコードして表示する際、インテル® PXA27x プロセッサは、最大限可能な周波数と電圧で動作できます。その場合、図 2 に示すように、アプリケーションへのサービス提供に必要な実行バーストの間、多くのアイドル時間が存在します。アイドル・プロファイラのみの場合、このアイドル時間を検出できないか（極めて短時間の場合）、アイドル時間の検出でレイテンシを発生させることになります。オプションにより、アイドル時間情報をパワー・マネージャに通知するようアプリケーションを拡張すれば、ポリシー・マネージャは、アイドル時間中に消費電力を削減する上でより適切な決定が下せるようになります。

10.2 API

図 11 に、API を示します。

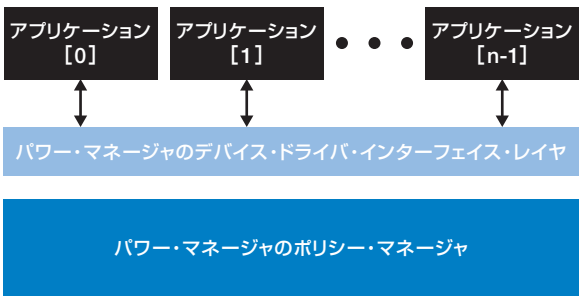


図 11. API

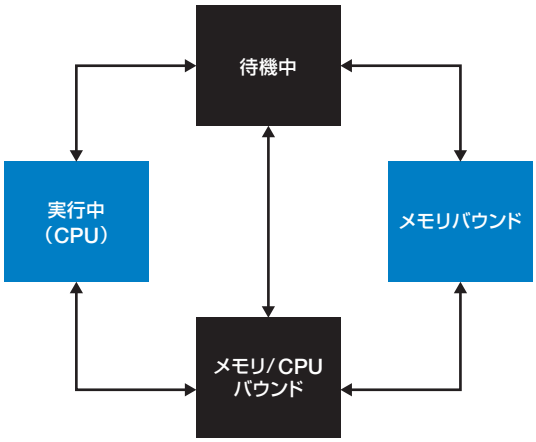


図 12. アプリケーションの電力ステート

API は、ポリシー・マネージャと各パワー・マネージャ・ソフトウェア認識アプリケーションとの通信を可能にするよう設計されています。アプリケーションは、パワー・マネージャ・ソフトウェアに登録され、電力ステート情報をパワー・マネージャ・ソフトウェアに提供する必要があります。

10.3 アプリケーションの電力ステート

図 12 に示すように、各アプリケーションは常時、4 つの電力ステートのいずれかに属している必要があります。

- **実行中：**このステートは、通常の電力ステートです。アプリケーションが命令を実行中であり、必要なデータをすべて利用できます。つまり、アプリケーションはデータストールを予期していません。この CPU バウンドの電力ステートの例としては、Dhrystone アプリケーションがあります。
- **待機中：**アプリケーションが周辺機器にポーリングを行って、応答を待機しています。またはアイドル中です。この I/O バウンドの電力ステートの例としては、ネットワーク・データを待機中のアプリケーションがあります。
- **メモリバウンド：**アプリケーションが大きなデータブロックを移動しています。このメモリバウンドの電力ステートの例としては、メモリコピー（MEMCPY）操作があります。
- **メモリ/CPU バウンド：**アプリケーションがメモリ・データ・ブロックを使用する複雑なアルゴリズムを実行しています。このメモリおよび CPU バウンドの電力ステートの例としては、ビデオゲームがあります。

11.0 インテル® PXA27x プロセッサ向けのパワー・マネージャ・ソフトウェアの例

```
typedef enum {
IPMErr = 0,
IPMNoErr =1
} IPMStatus;
typedef enum {
Run,
M13,
Standby,
Sleep
} CPUState;
typedef struct _IPMState {
CPUState CPUState;// Current Processor Mode
UINT32 CPUVoltage;// Current Core Voltage * 100
UINT32 CPUFrequency;// Current Core Frequency * 100
UINT32 PXFrequency;// Current Bus Frequency * 100
UINT32 SDClk0Frequency;// Current SDCLK0 Frequency * 100
UINT32 SDClk1Frequency;// Current SDCLK1 Frequency * 100
UINT32 SDClk2Frequency;// Current SDCLK2 Frequency * 100
} IPMState;
typedef enum {
Core,
MemClk,
LcdClk
} ClksEnum;
```

```
const UINT32 FreqArray[][3] = {// Core, Mem, LCD
62400, 20800, 10400,
52000, 20800, 10400,
41600, 20800, 10400,
31200, 20800, 10400,
20800, 20800, 10400,
31200, 10400, 5200,
15600, 10400, 5200,
10400, 10400, 5200
};
typedef struct _FreqStruct {
UINT32 Core;
UINT32 Mem;
UINT32 Lcd;
} FreqStruct;
typedef FreqStruct FreqArray [sizeof(BvdFreqArray)/
sizeof(UINT32)/(LcdClk+1)];
typedef enum {
Off,
On,
LowPower
} DevStatesEnum;
typedef struct _IPMDeviceInfo {
char DeviceId;// Device Identifier
DevStatesEnumDeviceState;// Current State of Device
} IPMDeviceInfo;
```

```
typedef struct _IPMDriverInfo {
USHORT DriverHandle;// Id for a registered driver
IPMDeviceInfo Devices[8];// Max of 8 devices
char NumDevices;// indicates #devices managed
char *CallBack; // callback function
DriverFreqSensitivity FreqSensitivity; // 32-bit
bitfield
DriverStateSensitvity StateSensitvity; // 32-bit
bitfield
} IPMDriverInfo;
typedef UINT32 DriverFreqSensitivity; // bit field based
on
FreqArray
typedef UINT32 DriverStateSensitvity; // bit field based
on
CPUState
```

12.0 まとめ

インテル® PXA27x プロセッサ・ファミリで初めて採用されたワイヤレス Intel SpeedStep® テクノロジは、CPU の負荷に応じてプロセッサの電力とパフォーマンスを動的に調整する機能を提供します。これにより、モバイル機器の消費電力を大幅に削減し、スタンバイ時間と通話時間を延長することができます。ワイヤレス Intel SpeedStep® テクノロジは、Intel XScale® マイクロアーキテクチャに組み込まれていた機能を強化したものであり、3 つの低電力ステートを新たに追加し、先進的なワイヤレス Intel SpeedStep® パワー・マネージャ・ソフトウェアを利用することにより、電力とパフォーマンスに対するエンドユーザのニーズに高度に対応します。このテクノロジは、電圧と周波数を即座に変更し、電力を節約する一方で、豊富なアプリケーションの実行に欠かせないパフォーマンスを提供できます。

詳細については、インテルの Web サイト http://www.intel.co.jp/jp/developer/を参照してください。

インテル株式会社
〒300-2635 茨城県つくば市東光台5-6
http://www.intel.co.jp/

2005 年 11 月

300577-001JA
JPN/0511/PDF/SE/CHG/HU/HN