



Application Power Management for Mobility

ホワイト・ペーパー

2002年3月20日



目次

1.	はじめに	4
1.1.	概要	4
1.2.	対象読者	4
2.	Application Power Management for Mobility	5
2.1.	バッテリー寿命の延長	5
1.1.	システムのスリープ / 稼働状態の切替えに対する処理	6
2.3.	システムのスリープ・モードへの移行の防止	6
2.4.	Application Power Management for Mobility の実装	7
2.4.1.	バッテリー寿命延長のための実装手順	7
2.4.2.	システムのスリープ / 稼働状態への移行に対処するための実装手順	7
2.4.3.	システムがスリープ・モードに入ることを防止するための実装手順	8
2.4.4.	概念実証アプリケーション	8
3.	Microsoft® Windows® Power Management API とメッセージ	9
3.1.	電力の状況	9
3.1.1.	GetSystemPowerStatus	9
3.1.2.	WM_POWERBROADCAST	10
3.2.	電力管理スキーム	11
3.2.1.	EnumPwrSchemes	12
3.2.2.	GetActivePwrScheme	12
3.2.3.	システムの電力管理スキームの変更	13
3.3.	実行状態	13
3.3.1.	SetThreadExecutionState	13
4.	Intel® Mobility Application Power Demo	14
4.1.	概要	14
4.2.	Application Power Management for Mobility の実装	15
4.2.1.	バッテリー寿命の延長	15
4.2.2.	システムのスリープ / 稼働状態への移行に対する処理	19
4.3.	インストール	19
4.4.	ソース・コード	19
5.	結論	20
	付録 A : Intel® Mobility Application Power Demo の実行ファイルとソース・コード	21



略語一覧

API	Application Programming Interface
DLL	Dynamic Link Library
MFC	Microsoft* Foundation Classes
MSDN	Microsoft* Developer Network
UI	User Interface

*その他の社名、製品名などは、一般に各社の商標または登録商標です。



1. はじめに

1.1. 概要

移動通信システムの将来的な成長にとって、優れたユーザ体験とバッテリー寿命の長さは決定的な意味を持ちます。ユーザ体験の改善とバッテリー寿命の延長を実現するためには、アプリケーション・ソフトウェアが主要な役割を果たします。このホワイト・ペーパーでは Application Power Management for Mobility の概念について検討を加え、アプリケーション・ソフトウェアがユーザ体験の改善とバッテリー寿命の延長をどのように実現するかを示しています。

このホワイト・ペーパーには、Application Power Management for Mobility のアプリケーションへの実装手順、利用可能な Microsoft* Windows* API (Application Programming Interface) の紹介、および Intel®ソフトウェア・ラボにおいて開発された概念実証アプリケーションの概要が記載されています。

1.2. 対象読者

このホワイト・ペーパーは、Windows* 98、Windows* 98SE、Windows* ME、Windows* 2K、および Windows* XP を含む Microsoft* Windows* オペレーティングシステムを対象としてアプリケーションを開発するアーキテクト、設計者、およびデベロッパを対象としています。またアプリケーションの定義や設計にかかわるマネージャ、検証エンジニア、技術マーケティング・エンジニアを対象とした、Application Power Management for Mobility の優れた概要も記載されています。

* その他の社名、製品名などは、一般に各社の商標または登録商標です。



2. Application Power Management for Mobility

Application Power Management for Mobility は、アプリケーションが実装すべき 3 つの電源管理機能を特定しています。これらの機能にはシステムがバッテリーにより駆動されているときのバッテリー寿命の延長、システムのスリープ / 稼働状態の切替えに対する処理、およびアプリケーション使用時におけるシステムのスリープ状態への移行の防止が含まれます。

2.1. バッテリー寿命の延長

システムがバッテリーにより駆動されている場合、アプリケーションは機能のパフォーマンス / 品質を落とすこと、バックグラウンド動作の停止、またはそれらの組合せによってバッテリー寿命を延長できます。これらの動作をアプリケーションに組み込むことにより、プロセッサのアイドル¹時間が延長されます。プロセッサの消費電力はアイドル時の方が動作²時よりもはるかに低いため、結果としてバッテリー寿命が長くなります。

バッテリー寿命の延長に関しては、そのためにパフォーマンス / 品質を低下させるべき機能や停止すべきバックグラウンド動作が定義されているわけではありません。これらの機能とバックグラウンド動作はアプリケーションごとに異なります。

この例としては次のようなものがあります。

- 旅行中に DVD の再生時間を延ばすため、ビデオやオーディオの再生品質を落とす。
- バッテリー残量がわずかなときは、ワードプロセッサのスペルチェックや文法の自動校正機能をオフにして文書を完成させる。
- オンライン時間を延ばすため、ウェブ・ベースの財務アプリケーションの自動株価表示機能による更新の頻度を落とす。

システムがバッテリーによって駆動されているときは、アプリケーションによってバッテリー寿命を延長することができます。またバッテリーの使用可能時間をどのような場合に延長するかのもアプリケーションによって定義できます。このような条件はその時点でのバッテリー状況、システムの電力管理スキーム³、またはその両方に依存します。

たとえば、次の条件のうち少なくとも 1 つが満たされている場合にはアプリケーションによってバッテリーの使用可能時間を延長できます。

- バッテリー残量の割合が一定の値よりも低い。
- バッテリー残量による使用可能時間が一定の値よりも短い。
- システムが特定の電力管理スキームを使用している。

アプリケーションのユーザ・インターフェイス (UI) には、いつ、およびどのようにして機能のパフォーマンス / 品質を低下させるか、バックグラウンド動作を停止するか、あるいはその両方を行うかを指定するためのオプションを配置できます。要件と環境についてはユーザ自身がかつともよく知っているため、ユーザによる指定は重要です。これらの設定をアプリケーション側で事前に設定し、UI から選択できないようにした場合、ユーザの使い勝手は悪くなる可能性があります。

¹ アイドル時とはプロセッサが命令を実行していない状態を指します。

² 動作時とはプロセッサが命令を実行している状態を指します。

³ Windows*オペレーティングシステムでは、電力管理スキームによって最適な電力消費を設定できます。電力管理スキームの詳細については Windows*のマニュアルを参照してください。



2.2. システムのスリープ / 稼働状態の切替えに対する処理

Windows*オペレーティングシステムでは、スリープ状態への移行はスタンバイまたは休止モードへの切替えを意味し、稼働状態への移行は休止またはスタンバイ・モードからの切替えを意味します。スタンバイと休止の 2 つのモードにより、通常のシステム終了 / 起動よりもはるかに速くシステムのオン / オフを行えます。

アプリケーションは、システムのスリープ / 稼働状態への移行に対処することによってユーザ体験を改善できます。システムのスリープ / 稼働状態への移行に対処するとは、アプリケーションが再起動を要求することなく、データを失うことなく、かつ他の状態を変化させることなく、これらの状態移行に対応する必要があることを意味します。さらに、優れたユーザ体験を確保するためには、アプリケーションはユーザとのやり取りなしにこれらの状態移行に対応する必要があります。

システムのスリープ / 稼働状態の切替えに対する処理に関しては、一般的な作業項目のリストは存在しません。必要な作業はアプリケーションに依存し、またアプリケーションごとに異なります。

考慮すべき項目は次のとおりです。

- スリープ状態への移行前に状態 / データを保存し、稼働状態への復帰後に状態 / データを回復する。
- スリープ状態への移行前にファイルや I/O デバイスなど開いているシステム・リソースのハンドルをすべて閉じる。
- スリープ状態への移行前に通信リンクをすべて切断し、稼働状態への復帰後にすべて確立し直す。
- 稼働状態への復帰後に (リモート・ファイルやリモート・データベースへの書き込みなど) すべてのリモート操作を同期化する。
- スリープ状態への移行前に (たとえばストリーミング・ビデオやファイルのダウンロードなど) 実行中のユーザ操作をすべて停止し、稼働状態への復帰後にすべて再開する。

2.3. システムのスリープ・モードへの移行の防止

Windows*オペレーティングシステムでは、スリープ・モードに移行するにはシステムをスタンバイまたは休止モードに切り替えます。バッテリー電力を節約するため、Windows*オペレーティングシステムは一定期間¹特定の動作が検出されない場合にシステムをスタンバイまたは休止モードに切り替えます。Windows*オペレーティングシステムが自動的に検出しない動作には、ディスクの動作、CPU の動作、およびビデオ・ディスプレイの動作があります。Windows*オペレーティングシステムが自動的に検出する動作は、キーボードからの入力、マウスからの入力、サーバの動作、およびウィンドウ・フォーカスの変更です。

ビデオ・プレーヤやプレゼンテーション・マネージャなどのアプリケーションでは、ユーザからの入力がないまま長期間ビデオ表示が行われ、その結果としてビデオ再生やプレゼンテーション中に Windows*オペレーティングシステムがディスプレイを停止することがよくあります。このような状況に対処するには、ディスプレイが停止するたびにシステム・ディスプレイを稼働状態に戻すか、あるいはディスプレイ停止までの待ち時間を長く設定してスリープ・モードへの移行を遅らせます。システム・ディスプレイを稼働状態に設定し直すごとにユーザ体験の品質は低下し、またスリープ・モードへの移行待ち時間を長く設定すると長期間システムが使用されない場合にバッテリー電力を浪費することとなり、いずれの対策も理想的なものとは言えません。

¹ Windows*オペレーティングシステムでは、電力管理スキームを使用して時間を指定できます。電力管理スキームの詳細については Windows*のマニュアルを参照してください。



オペレーティングシステムが検出しない動作をアプリケーションが実行している場合、オペレーティングシステムにその動作を伝えることによってスリープ・モードへの移行を防止できます。これによりユーザ体験が向上するだけでなくバッテリー管理も改善されます。

2.4. Application Power Management for Mobility の実装

このセクションではアプリケーションに Application Power Management for Mobility を実装するための手順を示しています。

2.4.1. バッテリー寿命延長のための実装手順

バッテリー寿命を延長するためのアプリケーションの実装手順は次のとおりです。

1. アプリケーションの機能のうち、バッテリー寿命を延長するためにパフォーマンス/品質を落としても構わないと思われるものを特定します。
2. アプリケーションのバックグラウンド動作のうち、バッテリー寿命を延長するために停止しても構わないと思われるものを特定します。
3. システムがバッテリーで駆動されている場合、アプリケーションによってバッテリー寿命を延長することができます。さらに、どのような状況のときにバッテリー寿命を延長するかの条件を定義できます。これらの条件はその時点でのバッテリー状況、その時点でのシステムの電力管理スキーム、またはその両方に基づきます。
4. このホワイト・ペーパーの「Microsoft® Windows® Power Management API とメッセージ」のセクションに目を通してください。このセクションはシステムの電源やバッテリーの状態、システムの電力管理スキームなど、システムの電力情報を取得するために Windows*オペレーティングシステム下で利用できる API とメッセージの概要を示しています。
5. 機能のパフォーマンス/品質を落とす、あるいはバックグラウンド動作を停止するタイミングと方法を示した UI を設計し、実装します。
6. システムの電力情報を取り出して機能のパフォーマンス/品質を落とすタイミングを決める、バックグラウンド動作を停止する、またはその両方を行うアプリケーション・ロジックを実装します。
7. 機能のパフォーマンス/品質を落とす、バックグラウンド動作を停止する、またはその両方を行うアプリケーション・ロジックを実装します。

2.4.2. システムのスリープ / 稼働状態への移行に対処するための実装手順

Windows*オペレーティングシステムは、システムがスリープ / 稼働状態に入ることを通知するメッセージをアプリケーションに送ります。アプリケーションはこれらのメッセージに応答し、電力状態の移行に対処するために必要な操作を実行します。

システムのスリープ / 稼働状態への移行に対処するためのアプリケーションの実装手順は次のとおりです。

1. システムがスリープ / 稼働状態に移行するときのアプリケーションの動作を調査します。Windows*オペレーティングシステムでは、スリープ状態への移行はスタンバイまたは休止モードへの切替えを意味し、稼働状態への移行はスタンバイまたは休止モードからの切替えを意味します。



2. アプリケーションが再起動を必要とする、データを失う、またはアプリケーションの状態が変化する原因となる問題点を特定します。
3. このホワイト・ペーパーの「Microsoft® Windows® Power Management API とメッセージ」のセクションに目を通します。このセクションはシステムのスリープ / 稼働状態への移行に対処するために Windows* オペレーティングシステム下で利用できる API とメッセージの概要を示しています。Windows* Power Management API の用語では、システムのスリープ / 稼働状態への移行は suspend および resume メッセージと呼ばれます。
4. 電力状態の移行通知に応答し、再起動を必要とすることなく、データを失うことなく、かつ状態を変更することなく状態移行に対処するために必要な操作を行うアプリケーション・ロジックを実装します。

2.4.3. システムがスリープ・モードに入ることを防止するための実装手順

システムがスリープ・モードに入ることを防止するためのアプリケーションの実装手順は次のとおりです。

1. オペレーティングシステムが検出しないアプリケーションの動作を特定します。
2. このホワイト・ペーパーの「Microsoft® Windows® Power Management API とメッセージ」のセクションに目を通します。このセクションには、システムがスリープ・モードに移行することを防止するために Windows* オペレーティングシステム下で利用できる API が記載されています。
3. オペレーティングシステムが検出しない動作をアプリケーションが実行しているときにシステムがスリープ・モードに入ることを防止するアプリケーション・ロジックを実装します。

2.4.4. 概念実証アプリケーション

Intel® ソフトウェア・ラボでは、Application Power Management for Mobility の概念を示すアプリケーションとして、Intel® Mobility Application Power Demo を開発しました。この概念実証アプリケーションは、バッテリー寿命の延長とシステムのスリープ / 稼働状態移行への対処のサポートをシミュレートします。

Intel® Mobility Application Power Demo とそのソース・コードは、対象となるアプリケーションに Application Power Management for Mobility を実装する際の参考としてお使いいただけます。詳細については「Intel® Mobility Application Power Demo」のセクションを参照してください。



3. Microsoft® Windows® Power Management API とメッセージ

Windows* 98、Windows* 98SE、Windows* ME、Windows* 2K、および Windows* XP を含む Microsoft* Windows* オペレーティングシステムに適用されます。

Application Power Management for Mobility の実装に必要な Microsoft* Windows* Power Management API とメッセージのみを取り扱います。すべての Microsoft* Windows* Power Management API とメッセージについては、Microsoft* Developer Network (MSDN) ライブラリを参照してください。

注：このセクションの内容の大半は MSDN ライブラリからの引用です。

3.1. 電力の状況

このセクションでは GetSystemPowerStatus API と WM_POWERBROADCAST メッセージについて説明しています。GetSystemPowerStatus API はシステムの電源や現在のバッテリー状況など、システムへの電力供給状況を知るのに使用されます。WM_POWERBROADCAST メッセージは、システムのスリープ / 稼働状態への移行およびシステムの電力状況に関する変化をアプリケーションに伝えるため使用されます。

3.1.1. GetSystemPowerStatus

GetSystemPowerStatus API はシステムへの電力供給状態を知るために使用されます。この状況とはシステムが AC と DC (バッテリー) のどちらで稼働しているか、バッテリーが現在充電中か否か、およびバッテリーの残量を意味します。

この API を使用するには **Windows.h** ヘッダ・ファイルをインクルードして **Kernel32.lib** ライブラリにリンクします。

```
BOOL GetSystemPowerStatus(LPSYSTEM_POWER_STATUS pSystemPowerStatus);
```

パラメータ:

LPSYSTEM_POWER_STATUS pSystemPowerStatus – 状態の情報を受け取る SYSTEM_POWER_STATUS 構造体へのポインタ。

戻り値:

この関数が成功した場合、戻り値はゼロ以外の値になります。失敗した場合には戻り値はゼロになります。詳細なエラー情報を取得するには GetLastError API を呼び出します (詳細については MSDN for GetLastError を参照してください)。

SYSTEM_POWER_STATUS 構造体

```
typedef struct _SYSTEM_POWER_STATUS {
    BYTE          ACLineStatus;
    BYTE          BatteryFlag;
    BYTE          BatteryLifePercent;
    BYTE          Reserved1;
    DWORD         BatteryLifeTime;
    DWORD         BatteryFullLifeTime;
} SYSTEM_POWER_STATUS, *LPSYSTEM_POWER_STATUS;
```



この構造体のメンバは次のとおりです。

- ACLineStatus – AC 電源の状態を示します。0 はオフライン (バッテリー / DC)、1 はオンライン (AC)、255 は不明を示します。
- BatteryFlag – バッテリーの充電状況を示します。1 は高、2 は低、4 は危険、8 は充電中、128 はシステム・バッテリーなし、および 255 は不明を示します。
- BatteryLifePercent – バッテリーの残量をパーセンテージで示します。この値は 0 から 100 までの範囲にあり、不明なときは 255 となります。
- Reserved1 – 予約済み。0 でなければなりません。
- BatteryLifeTime – バッテリーの残量を秒単位で表します。不明な場合は-1 となります。
- BatteryFullLifeTime – フル充電時のバッテリー寿命を秒単位で表します。不明な場合は-1 となります。

3.1.2. WM_POWERBROADCAST

WM_POWERBROADCAST メッセージはシステムのスリープ / 稼働状態への移行、およびシステムへの電力供給状況の変化をアプリケーションに伝えます。API マニュアルと Windows*オペレーティングシステムの用語規約に従うため、このセクションではシステムのスリープ状態への移行を suspend、稼働状態への移行を resume と呼びます。

注：Windows*オペレーティングシステムは、緊急の suspend が発生（例：バッテリーに障害が発生したとき）してもアプリケーションに通知しません。緊急の resume の通知（PBT_APMRESUMECRITICAL については下記参照）は、システムが緊急の suspend 状態から復帰した後にアプリケーションに送られます。このような状況では、アプリケーションは状態とデータを可能な限り回復する必要があります。

このメッセージを処理するには Windows.h ヘッダ・ファイルをインクルードします。

```

LRESULT CALLBACK WindowProc(
    HWND hwnd,           // handle to window
    UINT uMsg,           // message id (WM_POWERBROADCAST)
    WPARAM wParam,       // power-management event
    LPARAM lParam        // function-specific data
);

```

パラメータ:

wParam – 電源管理イベントを指定します。

PBT_APMBATTERYLOW – バッテリー残量が低下しています。バッテリー残量の低下を通知しない BIOS もあるため、システムによってはこのイベントは通知されません。

PBT_APMOEMEVENT – OEM 定義による BIOS 信号イベント。lParam は OEM 定義のイベント・コードを指定し、範囲は 0200h ~ 02Ffh です。OEM イベントを通知しない BIOS もあるため、システムによってはこのイベントは通知されません。

PBT_APMPOWERSTATUSCHANGE – 電力供給の状況に发生了变化がありました（たとえばバッテリー (DC) から AC に切り替えられた）。このメッセージは、バッテリーの残りの使用可能時間が 5 分未満となったとき、バッテリーの残量が全体の 10% 未満となったとき、およびバッテリー寿命が 3% 以上変化したときにも送信されます。アプリケーションはこのメッセージに対し、[GetSystemPowerStatus](#) 関数を呼び出してシステムの現在の電力供給状態を確認する必要があります。

PBT_APMQUERYSPEND – システムの suspend 許可が要求されました。Suspend 許可を与えたアプリケーションは、元に戻る前に suspend のための準備を行う必要があります。



lParam – 動作のフラグを指定する DWORD 値です。ビット 0 が 1 の場合、アプリケーションは suspend 状態に対してどのように準備するかの指示を求めることができます。それ以外の場合には、アプリケーションはユーザの介入なしに準備を行わねばなりません。その他のビットはすべて予約済みです。システムは PBT_APMSUSPEND 中の動作を完了するのに 20 秒しか認めていないため、20 秒以上を要する動作はすべてここで完了させる必要があります。Suspend 要求を許可するには TRUE、拒否するには BROADCAST_QUERY_DENY を返します。

PBT_APMQUERYSUSPENDFAILED – suspend の要求が拒否されました。このイベントは、直前の PBT_APMQUERYSUSPEND イベントに対してアプリケーションまたはドライバから BROADCAST_QUERY_DENY が返されたときに送信されます。アプリケーションはこれに対し、通常の動作を再開します。

PBT_APMRESUMEAUTOMATIC – システムがイベントを処理するため自動的に稼働状態に復帰したときに送信されます。アプリケーションがこれに対応するのは、ユーザによる介入なしにイベントを処理する場合だけです。PBT_APMRESUMEAUTOMATIC を送信した後にユーザによる作業を検出した場合、ユーザとのやり取りを全面的に開始できることをアプリケーションに伝えるため PBT_APMRESUMESUSPEND イベントを送信します。

PBT_APMRESUMECRITICAL – システムがバッテリーの障害など緊急の suspend 状態から回復しました。緊急の suspend は事前の通知なしに発生するため、イベントをアプリケーションが受け取る時にはそれまで取得できたリソースやデータにアクセスできなくなる場合があります。アプリケーションはこの状態を可能な限り回復する必要があります。

PBT_APMRESUMESUSPEND – システムが suspend 状態から回復しました。アプリケーションがこのイベントを受け取ることができるのは、システムが suspend 状態になる前に PBT_APMSUSPEND イベントを受け取った場合だけです。

PBT_APMSUSPEND – システムは作業を中断しようとしています。このメッセージは、システムが suspend 状態になる直前に送信されます。これはすべてのアプリケーションとドライバが直前の PBT_APMSUSPENDQUERY イベントに対して TRUE を返した場合にのみ送信されます。アプリケーションはこのイベントに対し、状態とデータの保存に必要なすべての作業を完了する必要があります。アプリケーションはこの作業を約 20 秒以内で完了しなければならず、20 秒が経過した後も作業が続けられている場合にはアプリケーションは中断されます。

lParam – イベントごとのデータ。詳細については wParam の情報を参照してください。大半のイベントについてはこのパラメータは予約済みであり、使用されません。

戻り値:

PBT_APMQUERYSUSPEND イベント 要求を許可する場合には TRUE、拒否する場合には BROADCAST_QUERY_DENY を返します。他のイベントについては戻り値はありません。

3.2. 電力管理スキーム

システムにインストールされているすべての電力管理スキームをリストアップし、現在のスキームを取り出す API は EnumPwrSchemes と GetActivePwrScheme です。

これらの API を使用するには Powrprof.h ヘッダ・ファイルをインクルードして Powrprof.lib ライブラリにリンクします。



3.2.1. EnumPwrSchemes

EnumPwrSchemes API はシステムにインストールされているすべての電力管理スキームをリストアップします。リスト中のそれぞれの電力管理スキームについて、EnumPwrSchemes はそのスキームの情報とともにコールバック関数を呼び出します。このコールバック関数の呼出しは同期的に行われ、すべてのスキームがリストアップされるまで戻りません。

```
BOOL EnumPwrSchemes(PWRSCHEMESENUMPROC lpfnPwrSchemesEnumProc, LPARAM lParam);
```

パラメータ:

PWRSCHEMESENUMPROC lpfnPwrSchemesEnumProc – リストアップされた各スキームに関するコールバック関数へのポインタです。

LPARAM lParam – コールバック関数に渡されるユーザ定義の値です。

戻り値:

関数が成功した場合、戻り値は ERROR_SUCCESS になります。失敗した場合には戻り値はゼロです。詳細なエラー情報を取得するには GetLastError API を呼び出します。

コールバック関数:

```
typedef BOOLEAN (CALLBACK* PWRSCHEMESENUMPROC)(
    UINT          uiIndex,          // power scheme index
    DWORD         dwName,          // size of the sName string, in bytes
    LPTSTR        sName,          // name of the power scheme
    DWORD         dwDesc,          // size of the sDesc string, in bytes
    LPTSTR        sDesc,          // description string
    PPOWER_POLICY pp,             // receives the power policy
    LPARAM        lParam           // user-defined value
);
```

名前と説明の文字列はヌルで終わる文字列であり、pp は電力管理スキームを含む POWER_POLICY 構造体へのポインタです。この POWER_POLICY 構造体にはそれぞれの電力管理スキームのポリシー設定が含まれません。電源ポリシーの設定は Application Power Management for Mobility には使用されないため、POWER_POLICY 構造体に関する情報はこのホワイト・ペーパーでは省略しています。POWER_POLICY 構造体の詳細については MSDN ライブラリを参照してください。

3.2.2. GetActivePwrScheme

GetActivePwrScheme API はアクティブな（現在使用中の）電力管理スキーム・インデックスを取り出します。

```
BOOL GetActivePwrScheme(PUINT puiID);
```

パラメータ:

PUINT puiID – アクティブな電力管理スキームのインデックスを受け取る変数へのポインタです。

戻り値:

この関数が成功した場合には戻り値はゼロ以外、失敗した場合にはゼロになります。詳細なエラー情報を取得するには GetLastError API を呼び出します。



3.2.3. システムの電力管理スキームの変更

Windows*オペレーティングシステムは、システムの電力管理スキームがプログラムまたはユーザによって変更された場合にもアプリケーションに通知しません。システムの電力管理スキームの変更への対処については、Intel® Mobility Application Power Demo は (GetActivePwrScheme API を使用して) システムの電力管理スキーム情報を定期的に取り出す Windows* Timer メッセージに依存します。Windows* Timer メッセージの詳細については MSDN ライブラリを参照してください。

3.3. 実行状態

このセクションでは、システムがスリープ・モードに入ることを防止するために使用される SetThreadExecutionState API について説明しています。

3.3.1. SetThreadExecutionState

SetThreadExecutionState API を使用することにより、アプリケーションはオペレーティングシステムに対して自らが使用中であることを伝え、したがってシステムがスリープ・モードに入ることを防止できます。

この API を使用するには **Windows.h** ヘッダ・ファイルをインクルードして **Kernel32.lib** ライブラリにリンクします。

```
EXECUTION_STATE SetThreadExecutionState(EXECUTION_STATE esFlags);
```

パラメータ:

EXECUTION_STATE esFlags – スレッドの実行要件を指定します。このパラメータは次の値を 1 つ以上取ることができます。

ES_SYSTEM_REQUIRED – システムに対し、通常は動作としてシステムに認識されない作業をスレッドが実行していることを伝えます。

ES_DISPLAY_REQUIRED – システムに対し、通常は動作としてシステムに認識されない作業をスレッドが実行していることを伝えます。

ES_USER_PRESENT – ユーザがいることをシステムに伝えます。ユーザが存在する場合、システムはそのユーザが設定した電源管理ポリシーの設定を使用します。存在しない場合にはシステムはディスプレイを稼働状態に戻さず、可能な限り早くスリープ状態に戻ります。

ES_CONTINUOUS – このフラグが他の 1 つ以上の状態フラグとともに使用された場合、設定されている状態を無限に継続するようシステムに伝えます。この無限の状態をリセットまたは解除するには、SetThreadExecutionState を再度 ES_CONTINUOUS フラグとともに呼び出します。

戻り値:

関数が成功した場合には戻り値は直前のスレッドの実行状態となり、失敗した場合には NULL が返されます。

たとえばプレゼンテーション・アプリケーションは、スライド・ショーを開始する前には SetThreadExecutionState を ES_SYSTEM_REQUIRED、ES_DISPLAY_REQUIRED、および ES_CONTINUOUS フラグとともに呼び出し、スライド・ショーが完了した後は SetThreadExecutionState を ES_CONTINUOUS フラグのみとともに呼び出します。



4. Intel® Mobility Application Power Demo

Intel®ソフトウェア・ラボは Application Power Management for Mobility の概念を示すためのアプリケーションとして Intel® Mobility Application Power Demo を開発しました。Mobility Application Power Demo とそのソース・コードは、対象となるアプリケーションに Application Power Management for Mobility を実装する際の参考としてお使いいただけます。

このセクションでは、Mobility Application Power Demo の概要、Mobility Application Power Demo がどのように Application Power Management for Mobility のサポートをシミュレートするか、および Mobility Application Power Demo とそのソース・コードの入手方法について説明しています。

4.1. 概要

Mobility Application Power Demo では、マイクロソフト社の Notepad と同様にテキスト文書のオープン、編集、および保存を行えます。このアプリケーションはバッテリー寿命の延長およびシステムのスリープ / 稼働状態への移行に対する処理のサポートをシミュレートします。システムのスリープ・モードへの移行の防止のサポートについてはシミュレートしません。

このアプリケーションのその他の機能は次のとおりです。

- 新規または既存の文書をプレーン・テキスト・フォーマットとして作成または開く。
- テキストのコピー、切り取り、および貼り付け。
- 文書への変更の取り消し。
- 既存文書への変更の保存。
- プレーン・テキスト・フォーマットによる新規文書の作成。
- 文書の印刷および印刷プレビュー。
- 最近開いた文書の追跡。
- バッテリーの状態とバッテリーの名称、タイプ、製造元などの情報の表示。
- 保存先などの保存時オプションの選択と保存されていないデータについての警告。

図 1 に示すこのアプリケーションのメイン UI には、アプリケーションの機能にアクセスするためのメニュー / ツール・バー、アプリケーションの状態を示すステータス・バー、およびテキスト文書を作成 / 編集するための編集ウィンドウが含まれます。

メニュー項目からは、次のアプリケーション機能を利用できます。

- [File] メニューからはテキスト・ファイルの作成、オープン、保存、および印刷を行います。
- [Edit] メニューからはテキストの切り取り、コピー、および貼り付けを行います。
- [View] メニューからはバッテリーの状態情報の表示、およびステータス / ツール・バーの表示 / 非表示の切替えを行います。
- [Options] メニューからは保存オプションと電力管理オプションの選択を行います。
- [Help] メニューからは Mobility Application Power Demo のバージョン情報と概要を表示できます。

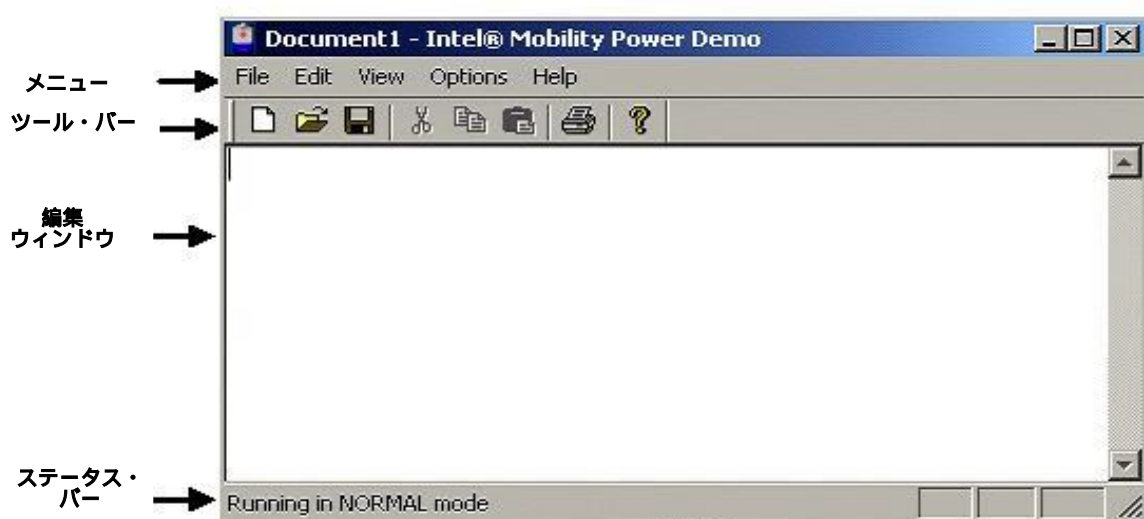


図 1

4.2. Application Power Management for Mobility の実装

Mobility Application Power Demo による Application Power Management for Mobility の実装は単なるシミュレーションです。このアプリケーションの概念を示すために使用されているオプションや UI はその概要を示すためのものであり、設計のガイドラインとして使用することを意図したものではありません。

4.2.1. バッテリー寿命の延長

バッテリー寿命を延長するため、Mobility Application Power Demo は次の機能を実装しています。

- 2つの動作モード：Normal モードと Low Power モード。
- 電力管理オプション UI：いつ、どのようにバッテリー寿命を延長するかをユーザーが指定できます。

Normal モードと Low Power モードの違い

Mobility Application Power Demo はバッテリー寿命を延長するときには Normal モードから Low Power モードに切り替わり、延長しないときは Low Power モードから Normal モードに切り替わります。このモードの切替えは機能を説明するためにのみ行われ、アプリケーションの実際の動作には影響しません。アプリケーションのステータス・バーには現在の動作モードが表示されます（図 2 参照）。



図 2

動作モードの
表示

電力管理オプション UI

Mobility Application Power Demo の電力管理オプション UI からは、アプリケーションをいつ Low Power モードに切り替えるか、および Low Power モードにおいてどのように動作させるかをユーザが指定することができます。この電力管理オプション UI は [Setting] タブと [Low Power] タブを含むプロパティ・シートになっています。このプロパティ・シートには、図 3 に示すように [Options] メニューの [Power] からアクセスできます。

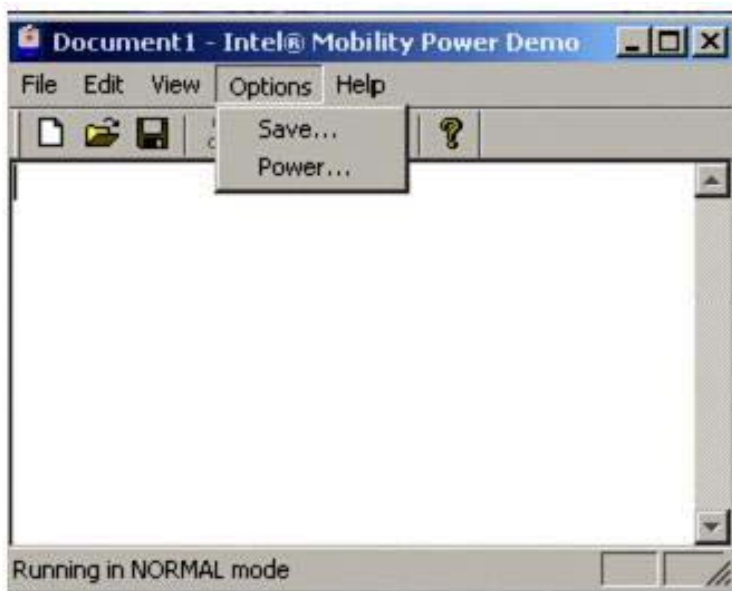


図 3



Settings タブ

図 4 に示す [Settings] タブでは、アプリケーションをいつ Low Power モードに切り替えるかの条件を設定できます。アプリケーションは、システムがバッテリーで駆動されており、このタブで指定された条件がすべて満たされたときに Low Power モードに切り替えられます。

設定できる条件は次のとおりです。

- 現在のバッテリー電力のレベルが指定された値未満に落ちたときにアプリケーションを Low Power モードに切り替えます。
- すべての電力管理スキームについて、または現在の電力管理スキームが選択された電力管理スキームに一致したときに Low Power モードに切り替えます。

[Settings] タブには現在の電源、バッテリー残量のパーセンテージ、バッテリーの残りの使用可能時間、および現在の電力管理スキームなどの情報も表示されます。

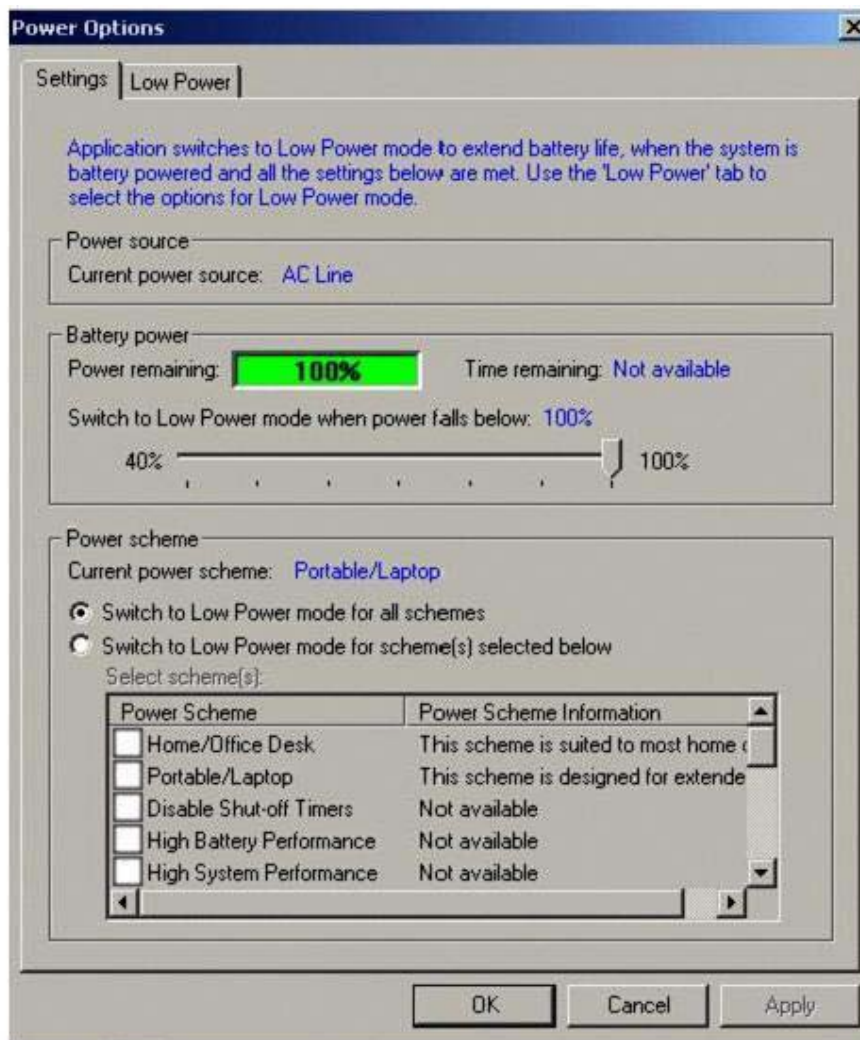


図 4



Low Power タブ

図 5 に示す [Low Power tab] タブからは、Low Power モードでのアプリケーションの動作を指定できます。このタブのオプションは、バッテリー寿命を延長するために機能のパフォーマンス / 品質を落とし、またバックグラウンド動作を停止するためのものです。これらのオプションは説明のためにのみ表示されており、Mobility Application Power Demo の実際の動作には影響しません。

[Low Power tab] タブのオプションは次のとおりです。

- Low Power モードにおいてビデオ再生の品質を落とすためのオプション。
- Low Power モードにおいてウェブからの情報更新頻度を落とすためのオプション。
- Low Power モードにおいてアプリケーションの特定の機能を無効にするためのオプション。

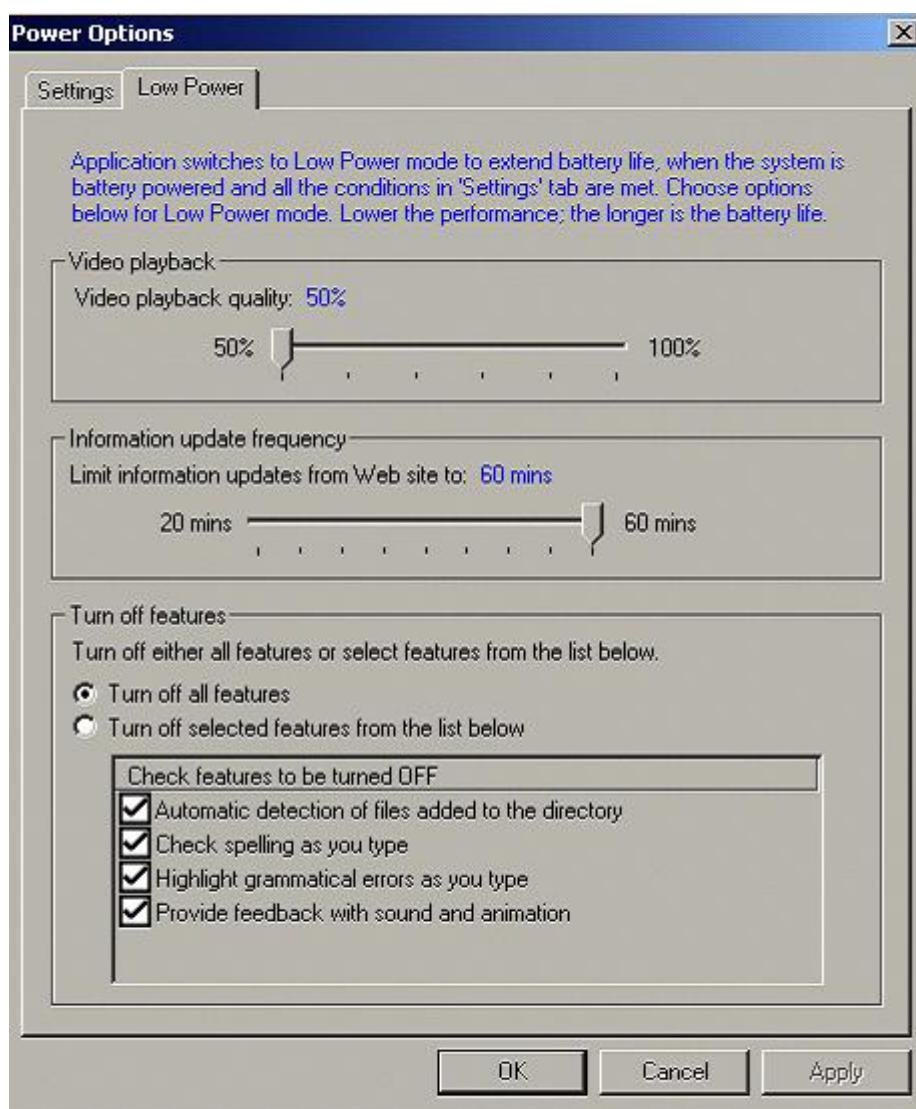


図 5



4.2.2. システムのスリープ / 稼働状態への移行に対する処理

Mobility Application Power Demo はシステムのスリープ / 稼働状態への移行に対する処理のサポートをシミュレートし、suspend または resume 操作準備を行っているというステータス・メッセージ（図 6 と 7 参照）を表示します。これらのメッセージは例証としてのみ表示されており、優れたユーザ体験を実現するにはシステムのスリープ / 稼働状態への移行はメッセージを表示することなく透過的に行う必要があります。

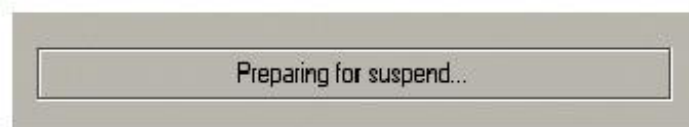


図 6

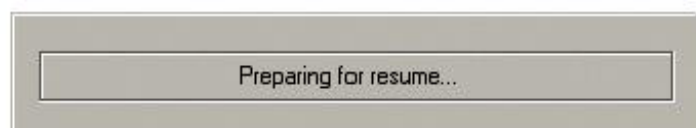


図 7

4.3. インストール

Mobility Application Power Demo をダウンロードするためのリンクは付録 A に記載されています。このダウンロード内容には Mobility Application Power Demo の実行ファイル ([pwrdemo.exe](#)) と Readme ファイル ([Read Me.htm](#)) が含まれています。対象とするシステムに Mobility Application Power Demo をインストールし、実行ファイルをダブルクリックしてアプリケーションを起動してください。ライブラリや DLL を追加する必要はありません。

Mobility Application Power Demo を実行するためのシステム要件は次のとおりです。

- Windows* (98、98SE、ME、2K、または XP) オペレーティングシステムを搭載したバッテリー駆動のノートブック¹。

4.4. ソース・コード

Mobility Application Power Demo のソース・コードをダウンロードするためのリンクは付録 A に記載されています。このダウンロード内容にはすべてのソース・ファイルとアプリケーションの設計ノート ([Design Notes.htm](#)) が含まれています。この設計ノートには、ソース・ファイルの内容に関する概要が含まれています。

Mobility Application Power Demo は C++ で作成されており、Microsoft* Foundation Classes (MFC) アプリケーション・フレームワーク・ライブラリを使用しています。アプリケーションのビルドには Visual Studio 6.0 IDE を使用してください。

¹ 注：このアプリケーションはデスクトップまたはバッテリー駆動でないノートブックでも動作しますが、アプリケーションが Normal モードと Low Power モード間でどのように切り替わるかは確認できません。



5. 結論

このホワイト・ペーパーに概要を示したように、アプリケーションへの Application Power Management for Mobility の実装はごくシンプルに行えます。実装時には次の項目に留意してください。

- バッテリー寿命を延長するためにパフォーマンス / 品質を落とすべきアプリケーション機能と、停止すべきバックグラウンド動作を特定します。
- システムのスリープ / 稼働状態への移行に伴ってアプリケーションが再起動を必要とする、データを失う、または状態を変更する原因となる問題点を特定します。
- オペレーティングシステムが検出しないアプリケーションの動作を特定します。
- このホワイト・ペーパーの「Microsoft® Windows® Power Management API とメッセージ」セクションに目を通してください。
- 設計開発の際には Intel® Mobility Application Power Demo アプリケーションとそのソース・コードを参考として使用してください。
- ユーザがバッテリー寿命を延長するために、いつ、およびどのように機能のパフォーマンス / 品質を低下させ、またはバックグラウンド動作を停止するかを指定できる UI を実装します。
- バッテリー寿命を延長し、システムのスリープ / 稼働状態への移行に対処し、オペレーティングシステムが検出しない動作をアプリケーションが実行しているときにシステムがスリープ状態に移行することを防止するためのアプリケーション・ロジックを実装します。



付録 A : Intel® Mobility Application Power Demo の 実行ファイルとソース・コード

Intel® Mobility Application Power Demo の実行ファイルは [TODO \(Insert the link for the download\)](#) から入手できます。

Intel® Mobility Application Power Demo のソース・コードは [TODO \(Insert the link for the download\)](#) から入手できます。