Mobile Platform Idle Power Optimization – Methodologies and Tools

Matthew Robben
Program Manager, Microsoft Corporation
Susumu Arai
Mobile Platform Architect, Intel Corporation

Session ID: EBLS003

Sponsors of Tomorrow.™ intel

Agenda

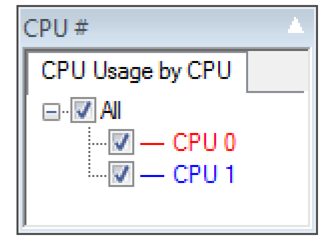
- Mobile platform energy efficiency goals
- Hardware considerations
- Software considerations
- Tools for idle power analysis

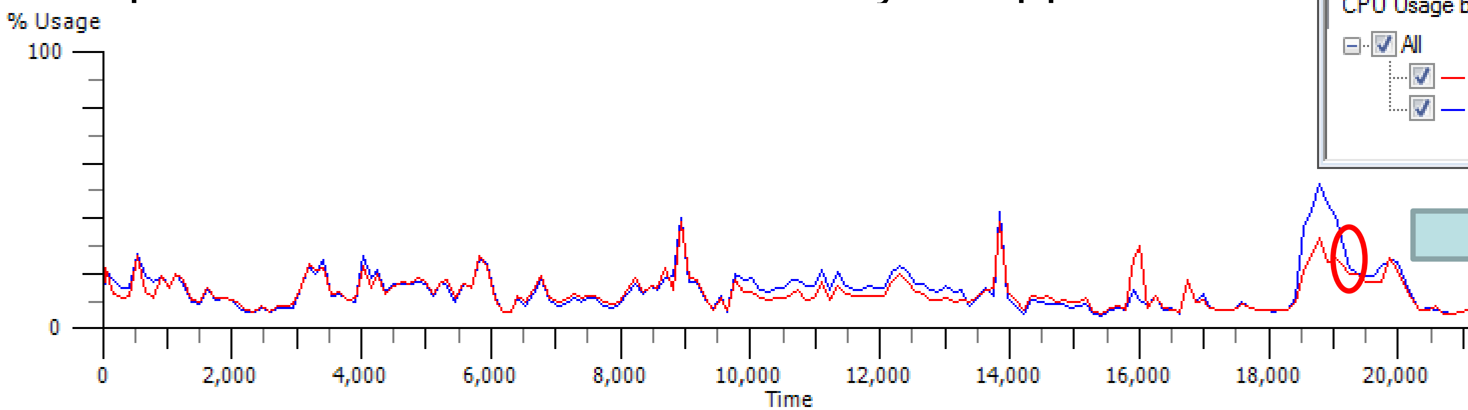
Mobile Platform Energy Efficiency Goals

Minimize power consumption while idle

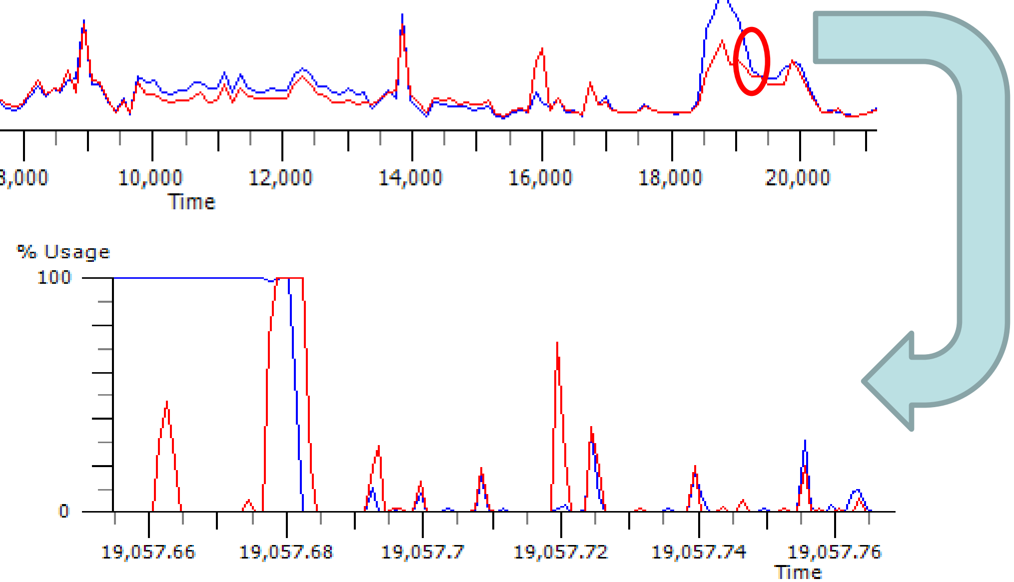
- Mobile client systems are idle (low CPU utilization) most of the time

Example: Office PC loaded with many IT applications





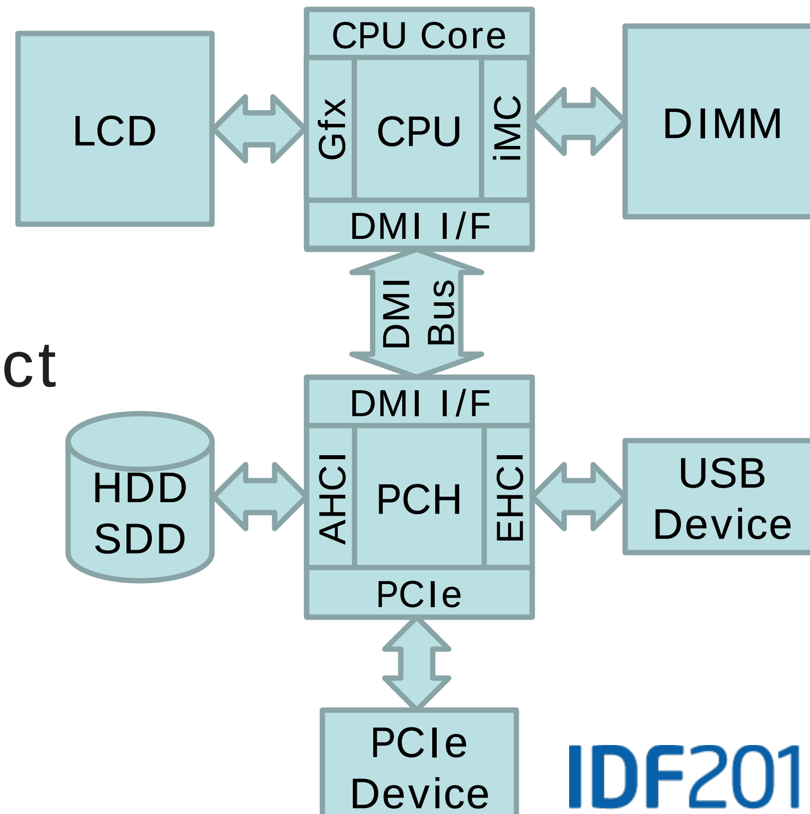
- Even with moderately busy workloads, there are a lot of low CPU utilization sections



Source: Intel Corporation

Platform Power Saving Features

- Today's mobile platform implements many power saving features
- But, many systems are not taking full advantage of them
  - Not properly configured
  - Bad component selection
  - Bad software activities
- One bad component can cause significant impact



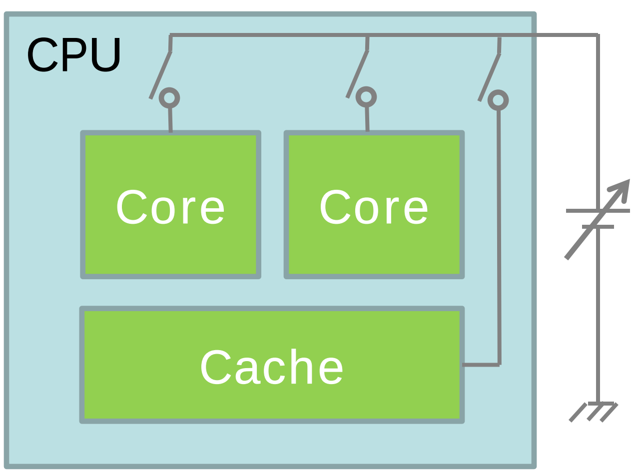
PCIe = PCI Express* Technology

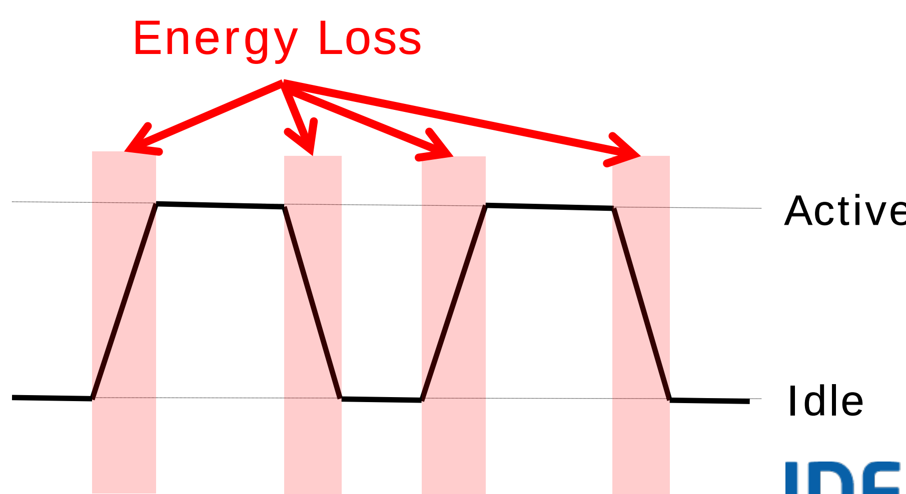
Agenda

- Mobile platform energy efficiency goals
- Hardware considerations
- Software considerations
- Tools for idle power analysis

CPU Power Saving

- CPU is a well power managed component
  - Dynamic power supply voltage
    Lower voltage when CPU utilization is low
  - Power gating
    Turn off power when CPU is idle to minimize leakage
  - CPU can adjust performance/power to the workload
- But, burns energy at each idle/active transition

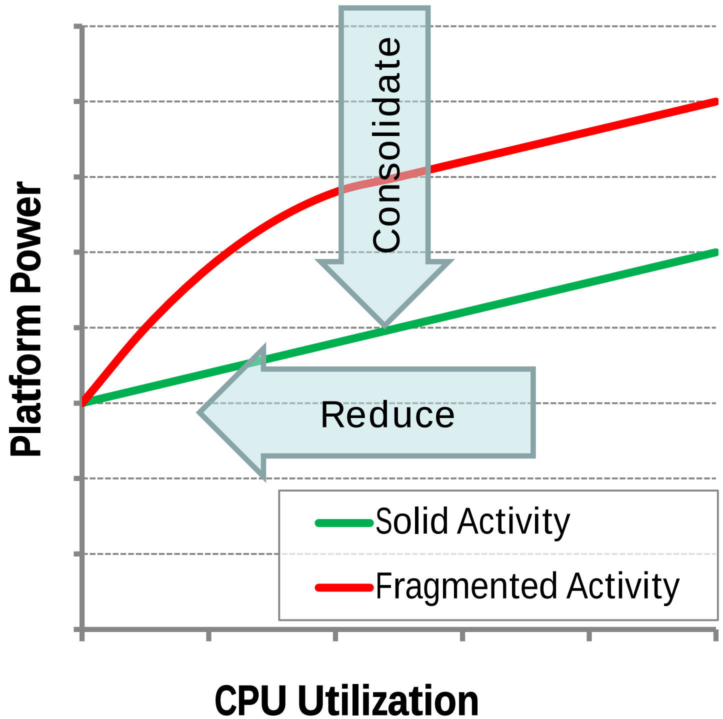




IDF2010 INTEL DEVELOPER FORUM

Software Impact to Platform Power

- Resource utilization (CPU cycle count) isn't the only factor
- Periodicity of the activity makes big impact
- Optimization needs to address both
- Details will be discussed in the software section



PCI Express* / SATA Power Saving

- Serial buses implement Link Power Management
  - L0s and L1 (ASPM) states for PCI Express* links
  - Partial and Slumber states for SATA links
- Maximize residency in the lower power states
- But, just setting enable bits isn't enough

  Actual residency determined by many factors
  - Traffic on the bus
  - Device and driver policy

Recommendations

- Select devices with adequate policy*
- Verify LPM state residency in the shipping configuration
- Configure driver policy if necessary

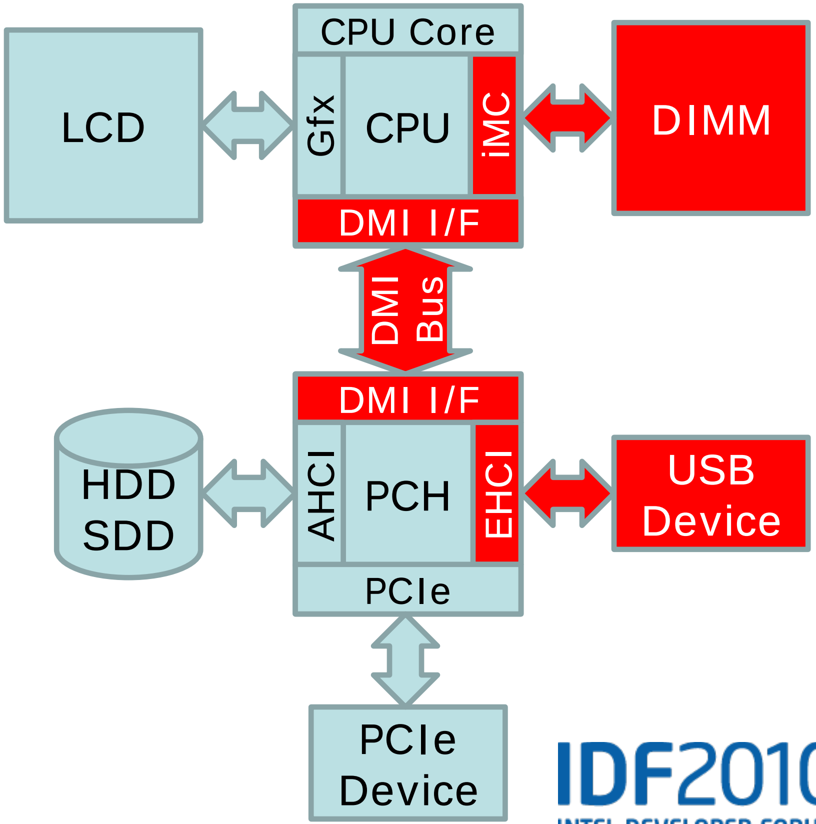
  Intel® Rapid Storage Technology (RST) has registry settings for SATA policy*

Please see the following white papers for details :

- Designing Energy Efficient SATA Devices (SATA devices)
- Energy-efficient platform devices (PCI Express* devices)

USB Device Power Saving

- USB operates on periodic polling and keeps large part of platform in active state
- USB device should implement Selective Suspend and stay in that state as long as possible



Please see the following white papers for details :
- Energy-efficient platform devices
- Making USB a more energy efficient interconnect

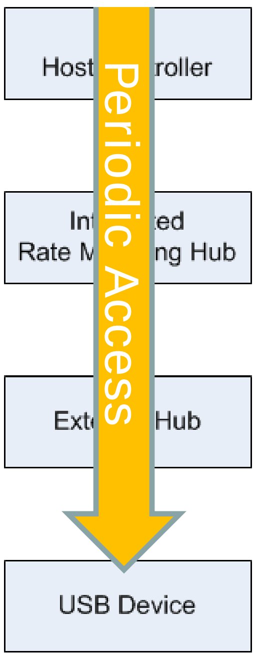
PCIe = PCI Express* Technology

USB Device Power Saving (Cont.)

- Periodic device polling diminishes the benefit of selective suspend

Recommendations

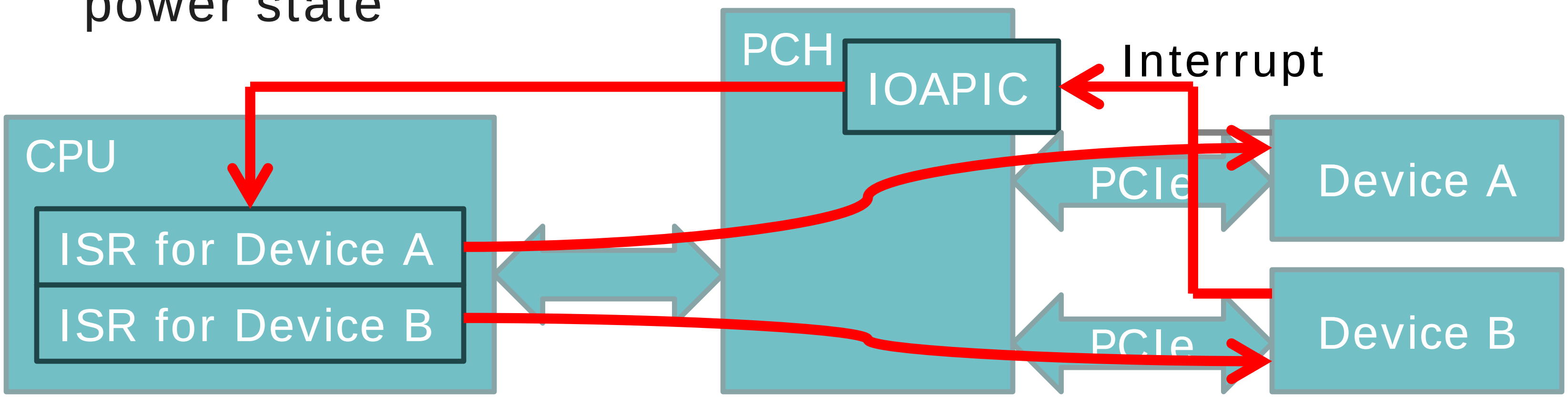
- Choose devices with selective suspend support
- Minimize access to the devices
- Place the device closer to host controller if periodic access is necessary



IDF2010 INTEL DEVELOPER FORUM

Interrupt Sharing

- Interrupt sharing increases platform activity
  - ISRs (Interrupt Service Routines) for multiple devices are executed to determine the source of interrupt
  - Each ISR accesses its hardware and wakes bus from low power state



Recommendations
- Make sure device and driver support MSI
- If not, assign dedicated line for devices with frequent interrupts

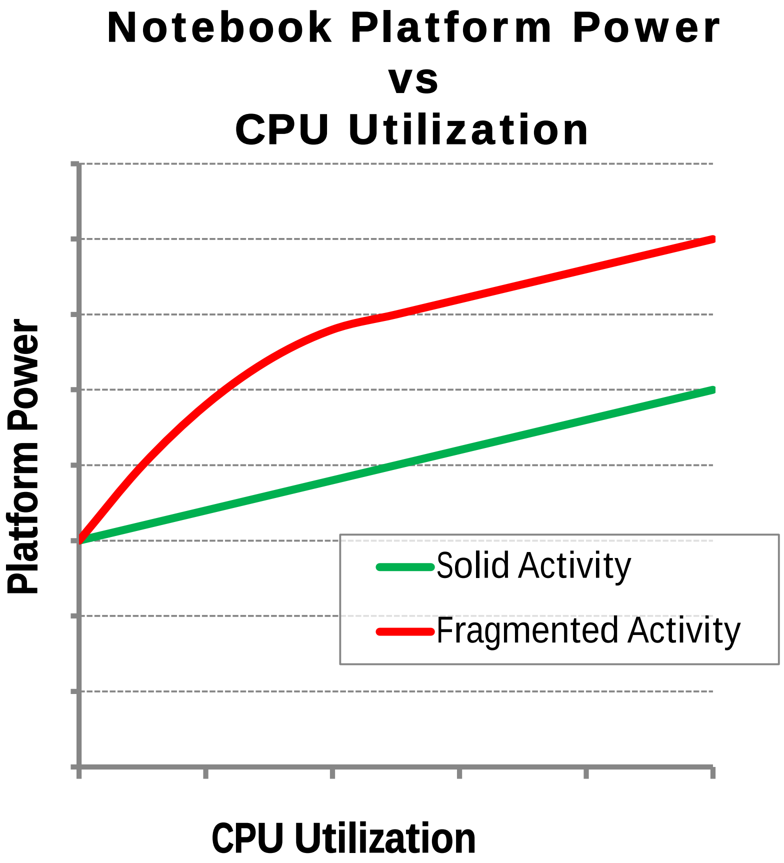
PCIe = PCI Express* Technology

Agenda

- Mobile platform energy efficiency goals
- Hardware considerations
- Software considerations
- Tools for idle power analysis

Optimize Application Behavior

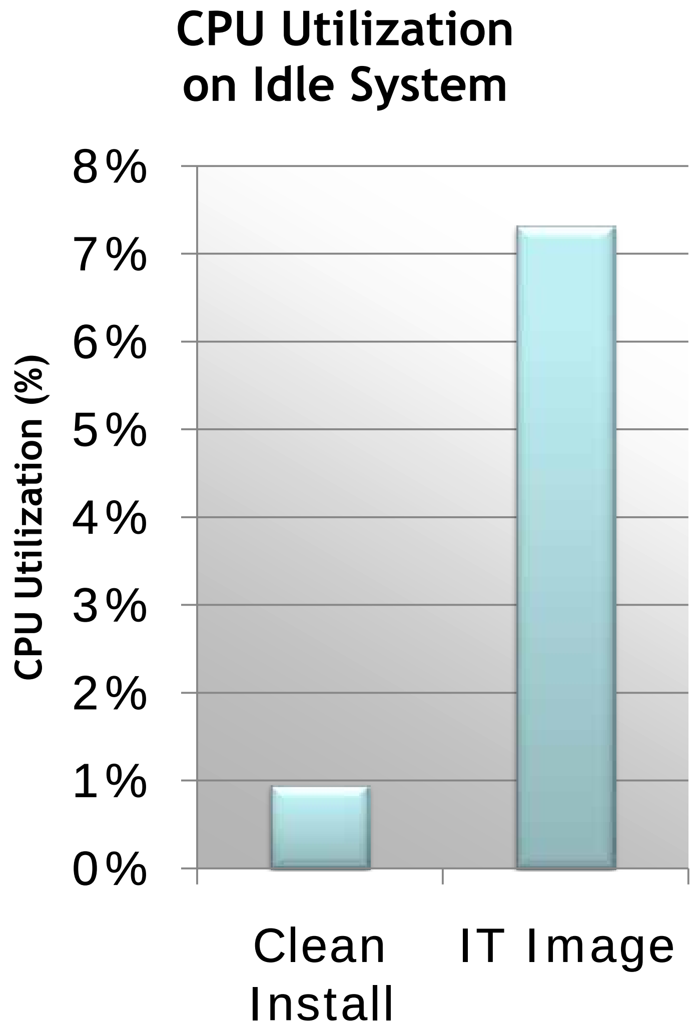
- Two major factors that determine platform power
  - Resource Utilization
    - § (e.g. CPU utilization)
  - Resource Usage Pattern
    - § Fragmented activities cause more power impact
- Focus on idle



Application Design Principles

Respect System Idle

- Idle dominates usage scenarios for client systems
  - Reducing idle power is essential for extending battery life
  - Windows* 7 made vast improvements



Source: Microsoft Corporation

IDF2010 INTEL DEVELOPER FORUM

Optimize Resource Usage

- Performance improvement = power improvement
- Architect event driven designs instead of polling or spinning
  - WaitForSingleObjectEx() or SleepEx()

```
void EatBatteryLife()
{
        HANDLE sharedResource = NULL;

        //your process waits for a file to be created via:
        while (sharedResource == NULL)
        {
                waitTime++;
                sleep(1);
        }
}
```

DO NOT DO THIS

WaitForSingleObjectEx() API Usage

```
//process 1's code
void UpdateSharedResource()
{
   //set sharedResource
   sharedResource = UpdateResource();

   // Set sharedResourceIsReadyEvent to
   // signaled
   SetEvent(sharedResourceIsReadyEvent);
}
```

```
//process 2's code
void ConsumeSharedResource()
{
   DWORD dwWaitResult;

   dwWaitResult = WaitForSingleObjectEx(
            sharedResourceIsReadyEvent,
            INFINITE,
            FALSE);  // indefinite wait

   switch (dwWaitResult)
   {
      case WAIT_OBJECT_0:
      //
      // TODO: use sharedResource
      //
      break;
      default:
      return 0;
    }
}
```

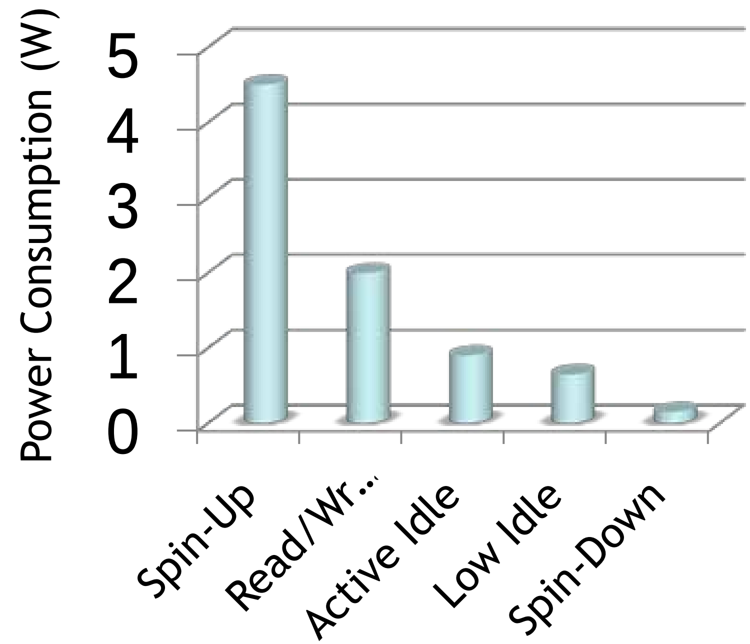
Optimize Resource Usage (2)

- Repainting GUI cascades work to graphics controller
  - 1 pixel change causes 10 VBI
  - Avoid animation icons in the system tray area
- Don't use WMI where Win32 APIs or .NET classes will suffice
  - Example: Repeated Win32_Directory enumeration vs. FileSystemWatcher class
- Choose event-driven APIs
  - Example: EvtSubscribe() instead of EventLogQuery()

Optimize Resource Usage (3)

- Assume all devices have power states
- Reduce device activity to enable low power states
  - Disk spin-down
  - Periodic, low-priority disk activity from applications should be on order of several hours
  - Batch I/Os where possible
  - Use volatile registry keys for transient information

Example HDD Power

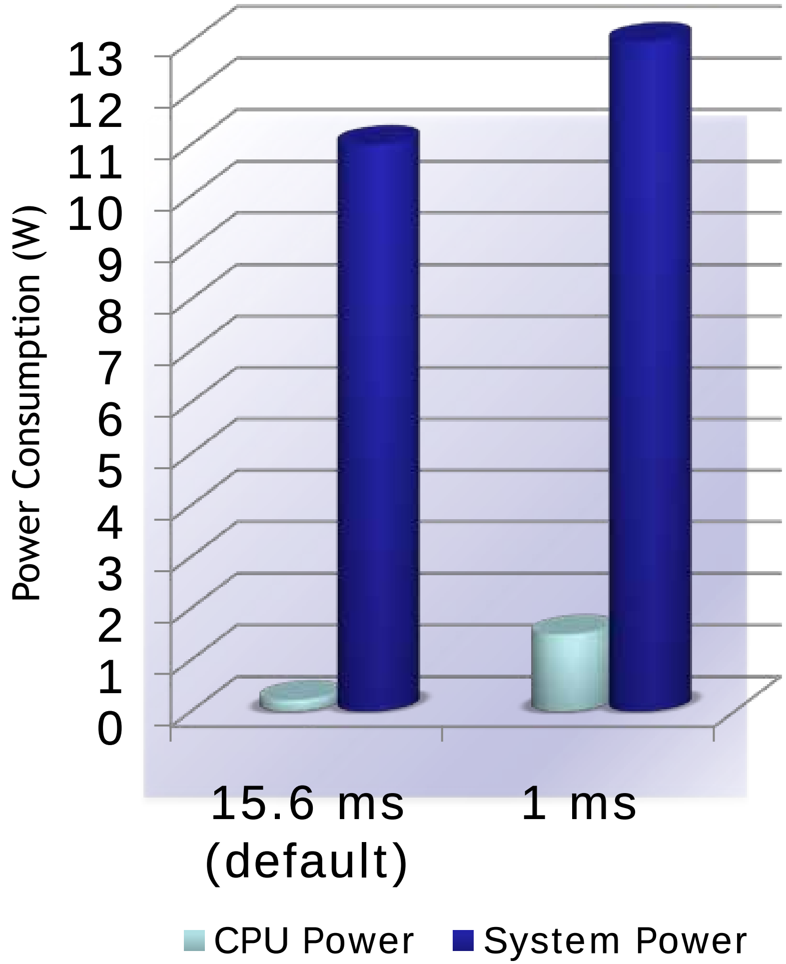


Source: Microsoft Corporation

IDF2010 INTEL DEVELOPER FORUM

Reduce Periodic Activity

- Avoid changing timer resolution from the default setting
  - Avoid reducing by using larger buffers
  - If you want more granular timestamps, use QueryPerformanceCounter()
- Media playback should use 10ms (or larger)
  - Limit request to as little codepath as possible
- Audio playback code should use event-driven, shared-mode WASAPI*

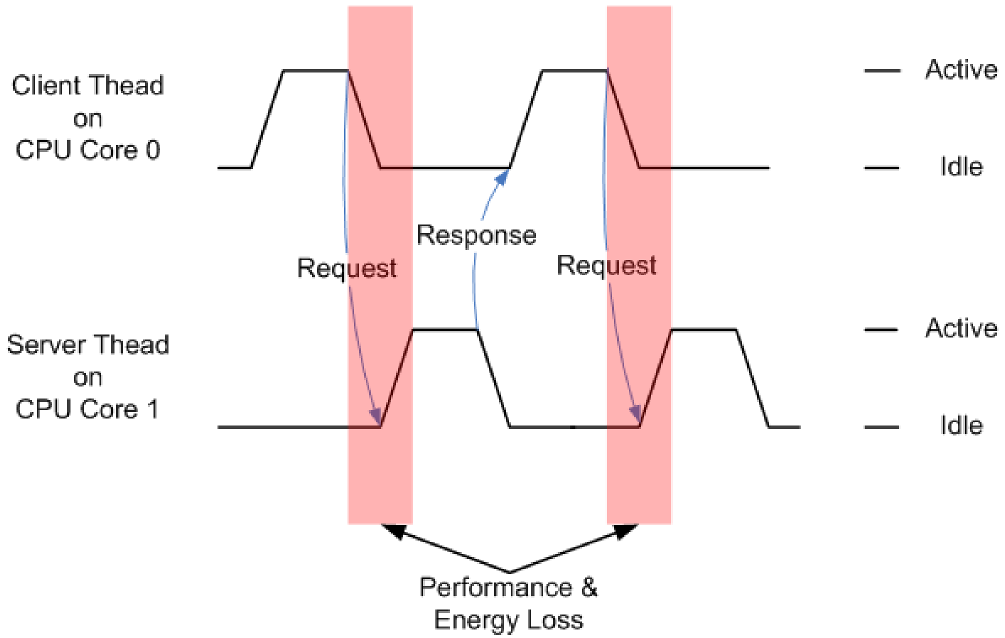


Source: Microsoft Corporation

* Please see Windows* 7 SDK sample code at
C:\Program Files\Microsoft SDKs\Windows\v7.0\Samples\multimedia\audio\

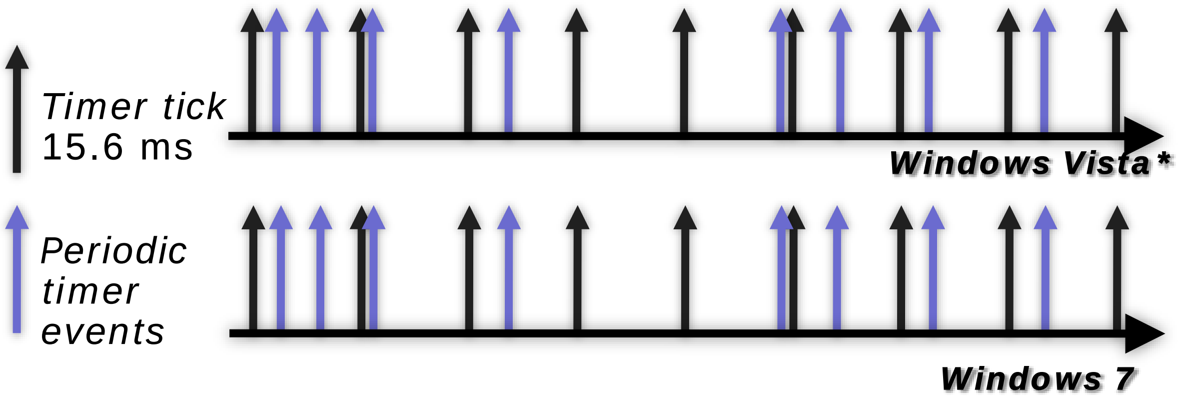
Reduce Periodic Activity (2)

- Thread ping-pong effects
  - Avoid extremely short duration 'work & signal' patterns
    - Use of RPC/COM can cause this
  - Battery Life Analyzer tool provides insight into thread behavior



Coalesce Timers

- Kernel mode and user mode timers should coalesce with other work on the system for minimum power impact
  - Need to engineer for timing tolerance
- New coalescing APIs in Windows* 7



IDF2010
INTEL DEVELOPER FORUM

SetWaitableTimerEx() API

- Replace calls to SetWaitableTimer() with this API
  - More efficient than a purely periodic timer
  - Tolerance parameter should scale with the timer period

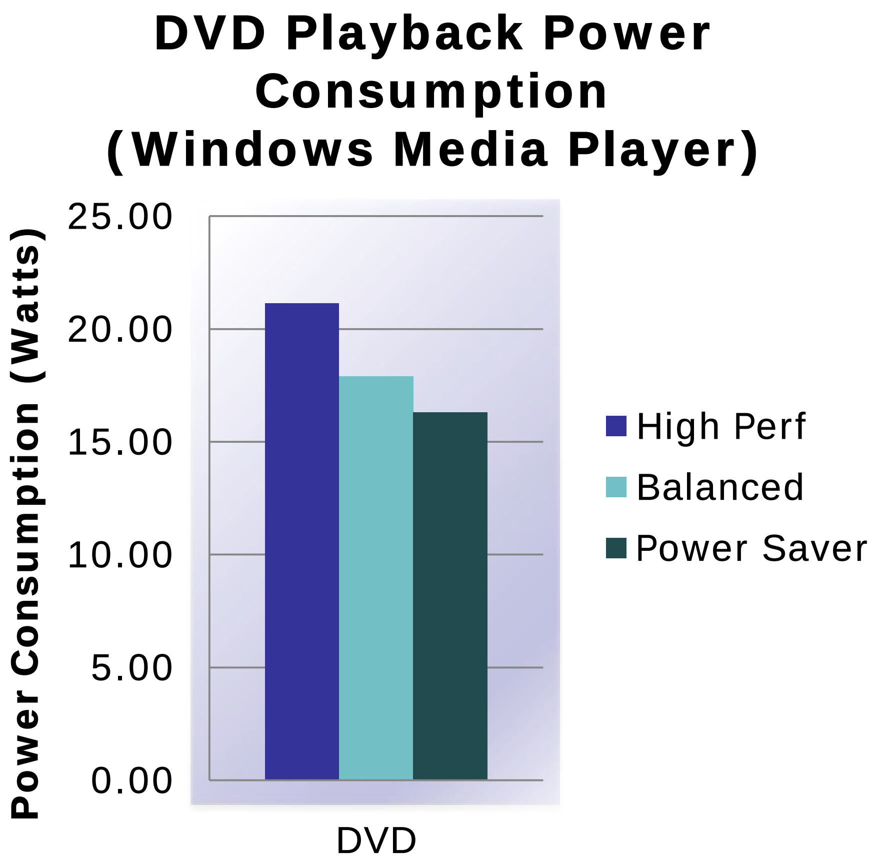
```
BOOL WINAPI SetWaitableTimerEx(
    __in     HANDLE hTimer,
    __in     const LARGE_INTEGER *lpDueTime,
    __in     LONG lPeriod,
    __in_opt PTIMERAPCROUTINE pfnCompletionRoutine,
    __in_opt LPVOID lpArgToCompletionRoutine,
    __in_opt PREASON_CONTEXT WakeContext,
    __in     ULONG TolerableDelay
    );
```

SetWaitableTimerEx() API Usage

```
void CreateAndSetPeriodicTimer()
{
   myTimer = CreateWaitableTimerEx(NULL,
                  TimerName,                   //string with chosen timer name
                  NULL,
                  TIMER_MODIFY_STATE);         //required security attribute to call
                                               //SetWaitableTimerEx

   bError = SetWaitableTimerEx(myTimer,
                  DueTime,             //UTC due time
                  10000,               //periodic timer duration is ten seconds
                  CompletionRoutinePointer,    //APC completion routine
                  ArgsToCompletionRoutine,     //completion routine arguments
                  WakeContext,         //only if waking the machine
                  1000);               //tolerable delay is one second

   //DO WORK

   bError = CancelWaitableTimer(myTimer);      //be sure to cancel periodic timers!
}
```

Bonus: Optimize for System State

- System state holds optimization potential for applications
  - Firewall application should do very little when the PC is not connected to network
- Register for power state change notifications
  - Use the callback to trigger behavioral changes



Source: Microsoft Corporation

IDF2010 INTEL DEVELOPER FORUM

RegisterPowerSettingNotification() API

- Allows you to register for change notifications on power settings
- Callback is a notification to change application behavior
  - Includes new power setting value

```
void MyApp::OnInit()
{
        hACDCSource = RegisterPowerSettingNotification(m_hWnd,
                                                       &GUID_ACDC_POWER_SOURCE,
                                                       DEVICE_NOTIFY_WINDOW_HANDLE);
}
void MyApp::OnDestroy()
{
        if (hACDCSource != 0)
                UnregisterPowerSettingNotification(hACDCSource);
}
```

Agenda

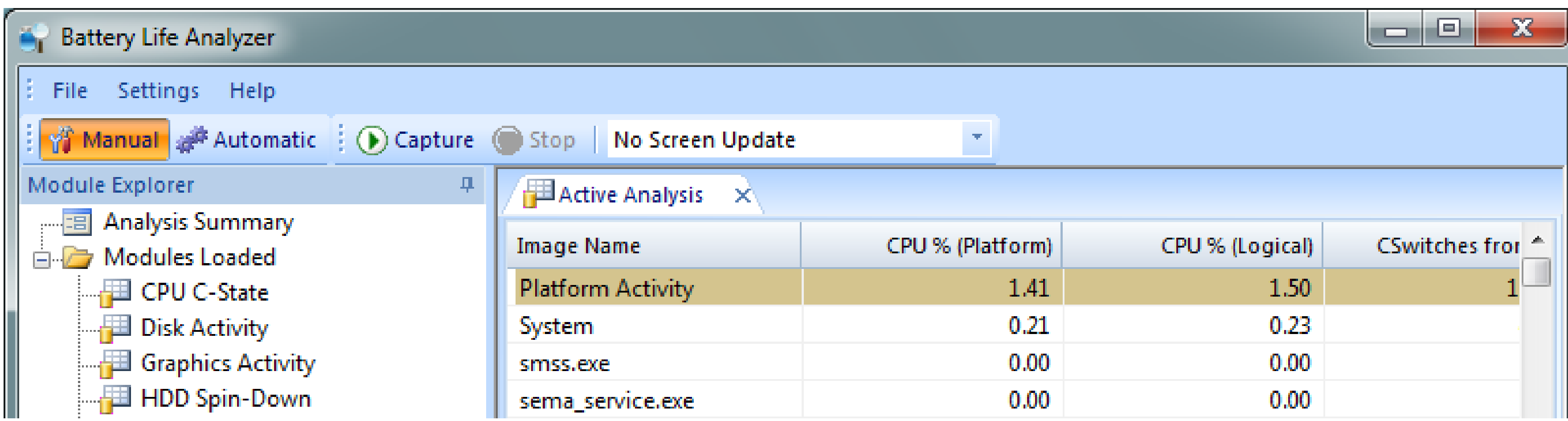
- Mobile platform energy efficiency goals
- Hardware considerations
- Software considerations
- Tools for idle power analysis

Choose the Right Tool

- Microsoft* Windows* “PowerCfg /energy”
  - OS Built-in command
  - Easy to use, suitable for identifying some common issues
- Microsoft* Windows* Performance Toolkit (xperf)
  - Built on ETW (Event Tracing for Windows) Technology
  - Good tool for deep performance / power analysis
- Battery Life Analyzer
  - New tool from Intel
  - Easy to use, suitable for identifying bad components
  - More detailed information on Intel mobile platforms
  - Built on ETW Technology

Battery Life Analyzer - Outline

- High level tool to identify battery life issues
  - Simple GUI application
  - In most cases, it takes only few mouse clicks
- Quantifies the impact of the issues
  - Where possible, power impact is estimated



Battery Life Analyzer - Features

- Hardware Analysis
  - CPU C-state residency
  - PCI Express*, SATA Link Power Management
  - USB selective suspend
- Software Analysis
  - Fine-grained CPU utilization information
  - Periodicity of the activity
  - Concurrency of multi-core activity
  - Graphics activity
  - Identify process causing HDD spin-ups

Battery Life Analyzer – CPU C-State

CPU C-State Example:

- High residency in the deepest C-state
- All software behaving well

CPU C-State

C-State Counters	Affinity Mask	Average (%)	Power Impact (W)
Package0 C0 - C1	0x0000000f	1.66	N/A (See SW Analysis)
Package0 C3	0x0000000f	1.94	0.021
Package0 C6	0x0000000f	96.40	0.000
Core0 C0 - C1	0x00000003	1.03	N/A (See SW Analysis)
Core0 C3	0x00000003	1.46	0.005
Core0 C6	0x00000003	97.52	0.000
Core1 C0 - C1	0x0000000c	1.06	N/A (See SW Analysis)
Core1 C3	0x0000000c	1.12	0.004
Core1 C6	0x0000000c	97.83	0.000

Battery Life Analyzer – SATA LPM

SATA Link Power Management

- Bad Example
  - DVD drive not entering Slumber state

SATA LPM

Controller/Drive Name	Port	% Active	% Partial	% Slumber	HIPM	DIPM	DIPM Enabled
Intel(R) 5 Series 6 Port SATA AHCI Controller (0/31/2)							
Fdfdsfdfjlkfkdlfd;fdjsfdfd	0	0.00	0.00	100.00	Y	Y	Y
Fdfdsfdfjlkfkdlfd;fdjsfdfd	2	0.00	100.00	0.00	Y	Y	Y

- Good Example
  - Same drive enters Slumber state after enabling host initiated Slumber in the registry

SATA LPM

Controller/Drive Name	Port	% Active	% Partial	% Slumber	HIPM	DIPM	DIPM Enabled
Intel(R) 5 Series 6 Port SATA AHCI Controller (0/31/2)							
Fdfdsfdfjlkfkdlfd;fdjsfdfd	0	0.00	0.00	100.00	Y	Y	Y
Fdfdsfdfjlkfkdlfd;fdjsfdfd	2	0.00	0.00	100.00	Y	Y	Y

Battery Life Analyzer – HDD Spin-up

Disk Activity Analysis

- Identify which process/routine is causing HDD spin-up

Disk Activity

Spin-up Time	Process	PID	TID	File	I/O
4:04:36.892 AM 7/27/2010	System	4	60	C:\Windows\System32\config\SYSTEM	FileIoFlush
4:14:25.552 AM 7/27/2010	taskhost.exe	1672	1944	C:\ProgramData\Microsoft\RAC\Temp\s...	FileIoFlush
4:46:34.082 AM 7/27/2010	explorer.exe	1984	2176	C:\Users\Administrator\AppData\Roami...	FileIoCreate
4:54:37.456 AM 7/27/2010	svchost.exe	824	2476	C:\Windows\CSC\v2.0.6\namespace	FileIoCreate
5:07:44.448 AM 7/27/2010	svchost.exe	792	1352	C:\Windows\system32\gpsvc.dll	FileIoCreate
5:42:04.019 AM 7/27/2010	svchost.exe	824	1640	C:\Windows\CSC\v2.0.6\namespace	FileIoCreate
5:49:37.395 AM 7/27/2010	svchost.exe	824	592	C:\Windows\CSC\v2.0.6\namespace	FileIoCreate

Stack Trace	
C:\Windows\system32\drivers\fileinfo.sys+0x000000000000adbc	
C:\Windows\system32\drivers\fileinfo.sys+0x0000000000007fff	
C:\Windows\system32\drivers\fltmgr.sys::FltReleasePushLock()	
C:\Windows\system32\drivers\fltmgr.sys::FltIsCallbackDataDirty()	
C:\Windows\system32\drivers\fltmgr.sys+0x00000000000016c7	
C:\Windows\system32\ntoskrnl.exe::IoReportHalResourceUsage()	

Summary

- Your product has a direct impact to the battery life of mobile platforms
- One bad product can ruin the customer experience
- Start looking at the impact to idle power
- Tools and more information are available from Intel and Microsoft

Call to Action

- Get tools
  - "PowerCfg /energy"
    Windows* 7 built in command
  - Microsoft* Windows* Performance Toolkit (xperf)
    Now included in Windows 7 SDK
  - Battery Life Analyzer
    Send e-mail to: BatteryLifeAnalyzer@intel.com
- Analyze your product and identify the issue
- Improve power efficiency of your product

Additional sources of information on this topic:

- Other Sessions
  - EBLS001: Interconnect Bus Extensions for Energy-Efficient Platforms
  - EBLS002: Impact of “Idle” Software on Battery Life
- White papers

  http://www.intel.com/technology/mobility/notebooks.htm
  - Designing Energy Efficient SATA Devices
  - Making USB a More Energy-Efficient Interconnect
  - Energy-Efficient Platforms – Designing Devices Using the New Power management Extensions for Interconnects
  - Energy-Efficient Platforms – Considerations for Application Software and Service

  http://www.microsoft.com/whdc/system/pnppwr/default.mspx
  - Mobile Battery Life Solutions Guide for Windows 7
  - Developing Efficient Background Processes for Windows
  - Using PowerCfg to Evaluate System Energy Efficiency
  - Windows Timer Coalescing

Legal Disclaimer

Risk Factors

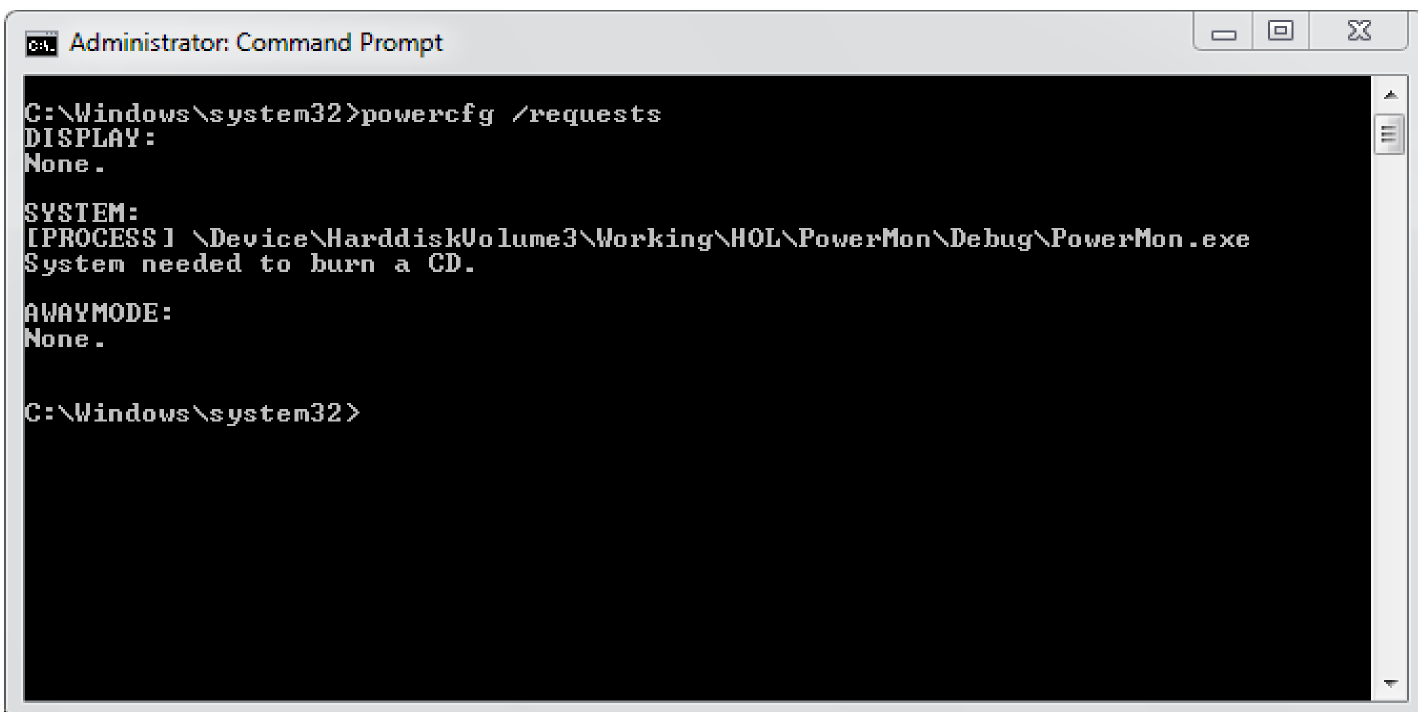
The above statements and any others in this document that refer to plans and expectations for the second quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Many factors could affect Intel's actual results, and variances from Intel's current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the corporation's expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel's and competitors' products; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Additionally, Intel is in the process of transitioning to its next generation of products on 32nm process technology, and there could be execution issues associated with these changes, including product defects and errata along with lower than anticipated manufacturing yields. Revenue and the gross margin percentage are affected by the timing of new Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel's response to such actions; defects or disruptions in the supply of materials or resources; and Intel's ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; start-up costs, including costs associated with the new 32nm process technology; variations in inventory valuation, including variations related to the timing of qualifying products for sale; excess or obsolete inventory; manufacturing yields; changes in unit costs; impairments of long-lived assets, including manufacturing, assembly/test and intangible assets; the timing and execution of the manufacturing ramp and associated costs; and capacity utilization. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. The majority of our non-marketable equity investment portfolio balance is concentrated in the flash memory market segment, and declines in this market segment or changes in management's plans with respect to our investment in this market segment could result in significant impairment charges, impacting restructuring charges as well as gains/losses on equity investments and interest and other. Intel's results could be impacted by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel's results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting us from manufacturing or selling one or more products, precluding particular business practices, impacting our ability to design our products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel's results is included in Intel's SEC filings, including the report on Form 10-Q for the quarter ended March 27, 2010.

Rev. 5/7/10

Backup Slides

PowerSetRequest() API

- Replaces setthreadexecutionstate()
- Allows you to issue availability requests for monitor & system
- Allows you to create a custom, localized reason string
- Does not prevent user-initiated sleep transitions



PowerSetRequest() API Usage

```
void KeepSystemAwake()
{
    // This example uses a simple, non-localized availablity request diagnostic
    string
    POWER_REQUEST_CONTEXT SimpleRqContext;
    SimpleRqContext.Version = POWER_REQUEST_CONTEXT_VERSION;
    SimpleRqContext.Flags = POWER_REQUEST_CONTEXT_SIMPLE_STRING;
    SimpleRqContext.Reason.SimpleReasonString = L“System needed to burn a CD.";

    HANDLE SimplePowerRequest = PowerCreateRequest(&SimpleRqContext);

    // Set a system request to prevent automatic sleep
    PowerSetRequest(SimplePowerRequest,PowerRequestSystemRequired);

    //
    // Do work here...
    //

    // Clear the request
    PowerClearRequest(SimplePowerRequest,PowerRequestSystemRequired);
}
```