您的潜力，我们的动力

Microsoft
微软（中国）有限公司

Windows Embedded从入门到精通系列课程

Windows CE电源管理入门篇

Sunny Cheng 程胜辅
Senior Technical Manager
Intrinsyc Software International Inc.
scheng@intrinsyc.com

msdn

MSDN Webcasts

Who Is Intrinsyc

您的潜力，我们的动力
Microsoft
微软（中国）有限公司

- Over 11 years experience in wireless, handheld and embedded device development (Founded in 1996)
- Located in Vancouver BC (HQ), Seattle WA, Cupertino, CA and Taipei TW
- System Integration for Tier 1 OEMs with 200+ engineering services projects
- Strong expertise in: Windows Mobile/CE, Mobile Linux, and Symbian
- 225 employees with 75% in development or engineering
- 2 business units offering both Services and Software
  - The Services unit provides engineering services
  - The Software unit developed our Windows CE based Soleus™ platform for consumer mobile devices
- Three-time winner of Microsoft Windows Embedded Partner of the Year (2002 x 2, 2007)
  - Microsoft MEDC 2007 - System Integrator Excellence Award
- Gold Windows Embedded CE Partner. Over 10 years experience on the Windows Embedded CE platform (starting with Windows CE 1.0)

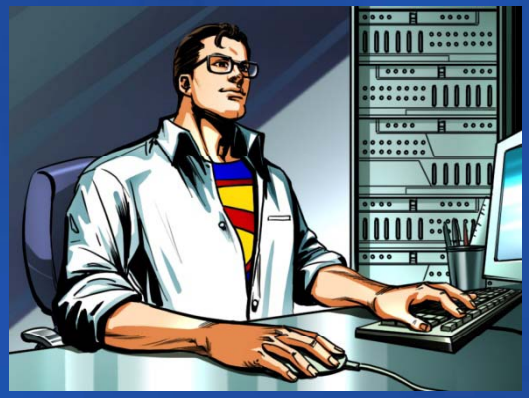
INTRINSYC®

Windows Embedded Gold Partner 2007 Excellence Awards Systems Integrator
Microsoft GOLD CERTIFIED Partner
Windows Embedded Gold Partner
ARM freescale semiconductor MARVELL SAMSUNG OMAP TEXAS INSTRUMENTS TECHNOLOGY Independent Technology Center symbian

msdn
Windows Mobile
MSDN Webcasts

Who Am I

- 1988年，西安电子科技大学计算机工程专业学士学位
- 1991年，航天部第三研究院工程硕士学位
- 航天部第三研究院，导弹制导系统开发
- TJNEC公司，NEC程控交换机软件开发，后期任职软件开发经理
- AirPatrol公司，任职资深软件工程师，专攻WiFi客户管理技术，独立开发了pocketWiNc，AirSafe Service, 和AirSafe Personal等产品。
- 加入Intrinsyc总部工作后，先任职资深软件工程师。主要工作领域
  - 电源管理
  - Bootloader
  - NAND, OneNAND技术
  - USB技术
- 现任职Intrinsyc总部资深技术经理，负责公司全球范围的技术方案制定，技术推广，技术支持等



微软中文技术论坛

——精彩生活每一天

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

http://forums.microsoft.com/china

本周活动更新：

- ✓ **Top 10 论坛英雄！**
- ✓ 畅谈我的2007

与众不同：

- ✓ **版主：50+ 微软最有价值专家（MVP）**
- ✓ **涵盖微软几乎所有产品线和知识库**
- ✓ **30+ 适合开发人员和 IT 专业人员技术板块**

City Event Sponsor Program
相聚同城技术培训

微软.NET俱乐部
.NET User Group

MVP Microsoft Most Valuable Professional

msdn

MSDN Webcasts

本次课程内容包括

- Windows CE Power Management
- Windows CE Power Management APIs
- Windows Mobile Power Management
- Application Power Management Consideration

收听本次课程需具备的条件

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

- Understand Windows CE device driver development concept
- Have knowledge of one or more embedded application processors, ARM or Xscale
- Have knowledge of Power Management concept

Level 200

内容

- Windows CE Power Management
  - What is Power Management
  - CPU Power State
  - System Power State
  - Device Power State
  - Power Manager
  - Windows CE PM Architecture
  - System Power States Transition
  - System-Device Power States Map
- Windows CE Power Management APIs
- Windows Mobile Power Management
- Application Power Management Consideration

What is Power Management?

- Power management involves getting more entertainment time out of the devices by optimizing battery life and reducing power consumption

decreased power consumption

longer battery life

more time to talk, play, listen

How to Achieve?

- Manage power consumption to extend the standby time
  - Windows CE Power Management
- Manage power consumption during specific device activities (under a load)
  - Use a DVFS solution

CPU Power State

- **Run**
  - All internal power domains and external power supplies are fully powered and functional. The processor clocks are running
- **Idle (Deep-idle)**
  - Clocks to the CPU Core are disabled
  - Peripherals in full speed
  - recovery is through interrupt assertion
  - Called Deep-Idle, when CPU core was running in 13M Mode
- **Standby**
  - Core and all peripherals stopped in low leakage state with context retained
  - SRAM retention is optional
  - No LCD operation
  - Recovery is through external and selected internal wake-up events to Run or Idle
- **Sleep**
  - Low voltage power supplies for core & peripherals are powered-off
  - I/O remains powered
  - Power Manager, OS Timer and RTC operate from I/O power supply
  - SRAM retention is optional
- **Deep-sleep**
  - Only RTC, Oscillators and related I/O pins are powered-on
  - RTC powered from backup battery input (e.g. coin-cell)
  - OS timer and PWRI2C may retain state or be powered-off

Note: Marvell PXA270 as an example

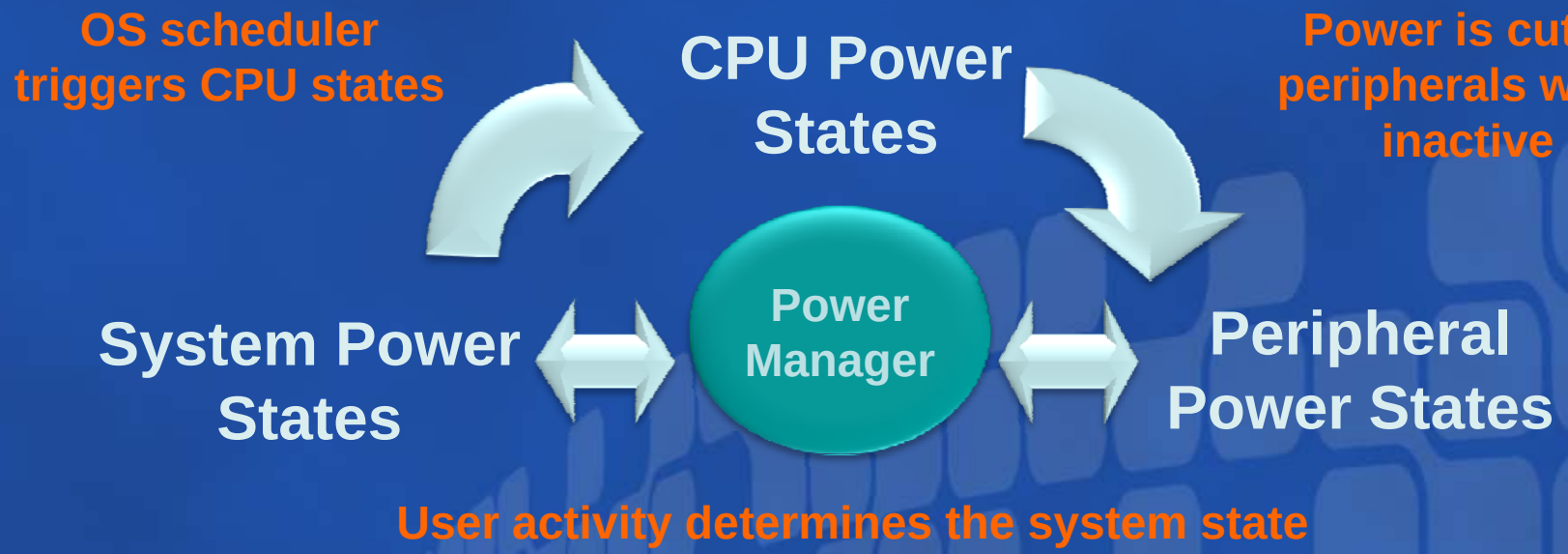
System Power State

- System power states vary by operating system implementation (CE vs. WM)
- Windows Embedded CE sample PM implementation
  - **On**
    - Mobile Device is on and is actively using
  - **UserIdle**
    - Mobile Device is using but not actively interacting with it
  - **SystemIdle**
    - Mobile Device is not directly using but processes are still active (display, audio, & touch drivers are off; CPU on, some device drivers are on)
  - **Suspend**
    - Mobile Device is asleep (RAM self-refreshed, CPU asleep, all device drivers are inactive)
  - **Off**
    - Mobile Device is completely off, no power consumed. (Not a real state)

Device Power State

- **Full On (D0)**
  - State in which the device is on and running. It is receiving full power from the system and is delivering full functionality to the user.
- **Low On (D1)**
  - Functional at a lower power or performance state than D0. D1 is applicable when the device is being used but where peak performance is unnecessary and power is at a premium.
- **Standby (D2)**
  - State in which the device is partially powered with automatic wakeup on request. A device in state D2 is effectively standing by.
- **Sleep (D3)**
  - State in which the device is partially powered with device-initiated wakeup if available. A device in state D3 is sleeping but capable of raising the System Power State on its own. It consumes only enough power to be able to do so; which must be less than or equal to the amount of power used in state D2.
- **Off (D4)**
  - State in which the device has no power. A device in state D4 should not be consuming any significant power. Some peripheral busses require static terminations that intrinsically use non-zero power when a device is physically connected to the bus; a device on such a bus can still support D4.

Optimizing Standby Time



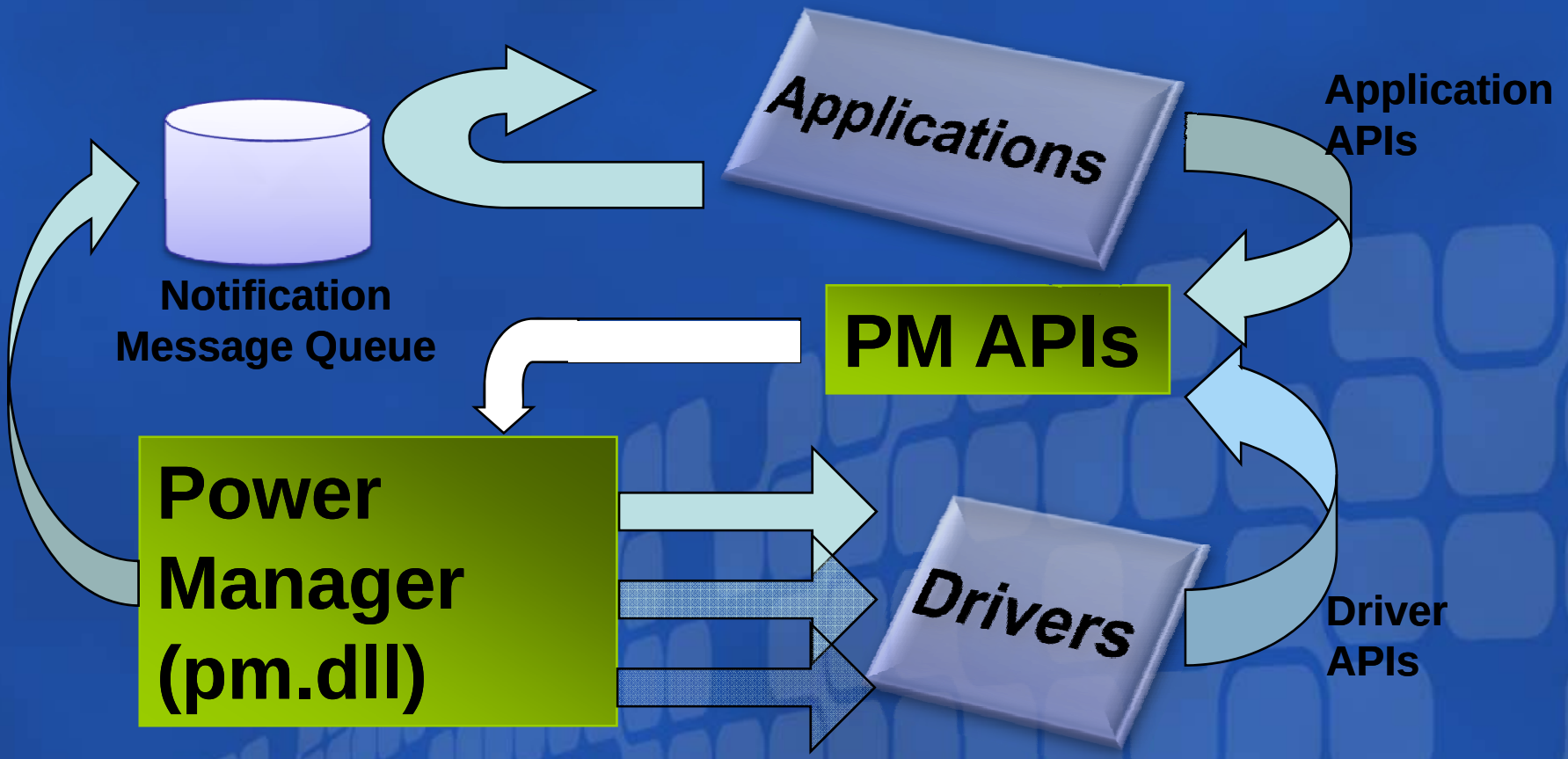
Power Manager

- Introduced in Windows CE 3.0 – PM.DLL
- Why? More system power states and more complicated device power behaviors
- Power Manager puts system into different power states based on predefined state machine
- Power Manager broadcasts system power state changes to other modules (devices, apps, etc.)
- Power Manager maps device power states into system power states

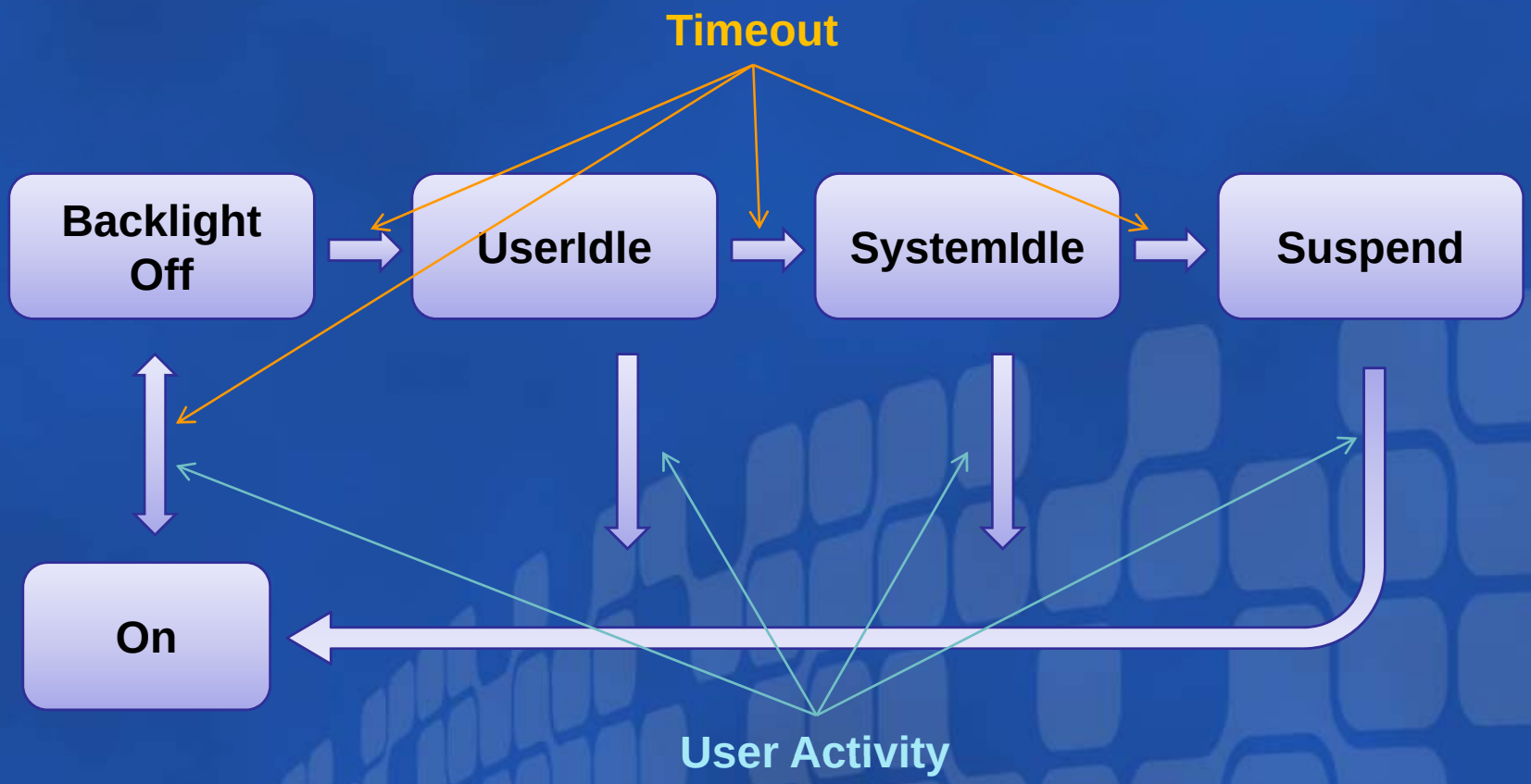
Power Manager Samples

- The minimum Power Manager
  - Enabled via SYSGEN_PMSTUBS
  - Stubs out all of the power management API (SetSystemPowerState et. al.)
  - Only supports suspend/resume via OEMPowerOff
- The full Power Manager
  - Enabled via SYSGEN_PM
  - API linked directly into DEVICE.EXE calls functions found in PM.DLL
  - Provides a library (DLL) for active (or self) power management or passive management
- Two reference Power Managers
  - \WINCE600\PUBLIC\COMMON\OAK\DRIVERS\PM\PDD\DEFAULT
    - Provides active and passive power management
    - Active management based on activity (or inactivity timers)
  - \WINCE600\PUBLIC\COMMON\OAK\DRIVERS\PM\PDD\PDA
    - The Power Manager found in Windows Mobile devices

CE PM Architecture



System States Transition (an Example)



Timeout Settings

- Drive the System States transition
- [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\Timeouts]
  - "ACUserIdle" = dword:3c; in seconds
  - "ACSystemIdle" = dword:12c; in seconds
  - "ACSuspend" = dword:0; in seconds
  - "BattUserIdle" = dword:3c; in seconds
  - "BattSystemIdle" = dword:b4; in seconds
  - "BattSuspend" = dword:12c; in seconds
  - "BatteryPoll" = dword:1f4; battery polling interval, in milliseconds
- “0” disables the timeout
- OEMs can adjust the value

System-Device Power States Map

- A States Map Example (Windows Mobile)

Device Drivers	States						
	On	Backlight Off	Screen Off	Unattended	Suspend	Resuming	Off
Backlight	D0	D4	D4	D4	D4	D4	D4
Display	D0	D0	D4	D4	D4	D4	D4
Audio	D0	D0	D0	D4	D4	D4	D4

- Drivers have to decide the most suitable state based on applications request (lowest to keep the performance) and the device power state (highest allowed)
- For example: BT driver should be in D3 or D4 state (and external hardware in low power condition) regardless of system state until BT application requests it to be on

内容

- Windows CE Power Management
- Windows CE PM APIs
  - Kernel-level APIs
  - Device-level APIs
  - Application APIs
- Windows Mobile PM
- Application PM Consideration

Kernel-level APIs

- Required kernel functions:
  - OEMIdle
    - Called when no threads are scheduled to run
  - OEMPowerOff
    - Called when suspending or turn off the system
- Optional kernel functions:
  - OALUpdateRescheduleTime
    - Called to update the system tick interval

OEMPowerOff

- Suspend the device
  - Responsible for placing the system in the lowest power state during suspend
    - CPU dependent
      - DRAM in self-refresh
      - CPU stopped
      - And etc.
  - Interrupts disabled (except wakeup sources)
  - After a wakeup interrupt, system state restored
- Or, power off the device

Suspend Sequence

- PM disables all PM-aware non-block drivers
- PM calls IOCTL_HAL_PRESUSPEND
- PM notifies file system
- PM disables all PM-aware block drivers
- System goes single threaded
- PM calls all legacy xxx_PowerDown callbacks
- No system calls in xxx_PowerDown
- These routines make drivers non-pageable
- PM calls OEMPowerOff
- OEM code that parks the CPU

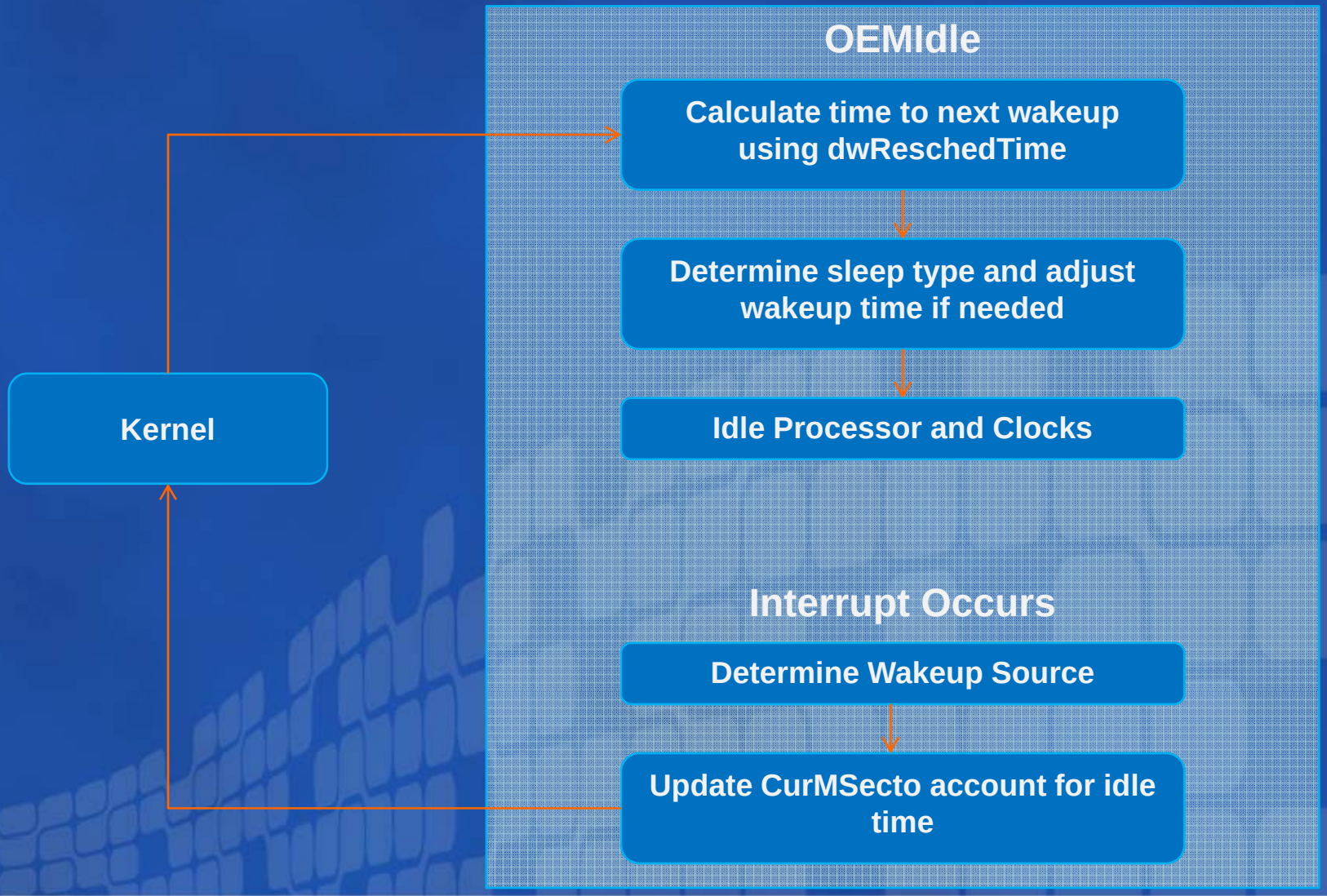
Resume Sequence

- Interrupt wakes up CPU to “boot address”
- Bootloader checks the wakeup flag
- Restore the context
- Continues executing where left off (OEMPowerOff)
- OEMPowerOff returns
- PM calls all legacy xxx_PowerUp routines
- System goes multi-threaded
- PM enables PM-aware block drivers
- PM notifies file system
- PM enables PM-aware normal drivers

OEMIdle

- Called by the scheduler when there are no threads ready to run. (Blocked waiting for input or event.)
- Responsible for placing the CPU in a low power state that can be rapidly exited
- Not possible to place DRAM in self refresh
- It’s a rest (not sleep), consider which CPU power state(s) to use

OEMIdle Execution Sequence



OEMIdle Example


```
void OEMIdle(DWORD dwIdleParam)
{
    // Slow down system clock
    SetSlowClock();
    // Stop the CPU clock
    *((DWORD *)SCDR) = PCK;
    // Fast clock resumed by ISR
}
```

c:\wince600\platform\common\src\common\timer\idle\idle.c

The Variable Tick Scheduler

- Scheduler calls OALUpdateReschedTime to change the system tick interval based on the next scheduled thread time
- In theory the tick ISR will never return SYSINTR_NOP
- Reduces the average power consumption because the CPU clock is stopped for longer periods of time
- Will be discussed in detail in the Advanced Course

Device Power Management

- Device Driver Power Management Tasks
  - Report power capabilities
  - Handle power requests
  - Power up the device
  - Power down the device
  - Manage device wake up events

What Is A “Device” to PM

- A “device” is an instance of a driver
  - A single DLL can be used multiple times for multiple instances of the same hardware (ex. serial port)
  - Physical drivers: Backlight, Bluetooth, Keyboard
  - Logical drivers: Filters (GPS), Converters
- All devices in the system are managed by the Device Manager (DM)
- The PM works in tandem with the DM, so it is able to determine
  - What interfaces a device supports
  - Devices entering/leaving the system
  - What power “class” a device belongs to

Understand Device Power State

- Driver is responsible for mapping state (Dx) to peripheral HW capabilities and performing applicable state transition on its physical device
- Physical devices do not have to support all of the device power states
- The only device states that all devices must support is Full On (D0)
- PM maps system power states to corresponding device power states
- Drivers must aggressively self-manage power

Group Devices by Interface Class

- The “IClass” registry key notifies an external agent, in this case PM.DLL, that a particular interface or API is available
- There are four standard IClass GUIDs defined in PM.H
  - #define PMCLASS_GENERIC_DEVICE TEXT("{A32942B7-920C-486b-B0E6-92A702A99B35}")
  - #define PMCLASS_NDIS_MINIPORT TEXT("{98C5250D-C29A-4985-AE5F-AFE5367E5006}")
  - #define PMCLASS_BLOCK_DEVICE TEXT("{8DD679CE-8AB4-43c8-A14A-EA4963FAA715}")
  - #define PMCLASS_DISPLAY TEXT("{EB91C7C9-8BF6-4a2d-9AB8-69724EED97D1}")

System-Device State Map Example

- Directly map to all devices or a particular one
  - [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\State\SystemIdle]
  - "Default"=dword:2 ; D2 (All devices)
  - “bkl1:”=dword:4 ; D4 (Backlight driver only)
  - "Flags"=dword:100000 ; POWER_STATE_IDLE
- Map to all the devices in an interface class
  - [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\State\SystemIdle\{EB91C7C9-8BF6-4a2d-9AB8-69724EED97D1}]
  - "Default"=dword:4 ; D4
- {EB91C7C9-8BF6-4a2d-9AB8-69724EED97D1} means Power-manageable display

Device-level APIs

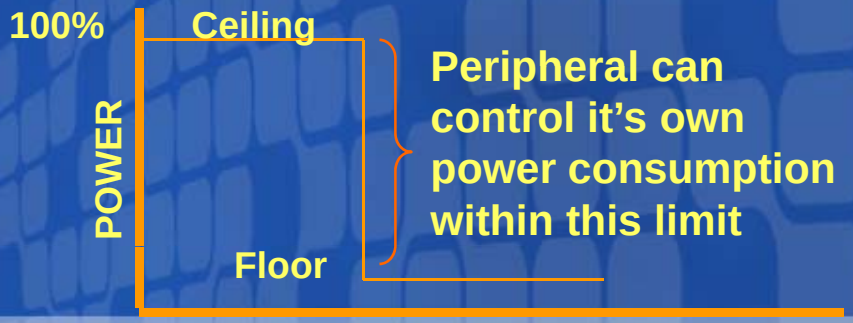
- The Power Manager uses IOCTL codes to communicate with a managed device driver
- The device driver calls DevicePowerNotify to request that Power Manager adjusts the power state of a device.

Device IOCTLs

- IOCTL_POWER_CAPABILITIES
  - Requests that the device inform Power Manager of supported power states, along with any associated characteristics
- IOCTL_POWER_QUERY
  - Checks whether the device is ready to enter a new device power state
- IOCTL_POWER_SET
  - Requests that the device update the power state information
- IOCTL_POWER_GET
  - Requests that the device inform Power Manager of the current device power state
- IOCTL_REGISTER_POWER_RELATIONSHIP
  - Notifies the parent device to register all controlled devices

Device Self Management

- Power Manager manages device drivers based on the system-to-device power states map
- OEMs define and optimize the map
- Power **ceiling** is the maximum power consumption limit determined by device power state
- Power **floor** is the minimum power requirement necessary to achieve the work. It is set by application or other peripheral.
- Power management allows peripherals to intelligently manage their own power as long as the system's power levels remain in the range between the **floor** and **ceiling**



Add PM to a Device Driver

- Make sure your device driver support stream interface
  - The stream interface functions enable other software modules to interact with peripherals as if they were files
- Add support for the power management I/O control codes (IOCTLs) to your stream interface driver.
- Notify Power Manager that your driver has power management
- Implement device power states in your driver
- Build your driver
- Debug your driver

Outside of Power Manager

- XXX_PowerUp/Down
  - Called by the Device Manager (independent of Power Manager) to suspend and resume a driver

Application APIs

- Power Manager API
  - Used to set or query the system or device power states (passive power management)
- Power notification API
  - Used to receive notification of power events. Power events are the result of active power management either by the Power Manager or by the device drivers themselves

Power Manager API

- GetSystemPowerState
  - Returns the name of the current system power state
- SetSystemPowerState
  - Requests that the Power Manager change the current system power state
- SetPowerRequirement
  - Requests that the Power Manager maintain a given device's power state at a minimum level
- ReleasePowerRequirement
  - Informs the Power Manager that it is no longer necessary to maintain a device's minimum power state as described in a previous call to SetPowerRequirement
- GetDevicePower
  - Returns a given device's current power state
- SetDevicePower
  - Request that the Power Manager change a given device's power state themselves

Power Notification API

- RequestPowerNotifications
  - Requests that the Power Manager send notification of power events
- StopPowerNotifications
  - Cancels a notification request made through RequestPowerNotifications

Power Notifications

- PBT_RESUME
  - Generated when the system resumes from a suspend state
- PBT_POWERSTATUSCHANGE
  - Generated when the system transitions to or from AC power
- PBT_TRANSITION
  - Generated when the Power Manager is performing a system power state change
- PBT_POWERINFOCHANGE
  - Generated when battery information or AC line status changes

内容

- Windows CE PM Architecture
- Windows CE PM APIs
- Windows Mobile PM
  - Power Manager is not changeable
  - Suspend/Resume Model
  - Always on Model
- Application PM Consideration

Windows Mobile PM

- Power Manager is not changeable
  - Even if you changed the PM code and built a new PM.DLL, the final PM.DLL in the flash image will be replaced by the pre-built PM.DLL
  - The system power state is fixed
    - Standby/Resume Model
    - “Always-on” Model

Standby/Resume Model

- For touch device (PPC, WM Professional)
- System power states
  - **On**
    - Normal operation state with user interactions
  - **BacklightOff**
    - Short period of inactivity, backlight is turned off (others can be as well)
  - **UserIdle**
    - Longer period of inactivity, all components in lower power usage mode
  - **ScreenOff**
    - Application requests the display be turned off (example: playing audio)
  - **Unattended**
    - Application requests it run without user knowledge
  - **Suspend**
    - Sleep state; No threads running; CPU idle; Can only wake up from a hardware wake source interrupt
  - **Resuming**
    - Initial state after wakeup from “Suspend”, for ignoring false wakeups

The “Always-on” Model

- For non-touch device (Smartphone, WM Standard)
- Simpler than the Suspend/Resume model
- System power states
  - **On**
    - Normal operation state with user interactions
  - **BacklightOff**
    - Short period of inactivity, backlight is turned off (others can be as well)
  - **UserIdle**
    - Longer period of inactivity, all components in lower power usage mode
  - **ScreenOff**
    - Application requests the display be turned off (example: playing audio)
  - **Off**
    - Device is completely powered off

Which One Is Better?

- Drop to the deepest possible power state (Suspend/Resume Model)
  - Greater latency for enter and exit of power state (usually several hundred milliseconds)
  - Device appears to be off. No software is running. Upon resume, timers continue where they left off.
- Drop to low power state with small transition latencies (“Always-on” Model)
  - Works better with background real-time communications (networks, VoIP, IM, WLAN, BT, …)
  - With frequent wakeups, this can provide better savings
- **“Always-on” Model is the WINNER**
  - **New processor technologies provide better CPU Idle performance**
  - **New H/W platform designs provide better power domain management**
  - **More aggressive device driver power management techniques available**

内容

- Windows CE PM Architecture
- Windows CE PM APIs
- Windows Mobile PM
- Application PM Consideration

Application PM Consideration

- Do I need to alter system or device power state?
  - Applications set the minimum power requirements to CPU and device drivers. For example, media player:
    - Requests audio, backlight and display to be on D0 for video playback
    - Requests only audio to be on D0 for audio playback
- Do I have to react when power state changes?
  - Register to receive power notification events, if it's necessary to do some special tasks due to power state changing
- Single poorly written application can ruin a great device
  - Using 1% of the CPU processing time and continuously running
  - Standby time will be reduced to 10%. (200h to 20h)

PM-Aware Applications

- Run the CPU as little as possible.
  - Low duty cycle is critical
- Consider wakeup overhead
  - Better to wake up 10 times/sec for 25 mS each than to wake up 100 times/sec for 2.5 mS each
- If not in the foreground, don’t use any CPU
  - General rule with some exceptions
    - Good for video playback
    - But not applicable to audio playback
- Use interrupts and event-driven wherever possible
  - Never polling
- Re-test whenever adding or updating 3rd party software or applications

CE电源管理开发资源

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

- **MSDN Power Manager Reference and APIs**

  http://msdn2.microsoft.com/en-us/library/aa923906.aspx

- **Samples**

  ..\public\common\oak\files\common.reg

  ..\public\common\oak\drivers\block\atapi\atapipm.cpp

- **Windows Mobile Team Blog**

  “Power To The Smartphone”

  http://blogs.msdn.com/windowsmobile/archive/2006/08/04/689069.aspx

  “Power To The PocketPC”

  “Power To The System”

  “Power To The People”

  “Power To The Developers part 1”

  “Power To The Developers part 2”

嵌入式开发资源

- **Windows Embedded**中文官方网站
  http://www.microsoft.com/china/windows/embedded
- **.NET Micro Framework**
  http://msdn2.microsoft.com/zh-cn/embedded/bb267253.aspx
- **Microsoft Robotics Studio**
  http://msdn2.microsoft.com/zh-cn/robotics/default.aspx
- 微软嵌入式开发者论坛
  http://forums.microsoft.com/china/default.aspx?ForumGroupID=493&SiteID=15
- 微软中国嵌入式开发者博客
  http://blogs.msdn.com/yunxu/
- **Mike Hall**的博客
  http://msdn2.microsoft.com/zh-cn/embedded/ Aa731228.aspx

微软启动新一轮“免费重考计划”

您的潜力，我们的动力

Microsoft 微软（中国）有限公司

- 作为对微软认证学习者的支持与鼓励，微软公司于2007年9月 15日启动新一轮免费重考计划。
- 如果您以前曾参与过该项计划，那您或许已经了解到该计划将为您顺利通过微软认证考试带来有效的保障。在计划推行期内，如果您在考试前注册获得免费重考考试券，那当您首次考试未通过时，您将获得一次免费重考的机会。

微软启动新一轮“免费重考计划”

您的潜力，我们的动力
Microsoft
微软（中国）有限公司

- 有关活动详情，请访问：

http://www.microsoft.com/china/msdn/events/featureevents/2007/Secondshot.aspx

- 赶快规划您的认证学习计划吧，现在注册即可获得免费重考考试券！

获取更多MSDN资源

您的潜力，我们的动力

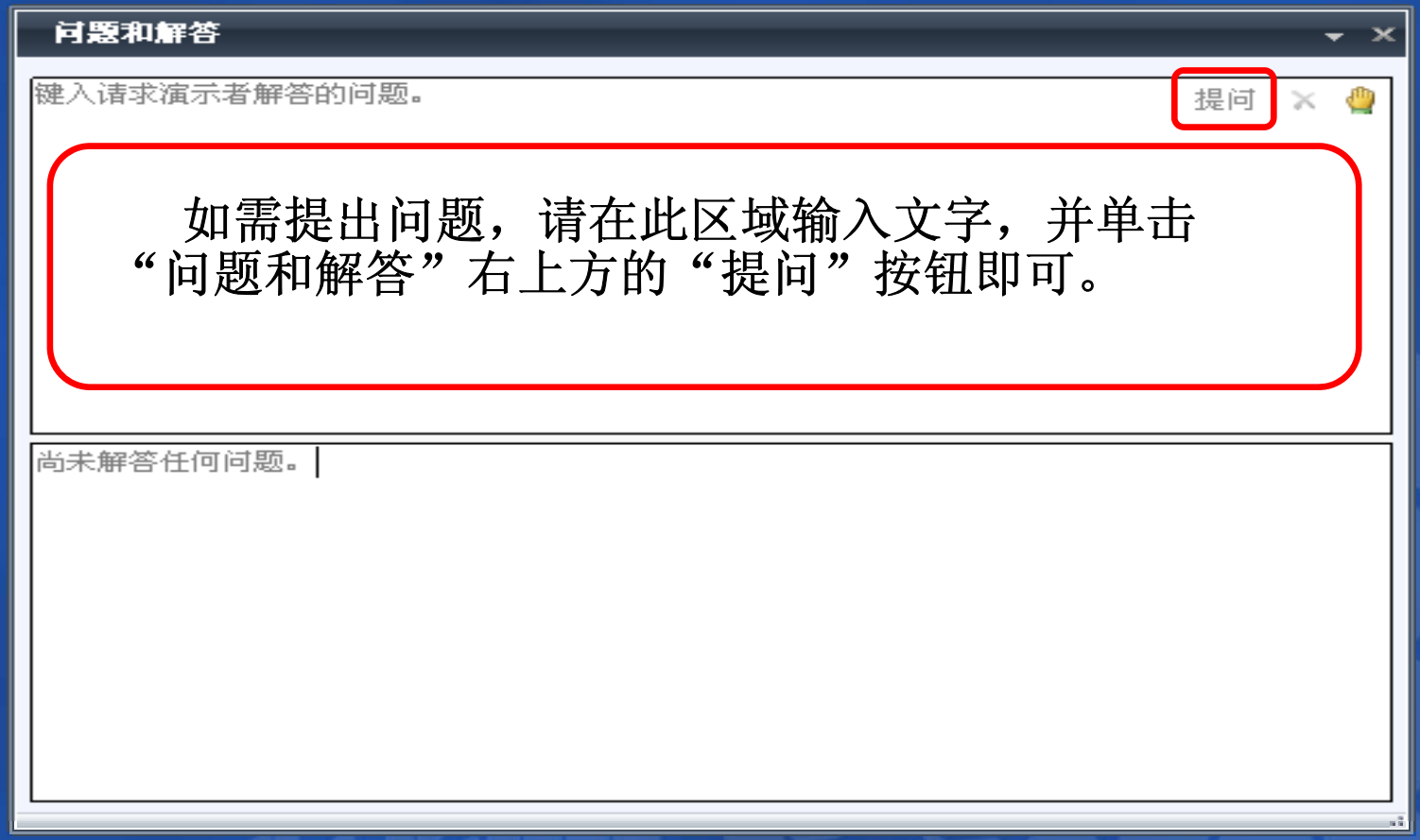
Microsoft
微软（中国）有限公司

- **MSDN中文网站**
  http://msdn2.microsoft.com/zh-cn
- **MSDN中文网络广播**
  http:// www.microsoft.com/china/msdn/webcast
- **MSDN免费中文速递邮件 (MSDN Flash)**
  http://msdn2.microsoft.com/zh-cn/flash
- **MSDN开发中心**
  http://msdn2.microsoft.com/zh-cn/developercenters
- MSDN图书中心
  http://www.microsoft.com/china/msdn/book

Question & Answer

您的潜力，我们的动力

Microsoft
微软(中国)有限公司



您也可以选择在微软中文技术论坛上寻求帮助，**MSDN**中文网络广播的讲师们会定期在论坛上为大家解答与课程相关的技术问题。

http://forums.microsoft.com/china

msdn

MSDN Webcasts


您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts