Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

99 | Worldwide Developers Conference

99 | Worldwide Developers Conference

Power Manager 2.0 for Mac OS 8

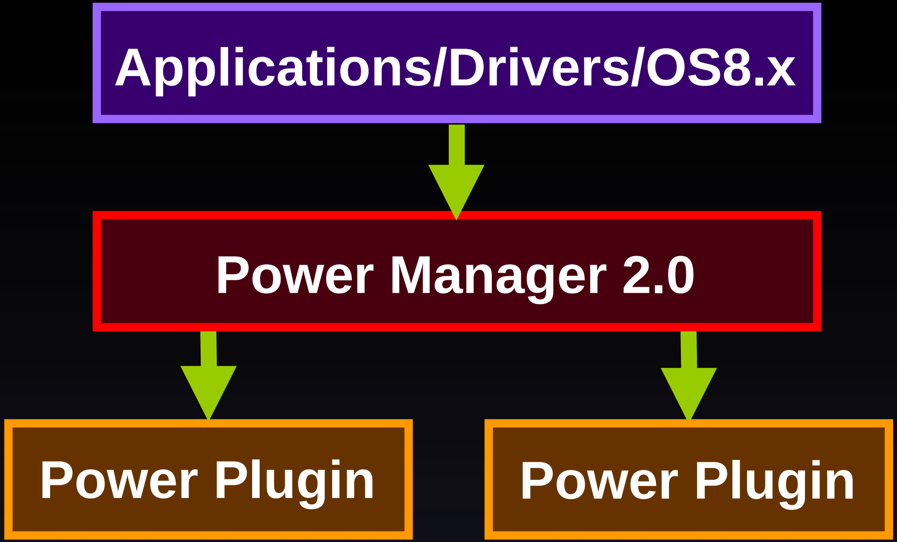
Scott A. Johnson

Software Engineer, CPU SW

Goals

- More aggressive power management
- Support new hardware features
- Support all hardware
- Support NewWorld ROM architecture

Overview


Applications/Drivers/OS8.x
Power Manager 2.0
Power Plugin
Power Plugin

Features

- PCI bus powered off during sleep
- Power source API
- Scheduled power events
- Processor temperature reporting
- And more!

Discovery

- Software must weak link against PowerMgrLib and check that routine symbols are defined (not equal to kUnresolvedCFragSymbolAddress)
- Use PMFeatures
- 68K-based code should check for gestaltPowerMgrVers >= 0x0200

Aggressiveness

- Mac OS 8.6 idles more aggressively to conserve power
- Software should:
  - Avoid spin loops/polling—use:

```
OSStatus PBWaitIOComplete( ParmBlkPtr ioPB,
                           Duration timeOut );
```

  - Use a non-zero sleep time for WaitNextEvent
  - Reduce use of VBLs and TM Tasks

PCI Bus Power Control

- Power will be removed from PCI slots during sleep in future hardware
- Device State must be saved and restored
    - Power Mgr will save and restore first 64 bytes of PCI configuration space
    - Drivers must save and restore device-specific state
- Revised drivers are required

PCI Characterization

- Power Mgr scans for power aware devices
- Gathers device power capabilities
  - *Does driver support PCI Power Off?*
  - *Does device have standby power requirements?*
- Registers Power Handlers in a prioritized DeviceSleepQueue

PCI Driver Changes

- Drivers must change in two ways:
- Export a new data structure, TheDriverPowerCapabilities, to indicate that the device is power management-aware and to describe power capabilities
- Implement a power handler to respond to power management requests

Power Capabilities

```
struct DriverPowerCapabilities {
    PowerCapsVersion            powerCapsVersion;
    PowerCapsFlags              powerCapsFlags;
    UInt32                      powerCapsStandbyPowerMilliWatts;
    UInt32                      powerCapsMinimumWakeTimeSeconds;
};
```

- Use kVersionOnePowerCapabilities

Power Capabilities

```
struct DriverPowerCapabilities {
    PowerCapsVersion        powerCapsVersion;
    PowerCapsFlags          powerCapsFlags;
    UInt32                  powerCapsStandbyPowerMilliWatts;
    UInt32                  powerCapsMinimumWakeTimeSeconds;
};
```

- kDriverPowerMgtAware
- kDriverPowerMgtUnderExpertControl
- kDriverHasPowerHandlerExport

Power Capabilities

```
struct DriverPowerCapabilities {
    PowerCapsVersion        powerCapsVersion;
    PowerCapsFlags          powerCapsFlags;
    UInt32                  powerCapsStandbyPowerMilliWatts;
    UInt32                  powerCapsMinimumWakeTimeSeconds;
};
```

- kDevicePowerCanBeRemovedForSleep
- kDeviceCanGeneratePMEDuringSleep
- kDriverSpecifiesStandbyPower

Power Capabilities

```
struct DriverPowerCapabilities {
    PowerCapsVersion         powerCapsVersion;
    PowerCapsFlags           powerCapsFlags;
    UInt32                   powerCapsStandbyPowerMilliWatts;
    UInt32                   powerCapsMinimumWakeTimeSeconds;
};
```

- Power used during sleep
    - Overrides PCI configuration space value if kDriverSpecifiesStandbyPower in powerCapsFlags is set

Power Capabilities

```
struct DriverPowerCapabilities {
    PowerCapsVersion        powerCapsVersion;
    PowerCapsFlags          powerCapsFlags;
    UInt32                  powerCapsStandbyPowerMilliWatts;
    UInt32                  powerCapsMinimumWakeTimeSeconds;
};
```

- Minimum value that a device must be powered on before being powered off again
- kUseDefaultWakeTime if less than 5 min

Power Handlers

- Routines called by the Power Manager
  - Get or set changes in power state
- Similar to today’s sleep queue procedures
- Implemented as DoDriverIO, an exported or registered routine

Power Handlers

- DoDriverIO
  - Selector: kPowerManagementCommand
  - ParmBlkPtr (a CntrlParam)
    - csCode = power mgt message
    - csData (first longword)
      - If kGet/SetPowerLevel, the power level

Power Handlers

- Export “DoDriverPowerManagement”

```
typedef long (*PowerHandlerProcPtr) (
        UInt32              message,
        PowerLevel *        powerLevel,
        UInt32              refCon,
        RegEntryID *        regEntryID );
```

  - refCon is only used by registered handlers

- Used for Open Transport and other drivers without a DoDriverIO entrypoint

Power Handlers

- Non-ndrv software can register a Power Handler using DriverServices
- If a RegEntryID is not provided, power handler is not prioritized and is run at beginning of the DeviceSleepQueue during sleep and at the end during wake

Power Handler Messages

- Request/Revoke
  - Only message that can be denied
  - If denied, sleep processes aborted
  - If not denied, complete or suspend pending I/O
  - Interrupts still enabled

Power Handler Messages

- Demand
    - Cannot be denied (result ignored)
    - Drivers save device state now
    - Interrupts are off

Power Handler Messages

- WakeUp
    - Interrupts still off
    - Restore device state
    - Resume suspended I/O

Power Handler Messages

- kDozeRequest, kDozeRevoke, kDozeDemand, kDozeWakeUp
- kSleepRequest, kSleepRevoke, kSleepDemand, kSleepWakeUp
- kSuspendRequest, kSuspendRevoke, kSuspendDemand, kSuspendResume
- kGetPowerLevel, kSetPowerLevel

Power Handler Results

- noErr = Success!
- kPowerMgtRequestDenied
- kPowerMgtMessageNotHandled
    - Must return this for unknown messages

New Driver Services

```
OSStatus AddDevicePowerHandler (
                        RegEntryIDPtr              regEntryID,
                        PowerHandlerProcPtr        handler,
                        UInt32                     refCon);

OSStatus RemoveDevicePowerHandler (RegEntryIDPtr regEntryID);
```

New Driver Services

```
OSStatus GetDevicePowerLevel (
                    RegEntryIDPtr           regEntryID,
                    PowerLevel *            devicePowerLevel);

OSStatus SetDevicePowerLevel (
                    RegEntryIDPtr           regEntryID,
                    PowerLevel              devicePowerLevel);
```

Power Source API

- You can register power sources you provide to the system
- The Power Manager will use those sources in its power summary calculations
- Examples: UPS Backup, Solar Panels, etc.

Power Source API

- PowerSource Data Structure

```
struct PowerSource {
   PowerSourceVersion        sourceVersion;
    PowerSourceID sourceID;
    OptionBits               sourceAttr;
    OptionBits               sourceState;
    UInt32                   currentCapacity;
    UInt32                   maxCapacity;
    UInt32                   timeRemaining;
    UInt32                   timeToFullCharge;
    UInt32                   voltage;
    SInt32                   current;
};
```

Power Source API

- Routines

```
OSStatus AddPowerSource (PowerSource * ioSrc);

OSStatus RemovePowerSource (PowerSourceID * inID);

OSStatus UpdatePowerSource (PowerSource * ioSrc);
```

Scheduled Power Events

The Power Manager can perform certain events based on a schedule provided by the user via Energy Saver or by software using this new API.

Scheduled Power Events

- Scheduled Power Event Types include Sleep, Shutdown, Wake, and Startup
- Not all event types are available on every machine, check with PMFeatures
- Get/SetWakeupTimer and Get/SetStartupTimer are supported but not preferred

Scheduled Power Events

- User Notification
    - Alert (user can cancel event)
    - Flashing Icon
    - Sound

Scheduled Power Events

- ScheduledPowerEvent Data Structure

```
struct ScheduledPowerEvent {
    Boolean                              eventEnabled;
    ScheduledPowerEventFreq              eventFreq;
    ScheduledPowerEventType              eventType;
    ScheduledPowerEventVersion           eventVersion;
    ScheduledPowerEventTimeRec           eventTime;
    ScheduledPowerEventNotifyRec         eventNotification;
};
```

Scheduled Power Events

```
OSStatus SetScheduledPowerEvent (ScheduledPowerEvent * ioEvent);

OSStatus GetScheduledPowerEvent (ScheduledPowerEvent * ioEvent);
```

Processor Temperature Reporting

- Routine to obtain core processor temperature (reported in Celsius):

  ```
  UInt32 GetCoreProcessorTemperature (MPCpuID inCpuID);
  ```

- Requires Multiprocessing API Library 2.0 present in Extensions folder

Example

- Obtaining Core Temperature

```
MPCpuID          cpuID;
OSStatus         err;
UInt32           temp;

cpuID = kInvalidID;
for ( err = MpGetNextCpuID(kInvalidID, &cpuID); err == noErr;
      err = MpGetNextCpuID(kInvalidID, &cpuID) )
         {
         temp = GetCoreProcessorTemp (cpuID);
         // do something ...
         }
```

Please Note…

- All APIs discussed are preliminary and subject to change

For More Information or Feedback…

- *Inside Macintosh: Devices*, “Power Manager”
- *PCI Bus Power Management Interface Specification*, Revision 1.1, PCI SIG
- E-mail: powermgr@apple.com
- Discussion list: powermgr@isg.apple.com

Related Session

What's New: NanoKernel
Multitasking that cares

Hall A1
Fri., 10:15

Think different.™

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

99 | Worldwide Developers Conference