Technotes

Download Acrobat file (K)	Download AppleWorks file (42K)

File Access & the Power Manager:Headaches & Cures

Technote 1039 **MARCH 1996**

If you are building an application that communicates with storage devices by bypassing the Macintosh file system, you may encounter some unexpected problems related to interaction with the Power Manager. This Note discusses some elementary but often overlooked ways to prevent these problems from occurring.

This Note is important for developers who write disk formatters, diagnostics or applications that access disks directly.

Contents

Meet the Power Manager on the Desktop

Traditionally, only portable Macintosh computers contained the combination of software and hardware components that allowed the system to conserve battery power by shutting down various subsystems. Developers who wrote specifically for these machines understood the need to interact with the Power Manager.

In an attempt to set ceilings on computer energy use, the Environmental Protection Agency created a set of specifications known as the Energy Star Program. In order to meet these specifications, the Power Manager was also introduced on Macintosh desktop computers.

One of the ways that the Power Manager functions is to conserve power by spinning down any attached disk drives after a specified period of file system inactivity.

Assuming your application uses standard file system calls, this interaction will be transparent. At worst, your app will only experience a few seconds delay on power up.

Problems with Apps That Bypass the File Manager

Some applications can't use the File Manager for disk I/O. For instance, drive diagnostics, formatters, or even certain performance-sensitive applications may have occasion to communicate directly with the SCSI or ATA manager to access the disk directly.

If your application bypasses the File Manager and doesn't periodically update the Power Manager's idle timer (e.g., by spinning the cursor), it is possible to trick the Mac OS into "thinking" that the computer is idle. If this inactivity persists for a long enough period of time and the disk spindown feature is enabled, the Power Manager will issue a command to shut down the disk subsystem.

If your app is not prepared for this, you may experience some unexpected behavior -- for example, read/write errors, or incompleted operations from your disk.

These problems can also plague users who install an older disk driver that doesn't properly handle powerdown. Some older drivers were simply never designed to interact with the Power Manager on desktop computers.

Mr. Disk Doctor, "It Hurts When I Do This... What's the Cure?"

The first thing your application should do if it intends to perform a long series of disk operations without any File Manager involvement is to determine if it is running on a Macintosh that supports the Power Manager. You can do this by gestalting for gestaltPowerMgrAttr and checking if the gestaltPMgrExists flag is true.

On standard configurations with one internal ATA or SCSI drive, the disk spindown is typically controlled by the Power Manager's HardDiskQueue activity timer. In the past, periodically calling idleUpdate() was sufficient to prevent disk spindown.

Yet on newer systems that include multiple internal and external drives, such as PCMCIA, expansion bay or even external SCSI, access to a single drive can prevent all other drives from being spun down. This is because the access will reset the one Power Manager spindown timer used by all the drives. What can also happen is that if you're using one timer, then all your drives will spin up at the same time. This strategy is very costly in terms of power consumption and does not promote the efficient battery use on PowerBooks.

On these systems the driver writer is compelled to employ an internal spindown timer instead of the Power Manger's timer, thus avoiding spindown synchronization.

The side effect of a driver using its own spindown timer is that IdleUpdate(), EnableIdle(), and DisableIdle() do not have an effect on the driver's timer as it does with the Power Manager's timer. The only way to reset or disable the driver's timer to prevent spindown is either to access the drive (via read or write) or disable spindown via SetSpindownDisable().

Even with this precaution, your application still needs to be prepared to handle any errors that might be caused by a disk drive spinning down. It's possible that some errant drivers won't even honor SetSpindownDisable().

How Does Apple Do It?

The Apple ATA driver manages its own spindown timer in such a way as to not cause a major disruption to your app. Its spins down the drive with the ATA Standby command instead of the ATA Sleep command. The ATA Standby does not deactivate the drive interface, allowing the drive to spin back up on the next access. On the other hand, the ATA Sleep does deactivate it and requires a soft reset before the drive can be used again. Consequently, your app is only inconvenienced by a spin up delay.

A Caveat

Unfortunately, this method was only recently incorporated into the Apple ATA driver. There is no guarantee that an earlier Apple or third-party disk driver will behave this way.

Summary

Power management functions are no longer limited to the realm of portables but are appearing now on desktop Macintosh computers. This can cause more than a few headaches if you plan to access the disk behind the File Manager's back.

If your application routinely bypasses the File System and accesses the drive through the ATA or SCSI Manager, it should temporarily disable the drive spindown. Even then, you should still anticipate and handle any errors caused by drive spindown.

Inside Macintosh: Devices, Chapter 6, Power Manager Reference

---

Technotes
Previous Technote | Contents | Next Technote