Technotes

Download Acrobat file (K)	Download AppleWorks file (44K)

Some Sound Advice: Getting the Most Out of the Sound Manager

Technote 1048 **JUNE 1996**

This Technote discusses issues that you need to be aware of if you are developing apps that use more than basic Sound Manager calls. The information in this Technote has been gleaned from a number of developer questions fielded by DTS, and focuses on issues that arise when dealing with sound input and sound output.

The information in chapters 2 and 3, "Sound Manager" and "Sound Input Manager," in *Inside Macintosh: Sound* is still accurate, but changes and new features require new documentation which this Technote is part of. This Technote covers information not found in the release notes for Sound Manager 3.0, 3.1, and 3.2. This Technote also clarifies the information in *Inside Macintosh: Sound* and points out how you can use the Sound Manager more efficiently.

Contents

Input Items

Over the last three years, in answering developer questions about the Sound Manager, DTS has gained some new insights into the Sound Manager's inner workings. In this section of the Technote, we share those insights.

On the surface, sound input appears to be programmatically similar to sound output. This, however, is not true. There are two reasons for this:

1. Sound input is still based on an old-fashioned driver model.

2. Sound output has been "modernized," beginning with Sound Manager 3.0, and uses the Component Manager extensively.

One Source of Input Problems: Different Hardware Implementations

Input problems can arise because of variations in sound input hardware from one machine to another. For example, a Macintosh Quadra 700 has 8-bit sound input while a Power Macintosh 7200 has 16-bit sound input. The contrast, however, is more subtle because a 7200 defaults to recording stereo 16-bit sound instead of 8-bit mono sound like the 700. Furthermore, even though the Quadra 840AV and most Power Macs support 16-bit sound, the 840AV supports many more and different sample rates than most Power Macs.

Important:
Don't assume that the default settings of the selected sound input device are what you're

expecting or what they've been on previous Macs or previous versions of the Sound Manager. Make sure that you set the input state to exactly what you want before you begin recording. Use the SPBSetDeviceInfo function to set the recording parameters. Use the SPBGetDeviceInfo function to make sure that the sound will be recorded exactly as you want it to be.

Simultaneous Play-and-Record

Simultaneous play-and-record is the act of recording one sound while playing another sound at the same time. This is a feature commonly requested in speaker phones.

Not all Macintosh computers can simultaneously play and record sound -- e.g., a PowerBook Duo 280c cannot play and record at the same time, while the Quadra 840AV can.

To determine if a particular Mac can play and record simultaneously, you need to call Gestalt with the 'snd ' selector and check for the gestaltPlayAndRecord bit being set. If this bit is not set and you are recording a sound, you will not be able to play a sound until the recording finishes. Likewise, if you are playing a sound, recording is not possible either. What this means to you is that if your application requires this functionality, you will need to check Gestalt and take the appropriate action.

Note:

Turning on the level meter (with SPBSetDeviceInfo and the siLevelMeterOnOff selector) is recording without saving the data. Therefore, for machines that cannot play and record simultaneously, the level meter must be off before you can play sounds.

Important:

Power Macintosh computers can only do simultaneous play and record if both the input and output channels are set to the same rate -- for example, 22.050 KHz. If you require this, make sure that input and output rates are the same.

Variations on Input Sample Rate Support

Don't assume that all Macs support the same input rates, and don't assume that they default to the same rates either. Even similar Macs can have very different capabilities.

There are variations in the input sample rates supported by different Macintosh computers. For example, the number of input rates supported by a Power Macintosh 7100 is two, while the number of input rates for a Power Macintosh 7500 is three, and a Quadra 840AV has seven. The Power Macintosh 7100 rates are 22.050 KHz and 44.1 KHz; the Power Mac 7500 also supports 11.025KHz; the Quadra 840AV rates includes those, plus an additional two -- 24 KHz and 48 KHz.

For most developers, and users, this is not an issue unless you have hard-coded in a particular sample rate. If this is the case, then you should rewrite your code to be more flexible in its choice of sample rates. Use SPBGetDeviceInfo with the siSampleRate selector to find the rate you like best and then use SPBSetDeviceInfo to set that rate. If you can't get a needed sample rate you require, you can use Sound Manager 3.2's conversion routines to convert between the rate you can get and the rate that you need.

Having the Correct Buffer Size Matters

Sound recording works best if the buffer you are recording into is sized to be an integer multiple of the size of the recording device's interrupt buffer.

After you've set up the recording parameters, specifically mono versus stereo, and 8 bit versus 16 bits, call SPBGetDeviceInfo with the siDeviceBufferInfo ('dbin') selector to find the size of the machine's input buffer.

After determining the size, multiply this number by some integer of your choosing to size your application's recording buffer. This way, at interrupt time you are copying whole buffers, and you and the Sound Manager don't have to worry about partial buffers.

This helps to optimize how the Sound Manager moves data out of the recording hardware and is a speed optimization.

Variations in Sound Input Hardware

The sound input hardware on all Power Macintosh 5200/5300/6200/6300 computers was changed during its lifetime. Some of these Macs can only record 8-bit sounds, while others can also record 16-bit sounds. The machine you have in your lab will be one or the other of these; you need to make sure that you write code that will work on both. To this end, use SPBGetDeviceInfo with the siSampleSizeAvailable ('ssav') selector to find out what bit depth the Mac you are running on is capable of recording sounds at, or better yet, write code that doesn't care what the bit size is.

Important:

There are sound hardware differences among these four classes of machines: the Power Macintosh 5200/5300/6200/6300 computers. Just because these are Power Mac machines, you should not assume that all contain 16-bit input and output hardware. For example, the Macintosh Performa 5215CD contains both 16-bit input and output hardware. However, if you call Gestalt on that particular machine, it returns gestalt16BitSoundIO not set (i.e., false), this is incorrect. You should instead check the available sample sizes by using the siSampleSizeAvailable selector.

Important:

The siActiveChannels selector used with the SPBGetDeviceInfo and SPBSetDeviceInfo calls is not supported on the Power Macintosh 5200/5300/6200/6300 series Macintoshes, and may not be supported on other current or future Macintosh models. If you make this call and it returns the error siUnknownInfoType, then you know that this selector is not supported. Don't panic: you will be recording always from both channels.

On PCI Power Macintoshes, using the siActiveChannels selector with a value of 0x01 or 0x02 (which would turn on only the left or right channel, respectively) does not work before System 7.5.3. If you are on a PCI Power Mac running system 7.5.2, the Sound Manager version does not matter: do not set siActiveChannels to anything but 0x03, which is the same as leaving it at its default. If you are on a PCI Power Mac running 7.5.3 or later, you can use the siActiveChannels selector and set it to 0x01, 0x02, or 0x03.

If you need to record from only one channel of an input source on a stereo Macintosh and siActiveChannels is not available, record in stereo (siNumberChannels set to 2) and have your buffer routine only copy out the desired channel. This is the most effective and memory-frugal workaround.

Portable Issues

The PowerBook 5300 does not support microphone level input as most other Macs do. That 1/8" jack in the back that looks suspiciously like a microphone jack isn't - it's a line input jack. The PowerBook 5300 requires line-level input to record sounds from this jack, such as the input levels that are carried on RCA type jacks. Some Macintoshes shipped with an RCA-to-microphone level adapter, but this won't work on the PowerBook 5300.

You'll need to buy a RCA-to-mini 1/8" jack to get sound into a PowerBook 5300 if you don't want to use the built in microphone. Recording must be done from a source that outputs line level signals, such as a standard CD player.

The PowerBook Duo 2300's sound hardware is capable of recording stereo sound, but since its built-in microphone only records in mono, and the 2300 doesn't have a line-in jack, you have to connect it to an external Duo Dock to get line-in input. The current Apple Duo Docks that provide sound input jacks only provide mono input connectors. Unless a third party makes a dock with a stereo sound input jack, there just isn't a way to get from hear (pun intended) to there.

Output Items

Sound Managers version 3.0 and later extend the capabilities of the output of sound, offering 16-bit sound and better sample rates along with more a flexible component architecture.

Return the Right Thing

If you are calling SndSoundManagerVersion, MACEVersion, MIDIVersion, SpeechManagerVersion, or SPBVersion, make sure that the return types are a NumVersion instead of a long. Change your headers if necessary.

These are currently the only calls that should return a NumVersion.

Trouble with Void Pointers

The SetSoundOutputInfo, SndSetInfo, and SPBSetDeviceInfo all take a void* as their last parameter.

Because the last parameter of these calls is a void*, it looks as if you need to pass a pointer to the value you are setting, but this is not the case if what you are passing is 4 bytes or less in size. If you are passing a parameter that is larger than four bytes, you must pass a pointer to it.

If you are calling GetSoundOutputInfo, SndGetInfo, or SPBGetDeviceInfo and it is returning a value that is 4 bytes in size or less, it does not return a pointer to that value, it just returns the value.

Playing a Compressed Sound Using SndPlayDoubleBuffer

Beginning with Sound Manager 3.0, if you're using SndPlayDoubleBuffer to play sounds, you need to use the SndDoubleBufferHeader structure for uncompressed sounds. Use the SndDoubleBufferHeader2 structure to play both compressed and uncompressed sounds.

SndPlayDoubleBuffer takes a pointer to a structure of type SndDoubleBufferHeader. To play a compressed sound using the SndDoubleBufferHeader2 structure, simply cast it to a SndDoubleBufferHeaderPtr when you call SndPlayDoubleBuffer. The SndPlayDoubleBuffer function is "smart" enough to figure out the rest.

Note:

Starting with Sound Manager 3.0 SndPlayDoubleBuffer is able to play 16 bit sounds. See the *SndMgr 3.0 release note* for more information

Too Much of a Good Thing

Some applications may be calling some Sound Manager calls more often than they need to, specifically, GetDefaultOutputVolume. Some applications call this function in their main event loop when this is probably not necessary. Sound volume levels rarely change ten or more times a second -- in fact, they rarely change. The proper time to call GetDefaultOutputVolume is right before you go to play a sound to see if you need to adjust the output device's volume, or to adjust the input or output gain of a device. This saves you a little time in your event loop and may make your application a bit faster.

Note:

You probably shouldn't be changing the output volume, making the GetDefaultOutputVolume unnecessary. Take the user's word for the output volume they want.

Trouble with Bits and Bytes

In the SndDoubleBufferHeader2 structure, there is a field called dbhSampleSize which is the sample size in bits of the sound. *Inside Macintosh:Sound* page 2-111 says that dbhSampleSize should be set to 0 if the sound is compressed. This is incorrect: dbhSampleSize should always be set to the actual uncompressed sample size of the sound. *Inside Macintosh:Sound* page 2-112 says that the dbhPacketSize field needs to be set to the number of bits per packet; it needs to be set to the number of **bytes** per packet, however.

Outta Time--The kNonRealTime Flag

This only applies to those developers who are writing sound output components. It is expected that all sound components work in real time.

Beginning with Sound Manager 3.1 the kRealTime flag was renamed to kNonRealTime. This flag tells the Sound Manager that your sound component can also work in non-real time, i.e., it can do a better job if only given the extra time.

This means that when the component is part of a QuickTime output chain or a Sound Manager 3.2 conversion chain, it can take the extra time it needs to do a better job.

Hold That Memory

Because we expect sound to occur in realtime, smoothly and without hiccups or gaps, if you turn on Virtual Memory (VM), you're going to experience problems. When VM pages it turns off interrupts for a long period of time and without interrupts enabled, the Sound Manager is not able to service the sound hardware. If interrupts are disabled for too long, the sound will stop playing and then resume when VM re-enables interrupts.

You want to make sure that VM at least never pages out your sound's buffers. Even if it doesn't page out your buffers, this will not assure you that sound will play uninterrupted, but it is the best that you can do. To this end, use HoldMemory to hold the memory that your sound's buffers are in. HoldMemory is explained in the chapter "Virtual Memory" in *Inside Macintosh: Memory*. Macintosh Technical Note ME 09 has valuable information about how to deal with VM and the limitations it imposes.

Be sure you call UnholdMemory on each block when you are done with it.

Why Caching is Not Useful

Setting the noCacheBit (which is defined in FSM.h) to disable caching of reads when you're reading a sound from disk is a good idea. Since many sounds are larger than will fit into the cache, and most sounds are only played once, caching them is of no benefit and not worth the overhead. Refer to Macintosh Technical Note FL 16 for information on how to use this feature.

Summary

Because there are real hardware differences among various Macintosh computers, you may not be taking full advantage of sound in your application. To avoid sound input and output problems, your app needs (1) to call Gestalt and SPBGetDeviceInfo to determine what features and capabilities are present in both the hardware and the Sound Manager, and (2) to pay attention to the topics and issues addressed in this Technote.

- *Inside Macintosh: Sound*
- The Sound Manager 3.0, 3.1, and 3.2 Release Notes on Developer CD Series.
- Macintosh Technote ME 9 - Coping With VM and Memory Mappings
- Macintosh Technote FL 16 - File Manager Performance and Caching

---