Obtaining Information About Sound Features What you need to know

Developments in the sound hardware available on Macintosh computers and in the **Sound Manager** routines that allow you to drive that hardware have made it imperative that your application pays close attention to the sound-related features of the operating environment. For example, the routines that provide continuous play from disk operate only on machines that are equipped with an Apple Sound Chip that have the enhanced **Sound Manager**. So before issuing any play-from-disk calls, you should check to make sure that the target machine provides the features you need.

Similarly, the ability to have multiple channels of sound open simultaneously makes it important that you monitor the load placed on the CPU by those channels. The enhanced **Sound Manager** provides several new routines that you can use to determine information about open sound channels and the amount of CPU loading they create.

To make appropriate decisions about the sound you want to produce, you may need to know some or all of the following types of information:

- whether the machine can produce stereophonic sounds
- whether the internal speaker mixes both right and left channels of sound
- whether the sound input routines and hardware are available
- whether multiple channels of sound are supported
- how much CPU load is produced by a single channel of sound or by all channels of sound
- whether the system beep has been disabled
- how much of the available processing power a new channel of sound would consume
- whether a sound playing from disk is active or paused
- how many channels of sound are currently open

To determine how much of the available processing power a new channel of sound would consume, you can use the loadCmd sound command, described earlier in **Managing the CPU Load**. The following sections describe how to use the **Gestalt** function and new **Sound Manager** routines to determine these other types of information.

Obtaining Information About Available Sound Features

You can use the **Gestalt** function with the gestaltSoundAttr selector to determine whether various new **Sound Manager** capabilities are present (for example, whether the machine can produce stereophonic sounds and whether it can mix both left and right channels of sound on the external speaker). Currently, **Gestalt** returns a bit field that may have some or all of the following bits set:

```
gestaltStereoCapability         //stereo capability present
```

```
gestaltStereoMixing //stereo mixing on internal speaker
gestaltSoundIOMgrPresent     //sound input routines available
gestaltBuiltInSoundInput     //built-in input device available
gestaltHasSoundInputDevice   //sound input device available
```

If the bit gestaltStereoCapability is TRUE, the available hardware can play stereo sounds. The bit gestaltStereoMixing indicates that the sound hardware of the machine mixes both left and right channels of stereo sound into a single audio signal for the internal speaker. The gestaltSoundIOMgrPresent bit indicates that the new sound input routines are available, and the gestaltBuiltInSoundInput bit indicates that a built-in sound input device is available. The gestaltHasSoundInputDevice bit indicates that some sound input device is available.

Obtaining Version Information

The **Sound Manager** provides functions that allow you to determine the version numbers of the **Sound Manager** itself and of two distinct subsets of the **Sound Manager** routines, the MACE compression and expansion routines and the sound input routines. Generally, you should avoid trying to determine which features or routines are present by reading a version number. Usually, the **Gestalt** function discussed in the previous section provides a better way to find out if some set of features, such as sound input capability, is available. In some cases, however, you can use these version routines to overcome current limitations of the information returned by **Gestalt**.

All three of these functions return a value of type NumVersion that contains the same information as the first 4 bytes of a resource of type 'vers'. The first and second bytes contain the major and minor version numbers, respectively; the third and fourth bytes contain the release level and the stage of the release level. For most purposes, the major and minor release version numbers are sufficient to identify the version. (See the section **Finder Interface** for a complete discussion of the format of 'vers' resources.)

You can use the **SndSoundManagerVersion** function to determine which version of the **Sound Manager** is present. The **Sound Manager** provided with system software version 6.0.7 contains the routines supporting multichannel sound, play from disk, and channel status inquiries.

You can use the **MACEVersion** function to determine the version number of the available MACE routines (for example, **Comp3to1**).

You can use the **SPBVersion** function to determine the version number of the available sound input routines (for example, **SndRecord**). If **SPBVersion** returns a value that is greater than 0, then the sound input routines are available.

Obtaining Information About a Single Sound Channel

You can use the **SndChannelStatus** function to obtain information about a single sound channel and about the status of a disk-based playback on that channel, if one exists. For example, you can use **SndChannelStatus** to determine if a channel is being used for play from disk, how many seconds of the sound have been played, and how many seconds remain to be played.

One of the parameters required by the **SndChannelStatus** function is a

pointer to a sound-channel status record, **SCStatus**,which you must allocate before calling **SndChannelStatus**.

The code below illustrates the use of the **SndChannelStatus** function. It defines a function that takes a sound-channel pointer as a parameter and determines whether a disk-based playback on that channel is paused.

```
//Determining whether a sound channel is paused

// Assuming inclusion of MacHeaders
#include <Sound.h>

// Prototype routine like this prior to calling it
Boolean ChannelIsPaused(SndChannelPtr);

Boolean ChannelIsPaused (SndChannelPtr chan)
{
    OSErr myErr; // temporary error variable
    SCStatus mySCStatus;                    // status of channel

    // Get Status
    myErr = SndChannelStatus (chan, sizeof(SCStatus), &mySCStatus);

    // if no error set boolean return value from status information
    if ( !myErr )
        return mySCStatus.scChannelPaused;

    return FALSE;
}
```

The function defined here simply reads the *scChannelPaused* field to see if the playback is currently paused.

Obtaining Information About All Sound Channels

You can use the **SndManagerStatus** function to determine information about all the sound channels that are currently allocated by all applications. For example, you can use this function to determine how many channels are currently allocated.

One of the parameters required by the **SndManagerStatus** function is a pointer to a **Sound Manager** status record, **SMStatus**,which you must allocate before calling **SndManagerStatus**.

The code below illustrates the use of **SndManagerStatus**. It defines a function that returns the number of sound channels currently allocated by all applications.

Determining the number of allocated sound channels

```
// Assuming inclusion of MacHeaders
#include <Sound.h>
```

```
// Prototype routine like this prior to calling it
short NumChannelsAllocated(void);

short NumChannelsAllocated()
{
    OSErr myErr;    // temporary error variable
    SMStatus mySMStatus;    // status record for sound manager

    //  get status of Sound Manager
    myErr = SndManagerStatus (sizeof(SMStatus), &mySMStatus);

    // if no error occured, set number of channels to return
    if ( !myErr )
        return mySMStatus.smNumChannels;

    return 0;
}
```