Sound Commands

Creating sounds

The **Sound Manager** provides routines that allow you to create and dispose of sound channels and send control information directly to synthesizers. These routines allow you to manipulate sound channels and synthesizers, but (except for the **SndPlay** and **SndStartFilePlay** functions) they do not directly produce any sounds. To produce actual sounds, you need to issue sound commands. A **sound command** is an instruction to a synthesizer to produce sound, modify sound, or otherwise assist in the overall process of sound production.

You can issue sound commands in several ways. You can send sound commands one at a time into a sound channel by repeatedly calling the **SndDoCommand** function. Or you can bypass a sound channel altogether and send commands directly to a playback synthesizer by calling the **SndDoImmediate** function. You can also issue sound commands by calling the function **SndPlay** and specifying a sound resource of type 'snd ' that contains the sound commands you want to issue. A sound resource can contain any number of sound commands and can be used to send commands and data to any of the available sound synthesizers. As a result, you may be able to accomplish all sound-related activity simply by creating sound resources and calling SndPlay in your application. See **Sound Resources** for details on the format of an 'snd ' resource.

Generally speaking, no matter how they are issued, all sound commands are eventually sent to the playback synthesizer, which interprets the commands and plays the sound on the available audio hardware. All synthesizers are designed to accept the most general set of sound commands, although some commands are specific to a particular synthesizer. Here is the structure of a generic sound command:

```
#include <Sound.h>

typedef struct SndCommand { Description
unsigned short              cmd;     command number
short          param1;               first parameter
long           param2;               second parameter
} SndCommand;
```

Commands are always 8 bytes in length. The first 2 bytes are the command number, and the next 6 make up the command's options. The format of the last 6 bytes depends on the command in use, although typically those 6 bytes are interpreted as an integer followed by a long integer. For example, an application can install a wave table into a sound channel by using **SndDoCommand** with a sound command whose cmd field is the waveTableCmd constant. In that case, the *param1* field specifies the length of the wave table, and the *param2* field is a pointer to the wave-table data itself. Other sound commands may interpret the 6 parameter bytes differently or may not use them at all.

The sound commands available to your application are defined by constants.

nullCmd	do nothing
quietCmd	stop a sound that is playing
flushCmd	flush a sound channel
reInitCmd	reinitialize a sound channel

waitCmd	suspend processing in a channel
pauseCmd	pause processing in a channel
resumeCmd	resume processing in a channel
callBackCmd	execute a callback procedure
syncCmd	synchronize channels
availableCmd	see if initialization option available
versionCmd	determine synthesizer version
totalLoadCmd	report total CPU load
loadCmd	report CPU load for a new channel
freqDurationCmd	play a frequency for specified duration
restCmd	rest a channel for specified duration
freqCmd	change the pitch of a sound
ampCmd	change the amplitude of a sound
timbreCmd	change the timbre of a sound
getAmpCmd	get the amplitude of a sound
waveTableCmd	install a wave table as a voice
soundCmd	install a sampled sound as a voice
bufferCmd	play a sampled sound
rateCmd	set the pitch of a sampled sound
getRateCmd	get the pitch of a sampled sound

For details on each sound command, see the relevant sections in **Using the Sound Manager**.