Specifying Callback Routines Associate a routine with a channel

The **SndNewChannel** function allows you to associate a completion routine or callback procedure with a sound channel. This procedure is called whenever a callBackCmd command is received by the synthesizer linked to that channel, and the procedure can be used for various purposes. Generally, your application uses a callback procedure to determine that the channel has completed its commands and to arrange for disposal of the channel. The callback procedure cannot itself dispose of the channel because it may execute at interrupt time. A callback procedure can also be used to signal that a channel has reached a certain point in the queue. Your application may wish to perform particular actions based on how far along the sequence of commands a channel has processed. This allows you, for example, to synchronize sound output with other actions in the computer.

A callback procedure has the following syntax:

pascal void **MyCallBack (**SndChannelPtr chan, SndCommand cmd);

When called, the procedure is passed two parameters-a pointer to the sound channel that received the callBackCmd command and the sound command that caused the callback procedure to be called. Applications can use *param1* or *param2* of the sound command as flags to pass information or instructions to the callback procedure. If a callback procedure is to use an application's global data storage, it must first reset A5 to the application's A5 and then restore it on exit. For example, the Listing below illustrates how to set up a callBackCmd command that contains the required A5 information. The **InstallCallBack** function defined there must be called at a time when the application's A5 world is known to be valid.

Issuing a callback command

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

#define kWaitIfFull TRUE     // wait for room in queue
#define kSoundComplete  1    // last command in channel;  Application defined

// Prototype routine like this prior to calling it
OSErr InstallCallBack(SndChannelPtr);

OSErr InstallCallBack (SndChannelPtr mySndChan)
{
    SndCommand mySndCmd;       // command record

    mySndCmd.cmd = callBackCmd;    // install the callback command
    mySndCmd.param1 = kSoundComplete;    // last command for this channel
    mySndCmd.param2 = SetCurrentA5();     // pass the callback the A5
    return SndDoCommand (mySndChan, &mySndCmd, kWaitIfFull);
}
```

In this function, kSoundComplete is an application-defined global constant that indicates that the requested sound has finished playing. You could define it

like this:

```
#define kSoundComplete =        1;          //sound is done playing
```

Because *param2* of a sound command is a long integer, you can use it to pass the application's A5 to the callback procedure. That allows the callback procedure to gain access to the application's A5 world. The sample callback procedure defined in the code below can set A5 to access the application's global variables when the synthesizer receives this command.

Here, the callback procedure simply sets a global variable that indicates that the callback procedure has been called. The application can then read that variable to determine that it can safely dispose of the corresponding sound channel. The functions **SetCurrentA5** and **SetA5** are documented in the **Memory Manager**.

> **Note:** These callback routines are called at interrupt time and therefore must not attempt to allocate, move, or dispose of memory, dereference an unlocked handle, or call other routines that do so.

```
// Listing:Defining a callback procedure


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

#define kSoundComplete 1  //  Application defined; last command in channel
Boolean gCallBackPerformed;   // global flag for call back

// Prototype call back procedures as follows
// Note that since this routine is a callback that will be called
// from the ToolBox, it must use pascal calling conventions
pascal void SampleCallBack(SndChannelPtr,SndCommand);

pascal void SampleCallBack (SndChannelPtr theChan, SndCommand theCmd)
{
    long myA5;      // variable to hold current value of A5

    // if this the last command, then set up a global flag to note as such
    if  (theCmd.param1 = kSoundComplete) {
        myA5 = SetA5(theCmd.param2);     // set my A5
        gCallBackPerformed = TRUE;       // set a global flag
        myA5 = SetA5(myA5);              // restore the current A5
    }
}
```

> **Assembly-language note:** A callback procedure is a high-level language procedure and must preserve all registers other than A0-A1 and D0-D2.