

# New Technical Notes

Macintosh



®

---

Developer Support

## Sound Manager Q&As

**Toolbox M.TB.SoundMgr.Q&As**

Revised by: Developer Support Center

October 1992

Written by: Developer Support Center

October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

|New Q&As and Q&As revised this month are marked with a bar in the side margin.

---

## How to play Macintosh sounds in a continuous sequence

Written: 11/15/90

Last reviewed: 3/18/92

When I open a Macintosh sound channel under System 6.0.5, I can use SndPlay to play sampled sounds, one at a time. When I'm done, I close the channel. Under System 6.0.7, after the first sound is played, no other sounds will play. If I close and reopen the channel, I can play another sound, but only one (until I close and reopen the channel again). Why do sound routines that functioned normally in System 6.0.5 seem to operate incorrectly in System 6.0.7?

---

The problem is documented in the Sound Manager chapter of Inside Macintosh Volume VI. If you have a 'snd ' resource that has a synth defined in it, then you cannot open your own channel with a synth and then call SndPlay. Because your code is playing the sound once, what is happening is that you have the channel set up correctly but you are trying to use it twice. The only way you can play a 'snd ' with a synth is to create and dispose of the channel each time, as in the following code (the resource used is just a SoundEdit resource):

```
#include    <Sound.h>

#define     TRUE     0xFF
#define     FALSE    0

main()
```

```
{
    Handle      Sound;
    SndChannelPtr chan;
    int         i;
    OSErr       err;

    Sound = GetResource ('snd ', 100);
    if (ResError() != noErr || Sound == nil)
        Debugger();

    for (i = 0; i < 3; ++i) {
        chan = nil;
        err = SndNewChannel (&chan, 0, 0, nil);
        if (err != noErr)
            Debugger();

        err = SndPlay (chan, Sound, FALSE);
        if (err != noErr)
            Debugger();

        err = SndDisposeChannel (chan, FALSE);
        if (err != noErr)
            Debugger();
    }
}
```

If you want to play multiple sounds one after another, you may be able to do it by using the `bufferCmd`. You can just keep pumping sounds to it and it will continue to play them.

The way to tell if your 'snd ' has a synth is to check the third word in the 'snd ' file. If the value is a 5 then you have the synth set to `sampldSynth`. A format 2 snd is assumed to be for the `sampldSoundSnth`. In the case of `SoundEdit` resources, you do get a synth. If you take this route, you will need to make sure that all the 'snd ' resources have been changed and you will want to define the synth when you create the channel.

You can avoid having to dispose of the channel after each play if you make sure that each 'snd ' does not have a synth ID and then create the channel with the proper synth ID and play the 'snd ' using `SndPlay`. The one problem with this is that you do not know when the sound is finished playing is you do the asynchronously.

Note the this problem of using `SndPlay` more than once on a channel has been fixed in System 7.

## Sound Manager versus Sound Driver

Written: 11/6/90

Last reviewed: 3/18/92

We generate sounds using the Four-Tone Synthesizer (`StartSound`, `StopSound`). Our application must run on all Macintosh computers and all system software versions starting with System 6.0. According to an early version of *Inside Macintosh*, Volume VI, the Sound Manager's Four-Tone Synthesizer does not perform as expected on some Macintosh systems. When should we use the Sound Manager and when should we use the Sound Driver?

---

The Sound Driver is no longer supported, as of System 6.0.7. The Four-Tone Synthesizer in the Sound Manager released with 6.0.7 does not work correctly for non-Apple Sound Chip

(ASC) machines (Macintosh Plus, Macintosh SE, Macintosh Classic, and Macintosh LC). This has been fixed in System 7. If you need to use the Four-Tone Synthesizer for non-ASC machines running 6.0.7, you could try the Sound Driver with 6.0.7 on the chance it'll work for your purpose. Use the Sound Driver for non-ASC machines running System 6.0.5 and earlier. Note that there ARE problems when mixing the use of the Sound Driver and the Sound Manager. Also note that SysBeep is a call to the Sound Manager!

### **Specifying a Macintosh Sound Manager frequency**

Written: 11/20/90

Last reviewed: 3/18/92

Why don't the Macintosh System 6.0.5 Sound Manager's noteCmd and freqCmd allow you to specify the frequency by specifying a three-byte frequency value, with the least significant byte representing the fractional part of the frequency? According to Inside Macintosh, Volume V, page 483, you should be able to specify frequency in this way, as well as by the selection of a musical note enumeration from an even-tempered scale (values 1 through 127). The Sound Manager chapter in Volume VI of Inside Macintosh does not discuss frequency specification at all. Is support for this very valuable feature forthcoming?

---

We are currently unsure if the second method for specifying a frequency will ever be supported. The problem is that there is no clean way to do the byte sharing which was suggested in Inside Macintosh. If we come up with some way to support this and still have a reasonable range of frequencies which can be played, then we will do it.

### **Which Macintosh systems support internal speaker stereo mixing**

Written: 1/2/91

Last reviewed: 3/18/92

Why do I get only the left channel of a stereo sound out of my Macintosh IIcx?

---

The only Macintosh models that currently combine the two stereo channels into one for playback out of the internal speaker are the Macintosh SE/30 and the IIsi. All others use just the left channel. If you would like to check for the machine's ability to do mixing, you can use Gestalt. This is documented in Inside Macintosh Volume VI, page 22-70.

### **Macintosh code and instructions for generating dial tones**

Written: 1/17/91

Last reviewed: 3/18/92

My program lets a user dial the phone if a modem is connected to the user's Macintosh. How can I add the option of dialing without using the modem, just like HyperCard does? Is generating dial tones a standard function of the Sound Manager?

—

So, ya wanna do dial tones just like HyperCard? Well, that's not so hard (as long as you are patient enough to read through all the Sound Manager documents and beat your head against the wall for half an hour, like I did). The technique is simple: lift the 'snd ' resources from HyperCard and use them to generate dial tones! (Yes, that's OK, dtmf tones are generic sounds; just don't lift any of the other sounds!)

Start by using ResEdit to copy 'snd ' resources #130-145 from the HyperCard program file (I got them out of 2.0). Next use the Sound Manager to play them and to use them as an instrument definition for the samplesynth. Since this can be tricky, and since you may or may not want dive deep into the Sound Manager, I am including a generic subroutine that I wrote that will “dial” for you. All I ask is that if you use it “as is” in your application, please remember me in your prayers.

How to use the dial procedure:

To call the dial procedure simply pass it a string and it will do the rest. All digits A thru D, # and \* play the proper tone for the touch tone key; any other characters generate a 1/2 second delay. Each tone is sounded for about 1/4 second. The interface to it is simply: Procedure Dial(theNum:Str255). Please note that in order to keep this short, very little error checking has been added, please add any/all that you need.

And now for the fabulous source code:

```
PROCEDURE Dial(theNum: Str255);

VAR
    i                : Integer;
    theSound         : Handle;
    theResNum        : Integer;
    dummy            : LongInt;
    err              : OSErr;
    chan             : SndChannelPtr;
    cmd              : SndCommand;

BEGIN
    IF Length(theNum) = 0 THEN
        Exit(Dial);

    chan := NIL;           {allocate a channel}
    err := SndNewChannel(chan, sampledSynth, 0, NIL);
    IF err <> noErr
        Exit(Dial);

    FOR i := 1 TO Length(theNum) DO BEGIN
        CASE theNum[i] OF
            '0'..'9': theResNum := ord(theNum[i]) - ord('0') + 130;
            'A'..'D': theResNum := ord(theNum[i]) - ord('A') + 140;
            '*': theResNum := 144;
            '#': theResNum := 145;
            OTHERWISE theResNum := 0;
        END;

        IF theResNum <> 0 THEN BEGIN
            {Load the proper sound resource}
            theSound := GetResource('snd ', theResNum);
            HLock(theSound);

            {See sound manager docs for resource format of}
```

```
{type 2 sounds for more info}  
cmd.cmd := soundCmd;
```

```
cmd.param1 := 0;

{This next line ASSUMES it was a format 2 'snd ' resource}
{This will not work for other 'snd ' resources}
cmd.param2 := ORD4(theSound^) + 14;
err := SndDoImmediate(chan, cmd);

{Now send the freqDurationCmd to play a middle C}
{that's how the instrument was defined in HyperCard}
cmd.cmd := freqDurationCmd;
cmd.Param1 := 4000;                                {2 seconds in duration}
cmd.Param2 := $FF000000 + 60;                        {Loud + Middle c}
err := SndDoImmediate(chan, cmd);

END ELSE
    Delay(30, dummy);    {all nonDTMF characters produce a delay...}
END;
err := SndDisposeChannel(chan, false);
HUnlock(theSound);
END;
```

## Macintosh sndRecord filter proc structure

Written: 8/30/91

Last reviewed: 3/18/92

What is the structure of a filter proc for SndRecord?

---

The filter procs used by SndRecord are identical to those used by the standard modal dialog. You can find information on writing filter procs in Inside Macintosh Volume I, page I-415. For sample code, take a look at the Macintosh Technical Notes "Customizing Standard File" and "Standard File Tips," and Sample Code #18 called StdFile.c

## SndPlayDoubleBuffer doesn't work on Macintosh Plus or Classic

Written: 8/30/91

Last reviewed: 3/18/92

The call SndPlayDoubleBuffer doesn't produce sound on the Macintosh Plus or Classic because neither model is equipped with an Apple Sound Chip (ASC). The following is a short blurb taken from Inside Macintosh Volume VI, page 22-3 (second paragraph): "...the continuous play-from-disk routines and the ability to produce concurrent multiple channels of sampled sound are currently available only on machines equipped with the ASC."

You can fake your own double-buffering routine, replacing SndPlayDoubleBuffer, by issuing a string of buffering commands followed immediately by a callback routine. While this task takes a bit 'o coding time, the method will work on all Macintosh models.

Alternatively, you can use the Gestalt function to test for the Apple Sound Chip (ASC), and then use SndPlayDoubleBuffer only with Macintosh systems that have the ASC.



## **SndDoImmediate versus SndDoCommands**

Written: 9/16/91

Last reviewed: 3/18/92

We want to allow the user to change the sound volume while a lengthy (many seconds or minutes) sound is playing.

—

If the sound that is playing belongs to your application, then you should be doing a `sndDoImmediate` with an `ampCmd` to change the amplitude of the current sound. (See Inside Macintosh Volume VI, pages 22-50 thru 22-52, for information.) You should not set the system wide volume to accomplish this. Only the “new” Sound Manager running on a machine with the Apple Sound Chip supports amplitude change.

### **SndStartFilePlay 20,000-byte minimum buffer size**

Written: 10/8/91

Last reviewed: 3/18/92

What’s the minimum buffer size for `SndStartFilePlay`?

—

Currently `SndStartFilePlay` needs a buffer of at least 20,000 bytes (approximately 20K) to output sound without problems on the slowest Macintosh models.

### **Playing 16-bit sound data from a Macintosh disk**

Written: 10/22/91

Last reviewed: 3/18/92

What’s the preferred method for playing sound from disk: with a `doubleback` routine and `SndPlayDoubleBack` or though the queue with our own routines? Does our 16-bit audio program have to convert the audio to 8-bit before playing through the Macintosh speaker or does the `DoubleBackPlay` or `StartSoundFilePlay` file playing routines do this for us?

—

The Sound Manager is not designed at this time to handle 16-bit sound data. You will need to convert the 16-bit data before giving it to the Sound Manager. The best method for playing from disk will be to make a `doubleback` routine and use `SndPlayDoubleBack`. You shouldn’t need to completely roll your own, but you’ll need to take care of getting the data from disk, converting it to 8-bit data and passing the data to the Sound Manager to be played. QuickTime will allow for the playback of 16 bit data.

### **SndStartFilePlay: no NIL sound channel for asynchronous plays**

Written: 12/19/91

Last reviewed: 3/18/92

SndStartFilePlay always returns a -205 badChannel error after the following call:

```
sndChanPtr := NIL;  
sndBufferPtr := NIL;
```

```
sndSelectionPtr := NIL;  
sndCompletionProc := NIL;  
sndErr := SndStartFilePlay(sndChanPtr, kSndFRefNum, resID,  
                           kSndBufferSize, sndBufferPtr,  
                           sndSelectionPtr, sndCompletionProc, async);
```

What's wrong here?

---

You need to pass SndStartFilePlay a valid, initialized sound channel if you plan on calling it asynchronously. The feature of passing NIL in to have the routine allocate a channel for you *\*only\** works if the asynchronous parameter is false.

## **Where to get Macintosh sound compression specs**

Written: 2/28/92

Last reviewed: 2/28/92

Where can I get technical specifications for Macintosh 4:1 sound compression?

---

The 4:1 and 8:1 sound compression techniques are part of Farallon's software, not part of Apple's system software. You must contact Farallon for technical specifications. You can contact Farallon at:

Farallon  
1321 Wakarusa Dr. #2010  
Lawrence, KS 66049  
(913) 843-8101

The 3:1 and 6:1 sound compression techniques are Apple's. Technical specification for both are in the MACE Specifications, available through APDA (part number M0241LL/A). You can contact APDA at:

APDA  
Apple Computer, Inc.  
20525 Mariani Avenue, Mail Stop 33-G  
Cupertino, CA 95014

AppleLink: APDA  
Phone:  
U.S. 800-282-2732  
Canada: 800-637-0029  
International: 408-562-3910  
FAX: 408-562-3971

## **Playing a sound from a Macintosh interrupt**

Written: 4/7/92

Last reviewed: 4/7/92

I queue a deferred task (which is already loaded and locked—the VBL doesn't move memory) to play a sound when my vertical retrace task checks for a particular condition and this

condition is met. The VBL is installed at INIT time, the deferred task is preloaded, and the sound is preloaded and locked. This works fine once the system has completed its startup, but if the VBL detects the condition before startup has completed, such as at INIT time, the Macintosh will freeze. Does the Deferred Task Manager need to be initialized somehow? Is there any Deferred Task Manager documentation besides what's in Inside Macintosh Volume V?

—

There's no additional information available on the Deferred Task Manager. Inside Macintosh Volume V covers it completely (however briefly). The Deferred Task Manager does very little; it's designed primarily to allow your interrupt routine to have interrupts re-enabled before processing information. Why is this important? Well, system services like AppleTalk require that interrupts not be disabled for much more than 1/1000th of a second usually. This means if you are doing something large during your interrupt routine (when all interrupts may be turned off), you could delay the AppleTalk interrupt so long that data gets lost. The Deferred Task Manager simply allows you to schedule a routine to get run at interrupt time, but while interrupts are enabled. This deferred task has to abide by two fundamental rules: (1) it must be interruptable—that is, you should not have to do anything such as timing-critical hardware access that can get screwed up with interrupts on, and (2) it must adhere to all the same tool/OS rules that an interrupt routine must follow. This means you cannot call any routine that may move memory; if you do, you can damage the current heap zone. If you image the flow of control for an interrupt/deferred task it looks something like this:

```
Interrupt stops application in the middle of doing something
      |
      System Interrupt handler decides who gets the interrupt
      (interrupts off)
      |
      Your interrupt handler gets called, decides that too much time needed
      You defer the task and return (interrupts still off)
      |
      System interrupt handler calls the rest of the interrupt routines
      that need to be called (interrupts still off)
      |
      System interrupt handler turns interrupts on (interrupts on)
      |
      Your deferred task gets run (interrupts on, but memory moving bad)
      |
      Application resumes and completes what it was in the middle of
```

There's no way to start the Deferred Task Manager; it's always running. The Sound Manager also is always running. The reason why this is particularly bad at init time for you is that when making the mistake of allocating memory at interrupt time, you can get away with it safely if the heap you're using (system heap) is not the application's current heap zone. At init time, the system heap is the current heap zone for the most part until an app gets launched. At init time your chances of harming the system by moving memory at interrupt time increases dramatically.

Inside Macintosh Volume VI documents both SndPlay and SysBeep as calls that move memory and cannot be used at interrupt time. Also, by passing a NIL channel to SndPlay you are forcing memory allocation every time, since the channel record needs to get allocated; you would be a little safer by pre-allocating a sound channel. Unfortunately SndPlay can still move memory, and many machines (68000 based) can only have one channel allocated at all, so you would then be robbing the system of its only sound channel by pre-allocating it.

Your best bet if you want a sound to play from an interrupt is to use the Notification Manager (Chapter 24 of inside Macintosh Volume VI) The Notification Manager is primarily designed

so background applications and interrupt tasks (like VBLs) can get the user's attention with a flashing icon, sound, dialog or any combination of the above. And you can simply pass it any sound handle you might want to use. The Notification Manager will simply delay your sound request until the system can play it back and not hurt anything.

## **Correct value of Macintosh initNoInterp**

Written: 5/28/91

Last reviewed: 3/18/92

Is there an error in Inside Macintosh Volume VI where it claims that both `initNoInterp` and `initChan0` have the same value of \$0004?

—

You're right, they're the same. According to the guy who is doing time attempting to straighten out the Sound Manager, "`initChan0` is ONLY used by the `waveTableSynth`, `initNoInterp` is ONLY used by the `sampledSynth`." I'm always grateful for meaningful constant names, aren't you?!

## **Determining Macintosh Sound Manager version**

Written: 8/30/91

Last reviewed: 3/18/92

The call `SndSoundManagerVersion` returns 2018000 for the System 7.0 Sound Manager and 2006000 for the System 6.0.7 Sound Manager, but it crashes if earlier Sound Manager versions are in use. `SndSoundManagerVersion` is a selector based trap call which calls the trap `_SndDispatch`. Before calling `SndSoundManagerVersion` you need to check if `_SndDispatch` is implemented. If so, then you can make the call. Otherwise you have the "old" version of the Sound Manager.

## **Using the Sound Manager to produce desired frequencies**

Written: 3/16/92

Last reviewed: 3/18/92

To generate arbitrary frequencies using sampled sounds, start with the original sample rate and know what frequency was sampled. For example, to play a pitch of 440 Hz sampled at 11 kHz as a 460 Hz pitch, use the ratio of 460/440 as a fixed point number and multiply this ratio by the original sample rate of 11 kHz. Put this product into the sound header's sample rate and use it with a `bufferCmd` to play your sound back at 460 Hz.

Consider overflows in your equations. Fixed point numbers have an upper range of 65K, otherwise they become negative numbers. Therefore, some sample rates and pitches cannot be created. You will not be able to take a 22 kHz sampled sound and play it back three octaves higher. You could use the Standard Apple Numeric Environment (SANE) to calculate the number with more precision, but would need to be careful with overflow. Also, you cannot use SANE at interrupt level.





Inside Macintosh Volume VI, which documents the Sound Manager, defines supported frequency values as 0-127. Arbitrary frequencies are not supported. The freqCmd is exactly the same as the freqDurationCmd but does not have an associated duration. The Sound Manager has no way of determining the original frequency of the sound. A baseNote of 56 may be a pitch of 440 Hz, but if the original pitch wasn't in equal temperament (a pitch represented by notes of a piano) it's impossible to represent this in the Sound Header.

Additionally, to support other frequencies a new sound command is needed. The freqCmd is documented as having its high byte used for amplitude, leaving three bytes to represent a frequency, but it should be a Fixed type number. This need was not realized, though, until after the MPW interface was frozen and an agreement had been made with developers not to change the API.

The Sound Header was changed when the original Macintosh Audio Compression and Expansion (MACE) package was released. This was done in 1989. The MPW Sound interfaces have been incorrect since then. The former field "baseNote" which was a word (with the high byte always being ignored) is now two fields. This change causes no problems for existing applications, but applications being compiled against the System 7 interfaces may need to be changed.

### **How can I play more than one Macintosh sample at a time?**

Written: 5/3/89

Last reviewed: 3/18/92

How can I play more than one Macintosh sample at a time?

—

Unfortunately, the Sound Manager prior to System 6.0.7 can play only one sample at a time. A basic limitation of the Sound Manager was that it only allows a single channel open with the sampled sound 'snth' and the note 'snth.' The wave table 'snth' allows up to four channels at a time, but doesn't provide the same capabilities as the sampled sound 'snth.' Using System 6.0.7 or later (which includes the "new" Sound Manager) you can play multiple sampled sounds at once by opening multiple sampled sound channels.

### **Why Macintosh system crashes when StartSound is launched twice**

Written: 5/3/89

Last reviewed: 3/18/92

My Macintosh application calls StartSound when it starts. The second time it is launched the system hangs. Why?

—

A bug in the InitApplZone trap is the cause of the bomb at the second launch. To avoid the

problem, simply call `StopSound` before exiting the program. It is also a good idea to call `StopSound` before calling `StartSound`, in case another program called `StartSound` but never called `StopSound`. The “glue” in MPW 2.0.2 and later versions fixes this problem.

Note that the Sound Driver is no longer supported. You should be using the Sound Manager to play sounds, and not the older driver calls such as `StartSound`.

X-Ref:

Macintosh Technical Note “How to Produce Continuous Sound without Clicking”

## **Changing the pitch of a sample in Macintosh Sound Manager**

Written: 5/3/89

Last reviewed: 3/18/92

How do I specify different pitches for the sampled sound synthesizer of the Macintosh Sound Manager?

—

You should be using the `freqDurationCmd` to specify a pitch. Note that these pitches are specified as MIDI note values. If you need pitches other than the pitches on the piano keyboard, then you'll have to calculate a new sample rate. (The `baseNote` field is only used internally by the Sound Manager.) You can change the sample rate in the sound header and play it back at any rate.

X-Ref:

Macintosh Technical Note “How to Produce Continuous Sound without Clicking”

## **Specifying different frequencies for the Macintosh Note Synth**

Written: 45/3/89

Last reviewed: 3/18/92

How do I specify different frequencies for the Macintosh Note Synth?

—

The Note Synth sample code on page 484 of Inside Macintosh Volume V is incorrect. You should follow the 'snd ' method described on page 492. The pitch value should be specified in decimal notation as explained on page 483. Also note that the interface files shipped with pre-3.0 releases of MPW may be incorrect. The following code produces a note:

```
{ Warning -- no error checking done here }
```

```
Procedure DoSound;
```

```
VAR
```

```
    myErr : osErr;  
    amp, pitch : LONGINT;  
    myChannel : sndChannelPtr;  
    myCmd : sndCommand;
```

```
BEGIN
```

```
    amp := $FF000000;  
    pitch := 60;  
    myChannel := NIL;  
    WITH myCmd DO BEGIN  
        cmd := noteCmd;  
        param1 := 1000;  
        param2 := amp + pitch;
```

```
END;
```

```
myErr := SndNewChannel(myChannel, 1, 0, NIL);
```

```
myErr := SndDoImmediate(myChannel, myCmd);  
myErr := SndDisposeChannel(myChannel, TRUE);  
END;
```

X-Ref:

Macintosh Technical Note “How to Produce Continuous Sound without Clicking”

## **How to minimize clicking sound when using Mac Sound Manager**

Written: 5/3/89

Last reviewed: 3/18/92

When I use the Sound Manager with MultiFinder, I hear clicks before and after my sound plays. How can I stop the clicks?

—

The clicking is a hardware problem. The speaker clicks whenever the sound is turned on or off. There’s nothing you can do about this. The Macintosh Tech Note “How To Produce Continuous Sound Without Clicking” explains a technique that minimizes the clicking.

Because the sound hardware can’t be shared among processes, if your program is using the Sound Driver and another process attempts to use the Sound Manager, it’s likely to crash the Mac. This includes SysBeep (in System 6.0 and later, SysBeep calls the Sound Manager). For this reason, your application should close its sound channel when it receives a “suspend” event and when it terminates.

In systems without an Apple Sound Chip (the Macintosh Plus and SE), the Sound Manager will either skip or repeat bytes, based on the sample and playback rates. If you use a buffer size that is a multiple of 370 bytes and the bufferCmd, the clicking will be minimized, because the Sound Manager keeps an internal 370-byte buffer. You don’t need to use a VBL task (as the this Technical Note suggests) when using the Sound Manager—simply call the bufferCmd for every chunk of data. The Sound Manager maintains a queue of all the commands. If 128 commands aren’t enough, you can use the callBackCmd to find out when your last command was processed, or you can create your own sound channel and increase the number of commands held in its queue.

X-Ref:

Macintosh Technical Note “How to Produce Continuous Sound without Clicking”

## **Macintosh Sound Manager changes in System 6.0.4**

Written: 5/14/90

Last reviewed: 3/18/92

What changes have been made to the Macintosh Sound Manager?

—

System 6.0.3 Sound Manager changes: none

## System 6.0.4 Sound Manager changes:

- The current Sound Manager is in the ROM of the Portable and IIfx.

- Sound Manager will not crash if the application calls `ExitToShell` without disposing its channels.
- Sound Manager and Sound Driver conflicts have been resolved. While the Sound Driver is in use, the Sound Manager will not be available.
- `SysBeep` will flash the menu bar when a sound is active (`SoundActive = TRUE`). This sometimes crashed on the Macintosh Plus and SE.
- `SndNewChannel` will return an error if an attempt is made to allocate a second sampled sound channel. Previously, it would return a non-functioning channel and `noErr`.
- Sound is no longer available in HyperCard XCMDs due to changes made in System 6.0.4 and HyperCard 1.2.5. HyperCard users have only the Play command. This was always unsupported by XCMD, but Sound Driver calls often worked. It was a risk and often crashed, but these conflicts were resolved in the Sound Manager. Now HyperCard will have to be revised to support an XCMD using sound.
- A new bug was created in 6.0.4. `SndPlay` will install a `callBackCmd` into the channel when a sampled sound 'snd ' resource is being used. This causes the user's call-back routine, if present, to be called unexpectedly. The call-back routine can witness this additional `callBackCmd` by looking at its user parameters. Here's what the bogus `callBackCmd` being received by the call back procedure looks like:

```
cmd    = callBackCmd
param1 = snd handle state bits, result of GetHState(sndHandle)
param2 = handle to 'snd '
```

This happens when you supply a sound channel to `SndPlay` with a format 2 'snd ' and specify `aSynch`. It looks like this:

```
SndNewChannel(newChan, 0, 0, myCallBackProc);
SndPlay(newChan, format2Snd, aSynch);
```

The `callBackCmd` has two parameters that are for the programmer's use. The call-back routine is called at interrupt time, and the routine needs to set a global flag. As a solution, store a unique value in `param1` before issuing the `callBackCmd`. While in the `callBack` procedure, test for this value to determine if it is the correct `callBackCmd`.

### **SetupAIFFHeader bug**

Written: 4/1/92

Last reviewed: 5/21/92

I may have found a bug in the SetupAIFFHeader routine. When trying to create a 44.1 kHz AIFF file header, the sample rate is converted into a negative extended value. I think SetupAIFFHeader calls Fix2X which takes a signed fixed value and returns a signed extended value. Please let me know if this is a bug.

—



You are absolutely correct. The Sound Manager routine SetupAIFFHeader does call Fix2X which in turn assumes that it is being passed a signed Fixed number. Until the bug is fixed, DTS recommends that you fill in the sampleRate field of the header yourself "

### **How to get MACE algorithms**

Written: 11/9/90

Last reviewed: 3/18/92

What are the Macintosh Audio Compression & Expansion (MACE) algorithms that will allow me to translate the compressed digitized audio from my device into compressed 'snd ' resource format?

—

The MACE algorithms are proprietary, but they are licensable. Contact Apple's Software Licensing department for more information. You can reach them at (408) 996-1010 or at AppleLink SW.LICENSE.

### **Macintosh System 6.0.5/MACE problem fixed for 6.0.7**

Written: 2/8/91

Last reviewed: 3/18/92

After playing a MACE-compressed sound several times under System 6.0.5, my Macintosh SE eventually hangs. The same code works fine under System 6.0.4 and System 6.0.7. Do you have any clues as to what is causing this?

—

MACE was shipped originally as a strange hack to System 6.0.4. It was not revised for System 6.0.5 and as a result, it has never worked under 6.0.5. In System 6.0.7, MACE became part of the Sound Manager, so it should work fine on all systems from 6.0.7 on. Unfortunately, it is not going to ever work under 6.0.5.

### **MACE-compressed sound -201 error**

Written: 12/20/91

Last reviewed: 3/18/92

We get a -201 (not enough sound hardware available) when we try to get our backlit Macintosh Portable (not PowerBook) to play MACE-compressed sounds. What are we doing wrong?

—

You're probably either (1) incorrectly opening the sound channel or (2) trying to open too many sound channels or (3) being honest and using the InitMACE3 or InitMACE6 options

to SndNewChannel. Take a look at how SoundApp (on the Developer CD Series CD) deals with MACE sounds.

The InitMACE3 and InitMACE6 options are used only when the Sound Manager is trying to figure out how much of the processor time it can use; they have no bearing on the actual playing of the sounds. By not specifying these initialization options, the Sound Manager will

give you a channel and happily play your compressed sound on the PowerBooks. This is a known “feature” of the Sound Manager.

### **MPW C sound.h header file fix**

Written: 11/21/89

Last reviewed: 12/17/90

I’m having trouble compiling & linking with the sound routines and MPW C 2.0.2.

—

The “sound.h” header file shipped with MPW C 2.0.2 was incorrect. The correct header file is available on AppleLink in the Developer Services Bulletin Board, and is also shipped with MPW 3.0.