
Chapter 13 Sound & the Sound Manager

Need help with the new sound manager! (with code).....	208
Synchronous sounds using the Sound Manager.....	209
Synchronous sounds using the Sound Manager.....	209
How to get rid of clicking?.....	210
Audio Phone Dialer Software.....	211
Simultaneously played digitized sounds.....	212

USENET Macintosh Programmer's Guide

From: stoms@castor.ncgia.ucsb.edu (David Stoms)

Subject: Re: Need help with the new sound manager! (with code)

In article <900004@hpcvcs1.HP.COM> stevem@hpcvcs1.HP.COM (Steve Miller) writes:

>OK boys and girls, I'm about at my wits end on the subject of playing 'snd'
>resources on the Mac. I'm trying to use snd's in a simple game where the
>sounds are played asynchronously while arcade style animation is occurring.

I found this in one of my text directories (I can't verify that this is right)----

In article <1354@ndmath.UUCP> milo@ndmath.UUCP (Greg Corson) writes:

>Would anyone happen to have a code fragment that illustrates how to setup
>the required callback routines so you can have SoundPlay play a sound in the
>background (async) mode?
>
>1. open the channel and specify a callback
>2. play a snd resource
>3. have the callback set a global flag
>4. have my main event loop close the channel when the flag is set.

This is the code I am using. There is NO guarantee or warranty with this code. Use at your own risk! This is NOT Apple DTS sanctioned code.

1. the main application needs to keep a global variable is a SndChannelPtr. Initialize it to NIL.
2. Pass in the handle to the snd resource that you want to play.

This should bootstrap to you greater understanding of the world of the Sound Manager.

```
PROCEDURE PlayIt(VAR chan: SndChannelPtr; theSound: Handle);
VAR
    err: OSErr;
    myWish: SndCommand;

BEGIN
    IF chan <> NIL THEN {cancel sound in progress}
    BEGIN
        err := SndDisposeChannel(chan, TRUE); {TRUE means now}
        IF err <> noErr THEN
        BEGIN
            DebugStr('Cannot dispose of old channel');
            Exit(PlayIt);
        END
        ELSE
            chan := NIL;
    END;

    err := SndNewChannel(chan, 0, 0, NIL);
    IF err <> noErr THEN
    BEGIN
        DebugStr('Cannot allocate new channel');
        Exit(PlayIt);
    END;

    IF theSound <> NIL THEN {we have a handle}
    IF theSound^ <> NIL THEN {and is not a nil handle }
    BEGIN
        err:=SndPlay(chan, theSound, TRUE);
        IF err <> noErr THEN
```

```
BEGIN
    DebugStr('Cannot play sound');
    Exit(PlayIt);
END;
```

USENET Macintosh Programmer's Guide

```
WITH myWish DO { set up a sound mgr command block }
BEGIN
  cmd := freeCmd; {to free self when done }
  param1 := 0;
  param2 := 0;
END; {with}

err := SndDoCommand(chan, myWish, FALSE);

IF err <> noErr THEN
BEGIN
  DebugStr('cannot add command');
  Exit(PlayIt);
END;
END;
```

• • •

From: Isr@Apple.COM (Larry Rosenstein)

Subject: Re: Synchronous sounds using the Sound Manager

In article <30976@ut-emx.UUCP> wras@walt.cc.utexas.edu (Steve Mariotti) writes:

```
> err = SndNewChannel(myChanPtr, sampledSynth, initMono, NULL);
> if (!err) {
>   err = SndPlay(myChanPtr, soundHandle1, FALSE);
>   err = SndPlay(myChanPtr, soundHandle2, FALSE);
> }
```

First, to play a sound while your programs continue, you need to pass TRUE for the last parameter. This specifies asynchronous playing, which is what you want.

Also, you can't (in general) play 2 snd resources with the same channel. When you play a resource, the synthesizer required by the resource is linked to the channel. If you link a synthesizer more than once, you will (at best) not get any sound, and will probably crash.

For this reason, it is best to dispose of a channel after the sound is finished playing. To do this pass a call back routine when you create the channel, and send a callBack command after the SndPlay.

For the same reason, you should pass 0 instead of sampledSynth when creating the channel, in case the resource you play wants to use a different synthesizer.

If you know the structure of the resources you are going to play (ie, they have no synthesizer information) then you can play several sounds on the same channel. Or you can parse the sound resource to extract the actual samples and play them.

All this information comes from the revised Sound Manager documentation, which I think is available on Apple.COM (in /pub/dts/mac/docs). I've also got sample MPW Pascal code which implements an asynchronous SndPlay call.

Larry Rosenstein, Apple Computer, Inc.

• • •

From: cak3g@astsun9.astro.Virginia.EDU (Colin Klipsch)

Subject: Re: Synchronous sounds using the Sound Manager

Summary: Careful with SndNewChannel!

In article <8548@goofy.Apple.COM> Isr@Apple.COM (Larry Rosenstein) writes:

Sound & the Sound Manager

>In article <30976@ut-emx.UUCP> wras@walt.cc.utexas.edu (Steve Mariotti)
>writes:

USENET Macintosh Programmer's Guide

```
>> err = SndNewChannel(myChanPtr, sampledSynth, initMono, NULL);
>> if (!err) {
>>     err = SndPlay(myChanPtr, soundHandle1, FALSE);
>>     err = SndPlay(myChanPtr, soundHandle2, FALSE);
>> }
```

A minor point that I missed on first reading of the Sound Manager chapter: the first parameter is a VARIABLE sound channel pointer; i.e. the calling definition of SndNewChannel goes like:

```
function SndNewChannel (VAR: chanPtr; ...)
```

In assembly language this requires extra care. You must be sure that you:

- a) set your chanPtr to NULL or to the pointer of your own created pointer-block (using _NewPtr, size 1060 bytes);
- b) pass the ADDRESS of your chanPtr to _SndNewChannel, NOT the pointer itself.

So it goes like this. . . (with the appropriate push/pop macros)

```
clr.l myChanPtr(A5); if you're lazy; otherwise make your own
push.w #0           ; space for the function result
pea myChanPtr(A5); NOT push.l myChanPtr(A5)!
push.w #0           ; let SndPlay link a synthesizer typically
push.l #0           ; (is this a longint? my docs aren't handy)
pea MyCallBack      ; you definitely want this if you're going
                    ; to be playing asynchronously
_SndNewChannel
pop.w D0            ; error, if any
...                ; and on and on
```

```
MyCallBack      ; This routine will execute in response to a callBack
                ; command, which you pass to your channel with _SndDoCommand
                ; after you're finished with the channel. This routine
                ; should set a flag somewhere indicating the channel
                ; is to be killed with _SndDisposeChannel as soon as is
                ; convenient, but this routine should NOT do the disposing
                ; itself, since it is not allowed to use the Memory Manager.
```

I realize the original poster was using Pascal, not assembly language, but I hope this can help anyway.

```
>Also, you can't (in general) play 2 snd resources with the same channel.
>When you play a resource, the synthesizer required by the resource is
>linked to the channel. If you link a synthesizer more than once, you will
>(at best) not get any sound, and will probably crash.
```

Precisely, although I have never understood why the Sound Manager works in this way. Why can't SndPlay check to see if a particular synthesizer has already been linked to this channel, and if so, don't try relinking? SndPlay could be much more robust, it seems to me, but then again I wouldn't want to have to write it. . .

Colin Klipsch

•••

From: oster@well.sf.ca.us (David Phillip Oster)
Subject: Re: How to get rid of clicking?
Keywords: Sound Driver

In article <1990Jun4.020601.9644@msuinfo.cl.msu.edu> gurney@cpsin2.uucp (Eddy J Gurney) writes:

```
>Also, while I'm posting... has anybody come up with a way to specify a
>FREQUENCY to FTSoundRec.sound1Rate instead of the rate? IM II gives the
>equation:
```

```
> frequency = FixMul(rate,FixRatio(22257,256))
>I solved this for frequency, and got: (Actually, I had my HP-48SX do it. I
>didn't want to be flamed for not solving correctly :-))
> rate = FixRatio(frequency,FixRatio(22257,256))
```

USENET Macintosh Programmer's Guide

in floating point, this is:

```
rate = frequency / (22257. / 256.);
```

Or, in fixed point:

```
rate = FixDiv(FixRatio(frequency, 1), FixRatio(22257,256))
```

If the FixRatio(frequency, 1) offends you, you can also say:

```
#define FIXED(f) ( ((long) (f)) << 16)
and say    FIXED(frequency)
```

--

-- David Phillip Oster -

•••

From: oster@well.sf.ca.us (David Phillip Oster)

Subject: Re: Audio Phone Dialer Software

In article <3397@ssc-vax.UUCP> housen@ssc-vax.UUCP (Kevin Housen) writes:

```
>he needs some software (preferably a DA) to generate
>appropriate dial tones. This would be pretty easy to
>write, but I wanted to check first to see if there are
>any PD/shareware DAs that will do the trick.
```

Both hypercard and Address Book Plus (a commercial product from Power Up Corporation, that I wrote) dial the phone by generating audio tones. AB+ uses a d.a., but it is commerical software, not shareware or public domain.

You can just hold the reciever up to the Mac on later Macs, but my testing shows that not all raw MacPluss have enough volume to generate all the tones reliably enough for the phone system. Experiment.

To dial the phone, you need two simultaneous tones. Hypercard and I use the Sound Driver's 4 voice synthesizer to produce two pairs of tones by doubling up the voices. Apple Developer Tech Support has been saying for some time that they want people to stop using Sound Driver and do everything with Sound Manager. Unfortunately, at least as of System 6.0.4, 4 voice sound was not supported by Sound manager on Macpluss. Here are the numbers from my dialer:

```
typedef struct Freq2 {
    Fixed a, b;
}Freq2;

/* These are the frequencies for the touch tone phone pad.
   Source: The TV Typewriter Cookbook, Don Lancaster, p.178
   Units for the table in comments is Hertz.
   The actual table was generated by applying the formula:
   hz = 1000000 / ( 44.93 * 256 / rate )
   to the table to get a new table I can use at interrupt time.
   This table uses the same values hypercard does.
*/
static Freq2 freqs[] = {
    { 0x000F5DDE, 0x000AD2C9}, /* { 1336, 941}, /* 0 */
    { 0x000DE7EA, 0x00080453}, /* { 1209, 697}, /* 1 */
    { 0x000F5DDE, 0x00080453}, /* { 1336, 697}, /* 2 */
    { 0x0010FD06, 0x00080453}, /* { 1477, 697}, /* 3 */
    { 0x000DE7EA, 0x0008DB46}, /* { 1209, 770}, /* 4 */
    { 0x000F5DDE, 0x0008DB46}, /* { 1336, 770}, /* 5 */
    { 0x0010FD06, 0x0008DB46}, /* { 1477, 770}, /* 6 */
    { 0x000DE7EA, 0x0009CCB9}, /* { 1209, 852}, /* 7 */
    { 0x000F5DDE, 0x0009CCB9}, /* { 1336, 852}, /* 8 */
    { 0x0010FD06, 0x0009CCB9}, /* { 1477, 852}, /* 9 */
    { 0x0012C9B2, 0x00080453}, /* { 1633, 697}, /* A */
```



```
{ 0x0010FD06, 0x0008DB46},    /* { 1477, 770},    /* B */
{ 0x0010FD06, 0x0009CCB9},    /* { 1477, 852},    /* C */
{ 0x0010FD06, 0x000AD2C9},    /* { 1477, 941},    /* D */
```

USENET Macintosh Programmer's Guide

```
    { 0x000DE7EA, 0x000AD2C9},    /* { 1209, 941},    /* * */
    { 0x0010FD06, 0x000AD2C9}    /* { 1477, 941},    /* # */
};
--
-- David Phillip Oster -
```

...

From: jmunkki@hila.hut.fi (Juri Munkki)
Subject: Re: Simultaneously played digitized sounds
Keywords: divide

In <25974@pasteur.Berkeley.EDU> eta@ic.Berkeley.EDU.UUCP (Eric T. Anderson):

```
>Hey -- be careful. You know you're throwing away bits for no reason
>if you pre-divide. How about add first and then post-divide? On a
>Mac, doing byte-addition is just as quick as 16-bit addition (except
>if you're adding constants). Right?
```

Every cycle counts when you are doing things like this. I found it optimal to pre-divide. I also used only 11 Khz sound. Doing two 11 Khz sounds withpre-divide doesn't add all the much overhead, so my animation routine were not hurt too badly. Two sound channels is infinitely better than just one.

Juri Munkki Macintosh Support jmunkki@hut.fi PS /