Sounder 2.01: Digitalized Sound Player for Windows 3.0

(See the "Changes from Sounder 1.0" section if you are familiar with Sounder 1.0)
(See the "Changes from Sounder 2.01" section if you are familiar with Sounder 2.0)

Sounder allows you to play digitalized sound files recorded in the popular Macintosh sound format on your PC. Better yet, no sound board is required, although a 10 MHz or faster machine is. This program was inspired by programs like REmac and Talk that operate on the principle of speeding up the system clock to a ridiculous speed, turning off everything else, and faking dynamic modulation with the imperfect dynamics of the PC speaker system and the human ear. For such a hack, the sound is actually pretty good, although the quality depends on the actual sound file and also may vary from system to system and speaker to speaker.

Sound files can be gotten via anonymous FTP from terminator.cc.umich.edu, sumex.stanford.edu (these are Mac files, so you'll need access to BinHex and Unstuffit as well as a Mac and something called Apple File Exchange), and other sites. Any Mac-format file should work. The newsgroup alt.tv.simpsons seems to generate a few sound bites from recent episodes now and then, too.

The format of Mac sound files is simply a string of bytes, each of which represents the volume at a given sampling time. Most Mac sound files are recorded at either 11 or 22 KHz, and Sounder can handle these as well as 7.33 and 5.5 KHz files. The de facto extension for these files is .SOU, which I see no reason to change, but...

Since Mac sound files by default carry no frequency or other information (at least in their PC incarnation), it is necessary to include this information on the command line when using Sounder. The Program Manager can handle this task very well, but to make life easier in the future, I propose an ad hoc file format, the .SND format, described later. When a .SND file is played by Sounder, frequency and such information internal to the file is used. It is relatively easy to convert a .SOU to a .SND file. Also, the internal .SND parameters can be overridden by passing command-line arguments. A DSOUND sound resource format is also described later.

Disk Contents

You should have the following files; if not, download a complete version of the archive:

SOUNDER.EXE	Program to play sounds
SOUNDER.WRI	This file
SNDCNTRL.EXE	DSOUND Control Panel program
DSOUND.DLL	Digital sound driver
DSOUND.LIB	See DSOUND.DLL section below
DSOUND.H	Ditto.
BEAM.SOU	Sample sound file ("Beam me up,

Scotty")
20TH_CEN.SND — Sample .SND file (20th Century Fox intro)
ILLCHEER.SOU — Another sound file

Using Sounder

To use Sounder, copy DSOUND.DLL into your Windows system directory (\WIN\SYSTEM, usually) and SOUNDER.EXE and SNDCNTRL.EXE to somewhere on the PATH. I use a \SOUND directory for sound files and SOUNDER.EXE. If you're like me and most Mac users, you'll probably collect several megabytes of The Simpsons, Star Trek, Mission Impossible, and other oldies.

The DSOUND Control Panel should then be added to an appropriate Program Manager group for easy access, using the File New menu option.

Before trying to play a sound, it is necessary to set a few parameters using the DSOUND Conrol Panel *especially if you are using 386 Enhanced mode.* The correct values for the parameters will depend on which mode you are running Windows in. Note that they must be set for each mode you plan to use *while Windows is in that mode.*

The DSOUND Control Panel will tell you which mode you're currently running in. Also verify that the DSOUND version is 2.0. The Control Panel will not work with DSOUND.DLL 1.0 (shipped with Sounder 1.0).

If you are in **Real** or **Standard** mode, initially set the volume to 10 (use the slider above it) and check the Use Timer box. Press the Test button. Adjust the volume to your liking (note that the volume of individual sounds can be set as well; the test sound is a "typical" sound). If the sound has a lot of static, sounds unrecognizable, or otherwise is not Bart saying, "Oh, come on, I'll cheer you up," it may be necessary to use a delay value instead of the timer as explained for 386 Enhanced mode.

If you are in **386 Enhanced Mode**, the Set Timer box will be grayed. It will be necessary to set a delay value that is specific to your machine and hardware. DSOUND cannot figure this out on its own due to limitations of 386 Enhanced mode. 16 works well with a 386sx-20. Use the Test sound to determine if the tempo of the playback is correct. If the sound plays back too quickly, bump up the delay value; if it's too slow, bump the delay down. Adjust the volume to your liking.

When done, press the Exit button. This will save stuff in WIN.INI as described later on if you're curious.

You should then associate all *.SND and *.SOU files with Sounder so that you can double-click on the sounds to play them. Do this by either hacking the WIN.INI file or by selecting such a file in the File Manager and choosing File Associate... to associate them with SOUNDER.EXE. Sound files can then be dragged into a Program Manager group where they'll assume the icon of Sounder (a bit stream

going into a speaker, and a waveform coming out). In the Properties... dialog box, command line parameters specifying the format of the sound file should be included, as described below.

Sounder is invoked with the following command-line syntax:

SOUNDER *sound-file* [S] *frequency volume shift sample_type*

where *sound-file* is a .SOU or .SND file, S is an optional Save flag (described later),*frequency* is the desired playback frequency in Hertz (default: 22000), *volume* is the playback volume parameter (default: 10), *shift* can be altered to compensate for poor recordings or different hardware (default: 4), and *sample_type* is the type of sample data (*sample-type* is currently ignored, but should be 0 for future compatibility if specified). If a .SND file is specified and 'S' (or 's') is specified after the filename and before the other parameters, the settings specified will be permanently saved in the file; no sound will play. Fewer than 4 parameters can be passed, but they'll be assigned in the order above. That is, if you assign a value to a parameter, all preceeding parameters must be passed as well to act as "place holders". *All* parameters should be passed if the S option is included, otherwise default values will be used.

Instead of spaces, commas and tabs are valid delimiters.

If you have placed SOUNDER on the PATH and associated .SOU and .SND files with it, you can omit SOUNDER from the command line. This will be the case with sound files dragged into the Program Manager. It's easiest to drag the sound files into a Sound group and simply add the frequency and volume parameters in the Command line field of the Properties... dialog box:

Sounder has no window, and the only non-error feedback is that the system pauses while the sound plays.

Note: Because of the way Windows parses the run= statements in WIN.INI, a statement like

run=20th_cen.sou 11000

causes the sound to be played at the default 22000 Hz and a file named "11000" to be run (with an error, of course). This is probably not what you want. I suggest creating and using .SND files instead, as will be described later:

run=20th_cen.snd

This will allow you to have a start-up sound.

Errors:

Sounder will complain if:
- The wrong version of DSOUND.DLL is detected
- It can't find the sound file specified, or the file cannot be opened
- There was no sound file specified
- There is not enough memory to load the sound file (all of it at once, unfortunately for you real-moders out there)
- Any other file-related errors it catches spring up.

Getting the Paramters Right (Once the Control Panel Settings Have Been Verified)

If the sound file seems to be playing too slowly, increase the frequency; likewise, if it's played too quickly, decrease the frequency. Although any frequency can be passed, currently sounds are played only at 22000, 11000, 7333 and 5500 Hz. The closest of these to the specified frequency is used.

If there is a lot of noise, "static," or a high-pitched whine, try adjusting the shift value, which can help "center" the sound samples within the dynamic range of your hardware. 4 is default, but small variations do make a difference with some recordings.

The playback volume can be changed with the volume parameter, but there is an upper limit to how loud a sound file can be played before the quality of the sound decays (a limitation of the PC speaker). This depends on the sound file and could be from 30 to 100. There is no lower limit, except that at 0 nothing will be heard except background noise.

Converting a .SOU (parameterless sound) file to a .SND (embedded parameter) file

The 'S' (Save) option described above may be used to create a .SND file. Simply rename or copy a .SOU file to have the .SND extension, then run Sounder with the 'S' options and the desired frequency, volume, shift, and sample-size. This can be done with the File Run... option in the Program manager; for example:

BEAM.SND S 11000 15 4 0

could be used as the Command line to save the settings in BEAM.SND after renaming BEAM.SOU to BEAM.SND, assuming *.SND files are associated with Sounder as they should be.

A very small amount of sound information will be overwritten using this approach, but it'll be too slight to notice. I plan on having a better .SOU to .SND conversion utility later, if need be...

Changes from Sounder 1.0

The only file that's been changed since Sounder 1.0 is DSOUND.DLL. The changes to DSOUND.DLL allow for operation in 386 Enhanced mode (Yay!!!).

Your WIN.INI file should have the following section and keys in it:

[DSOUND]
EDelayValue=15
SDelayValue=0
RDelayValue=0

The settings that appear above are probably a good first stab at getting the correct values; they work on my 386sx-20. They can be set with the DSOUND Control Panel.

Each value refers to a delay value that is used to pace the replay of the sound; the larger the value, the slower the replay will be. The value of 0 is special: it means to use the internal timing hardware to get the correct replay pacing instead of a loop. Checking the Use Timer box sets the entry for the current mode to 0 in WIN.INI. The timer approach is the same as what was used in DSOUND.DLL 1.0, and should be used if possible.

The delay value that is used depends on which mode Windows is running in. EDelayValue is used in Enhanced mode, SDelayValue in Standard mode, and RDelayValue in Real mode. In general, SDelayValue and RDelayValue should be 0 (but see below). EDelayValue cannot be 0; you'll hear a beep and Sounder will refuse operation if this is the case.

Some computers seemed to have problems with Sounder 1.0--I suggest trying a non-zero delay value for Standard and Real modes.

An incompatibility with ATM in Standard mode was fixed. It *was* my fault, but only under very specific circumstances did Windows mind.

Changes from Sounder 2.0:

This release is basically a bug fix and clean-up. The main problem was that the speaker would come back to life if Sounder had been played in Windows (Enhanced mode), then a DOS box played a sound, then you returned to Windows. The culprit was a few missing lines of assembly code that weren't copied from the Standard mode driver. Yuk.

There was also a problem in the Control Panel that no one complained about, but it is fixed. It involved coordinating the Use Timer button and the Timer Value edit control better. Another "bug" was that the version number was displaed as 2.x instead of 2.xx, causing this version to appear to be 2.1 in the Control Panel.

Ugly News:

Sounder stops everything dead in its tracks while the sound plays. No mouse, no communication downloads, no nothing. The floppy light stays on. The wait cursor stays on because Windows *can't* turn it off after loading the file. 22 KHz is fast, too fast to let things like these happen. Sounder needs to time things very precisely, and other interrupts would result in clicks and clucks in the speaker. Note that the system time also freezes; I could fix this if it were really a problem.

This is why it's not possible for Sounder to figure out the correct delay values discussed above: while playing sounds, Sounder cannot tell how much time has elapsed, so it can't tell if it played the sound too quickly or not!

Future Attractions:

What's up with DSOUND.DLL? I'd love to see something like SoundMaster for the Mac under Windows. I may even write it myself. But don't hold your breath. DSOUND.DLL has all the playback code in it, so if anyone wants to write a shareware/PD Windows sound program, feel free to use DSOUND.DLL.

DSOUND.DLL (For Windows Programmers Mainly!)

Don't you always hate it when a neat app comes out that has a widget you'd like to use, only you can't. Me, too. So I isolated Sounder's playback code into a .DLL file. Applications that you write can access DSOUND.DLL. The main function in DSOUND.DLL as prototyped in DSOUND.H is:

> void FAR PASCAL PlaySound(*lpSound, dSize, uFrequency, uSampleSize, uVolume, uShift*)

lpSound is a long pointer to a chunk of memory with the samples in it, one byte each. This should be locked but need NOT be in real address space. Page-locking, though not needed, will prevent the sound from stuttering as the sample bytes are paged into memory. lpSound is strictly speaking a char huge *; samples can be much longer than 64K, up to the maximum size GlobalAlloc'able under the current mode.

dSize is an unsigned long that tells how many samples there are in lpSound. Currently, this should be the size of the sound data in bytes. Be accurate; Windows complains (dies) if DSOUND.DLL accesses memory you don't own.

uFrequency is the desired playback frequency. Currently 22 KHz, 11KHz, 7.33 KHz, and 5.5 KHz are supported. However, to be open-ended about this all, all frequencies are "valid," and the nearest one is used. That is, 9 KHz will play at 11 KHz, and 44 KHz will play at 22 KHz. I'll probably add 44 KHz functionality if I can optimize the play loop enough.

uSampleSize is a code for the sample size. Currently only 8-bit linear is supported, which has the code 0. This parameter is currently ignored, but future versions may support 16-bit sample sizes, log scales, and such.

uVolume adjusts the dynamics of the sound before playing. Basically, the amplitude of the sound centered around a middlepoint is expanded/compressed by the factor (uVolume/10). Thus, a volume of 5 is about half as loud, and 20 is twice as loud. In theory, that is--the PC speaker is not all that dynamic.

uShift is an amount added to each sample before being processed (but after the volume has been accounted for). This wouldn't be useful on a real sound chip, but it allows for compensating different speaker hardware in this hack. 3-5 seem to work best; 4 is the default Sounder uses.

Also defined in DSOUND.DLL is:

```
int FAR PASCAL GetDSoundVersion()
```

which returns the version of DSOUND.DLL as an integer whose upper byte is the major revision and whose lower byte is the minor revision. Currently this is 2.00, which supports the features described above in all current operating modes of Windows 3.0.

New to version 2.0 are:

```
void FAR PASCAL SetDelayValue(int d)
int FAR PASCAL GetDelayValue()
void FAR PASCAL SetVolume(int v)
int FAR PASCAL GetVolume()
```

which can be used to get and set the delay value and/or volume dynamically. It doesn't change the value in WIN.INI, and setting the delay value to 0 in Enhanced mode is bad. These are mainly for use by the Control Panel, but are available for other programs.

To use DSOUND.DLL by linking implicitly, DSOUND.H should be included in your source files that access it and *either* DSOUND.LIB should be linked with your files or you should put:

```
IMPORTS
        PlaySound=dsound.1
        GetDSoundVersion=dsound.2
        SetDelayValue=dsound.3
        GetDelayValue=dsound.4
        SetVolume=dsound.5
        GetVolume=dsound.6
```

in you .DEF file. As I eventually learned from Martin ???, doing both these causes problems!

DSOUND.H prototypes all exported functions and defines a few useful constants.

There are other ways of dynamically linking a DLL that vary from environment to environment; see the appropriate documentation for details (i.e. Actor).

.SND Files and Other Formats

The format of a .SND file is quite simple; the file has the following words, followed by the sample bytes:

word 0 Sample size code (see uSampleSize above)
word 1 Frequency to play back at (see uFrequency above)
word 2 Volume to play at (see uVolume above)
word 3 Shift (see uShift above)

Others have proposed a more robust file/clipboard/resource format, one which has a magic number, description field, and such. For clipboard formats, Martin Hepperle's Sound Tool uses the following format, to be registered under CF_SOUND. This format should also work for sound resources. The actual sample bytes follow the header.

```
#define  FILENAME 96          /* max. length of a filename */
typedef struct sound_tag
 {
  GLOBALHANDLE hGSound;      /* not used for clipboard transfer */
  DWORD dwBytes;             /* length of complete sample */
  DWORD dwStart;             /* first byte to play from sample */
  DWORD dwStop;              /* first byte NOT to play from sample */
  unsigned short usFreq;
  unsigned short usSampleSize;
  unsigned short usVolume;
  unsigned short usShift;
  char szName[FILENAME];         /* name of sound */
 } SAMPLE;

usFreq must have one of the following values:
{ 5500, 7330, 11000, 22000 }
```

For disk files, a "magic number" like "DSOUND", control Z, followed by the header above would probably work well.

Terms:

"While the prospect of making a few bucks off this is tempting, reality says that no one pays shareware fees," I wrote in the docs for version 1.0. Proof: only two people registered, even at $1. The license to use it is now $2. Maybe by the time I'm at version 10 I'll actually make a bit of money on this! Checks can be sent to my P.O. Box, and as far as I'm concerned, you need only pay once for Sounder and any possible upgrades.

If you want to distribute DSOUND.DLL with a program that needs it,

here are the terms:

o If your program is freeware, distribute all programs in this
  distribution with it. Mention that the DSOUND.DLL driver is not
  freeware or public domain.
o If your program is shareware, it should be made clear that licensing
  DSOUND.DLL is not included in the cost of licensing your program.
o For commercial applications, contact me.

Oh, and if you do distribute this, don't modify it. It's hard enough fixing my own bugs...

Acknowledgements

I loved REmac, only it didn't work under Windows. I'm not exactly a DOS-type person anymore. A snoop or two with Debug suggested it could be done under Windows after extensive rewriting to make it protected-mode friendly and all. A few other programs (TALK/PLAY) showed how to get around a few other problems. I'm indebted to the authors of these programs for "showing that it could be done."

Kudos also to Jackie Burmas, Jeffrey Belt, and David Feinleib, the only people I know of who registered (i.e. sent $$$ for) for Sounder 1.0. Many thanks to them!

Thanks also to Martin Hepperle, for finding out that you can't IMPORT and link import libraries at the same time. Keep an eye out for his Sound Tool , a great digital sound editor.

Compatibility

Sounder and DSOUND have been tested on a IBM PS/2 50 and 80, Zenith 386/33, a C + T-based 386sx, and a C + T-based 286-12. It does mingle with a lot of hardware-specific stuff, but everything it does is documented and otherwise kosher.

Trivia

On Micro Channel machines, the timer interrupt condition must be cleared with a write to port 61H with bit 7 set to prevent repeated interrupts. Thing is, this doesn't have to be done in real mode, at least on the PS/2 50. I've noticed that a lot of programs (i.e. REMac) don't clear the interrupt condition at all. Be warned, and be sure to test on nonstandard architectures like Micro Channel.

Aaron Wallace
P.O. Box 13012
Stanford, CA 94309-3020

aaron@jessica.stanford.edu