Details

The wave's time per point, as determined by its X scaling, must be a valid sampling rate. A value of 1/44100 (CD standard) is typical.

Sound waves can be 8, 16, or 32-bit signed integer waves, or single-precision floating point waves:

- 8-bit integer waves support a sample range of -128 to +127.
- 16-bit inter waves support a sample range of -32768 to +32767.
- 32-bit integer waves are used to hold both 24-bit and 32-bit audio. Use /BITS=24 to tell PlaySound that the wave contains 24-bit instead of 32-bit values.
- 32-bit floating point waves support a range of -1 to +1.

Do not use complex waves with PlaySound.

To play a stereo sound, provide a 2 column wave with the left channel in column 0. Actually, the software will attempt to play as many channels as there are columns in the wave. You can also use multiple1D waves with the /A flag. To use this method, enclose the list of 1D waves in braces

With the /A flag, the sound plays asynchronously (i.e., the command returns before the sound is finished). If another command is issued before the sound is finished then the new command will wait until the last sound finishes. A PlaySound without the /A flag can play on top of the current sound. The transition between sounds should be seamless on Macintosh but may be slightly delayed on Windows.

It is OK to kill the sound wave immediately after PlaySound returns even if the /A flag is used.

Examples

Under Windows, support for sound is somewhat idiosyncratic so these sound examples may not work correctly with your particular hardware configuration.

```
Make/B/O/N=1000 sineSound                   // 8 bit samples
SetScale/P x,0,1e-4,sineSound               // Set sample rate to 10Khz
sineSound= 100*sin(2*Pi*1000*x)             // Create 1Khz sinewave tone
PlaySound sineSound
```

The following example will create a rising pitch in the left channel and a falling pitch in the right channel:

```
Make/W/O/N=(20000,2) stereoSineSound        // 16 bit data
SetScale/P x,0,1e-4,stereoSineSound         // Set sample rate to 10Khz
stereoSineSound= 20000*sin(2*Pi*(1000 + (1-2*q)*150*x)*x)
PlaySound/A stereoSineSound                 // 16 bit, asynchronous
```

Multichannel sounds as in the previous example but from multiple 1D waves:

```
Make/W/O/N=20000 stereoSineSoundL,stereoSineSoundR   // 16 bit data
SetScale/P x,0,1e-4,stereoSineSoundL,stereoSineSoundR// Set sample rate to 10Khz
stereoSineSoundL= 20000*sin(2*Pi*(1000 + 150*x)*x)// rising pitch in left
stereoSineSoundR= 20000*sin(2*Pi*(1000 - 150*x)*x)// falling in right
PlaySound/A {stereoSineSoundL,stereoSineSoundR}   // two 1D waves
```

See Also

SoundLoadWave, **SoundSaveWave**

pnt2x

pnt2x(*waveName*, *pointNum*)

The pnt2x function returns the X value of the named wave at the point *pointNum*. The point number is truncated to an integer before use.

For higher dimensions, use **IndexToScale**.

Details

The result is derived from the wave's X scaling, not any X axis of a graph it may be displayed in.

If you would like to convert a fractional point number to an X value you can use:
`leftx(`*waveName*`)+deltax(`*waveName*`)*`*pointNum*.

See Also

DimDelta, **DimOffset**, **x2pnt**, **IndexToScale**

Waveform Model of Data on page II-62 and **Changing Dimension and Data Scaling** on page II-68 for an explanation of waves and dimension scaling.