

Sound

COLLABORATORS

	<i>TITLE :</i> Sound		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 23, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Sound	1
1.1	Sound V1.03	1
1.2	allocatesoundchannels	1
1.3	changesoundperiod	2
1.4	changesoundvolume	2
1.5	copysound	3
1.6	createsound	3
1.7	decodesound	4
1.8	freesound	4
1.9	freesoundchannels	5
1.10	getsoundlength	5
1.11	initsound	6
1.12	loadsound	6
1.13	peeksounddata	6
1.14	playsound	7
1.15	pokesounddata	7
1.16	savesound	7
1.17	setsoundchannels	8
1.18	setsoundperiod	8
1.19	setsoundvolume	9
1.20	soundfilter	9
1.21	stopsound	9

Chapter 1

Sound

1.1 Sound V1.03

PureBasic - Sound V1.03

Managing sounds will never be so easy and so fast than with this library. Believe us, we push the Amiga hardware to the max, to allow system compliant but instant sound replay. It simply use the 4 stereo audio channels for efficient output. The sound format is the IFF/8SVX, standard of the Amiga.

Commands summary:

- AllocateSoundChannels
- ChangeSoundPeriod
- ChangeSoundVolume
- CopySound
- CreateSound
- DecodeSound
- FreeSound
- FreeSoundChannels
- GetSoundLength
- InitSound
- LoadSound
- PeekSoundData
- PlaySound
- PokeSoundData
- SaveSound
- SetSoundChannels
- SetSoundPeriod
- SetSoundVolume
- SoundFilter
- StopSound

Sound Demo

1.2 allocatesoundchannels

SYNTAX

```
Result.w = AllocateSoundChannels(Channels.w)
```

FUNCTION

Use this function to allocate one or more channels.

Channels

This mask specify which channels that should be allocated.

1 = try to allocate channel 0

2 = try to allocate channel 1

4 = try to allocate channel 2

8 = try to allocate channel 3

When added together:

9 = try to allocate channel 0 and 3

15 = try to allocate all channels

Result

If this mask is TRUE it show which the channels are that is allocated else it is FALSE and none of the wanted channels are allocated.

1.3 changesoundperiod

SYNTAX

```
Result.w = ChangeSoundPeriod(#Sound.w,Period.w)
```

FUNCTION

Use this function to change the period when the sound is actually played.

The Period set in the #Sound object, that is used when the sound is started, is not affected with this call.

#Sound

The sound to change period for.

Period

The new period value.

Result

This mask show which are the affected channels that the period is changed for.

1.4 changesoundvolume

SYNTAX

```
Result.w = ChangeSoundVolume(#Sound.w,Volume.w)
```

FUNCTION

Use this function to change the volume when the sound is actually played.

The Volume set in the #Sound object, that is used when the sound is started, is not affected with this call.

#Sound

The sound to change volume for.

Volume

The new volume value.

Result

This mask show which are the affected channels that the volume is changed for.

1.5 copysound

SYNTAX

```
Result.l = CopySound(#Sound.w, #Sound.w)
```

FUNCTION

This function make a complete copy out of another #Sound object.

#Sound

The sound to copy.

#Sound

The new sound.

Result

If this is TRUE the copy of the #Sound have been done else it is FALSE.

1.6 createsound

SYNTAX

```
Result.l = CreateSound(#Sound.w, Length.l)
```

FUNCTION

This function create a new #Sound object and all of it's sound data is set to zero. Period, Volume and Channels are also set to zero.

To read and write the sound data use PeekSoundData() and PokeSoundData(). Period, Volume and Channels have to be set with the appropriate statement.

#Sound

The sound to create.

Length

The length of the sound data.

It's useless to specify any value higher than 128K here as the Amiga hardware doesn't support it.

Result

If this is TRUE the #Sound object are created else it is FALSE.

1.7 decodesound

SYNTAX

```
Result.w = DecodeSound(#Sound.w,Pointer.l)
```

FUNCTION

This function initialize a #Sound object from the IFF sound file included into the program.

- * Period are set to value taken from IFF file.
- * Volume is set to 64.
- * Channels is set to 15.

If the #Sound object is already initialized it must first be freed with a call to FreeSound() else it stay in memory and there will be no possibility to play it.

#Sound

The sound to use.

Pointer

A pointer to the included IFF sound file.

Result

This is TRUE if the #Sound object could be initialized from the included IFF sound file else it is FALSE.

1.8 freesound

SYNTAX

```
FreeSound(#Sound.w)
```

STATEMENT

This statement free a #Sound object that is initialized with LoadSound() or DecodeSound().

If no fast memory is available then the program end up in chip memory and by that a #Sound object that is initialized with DecodeSound() is valid until the object is initialized

again with `LoadSound()` or `DecodeSound()`. Which mean, such object could never be freed.

`#Sound`
The sound to free.

1.9 freesoundchannels

SYNTAX

`Result.w = FreeSoundChannels(Channels.w)`

FUNCTION

This function frees one or more channels.

`Channels`

This mask specify which channels that should be freed.

1 = free channel 0
2 = free channel 1
4 = free channel 2
8 = free channel 3

When added together:

9 = free both channel 0 and 3
15 = free all channels

`Result`

If this mask is TRUE it show which channels that are freed else it is FALSE and which mean; they are not freed because they have never been allocated.

1.10 getsoundlength

SYNTAX

`Result.l = GetSoundLength(#Sound.w)`

FUNCTION

This function gets the length of a `#Sound` object.

Use it for calculations so `PeekSoundData()` and `PokeSoundData()` not read and write outside of the actual sound data.

`#Sound`
The sound to use.

`Result`

The length of the sound data.

1.11 initsound

SYNTAX

```
Result.w = InitSound(Sounds.l)
```

FUNCTION

This function is the initroutine, it set up all needed stuff, and it must be called before all other functions. It could only be called once.

Sounds

This is how many #Sound objects that is wanted.

Result

If this is TRUE the call was successful else it is FALSE and then no other functions could be called.

1.12 loadsound

SYNTAX

```
Result.b = LoadSound(#Sound.w,FileName$)
```

FUNCTION

Use this function to initialize a #Sound object from an IFF sound file stored on disk.

- * Period are set to value taken from IFF file.
- * Volume is set to 64.
- * Channels is set to 15.

If the #Sound object is already initialized it must first be freed with a call to FreeSound() else it stay in memory and there will be no possibility to play it.

#Sound

The sound to use

FileName

This is the full path to the IFF sound file.

Result

It will be TRUE if the #Sound object could be initialized from the specified IFF sound file else it is FALSE.

1.13 peeksounddata

SYNTAX

```
Result.b = PeekSoundData(#Sound.w,Position.l)
```

FUNCTION

This function read some sound data from a #Sound object.

#Sound
The sound to use.

Position
This is where in the sound data the read should be done.

Result
The data read, it could range between -128 and 127.

1.14 playsound

SYNTAX
`Result.w = PlaySound(#Sound.w, Repeat.w)`

FUNCTION
This function play the specified #Sound object and it use the values; Period, Volume and Channels set in the object.

#Sound
The sound to play.

Repeat
To have the sound played one or more times and then stoped set this to a positive none zero value, if the sound should be repeated forever set it to minus.

Result
This mask show the actual channels that is used for this sound, zero indicate that the sound is not played.

1.15 pokesounddata

SYNTAX
`PokeSoundData(#Sound.w, Position.l, Data.b)`

STATEMENT
This statement write some data to a #Sound object.

#Sound
The sound to use.

Position
This specify where in the sound data the write should be done.

Data
The data to write, should range from -128 to 127.

1.16 savesound

SYNTAX

```
Result.b = SaveSound(#Sound.w,FileName$)
```

FUNCTION

This function save a #Sound object to disk as a IFF sound file.

#Sound

The sound to save.

FileName

This is the full path to where the IFF sound file should be saved.

Result

It will be TRUE if the #Sound object could be saved to disk as the specified IFF sound file else it is FALSE.

1.17 setsoundchannels

SYNTAX

```
SetSoundChannels(#Sound.w,Channels.w)
```

STATEMENT

This statement set channel mask for the #Sound that is used when the sound is played.

Channel mask are set to 15 when the #Sound is initialized with LoadSound() or DecodeSound().

#Sound

The sound to set channels for.

Channels

This is a mask that specify which channels that should be used for this object.

1 = use only channel 0

2 = use only channel 1

4 = use only channel 2

8 = use only channel 3

When added together:

5 = use channel 0 and 2

15 = use all channels

1.18 setsoundperiod

SYNTAX

```
SetSoundPeriod(#Sound.w,Period.w)
```

STATEMENT

This statement set Period in the #Sound object that is used when the sound is played.

The Period will be correctly set to the right value, taken from IFF file, when #Sound object is initialized with LoadSound() or DecodeSound().

#Sound

The sound to set period for.

Period

The period, it should be the final value as no calculations at all is done on this.

1.19 setsoundvolume

SYNTAX

```
SetSoundVolume(#Sound.w,Volume.w)
```

STATEMENT

This statement set Volume in the #Sound object that is used when the sound is played.

The Volume are set to 64 when the #Sound object is initialized with LoadSound() or DecodeSound().

#Sound

The sound to set volume for.

Volume

The volume, should range between 0 and 64.

1.20 soundfilter

SYNTAX

```
SoundFilter(ON/OFF)
```

STATEMENT

This statement turn the audiofiler on or off.

ON/OFF

Set this to TRUE to turn the audiofilter ON or set it to FALSE to turn audiofiler OFF.

1.21 stopsound

SYNTAX

```
StopSound(#Sound.w)
```

STATEMENT

Use this statement to stop a sound.

#Sound

The sound to stop.
