Sound

Paul Manias

COLLABORATORS			
	TITLE : Sound		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	July 26, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sound

1.1 Sound.GPI

```
Name:          SOUND.GPI AUTODOC
Version:       0.6 Beta.
Date:          2 May 1997
Author:        Paul Manias
Copyright:     DreamWorld Productions, 1996-1997.  All rights reserved.
Notes:         This  document is still being written and will contain errors
               in  a  number  of  places.  The information within cannot be
        treated as official until this autodoc reaches version 1.0.
```

1.2 Sound.GPI Functions

```
 SOUND.GPI
 AllocAudio()
 AllocSoundMem()
 FreeAudio()
 InitSound()
 FreeSound()
 FreeSoundMem()
 GetSound()
 CheckChannel()
 PlaySound()
 PlaySoundDAC1()
 PlaySoundDAC2()
 PlaySoundDAC3()
 PlaySoundDAC4()
 PlaySoundPri()
 PlaySoundPriDAC1()
 PlaySoundPriDAC2()
 PlaySoundPriDAC3()
 PlaySoundPriDAC4()
 SetVolume()
 FadeVolume()
 InitPlayer()
 PlayMOD()
 StopPlayer()
```

1.3 Sound.GPI/AllocAudio

```
NAME  AllocAudio -- Attempt to allocate the audio channels.

SYNOPSIS
  ErrorCode = AllocAudio()
     d0

  ULONG AllocAudio(void)

FUNCTION
  Attempts  to  allocate all the audio channels for your own use.  If
  the  function  cannot  get  the  channels,  it  will return with an
  errorcode  of ERR_INUSE.  If the call is successful (NULL) then you
  can safely use all the sound functions within GMS.

  This function should be called at the start of your program, and if
  successful you must call FreeAudio() before your program exits.

RESULT  ErrorCode - NULL if successful or ERR_INUSE if unsuccessful.

SEE ALSO
  FreeAudio
```

1.4 Sound.GPI/FreeAudio

```
NAME  FreeAudio -- Free the audio channels for system use.

SYNOPSIS
  FreeAudio()

  void FreeAudio(void)

FUNCTION
  Frees  the  audio  channels  so that the system can use them again.
  You cannot make use of any of the audio channels after calling this
  function.

SEE ALSO
  AllocAudio
```

1.5 Sound.GPI/InitSound

```
NAME  InitSound -- Initialise a sound structure for the play routines.

SYNOPSIS
  Sound = InitSound(Sound/TagList/List/ObjectList)
    d0                          a0

  struct Sound * InitSound(struct Sound *)

  struct Sound * InitSoundTags(ULONG tag, ...)
```

FUNCTION
This function will initialise a sound for use in the play routines. Its main job is to load and assess the sound header, and fill in any missing fields. It can also unpack sounds in some cases.

If the following fields in the Sound structure are detected as being NULL, InitSound() will fill them in for you:

Data
Length
Period
Volume

If you don't want some or all of these fields written too, simply fill them in before-hand. This is imperative if the sound is in RAW format, for obvious reasons.

Lists and object lists are fully supported by this function, just pass a pointer to one of these instead of a Sound. (See lists).

NOTE If the sound is in RAW format, then this function will have little effect, so you should set most of the fields yourself.

INPUTS Sound - Pointer to a single sound structure, or for multiple initialisations, a list of Sound's.

SAM_Channel
The channel that you want to play through. Acceptable channel numbers are 0, 1, 2 and 3 (a total of 4 available channels).

SAM_Priority
The priority of your sound goes here. This field is used by the PlaySoundPri() function to determine if your sound should be played when the channel is busy. Naturally, higher values are played over samples with lower values.

SAM_Header
Points to the very start of the sample, which in most cases will be the something like an IFF 8SVX header. If the sample data is RAW then simply point to the start of the data here.

SAM_Data
This field points to the actual data that is going to be played. InitSound() will fill this field in for you if you initialise it to 0.

SAM_Length
The length of the sample data in words. This field will be filled in for you if the sound has a recognised header (eg IFF).

SAM_Octave
The octave at which to play this sample. The highest pitched value is OCT_G0S, the lowest is OCT_A7S. There are about 94 available settings, see games/sound.i to look at the complete list.

SAM_Volume

```
  The volume of the sound, which lies in the range 0 - 100.  A volume
  of zero will not be heard, a volume of 100 is the loudest.

  SAM_Attrib
  Specifies the attributes for the sound.

    SBIT8   - Sound data is 8 bit (only set this if raw).

    SBIT16  - Sound data is 16 bit (only set this if raw).

    SMODVOL - Modulates the volume with the next channel.

    SMODPER - Modulate the sound's period with the next channel.

    SREPEAT - Repeats the sample forever.

  SAM_File
  If your sound is contained on disk, place a pointer to the filename
  here.   This  will  cause InitSound() to load the sound data in for
  you  (via  a  call  to SmartLoad()) and fill in the Header and Data
  fields.   The  rest  of  the  initialisation procedure will then be
  carried out.

SEE ALSO
  FreeSound
```

1.6 Sound.GPI/FreeSound

```
NAME  FreeSound -- Free any allocations made in an initialised sound.

SYNOPSIS
  FreeSound(Sound)
            a0

  void FreeSound(struct Sound *)

FUNCTION
  Frees  any  allocations  made  in  the  initialisation  of  a Sound
  structure.   You have to call this function at some point for every
  initialised  Sound, otherwise resources may be withheld on the exit
  of your program.

  If  the  structure was allocated via GetSound() or a tag list, then
  the  structure  itself  will  be freed from memory (you will not be
  allowed to use it any longer).

  This function is fully supportive of lists and object lists.

INPUTS  Sound - Pointer to an intialised sound structure.

SEE ALSO
  InitSound
```

1.7 Sound.GPI/CheckChannel

```
NAME  CheckChannel -- Checks the current activity of a sound channel.

SYNOPSIS
  Status = CheckChannel(Channel)
    d0                  d0.w

  UWORD CheckChannel(UWORD Channel)

FUNCTION
  Checks  the  specified  channel  to  see if it has any data playing
  through  it.

INPUTS  Channel - Either 1, 2, 3 or 4.

RESULT  Status - The  current  status  of  the  channel,  a  result of NULL
    indicates that the channel is free.  A result of 1 indicates that
    the channel is busy.
```

1.8 Sound.GPI/PlaySound

```
NAME  PlaySound -- Play a sound through an audio channel.

SYNOPSIS
  PlaySound(Sound)
            a0

  void PlaySound(struct Sound *)

FUNCTION
  Plays  a  sound  according  to the settings in the sound structure.
  This  function  executes immediately, and ignores all channel/sound
  priorities.

  You  must  have initialised the sound structure before calling this
  function.

INPUTS  Sound - Pointer to a valid sound structure.

SEE ALSO
  PlaySoundDACx, PlaySoundPri, PlaySoundPriDACx
```

1.9 Sound.GPI/PlaySoundDACx

```
NAME  PlaySoundDACx -- Play a sound with ignorance to channel priorities.

SYNOPSIS
  PlaySoundDACx(Sound)
                a0

  void PlaySoundDACx(struct Sound *)
```

```
  Where 'x' is either 1, 2, 3 or 4, which is a direct reference to
  the channel number.

FUNCTION
  DAC stands for Direct Access to Channel. This is the fastest way
  to play a sound as it goes directly to that channel's sound
  routine, but it is not very easy to work with. This function
  exists for intelligently changing from full channel access for
  sound effects, to one channel access while music is playing.

  When setting up your sounds you should make sure that you use all
  four channels in your structures. If the music is off, use the
  normal PlaySoundPri() function. If the music is on, and if it uses
  all but one of the channels, use this function to re-route all the
  sound effects through the spare channel.

NOTE This function ignores sound priorities, and will play the sound
  regardless of wether the channel is busy or not.

INPUTS Sound - Pointer to a valid sound structure.

SEE ALSO
  PlaySound, PlaySoundPri, PlaySoundPriDACx, games/games.i
```

1.10 Sound.GPI/PlaySoundPriDACx

```
NAME PlaySoundPriDACx -- Play a sound ignoring the setting in SAM_Channel.

SYNOPSIS
  PlaySoundPriDACx(Sound)
                   a0

  void PlaySoundPriDACx(struct Sound *)

  Where 'x' is either 1, 2, 3 or 4, which is a direct reference to
  the channel number.

FUNCTION
  DAC stands for Direct Access to Channel. This is the fastest way
  to play a prioritised sound as it goes directly to that channel's
  sound routine, but it is not very easy to work with. This function
  exists for intelligently changing from full channel access for
  sound effects, to one channel access while music is playing.

  When setting up your sounds you should make sure that you use all
  four channels in your structures. If the music is off, use the
  normal PlaySoundPri() function. If the music is on, and if it uses
  all but one of the channels, use this function to re-route all the
  sound effects through the spare channel.

  This function supports prioritisation of sound effects.

INPUTS Sound - Pointer to a valid sound structure.
```

```
SEE ALSO
  PlaySoundDACx, PlaySound, PlaySoundPri, games/games.i
```

1.11 Sound.GPI/PlaySoundPri

```
NAME  PlaySoundPri -- Play  a sound if it can equal or better a channel's
      priority.

SYNOPSIS
  PlaySoundPri(Sound)
               a0

  void PlaySoundPri(struct Sound *)

FUNCTION
  Plays  a sound according to the settings in the sound structure, IF
  it equals or betters the channel's current priority setting.

  Prioritisation of sounds allows you to play sound effects according
  to their importance.  Make sure that you take care in ordering your
  sounds so that they play effectively!

  It is recommended that you use CHANNEL_ALL in the SAM_Channel field
  so  that  your  game  makes  maximum use of all the available sound
  channels.

INPUTS  Sound - Pointer to a valid sound structure.

SEE ALSO
  PlaySound, PlaySoundPriDACx, PlaySoundDACx, games/games.i
```

1.12 Sound.GPI/AllocSoundMem

```
NAME  AllocSoundMem -- Allocate memory for sample playback.

SYNOPSIS
  Memory = AllocSoundMem(Size)
    d0           d0

  APTR AllocSoundMem(ULONG Size)

FUNCTION
  Allocates a block of memory suitable for playing sound samples.

INPUTS  Size - The Size of the memory to allocate.

RESULT  Memory - Pointer  to  the  allocated  memory.  All  audio memory is
     formatted with 0's when allocated.  Returns NULL if error.

SEE ALSO
  FreeMemBlock
```

1.13 Sound.GPI/GetSound

```
NAME  GetSound -- Gets the latest version of the sound structure.

SYNOPSIS
  Sound = GetSound()
    d0

  struct Sound * GetSound(void);

FUNCTION
  Allocates the latest version of the GMS sound structure and returns
  it back to you.  The structure fields will be empty so that you can
  fill them out to suit your requirements.  Before your program exits
  you  will need to free the structure, this is automatically done in
  the FreeSound() function.

  You  have  to use this function if you do not want to use tag lists
  to  initialise  your  sound (remember that it is illegal to compile
  and use pre-initialised structures in GMS programs).

RESULT  Sound - Pointer to a new sound structure.

SEE ALSO
  InitSound
```

1.14 Sound.GPI/FreeSoundMem

```
NAME  FreeSoundMem -- Frees a block of sound memory.

SYNOPSIS
  FreeSoundMem(MemBlock)
      d0

  void FreeSoundMem(APTR MemBlock);

FUNCTION
  Frees a block of sound memory.

INPUT MemBlock - Pointer   to   a   Sound  memory  block  allocated  from
       AllocSoundMem() or AllocMemBlock()

SEE ALSO
```

1.15 Sound.GPI/

```
NAME

SYNOPSIS

FUNCTION
```

INPUT

RESULT

SEE ALSO