DLL Description

for

pei tel USB Audio Devices

Application Note 1202

Revision: 1.1

April 2016

pei tel Communications GmbH
A member of the peiker group
www.peitel.de

TABLE OF CONTENTS

0. HISTORY

Date	Revision	Author	Comments
Oct 2012	1.0	CM	First Release
April 2016	1.1	MP	Update (new VID)

Table 1: History

1. GENERAL

1.1 Disclaimer

All information and data contained in this document are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability.

Any new issue of this document invalidates previous issues. Further, pei tel Communications GmbH reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

1.2 Related Documents

No	Name	Comments
1	AN1002 Interfacing pei tel USB Audio Devices	Application Note 1002

Table 2: Related Documents

1.3 References

[1] **USB specification, USB HID specification, USB HID usages tables, etc.**
www.usb.org

[2] **MSDN Developer Center/Library**
http://msdn.microsoft.com/en-us/library

2. GLOBAL DEFINITIONS

All required drivers will be installed automatically when plugged into a free USB port the first time. You will be notified when installation has been completed.

In case that the driver is not found automatically, the driver can be downloaded from our website: www.peitel.de.

pei tel USB Audio devices will be recognized by Windows® Operating Systems as a **Composite Device** which consists of an **USB Sound Device** and a **USB HID Device** (Human Interface Device).

pei tel USB Audio devices use the **Vendor ID 0x2B2E.** The device name (product name) of all pei tel USB Audio devices will always be read as "**PTC USB**".

The Product ID is related to the type of product that is used. Refer to AN1002 for pre-defined Type of Products.

Since pei tel USB Audio devices are equipped differently, parts of this document may not be relevant. For instance, a PS12 USB does not have a speaker; hence adjusting speaker volume does not have an effect.

2.2 Files

HID and Sound/Mixer functionalities have been separated in two DLLs for reasons of clarity and logical comprehensibility. Depending on the environment used, the following files will be needed when using the DLLs:

ptcHid.dll This is the actual DLL file for HID communication and must be located in the directory your application is running from or the Windows® default DLL directory. The Windows® default DLL directory is usually C:\WINDOWS\SYSTEM32.

ptcHid.lib This is the library file for the ptcHid.dll, which your C linker will use to link your application to the DLL. This file must be located in the directory of your source code. This file is only required for C applications.

ptcSnd.dll This is the actual DLL file for the Sound/Mixer functions and must be located in the directory your application is running from or the Windows® default DLL directory. The Windows® default DLL directory is usually C:\WINDOWS\SYSTEM32.

ptcSnd.lib This is the library file for the ptcSnd.dll, which your C linker will use to link your application to the DLL. This file must be located in the directory of your source code. This file is only required for C applications.

ptcUSB.h This is the C header file which has the function prototyping for both DLLs to be used when communicating with the DLLs. This file must be located in the directory of your C source code. This file is only required for C applications.

modPtcUSB.bas This is the Visual Basic 6 (VB6) header file, which has the function prototyping in and useful constants that will be used when communicating with the DLL. This file should be located in the directory of your VB6 project and added to the project as a module. This file is only required for VB6 applications.

modPtcUSB.vb This is the Visual Basic 2008 (VB2008) header file, which has the function prototyping in and useful constants that will be used when communicating with the DLL. This file should be located in the directory of your VB2008 project and added to the project as a module. This file is only required for VB2008 applications.

3. USING DLLS IN YOUR APPLICATION

There are two ways to use a DLL in an application, they are **link implicitly** and **link explicitly**. Implicitly link is the easiest to use.

When **implicitly linking** (static loading) is used the DLL is loaded when your application loads. The disadvantage to this approach is that if a DLL to which the program references to is missing, the program will refuse to load.

Explicitly linking allows your application to load the DLL at a defer time with ***LoadLibrary()*** function.

The procedure for each linking type depends on the programming language and programming environment. Please refer to the documentation or help of your programming language and programming environment for more details if not given enough information in the following sections.

3.1 C Applications

When using **implicitly linking** in C applications the DLL is loaded when your application loads. To implicitly link to the DLL add the DLL library file (ptcHid.lib and ptcSnd.lib) to your project and include the DLL header file (ptcUSB.h) in your source code. Then simply call the DLL functions.

Explicitly linking allows your application to load the DLL at a defer time. For your application to Explicitly link to the DLL you will have to load the DLL with ***LoadLibrary()*** function. *LoadLibrary()* loads the DLL, if not already loaded, and returns a handle to the DLL. Then using the DLL handle and the DLL function name of interest, call ***GetProcAddress()*** to obtain the address of the DLL function. *GetProcAddress()* returns the function address, which you can use to call the DLL functions. When ending the application, the DLL should be unmapped from the address space by a call to ***FreeLibrary()***.

3.2 Delphi Applications

Implicitly linking means that your DLL is automatically loaded when the application that calls the DLL is executed. To use implicitly linking with Delphi, you declare a function or procedure that resides in the DLL with the **external** keyword.

After you have properly imported the function or procedure, you call it like any other function or procedure. This step presumes, of course, that the procedure or function has been exported from the DLL.

Explicitly linking can be done in the same way as it is done with C applications: Load the DLL with *LoadLibrary()* function. *LoadLibrary()* loads the DLL, if not already loaded, and returns a handle to the DLL. Then using the DLL handle and the DLL function name of interest, call ***GetProcAddress()*** to obtain the address of the DLL function. *GetProcAddress()* returns the function address, which you can use to call the DLL functions. When ending the application, the DLL should be unmapped from the address space by a call to ***FreeLibrary()***.

3.3 Visual Basic 6/Visual Basic 2008 Applications

When using **implicitly linking** in VB6 or VB2008 applications the DLL is loaded when your application's executable loads. For this method, the DLL file must be located in the application's directory or Window's® system directory. A call to a DLL function may fail when the application is started from VB6 IDE and the IDE may crash as well.

This can be avoided by using **explicitly linking**. Explicitly linking allows your application to load the DLL at a defer time, i.e. during form load event. For your application to explicitly link to the DLL you will have to load the DLL with ***LoadLibrary()*** function. *LoadLibrary()* loads the DLL, if not already loaded, and returns a handle to the DLL. When ending the application, the DLL should be unmapped from the address space by a call to ***FreeLibrary()*** i.e. in form unload event.

For both methods you must declare the DLL procedures (no return value) or functions (with return value) when using the DLL in a VB6 or VB2008 application. To declare DLL procedures or functions use the **Declare** statement to prototype each function.

All the DLL functions have been declared in the modPtcUSB.bas file. Simply add this file to your VB6 project and then call the DLL function from your code as you would any other function by keeping implicitly and explicitly linking methods in mind. Respectively, add the file modPtcUSB.vb to your project for VB2008 applications.

4. DLL FUNCTIONS

The prefix **"ptcHid"** shows that this is a function or procedure to a HID functionality. The prefix **"ptcSnd"** shows that this is a function or procedure to a Sound/Mixer functionality.

4.1 ptcHidGetDllVersion

This function retrieves the version of the HID DLL. May be important for service purposes.

- **Prototype C**
  ```
  char* ptcHidGetDllVersion(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetDllVersion(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetDllVersion Lib "ptcHid.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetDllVersion Lib "ptcHid.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  Version of DLL

4.2 ptcHidGetDllDate

This function retrieves the release date of the HID DLL in the format YYYY/MM/DD. May be important for service purposes.

- **Prototype C**
  ```
  char* ptcHidGetDllDate(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetDllDate(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetDllDate Lib "ptcHid.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetDllDate Lib "ptcHid.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  Date of DLL in format YYYY/MM/DD

4.3 ptcHidOpenDev

This function tries to find the specified device and if found, opens it. An opened device must be closed with a call to the DLL function *ptcHidCloseDev()* for a clean-up. We recommend to do this latest when exiting the main application.

- **Prototype C**

```
int ptcHidOpenDev(int vid, int pid);
```

- **Prototype Delphi**

```
function ptcHidOpenDev(vid: Integer;pid: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcHidOpenDev Lib "ptcHid.dll" _
(ByVal VID As Long, ByVal PID As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcHidOpenDev Lib "ptcHid.dll" _
(ByVal VID As Integer, ByVal PID As Integer) As Integer
```

- **Parameter(s)**

vid/VID	Vendor ID (for pei tel USB Audio devices this equals 0x2B2E)
pid/PID	Product ID (Refer to AN1002)

- **Returned value(s)**

-1	An open device has been unplugged
0	Device could not be opened. Possible reasons: - The device does not exist - The device is already open
1	Device could successfully be opened

4.4 ptcHidCloseDev

This function closes a previously opened device. This function must be called for a clean-up when a device has been opened with *ptcHidOpenDev()*. Since the *ptcHidCloseDev()* function does not harm if no device has been opened previously, we recommend to call this function when exiting the application.

- **Prototype C**
  ```
  int ptcHidCloseDev(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidCloseDev ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidCloseDev Lib "ptcHid.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidCloseDev Lib "ptcHid.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

0	Closing was not successful. Possible reasons: - The device is already closed - There is no connection to a device
1	Connection to device could be successfully closed

4.5 ptcHidGetVendorId

This function reads the Vendor ID from the device.

- **Prototype C**
  ```
  int ptcHidGetVendorId(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetVendorId ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetVendorId Lib "ptcHid.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetVendorId Lib "ptcHid.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

0	Invalid Vendor ID. Possible reasons: - The device is already closed - There is no connection to a device
Vendor ID	Vendor ID as read from the device. For pei tel USB Audio devices this should be read as 0x2B2E.

4.6 ptcHidGetProductId

This function reads the Product ID from the device.

- **Prototype C**
  ```
  int ptcHidGetProductId(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetProductId ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetProductId Lib "ptcHid.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetProductId Lib "ptcHid.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

0	Invalid Product ID. Possible reasons: - The device is already closed - There is no connection to a device
Product ID	Product ID as read from the device.

4.7 ptcHidGetVersionNumber

This function reads the Software Revision from the device.

- **Prototype C**
  ```
  int ptcHidGetVersionNumber(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetVersionNumber ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetVersionNumber Lib "ptcHid.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetVersionNumber Lib "ptcHid.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

0	Invalid Version Number. Possible reasons: - The device is already closed - There is no connection to a device
Number	Software Revision as read from the device

4.8 ptcHidGetSerialNumber

This function retrieves the Serial Number from the device. See section "2.1 Type Of Product" for coding of the serial number.

- **Prototype C**
  ```
  char* ptcHidGetSerialNumber(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetSerialNumber (): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetSerialNumber Lib "ptcHid.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetSerialNumber Lib "ptcHid.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  "" Empty string, if device not opened
  SN Serial Number of the device, coded as described in AN1002.

4.9 ptcHidGetDeviceName

This function retrieves the Device Name from the device. For pei tel USB Audio devices this will always be read as "PTC USB".

- **Prototype C**
```
char* ptcHidGetDeviceName(void);
```
- **Prototype Delphi**
```
function ptcHidGetDeviceName (): pchar;
```
- **Prototype VB6**
```
Public Declare Function ptcHidGetDeviceName Lib "ptcHid.dll" () As String
```
- **Prototype VB2008**
```
Public Declare Function ptcHidGetDeviceName Lib "ptcHid.dll" () As String
```
- **Parameter(s)**
  None
- **Returned value(s)**

""	Empty string, if device not opened
Name	Device Name. Should be "PTC USB".

4.10 ptcHidGetVendorName

This function retrieves the Vendor Name from the device.

- **Prototype C**
  ```
  char* ptcHidGetVendorName(void);
  ```
- **Prototype Delphi**
  ```
  function ptcHidGetVendorName (): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcHidGetVendorName Lib "ptcHid.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcHidGetVendorName Lib "ptcHid.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

""	Empty string, if device not opened
Name	Vendor Name

4.11 ptcHidGetInputReport

This function retrieves an Input Report from the device. For further information and coding of the Input Report refer to related document #1 "AN1002 Interfacing pei tel USB Audio Devices".

- **Prototype C**
```
int ptcHidGetInputReport(void);
```
- **Prototype Delphi**
```
function ptcHidGetInputReport ():Integer;
```
- **Prototype VB6**
```
Public Declare Function ptcHidGetInputReport Lib "ptcHid.dll" () As Long
```
- **Prototype VB2008**
```
Public Declare Function ptcHidGetInputReport Lib "ptcHid.dll" () As Integer
```
- **Parameter(s)**
  None
- **Returned value(s)**

-1	Invalid Input Report. Possible reasons: - The device is already closed - There is no connection to a device
Data	Input Report according current states of inputs. Refer to related document #1 "AN1002 Interfacing pei tel USB Audio Devices" for more information and coding of the Input Report.

4.12 ptcHidSetOutputReport

This function sends an Output Report to the device. For further information and coding of the Output Report refer to related document #1 "AN1002 Interfacing pei tel USB Audio Devices".

- **Prototype C**

```
int ptcHidSetOutputReport(int data);
```

- **Prototype Delphi**

```
function ptcHidSetOutputReport (data: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcHidSetOutputReport Lib "ptcHid.dll" _
(ByVal data As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcHidSetOutputReport Lib "ptcHid.dll" _
(ByVal data As Integer) As Integer
```

- **Parameter(s)**

  data Data for outputs to be set according to coding as described in related document #1 "AN1002 Interfacing pei tel USB Audio Devices".

- **Returned value(s)**

  0 Command not successful. Possible reasons:

  - The device is already closed
  - There is no connection to a device

  1 Command successful

4.13 ptcSndGetDllVersion

This function retrieves the version of the Sound DLL. May be important for service purposes.

- **Prototype C**
  ```
  char* ptcSndGetDllVersion(void);
  ```
- **Prototype Delphi**
  ```
  function ptcSndGetDllVersion(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcSndGetDllVersion Lib "ptcSnd.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcSndGetDllVersion Lib "ptcSnd.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  Version of DLL

4.14 ptcSndGetDllDate

This function retrieves the release date of the Sound DLL in the format YYYY/MM/DD. May be important for service purposes.

- **Prototype C**
  ```
  char* ptcSndGetDllDate(void);
  ```
- **Prototype Delphi**
  ```
  function ptcSndGetDllDate(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcSndGetDllDate Lib "ptcSnd.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcSndGetDllDate Lib "ptcSnd.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  Date of DLL in format YYYY/MM/DD

4.15 ptcSndInitMixerEnum

This procedure determines the Mixer ID for Playback and Recording line. With Windows XP both have identical ID, but with Windows Vista/7 Playback and Recording has been separated. The IDs are stored in the global variables MixerIDDstSpk (Speaker) MixerIDSrcMic (Microphone/WaveIn). We are just looking for pei tel USB Audio devices (PTC USB).

ATTENTION: This function needs to be called once prior any other call to a ptcSnd... function!!

- **Prototype C**
  ```
  void ptcSndGetDllDate(void);
  ```
- **Prototype Delphi**
  ```
  procedure ptcSndGetDllDate(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Sub ptcSndGetDllDate Lib "ptcSnd.dll" ()
  ```
- **Prototype VB2008**
  ```
  Public Declare Sub ptcSndGetDllDate Lib "ptcSnd.dll" ()
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  None

4.16 ptcSndGetWinVersion

This function retrieves the Windows® version and converts it to a string. This function is required to differentiate between the Windows® operating systems and adapt the behavior of the application, for example disable controls, if not supported by the operating system.

- **Prototype C**
  ```
  char* ptcSndGetWinVersion(void);
  ```
- **Prototype Delphi**
  ```
  function ptcSndGetWinVersion(): pchar;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcSndGetWinVersion Lib "ptcSnd.dll" () As String
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcSndGetWinVersion Lib "ptcSnd.dll" () As String
  ```
- **Parameter(s)**
  None
- **Returned value(s)**
  Windows Version. The following strings are possible:
  - *Windows 95*
  - *Windows 98*
  - *Windows ME*
  - *Windows NT 3.51*
  - *Windows NT 4*
  - *Windows 2000*
  - *Windows XP*
  - *Windows .NET Server*
  - *Windows Vista*
  - *Windows 7*
  - *Unknown Version*

4.17 ptcSndGetMasterVolume

This function reads current setting of speaker volume.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**
```
int ptcSndGetMasterVolume(void);
```
- **Prototype Delphi**
```
function ptcSndGetMasterVolume ():Integer;
```
- **Prototype VB6**
```
Public Declare Function ptcSndGetMasterVolume Lib "ptcSnd.dll" () As Long
```
- **Prototype VB2008**
```
Public Declare Function ptcSndGetMasterVolume Lib "ptcSnd.dll" () As Integer
```
- **Parameter(s)**
  None
- **Returned value(s)**

<0	Command not successful
Value	Current setting (usually in range from 0 … 65535)

4.18 ptcSndSetMasterVolume

This function sets speaker volume to specified value.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**

```
int ptcSndSetMasterVolume(int data);
```

- **Prototype Delphi**

```
function ptcSndSetMasterVolume (Value: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcSndSetMasterVolume Lib "ptcSnd.dll" _
(ByVal data As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcSndSetMasterVolume Lib "ptcSnd.dll" _
(ByVal data As Integer) As Integer
```

- **Parameter(s)**

  data/Value Speaker volume level in range from 0 … 65535

- **Returned value(s)**

  <=0 Command not successful

  1 Command successful

4.19 ptcSndGetMicVolume

This function reads current setting of microphone volume.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**
```
int ptcSndGetMicVolume(void);
```
- **Prototype Delphi**
```
function ptcSndGetMicVolume ():Integer;
```
- **Prototype VB6**
```
Public Declare Function ptcSndGetMicVolume Lib "ptcSnd.dll" () As Long
```
- **Prototype VB2008**
```
Public Declare Function ptcSndGetMicVolume Lib "ptcSnd.dll" () As Integer
```
- **Parameter(s)**
  None
- **Returned value(s)**

<0	Command not successful
Value	Current setting (usually in range from 0 … 65535)

4.20 ptcSndSetMicVolume

This function sets microphone volume to specified value.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**

```
int ptcSndSetMicVolume(int data);
```

- **Prototype Delphi**

```
function ptcSndSetMicVolume (Value: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcSndSetMicVolume Lib "ptcSnd.dll" _
(ByVal data As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcSndSetMicVolume Lib "ptcSnd.dll" _
(ByVal data As Integer) As Integer
```

- **Parameter(s)**

  data/Value Microphone volume level in range from 0 … 65535.

- **Returned value(s)**

  <=0 Command not successful

  1 Command successful

4.21 ptcSndGetVolumeMute

This function reads current mute state of speaker.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**
  ```
  int ptcSndGetVolumeMute(void);
  ```
- **Prototype Delphi**
  ```
  function ptcSndGetVolumeMute ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcSndGetVolumeMute Lib "ptcSnd.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcSndGetVolumeMute Lib "ptcSnd.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

-1	Command not successful
0	Mute state of speaker is "UNMUTED"
1	Mute state of speaker is "MUTED"

4.22 ptcSndSetVolumeMute

This function sets mute state of speaker to specified state.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**

```
int ptcSndSetVolumeMute(int mute);
```

- **Prototype Delphi**

```
function ptcSndSetVolumeMute (Mute: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcSndSetVolumeMute Lib "ptcSnd.dll" _
(ByVal mute As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcSndSetVolumeMute Lib "ptcSnd.dll" _
(ByVal mute As Integer) As Integer
```

- **Parameter(s)**

mute	0	Set speaker mute state to "UNMUTED"
	1	Set speaker mute state to "MUTED"

- **Returned value(s)**

0	Command not successful
1	Command successful

4.23 ptcSndGetMicVolumeMute

This function reads current mute state of microphone.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**
  ```
  int ptcSndGetMicVolumeMute(void);
  ```
- **Prototype Delphi**
  ```
  function ptcSndGetMicVolumeMute ():Integer;
  ```
- **Prototype VB6**
  ```
  Public Declare Function ptcSndGetMicVolumeMute Lib "ptcSnd.dll" () As Long
  ```
- **Prototype VB2008**
  ```
  Public Declare Function ptcSndGetMicVolumeMute Lib "ptcSnd.dll" () As Integer
  ```
- **Parameter(s)**
  None
- **Returned value(s)**

-1	Command not successful
0	Mute state of microphone is "UNMUTED"
1	Mute state of microphone is "MUTED"

4.24 ptcSndSetMicVolumeMute

This function sets mute state of microphone to specified state.

ATTENTION: Call at least once ptcSndInitMixerEnum() prior to this function.

- **Prototype C**

```
int ptcSndSetMicVolumeMute(int mute);
```

- **Prototype Delphi**

```
function ptcSndSetMicVolumeMute (Mute: Integer):Integer;
```

- **Prototype VB6**

```
Public Declare Function ptcSndSetMicVolumeMute Lib "ptcSnd.dll" _
(ByVal mute As Long) As Long
```

- **Prototype VB2008**

```
Public Declare Function ptcSndSetMicVolumeMute Lib "ptcSnd.dll" _
(ByVal mute As Integer) As Integer
```

- **Parameter(s)**

mute	0	Set microphone mute state to "UNMUTED"
	1	Set microphone mute state to "MUTED"

- **Returned value(s)**

0	Command not successful
1	Command successful

END OF DOCUMENT