Proposed Specification for MIXER API
copyright Media Vision (c) 1991,1992
version 1.0 -- Ken Nicholson

MCIMIXER.DRV	MCI interface to the mixer driver
MMMIXER.DLL	Multimedia Mixer DLL
MVMIXER.DRV	Media Vision Mixer driver

THE MULTIMEDIA MIXER DLL

The Multimedia Windows Mixer DLL, as proposed by Media Vision, is a library of generalized routines that provides a common, device-independent interface to software controllable mixers. Its design is based on the uses and functionality of real world audio mixers.

This DLL is responsible for loading the device-dependent mixer driver. In the past, the only device-dependent drivers available were for Media Vision hardware. Recently, a company has developed a mixer driver for the Sound Blaster Pro. The company is called Animotion and can be reached by calling (205) 591-5715. Their product, **MCS stereo,** provides a common interface for mixing on any device.

MMMIXER.DLL gives the MultiMedia application programmer control over a complete range of audio capabilites. Every conceivable audio mixing, patching, equalization and amplification need can be handled by this DLL and its API. It is capable of supporting mixing features far beyond those available with today's PC hardware.

WHY A MULTIMEDIA MIXER ?

The Multimedia PC Specification version 1 (May 13,1991) calls for "On-board analog audio mixing capabilities," requiring "... input from three (recommended four) sources and [must] present the sources as a stereo, line-level audio signal at the back panel. ... Individual audio source and master digital volume control registers and extra line-level audio sources are highly recommended."

Such hardware requires a set of standard functions calls (API's) that will handle volume changes in a device-independent and extensible way. Furthermore, there are a number of issues, apart from the setting of an input's or an output's volume levels, that should be handled in a standard way. Functions such as equalization, special effects, patch changes, device association, connection mapping, smooth timed transitions, power-up settings and device sharing are not addressed by Windows 3.1.

What is provided?

The WAVEOUT and MIDIOUT drivers have Get- and SetVolume entry points for control of a device's output volume. In addition, the Multimedia Extensions define an AUX device type that allows applications to get and set the volumes of additional devices. The

only two types of AUX devices defined are CD audio and auxillary input.

In the current design of the Multimedia Extensions, there is no link between an AUX device and the audio device associated with it. It may be assumed, when there is only one AUX device, that it controls the audio output of the CD-ROM drive. But in the presence of multiple AUX devices there is no mechanism for an application to determine which AUX device to control to change the CD volume. Varying AUX-to-Device association will cause chaos for application writers.

This raises the question of volume control for devices in general. Most consumer audio devices (cassette decks, turntables, video disk players, televisions) don't have a variable line output. These devices rely on a mixer or integrated amplifier to control the volume level. The term "attenuator" is applied to controls that vary the line output of a device.

Attenuation of PCM and MIDI audio output is not something users need concern themselves with and therefore volume control functions do not belong in waveout and midiout drivers; volume control is properly a mixing function. Users and multimedia authors will want to individually adjust the relative volumes of a number of device outputs and this is the primary need for a mixer. The Mixer API solves this problem by supporting volume controls on each of any number of mixer inputs as well supporting volume controls for each mixer output.

Some professional audio mixers feature a myriad of knobs to enhance or alter input audio. Among these are controls for bass, midrange and treble, controls to add reverb, effects and stereo-mono crossover. In fact, Media Vision's Pro-Audio Spectrum supports many of these features. The existing Multimedia Windows API's don't address these features.

Another feature of the MMMIXER.DLL API set is its ability to maintain and coordinate mixer patch information. A mixer device that supports software selectable input patching would allow any one of several audio devices to be "patched" to a mixer input. Similarly an application can interrogate the mixer API to find out which devices are connected to which mixer inputs.

Fading the volume up of one sound source while fading out the volume of another seems such an intrinsically necessary function for multimedia that we have incorporated it into the DLL. Without this feature, the application programmer would be required to send auxSetVolume messages in "ping-pong" fashion to the devices to be cross-faded.

Some users will have powered speakers with no volume control. MMWindows start-up sounds are capable of producing DEAFENING DECIBELS OF DIN. Mixer drivers conforming to this spec read WIN.INI to determine the desired start-up, or mixer reset, volume settings.

Frequently an application will fill the entire screen and on occasion a Multimedia application will suprise the user with unexpectedly loud audio. Such moments find the user frantically trying to bring up the volume control application in order to lower the volume. An equally dismaying situation is starting a multimedia application only to

discover that the volumes are too low for the sound to be audible.
The MMMIXER.DLL intercepts the CTRL-ALT-U and CTRL-ALT-D sequences and translates them into driver volume up and down calls. This allows the user to quickly change volumes up and down.

Media Vision has developed the Multimedia Windows DLL to address these and other issues in audio control on the MPC. We gratefully acknowledge Microsoft and especially the members of the Multimedia Windows Development Team for their advice and assistance in developing the Multimedia Mixer DLL and the Media Vision Mixer driver.

MEDIA VISION IMPLEMENTATION

The Multimedia Mixer DLL is capable of supporting virtually any type of mixing hardware. The limitations of the Pro AudioSpectrum mixing hardware should not be confused with the capabilities of the Multimedia Mixer DLL.

MVMIXER.DRV is implemented as a single Mixer device. This mixer has seven input lines and two output lines. The Pro AudioSpectrum 16 has new mixer circuitry that allows 8 input lines.

The standard input lines are by default patched to:

1) FM SYNTHESIZER	5) MICROPHONE
2) RECORD MONITOR	6) WAVE (PCM)
3) AUXILLARY INPUT	7) PC speaker
4) INTERNAL CD	8) Snd Blaster WAVE (mono PCM)

On the Pro AudioSpectrum and Pro AudioSpectrum Plus, Input #2 is the mix of all devices being routed to Output #2. For the Pro AudioSpectrum 16, Input #2 becomes a second internal CD connector.

Each input can be connected to one (but not both) of the two outputs.

1) PLAY	(Line Out)
2) RECORD	(Wave In)

Output #1, apart from supporting volume control, supports treble and bass, loudness, stereo enhance and mute.
Output #2 volume control of the second output is possible on the Pro-16 only.

Input #2 should not be patched to Output #2.
Input #6 should be patched to Output #2 during playback for proper waveform filtering. This is only necessary on the 8-bit Pro AudioSpectrum.

mixGetDevCaps returns the following structure:

```
MIXERCAPS MVMixer= {
  MM_MEDIAVISION,            // manufacturer id
  MM_PROAUD_MIXER,           // product id
  (VERSION) 0x100,           // version of the driver
  "Pro AudioSpectrum", // product name (NULL terminated string)
  7,                         // number of inputs  supported.
  2,                         // number of outputs supported.
  9,                         // number of input patches supported.
```

```
  3,                                    // number of output patches supported.
  MIXERCAP_MANUALPATCHSWITCH,           // supports some manual   patching
  NULL                        // reserved
  };
```

A Line Capabaility structure for the Pro Audio Spectrum Mixer
would look like this:

```
{
wNumber             =0;               // This is the current caps for input # 0
dwDeviceType=MIX_MICROPHONE      // Microphone currently patched;
              +MIX_USER_CONNECTED;// Requires user to plug it in (for prompting)
wNumSoftPatches=1;                // This input always patched to the Microphone
wPatchNumber=0;                             // the only mic
wNumChannels=2;                             // The mic is actually mono with a splitter
dwSupport   =MIX_SUPPORT_LRVOLUME       // supported functionality
szIOname    ="Line 1: Microphone Jack" // Input Name
szPname     ="MIC";                         // patch name
}
```

Each Mixer has a built in number of INPUT and OUTPUT patches. They
are referenced by an ordinal number starting from zero. Patch number
zero is always defined as NO CONNECT.

Here is the capability structure for Output #1:

```
{
wPatchNumber =1;                           // This patch's ordinal number
dwDeviceType =MIX_AMPLIFIER+          // device connected is amplifier
                MIX_LISTENER +             // all output patches are listeners
              MIX_USER_CONNECTED;  //  requires a cable connection
dwLineNumbers=1<<0;                        //  bit-field, lines connectable to
szPname[]    ="MASTER";                    // patch name (NULL terminated string)
dwAssociation=NULL;                        // HIWORD=type, LOWORD=device
dwReserved1  =NULL;                        // reserved for future use
}
```

//
//

MIXER API OVERVIEW

MIXER GENERAL FUNCTIONS:

mixGetNumDevs	- returns the number of mixer devices
mixGetDevCaps	- provides information on mixer device capability
mixOpen	- opens mixer device
mixClose	- closes mixer device
mixGetErrorText	- gets a text string corresponding to error number

MIXER PRESET FUNCTIONS:

mixMute	- global mixer mute
mixReset	- resets mixer inputs, volumes, patches etc.
mixGetState	- gets current state of the mixer device
mixSetState	- sets mixer state with optional timed fade
mixGetFadeStatus	- returns time remaining of current fadeprocess

// INPUT TO OUTPUT CONNECTION

mixGetConnections	- gets the connection map of an input or output
mixSetConnections	- sets input to output connection map

// LINE CONTROLS

mixGetControl	- gets the setting of a line's control
mixSetControl	- sets the level of a line's control
mixGetLineInfo	- gets functionality and current status of a line

// PATCH CALLS

mixGetPatch	- gets a line's patch number
mixSetPatch	- sets a line's patch information
mixGetPatchInfo	- returns information on a standard mixerpatch

// DEVICE ORIENTED CALLS

mixGetDeviceName	- converts device type to device name
mixGetDeviceLines	- finds which lines have a specified device
mixSetDeviceConnections	- performs abstract device connection
mixGetDeviceConnections	- determines device types connected

MIXER DEVICE GENERAL FUNCTIONS

mixGetNumDevs	- returns the number of mixer devices
mixGetDevCaps	- provides information on mixer device capability
mixOpen	- opens mixer device
mixClose	- closes mixer device
mixGetErrorText	- gets a text string corresponding to error number

---

Syntax — WORD mixGetNumDevs(void)
This function retrieves the number of mixer devices present in the system.

Parameters — None

Return Value — Returns the number of mixer devices present in the system

---

Syntax — mixGetDevCaps(wDeviceID, lpCaps, wSize)

Parameters

WORD wDeviceID
Identifies the mixer device to be queried.

LPMIXERCAPS lpCaps
Specifies a far pointer to a MIXERCAPS structure. This structure is filled with information about the capabilities of the device.

WORD wSize
Specifies the size of the MIXERCAPS structure.

Return Value — When the wDeviceID is non-zero, the return value will be MIXERR_NOERR (zero) if the function was successful. If wDeviceID is zero, the return value is the size of the drivers MIXERCAPS structure.

Comments — Use mixGetDevCaps to determine the number of mixer devices present in the system. The Device ID specified by wDeviceID varies from zero to one less than the number of devices present. Only wSize bytes(or less of information will be copied to the location pointed to by lpCaps. If wSize is 0, nothing will be copied and MMSYSERR_NOERROR is returned.

See Also — mixGetNumDevs

Syntax — mixOpen(lphMixer, wDeviceID, dwFlags)

This function opens a specified mixer device.

Parameters LPHMIXER lphMixer
Specifies a far pointer to an HMIXER handle. This location is filled with a handle identifying the opened mixer device.

WORD wDeviceID

DWORD dwFlags
Specifies flags for opening the device

Return Value Returns zero if the function was successful. Otherwise, it returns an error.

Comments Use mixOpen before making any control/enquiry calls to the mixer driver.

See Also mixClose

Syntax mixClose(hMixer)

Parameters HMIXER hMixer

Return Value Returns zero if the function was successful. Otherwise, it returns an error. Possible errors are:

MMSYSERR_INVALHANDLE
Specified device handle is invalid.

See Also mixOpen

Syntax mixGetErrorText(wError, lpText, wSize)

This function retrieves a textual description of the error identified by the specified error number.

Parameters WORD wError
Specifies the error number.

LPSTR lpText
Specifies a far pointer to a buffer which is filled with the textual error description.

WORD wSize
Specifies the length of the buffer pointerd to by lpText.

Return Value Returns the length of the string copied to zero if the function was successful.

MIXER PRESET FUNCTIONS

mixMute	- Global Mute. Causes all mixer devices to mute/unmute
mixReset	- resets mixer inputs, volumes, patches etc.
mixGetState	- gets current state of the mixer device
mixSetState	- sets current state of the mixer device
mixGetFadeStatus	- returns time remaining of current fadeprocess

---

Syntax mixMute(wFlag)

Toggles the state of all mixer devices to mute
or back to non-mute.

Parameters WORD wFlag

if wFlag is MIXMUTE_TOGGLE, mixMute will toggle
the mixer state
if wFlag == MIXMUTE_STATUS, mixMute will return
the current state of the global mute flag
Identifies the mixer device to be reset.

Return Value Returns:
MIXMUTESTATUS_MUTE when mixer is muted
MIXMUTESTATUS_NOMUTE when mixer is not muted

Syntax mixReset(hMixer)

Resets the mixer to a default state. The default
state is read from the WIN.INI file.

Parameters HMIXER hMixer
Identifies the mixer device to be reset.

Return Value Returns zero if the function was successful.
Otherwise, it returns an error.

See Also mixOpen, mixGetSetup, mixSetSetup

Syntax mixGetState(hMixer, lphMixerState, lpwSize)

Returns a handle to a structure containing the
current state of the mixer. The structure is

defined by the device.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPHANDLE lphMixerState
Specifies a far pointer to a handle to where the mixer state information is saved.

LPWORD lpwSize
Specifies a far pointer to a word where the size of the mixer state information is stored.

Return Value Returns zero if the function was successful. Otherwise, it returns an error. Possible errors are:

MIXERR_BADMIXERPTR null pointer to mixer

Comments This function is used to save the current state of the specified mixer device.

See Also mixSetState

Syntax mixSetState(hMixer,lpMixerState,wSize,dwTime, dwFlags,dwCallback);

Restores the mixer to a saved state.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPMIXERSTATE lpMixerState
Handle to a mixer state structure as returned by the mixGetState function.

WORD wSize
Size of the mixer state structure.

DWORD dwTime
The high word is delay time in tenths of seconds. The low word is the duration of the fade in tenths of seconds. A dwTime value of zero results in an instant mixer setting.

DWORD dwFlags
MIX_FADE_OVERRIDE override a fade in progress

DWORD dwCallback
Address of a procedure to be called when fade is complete. Note: this has not yet been implemented

Return Value Returns zero if the function was successful. Otherwise, it returns an error. Possible errors are:

MIXERR_FADEINPROGRESS

Comments If wSize is not correct, this function will be rejected.

See Also mixGetState, mixGetFadeStatus

Syntax mixGetFadeStatus(hMixer,lpdwTime)

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPDWORD lpdwTime
The high word is the delay time remaining in tenths of seconds.
The low word is the fade time remaining in tenths of seconds.
An lpdwTime value of zero indicates no fade is in progress.

Return Value Returns zero if the function was successful.

Comments Status indicators displaying the time remaining of a fade may wish to call this function upon receipt of any of the following MIXER MESSAGES:

WM_MIX_CONTROLCHANGED
WM_MIX_CONNECTIONCHANGED
WM_MIX_PATCHCHANGED

See Also mixSetState

// INPUT TO OUTPUT CONNECTION

mixGetConnections - gets the connection map of an input or output
mixSetConnections - sets input to output connection map

Syntax mixGetConnections(hMixer,WORD wLine, LPDWORD lpdwConnections);

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLine
The low byte indicates the mixer line to get the connections information for.
The high byte indicates whether the line is an input line or an output line. The following macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT

LPDWORD lpdwConnections
A far pointer to a DWORD where the connection information is to be stored. Each bit of the double word represents a mixer line. If the bit is 1, wLine is connected to that line.
For example, assume wLine=0x0000, indicating Input Line #1. If that Input line is connected to Output Line #0, bit 0 of *lpdwConnections will be set.
Logically, connection information can only be maintained for a mixer with a maximum of 32 inputs and 32 outputs.

Return Value Returns zero if the function was successful.
Possible Errors:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line

Comments

See Also mixSetConnections

Syntax mixSetConnections(Mixer,WORD wLine, DWORD dwConnections);

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLine
The low byte indicates the mixer line to get the

connections information for.
The high byte indicates whether the line is an input line or an output line. The following macros (equates) are used for the high byte:

MIX_INPUT
MIX_OUTPUT

DWORD dwConnections
This parameter specifies the connection map for the given line of the mixer. If wLine refers to an input Line, the outputs specified by this parameter will be connected to that input. Bit 0 refers to line 0, bit 1 to line 1, etc.

Return Value

Returns zero if the function was successful.
Possible Errors:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line

Comments

This function allows inputs to be selectively patched to one or more outputs. It also allows outputs to be connected to one or more inputs. This function allows the caller to say "connect input lines 1, 3 and 5 to output #1" or "connect input line 4 to outputs 1 and 2" If the hardware cannot support all connections requested, the mixer driver will connect the lower numbered lines first. Calling mixGetConnections after mixSetConnections is recommended for verification of requested connections.

See Also

mixGetConnections

// LINE CONTROLS
mixGetControl - gets the setting of a line's control
mixSetControl - sets the level of a line's control
mixGetLineInfo - gets support functionality of a line

Syntax mixGetControl(hMixer, wLineNum, dwControl, lpdwSetting);

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLineNum
The low byte indicates the mixer line to get the connections information for.
The high byte indicates whether the line is an input line or an output line. The following macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT

DWORD dwControl
Specifies the control to get the setting of.
Here is the current list of possible controls:

MIX_SUPPORT_LRVOLUME	left-right volume control
MIX_SUPPORT_ALC	Auto Level Control
MIX_SUPPORT_BMT	B-M-T equalization
MIX_SUPPORT_CROSSOVER	crossover change
MIX_SUPPORT_LOUDNESS	loudness equalization
MIX_SUPPORT_MUTE	channel mute
MIX_SUPPORT_REVERB	reverb
MIX_SUPPORT_STEREOENHANCE	stereo enhance
MIX_SUPPORT_CUSTOM1	custom effect #1
MIX_SUPPORT_CUSTOM2	custom effect #2
MIX_SUPPORT_CUSTOM3	custom effect #3

LPDWORD lpdwSetting
Specifies a far pointer to a location that will be filled with the current Control setting. For stereo controls, the high-order word of this location contains the left channel setting and the low-order word contains the right channel setting. A value of 0xFFFF represents full intensity and a value of 0x0000 is full cutout.

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line
MIXERR_NOTSUPPORTED	control not supported

See Also mixSetControl

Syntax mixSetControl(hMixer, wLineNum, dwControl, dwSetting)

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLineNum
The low byte indicates the mixer line to get the connections information for.
The high byte indicates whether the line is an input line or an output line. The following macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT
Specifies the input to set Control for

DWORD dwControl
Specifies the control to set
Here is the current list of possible controls:

MIX_SUPPORT_LRVOLUME	left-right volume control
MIX_SUPPORT_ALC	Auto Level Control
MIX_SUPPORT_BMT	B-M-T equalization
MIX_SUPPORT_CROSSOVER	crossover change
MIX_SUPPORT_LOUDNESS	loudness equalization
MIX_SUPPORT_MUTE	channel mute
MIX_SUPPORT_REVERB	reverb
MIX_SUPPORT_STEREOENHANCE	stereo enhance
MIX_SUPPORT_CUSTOM1	custom effect #1
MIX_SUPPORT_CUSTOM2	custom effect #2
MIX_SUPPORT_CUSTOM3	custom effect #3

DWORD dwSetting
Specifies a far pointer to a location that will be filled with the current Control setting. For stereo controls, the high-order word of this location contains the left channel setting and the low-order word contains the right channel setting. A value of 0xFFFF represents full intensity and a value of 0x0000 is full cutout.

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line
MIXERR_NOTSUPPORTED	control not supported

See Also mixGetControl, CONTROL SETTING NOTES

Syntax mixGetLineInfo(hMixer, wLineNum, lpInfo, wSize);

Retrieves information about the specified input's capabilities.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLineNum
The low byte indicates the mixer line to get the connections information for.
The high byte indicates whether the line is an input line or an output line. The following macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT

LPMIXERLINEINFO lpInfo
Specifiies a far pointer to be filled with the capability information for the specified input.

WORD wSize
Specifies the size of the LPMIXERLINEINFO structure.

Return Value Returns zero if the function was successful.

Comments

See Also

// PATCH CALLS
mixGetPatch - gets a line's patch number
mixSetPatch - sets a line's patch information
mixGetPatchInfo - returns information on a standard mixerpatch

---

Syntax mixGetPatch(hMixer,wLineNum, lpwPatchNum);

Returns information regarding a specific input patch
that can be selected into an input

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLineNum
The low byte indicates the mixer line to get the
connections information for.
The high byte indicates whether the line is an
input line or an output line. The following
macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT

LPWORD lpwPatchNum
Destination for the patch number. A value of -1
signifies a user-defined patch.

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line

Comments The range of wPatchNum must be from 0 to 1 less than
the number of software patches returned in
mixGetDevCaps.

See Also mixSetPatch, mixGetPatchInfo

Syntax mixSetPatch(hMixer,wLine, wPatchNum,lpPatch,wSize);

Allows the user to set the patch of an input to
another device.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wLine
The low byte indicates the mixer line to get the
connections information for.
The high byte indicates whether the line is an

input line or an output line. The following
macros (equates) are used for the high byte:
MIX_INPUT
MIX_OUTPUT

WORD wPatchNum
Specifies the patch number to set. Each mixer driver has a number of internal patches that are selected by this parameter.

LPPATCHINFO lpInfo
If this parameter is not NULL, the wPatchNum parameter is ignored and the PATCHINFO structure pointed to is used to set the patch information. The patch number assigned will be -1;

WORD wSize
Specifies size of PATCHINFO structure

Return Value Returns zero if the function was successful. Otherwise, it returns an error. Possible errors are:

MIXERR_INVALINPUT	illegal input line
MIXERR_INVALOUTPUT	illegal output line
MIXERR_PATCHMISMATCH	patch-to-line mismatch

Comments The MIX_USER_CONNECTED bit may be OR'd with the patch type to indicate a patch that is to be connected by the user rather than one that is selected via software control. Applications should check this bit at initialization time to advise users to make the external connection. If the patch number of a user-connected patch is illegal, the driver's default patch for that line will be used.

See Also mixGetPatchInfo, mixGetPatch

Syntax mixGetPatchInfo(hMixer,wPatchNum,lpInfo,wSize);

Returns information about a pre-defined patch.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

WORD wPatchNum
Specifies the patch number to set. Each mixer driver has a number of internal patches that are selected by this parameter.

LPPATCHINFO lpInfo
If this parameter is not NULL, the wPatchNum parameter is ignored and the PATCHINFO structure pointed to is used to set the patch information.

WORD wSize
Specifies size of PATCHINFO structure

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALPATCH patch number out of range

Comments The current patch information for a line is available by calling mixGetLineInfo. Now that patch information is stored in WIN.INI, the default patch information is automatically overridden. In most cases the wPatchNum will be -1. User defined patch information is not returned by this call.

See Also mixGetPatch, mixSetPatch, mixGetLineInfo

// DEVICE ORIENTED CALLS
mixGetDeviceName - converts device type to device name
mixGetDeviceLines - finds which lines have a specified device
mixSetDeviceConnections - performs abstract device connection
mixGetDeviceConnections - determines device types connected

Syntax mixGetDeviceName(dwDeviceType, lpDeviceName, wSize);

Returns information regarding a specific input patch that can be selected into an input

Parameters DWORD dwDeviceType
A 32-bit value indication the device type

LPSTR lpDeviceName
points to the destination for the device name

WORD wSize
buffers size pointed to by lpDeviceName
If wSize <= MIX_DEVICESHORTNAME, the three letter standard device mnemonic string will be copied to the buffer. In all cases wSize will be the limit of characters copied.

Return Value Returns zero if the function was successful.

Comments The device short name is intended for display in dialog boxes and in win.ini's mixer configuration settings.

See Also mixGetDeviceLines, mixGetDeviceConnections

Syntax mixGetDeviceLines(hMixer,lpDeviceLines);

/// Given a device type, this function will report the lines that the given
/// device is connected to. If the association is not NULL, only
/// lines with the same association will be reported. Otherwise, all
/// devices of the given type are reported on.
///

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPDEVICELINES lpDeviceLines
far pointer to DEVICELINES data structure

```
struct {
DWORD       dwDeviceType;  // aka technology
WORD  wNumDevices;  // return value: # lines with device found
DWORD       dwLines;       // return value: line map
DWORD       dwAssociation;  // for exclusive search
} DEVICELINES;
```

typedef DEVICELINES FAR *LPDEVICELINES;

Return Value Returns zero if the function was successful.

Comments Be sure that the dwAssociation element of the DEVICELINES structure is NULL unless you intend to find a specific device line of which the association has been established.

See Also mixGetDeviceConnections, mixGetPatchInfo

Syntax mixGetDeviceConnections(hMixer,lpDeviceConnect);

Given a device type, this function will return device types that the given device type is connected to. Input device types will yield reporting of output device types connected and vice versa. If the associationType is given, only devices with the same association will be reported. Otherwise, all devices of the given type are reported on.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPDEVICECONNECT lpDeviceConnect
long pointer to DEVICECONNECT structure

typedef struct{
DWORD dwInputDeviceType;
DWORD dwOutputDeviceType;
DWORD dwInputAssociation;
DWORD dwOutputAssociation;
} DEVICECONNECT;

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALSTRUCTPTR null lpDeviceConnect

Comments An output device type is one that can be connected to a mixer output and is distingquished by having the MIX_LISTENER bit set in its dwDeviceType field.

See Also mixSetDeviceConnections

Syntax mixSetDeviceConnections(hMixer,lpDeviceConnect)

Given two device types, input and output, this function will attempt to connect the two types. If the associationType and associationValue fields are not NULL, only devices with the same association will be connected.

Parameters HMIXER hMixer
Identifies the mixer device to be used.

LPDEVICECONNECT lpDeviceConnect
long pointer to DEVICECONNECT structure

typedef struct{
DWORD dwInputDeviceType;
DWORD dwOutputDeviceType;
DWORD dwInputAssociation;
DWORD dwOutputAssociation;
} DEVICECONNECT;

Return Value Returns zero if the function was successful.
Otherwise, it returns an error. Possible errors are:

MIXERR_INVALSTRUCTPTR null lpDeviceConnect

Comments

See Also mixGetDeviceConnections

CONTROL SETTING NOTES

The functions mixGetControl and mixSetInputControl
have a dwControl parameter and a dwSetting parameter.

Following the example set by Microsoft's volume settings, each
control setting will have associated with it a double word value
for specifying its setting.

dwControl	description	format
MIX_SUPPORT_VOLUME	volume control	LLLL:RRRR scalar
MIX_SUPPORT_LRVOLUME	left-right volume control	LLLL:RRRR scalar
MIX_SUPPORT_ALC	Auto Level Control	LLLL:RRRR on/off
MIX_SUPPORT_BMT	B-M-T equalization	--BB:MMTT scalar
MIX_SUPPORT_CROSSOVER	crossover change	MIXCROSSCAPS
MIX_SUPPORT_LOUDNESS	loudness equalization	LLLL:RRRR on/off
MIX_SUPPORT_MUTE	mute - don't change volume	LLLL:RRRR on/off
MIX_SUPPORT_REVERB	reverb	LLLL:RRRR scalar
MIX_SUPPORT_STEREOENHANCE	stereo enhance	LLLL:RRRR on/off
MIX_SUPPORT_CUSTOM1	custom effect #1	LLLL:RRRR on/off
MIX_SUPPORT_CUSTOM2	custom effect #2	LLLL:RRRR scalar

Scalars are unsigned values from 0-65535 except in the case of BMT
where they are unsigned values from 0-255. These values should be
interpolated to match the mixer hardware's scale.

On/Off values are ON for any non-zero value.

//

MCI INTERFACE

Media Vision is now providing an MCI driver for controlling the mixer hardware. For many of you, this will make the control of mixing functions much easier than it ever has been.

MCI, as you must know, stands for Media Control Interface. In the case of a mixer, there is no media (medium). When we speak of CDs, wave files, MIDI, we connote some sort of media transport, (ie. position within the data, support for PLAY, STOP, REWIND, etc)

A PLAY command sent to a mixer does not make sense. In fact most MCI commands have little meaning for a mixer.

Still, MCI is powerful. One of its outstanding features is its ability to convert strings into driver commands. We have, therefore, developed an MCI driver for Multimedia Windows.

The documentation for the MCI mixer is found in the file mcimixer.doc.

//

MIXER NOTES:

The mixSetState function allows timed fades. If it would be useful to developers we can add the capability for timed fades to the mixSetControl function. Let us know if you want this. Soon.

User-defined patches are not accessible through mixGetPatchInfo. The mechanism of user-defined patches may change in the future.

//
GLOSSARY:

line - a stereo input or output of a mixer

patch - the association of a line to a particular sound producing or recording device

connection - this term refers to a mixer's internal input-to-output routing of an audio signal

control - the capability to modify an audio signal (ie volume)

cross-fade - fading one or more controls up while fading other out

//