CL-PortAudio

January 16, 2012

Contents

1 The portaudio package **2**

1.1 Installation and Usage 2
1.2 Example 2
1.3 Introduction 6
1.4 Key abstractions: Host APIs, Devices and Streams 6
1.5 Initialisation, termination and utility functions 7
1.6 Host APIs 7
1.7 Devices 8
1.8 Streams 9
1.9 Opening and Closing Streams 9
1.10 Starting and Stopping Streams 9
1.11 The Read/Write I/O Method 10
1.12 Retreiving Stream Information 10
1.13 Error Handling 10
1.14 Conditions 10
1.15 Bitfields 12
1.16 Enums 12
1.17 Other functions 13
1.18 Other macros 31
1.19 Other classes 31

2 The portaudio-tests package **36**

2.1 Other functions 36

Chapter 1

The portaudio package

This package contains bindings to PortAudio. PortAudio is a free, cross-platform, open-source, audio I/O library.

1.1 Installation and Usage

```
git clone --depth 1
emacs
M+x slime
  (pushnew "path/to/cl-portaudio" asdf:*central-registry*)
  (ql:quickload :cl-portaudio)
  (ql:quickload :cl-portaudio-tests)
  (portaudio-tests:test-read-write-echo)
```

1.2 Example

In this section:

- with-audio
- with-default-audio-stream
- read-stream
- separate-array-to-channels
- merge-channels-into-array
- write-stream

```
(use-package :portaudio)
(defconstant +frames-per-buffer+ 1024)
(defconstant +sample-rate+ 44100d0)
```

```
(defconstant +seconds+ 15)
(defconstant +sample-format+ :float)
(defconstant +num-channels+ 2)
(defun test-read-write-converted-echo ()
 "Record input into an array; Separate array to channels; Merge channels into array; Play
 (with-audio
   (format t "~%=== Wire on. Will run ~D seconds . ===~%" +seconds+)
   (with-default-audio-stream (astream +num-channels+ +num-channels+ :sample-format +samp
     (dotimes (i (round (/ (* +seconds+ +sample-rate+) +frames-per-buffer+)))
       (ignore-errors (write-stream astream
                                    (merge-channels-into-array astream
                                                               (separate-array-to-channels
```

`with-audio` *&body body* [*Macro*]

Details

Execute body in PortAudio initialize/terminate environment.

`with-default-audio-stream` [*Macro*]
`(`*var* `num-input num-output &key (sample-format float) (`*sample-rate* `44100.0d0) (`*frames-per-buffer* `1024)) &`*body body*

Details

Execute body with opened and started stream VAR and shut down the stream after it is done. It is required use these macro in with-audio or initialize/terminate environment.

`read-stream` *pa-stream* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

Return Values

On success array of data will be returned, or raises input-overflowed if input data was discarded by PortAudio after the previous call and before this call.

Details

Read samples from an input stream. The function doesn't return until the entire buffer has been filled - this may involve waiting for the operating system to supply the data. Size of returned array equal to `(* frames-per-buffer channel-count)`.

See also

- open-stream

`separate-array-to-channels` *pa-stream array* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

array — Flat array, that is received from `read-stream` .

Return Values

(channelcount)-dimensional array of single-floats

Details

Separate flat array

See also

- open-stream
- read-stream

`merge-channels-into-array` *pa-stream channels* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

channels — Vector of vectors of floats, that contains data for all sound channels.

Return Values

Vector of data, that can be used with `write-stream` .

Details

Merge subarrays of (channelcount)-dimensional array to flat array.

See also

- open-stream
- write-stream

`write-stream` *pa-stream buffer* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

buffer — A array of sample frames. The buffer contains samples in the format specified by the `(stream-parameters-sample-format output-parameters)` field used to open the stream, and the number of channels specified by `(stream-parameters-num-channels output-parameters)`.

Return Values

On success NIL will be returned, or raises an error output-uderflow if additional output data was inserted after the previous call and before this call.

Details

Write samples to an output stream. This function doesn't return until the entire buffer has been consumed - this may involve waiting for the operating system to consume the data. Size of buffer should be equal to `(* frames-per-buffer channel-count)`.

See also

- open-stream

Note ignore-errors is used for ignoring output-underflowed error.

1.3 Introduction

PortAudio provides a uniform application programming interface (API) across all supported platforms. You can think of the PortAudio library as a wrapper that converts calls to the PortAudio API into calls to platform-specific native audio APIs. Operating systems often offer more than one native audio API and some APIs (such as JACK) may be available on multiple target operating systems. PortAudio supports all the major native audio APIs on each supported platform. The diagram below illustrates the relationship between your application, PortAudio, and the supported native audio APIs:

Image

PortAudio provides a uniform interface to native audio APIs. However, it doesn't always provide totally uniform functionality. There are cases where PortAudio is limited by the capabilities of the underlying native audio API. For example, PortAudio doesn't provide sample rate conversion if you request a sample rate that is not supported by the native audio API. Another example is that the ASIO SDK only allows one device to be open at a time, so PortAudio/ASIO doesn't currently support opening multiple ASIO devices simultaneously.

1.4 Key abstractions: Host APIs, Devices and Streams

The PortAudio processing model includes three main abstractions: Host APIs, audio Devices and audio Streams.

Host APIs represent platform-specific native audio APIs. Some examples of Host APIs are Core Audio on Mac OS, WMME and DirectSound on Windows and OSS and ALSA on Linux. The diagram in the previous section shows many of the supported native APIs. Sometimes it's useful to know which Host APIs you're dealing with, but it is easy to use PortAudio without ever interacting directly with the Host API abstraction.

Devices represent individual hardware audio interfaces or audio ports on the host platform. Devices have names and certain capabilities such as supported sample rates and the number of supported input and output channels. PortAudio provides functions to enumerate available Devices and to query for Device capabilities.

Streams manage active audio input and output from and to Devices. Streams may be half duplex (input or output) or full duplex (simultaneous input and output). Streams operate at a specific sample rate with particular sample formats, buffer sizes and internal buffering latencies. You specify these parameters when you open the Stream. Audio data is communicated between a Stream and your application via a user provided asynchronous callback function or by invoking synchronous read and write functions.

PortAudio supports audio input and output in a variety of sample formats: 8, 16, 24 and 32 bit integer formats and 32 bit floating point, irrespective of the formats supported by the native audio API. PortAudio also supports multichan-

nel buffers in both interleaved and non-interleaved (separate buffer per channel) formats and automatically performs conversion when necessary. If requested, PortAudio can clamp out-of range samples and/or dither to a native format.

The PortAudio API offers the following functionality:

- Initialize and terminate the library
- Enumerate available Host APIs
- Enumerate available Devices either globally, or within each Host API
- Discover default or recommended Devices and Device settings
- Discover Device capabilities such as supported audio data formats and sample rates
- Create and control audio Streams to acquire audio from and output audio to Devices
- Provide Stream timing information to support synchronising audio with other parts of your application
- Retrieve version and error information.

These functions are described in more detail below.

1.5 Initialisation, termination and utility functions

The PortAudio library must be initialized before it can be used and terminated to clean up afterwards. You initialize PortAudio by calling `initialize` and clean up by calling `terminate` . There is `with-audio` macro that does environment.

You can query PortAudio for version information using `get-version` to get a numeric version number and `get-version-text` to get a string.

The size in bytes of the various sample formats represented by the sample-format enumeration can be obtained using `get-sample-size` .

`pa-sleep` sleeps for a specified number of milliseconds. This isn't intended for use in production systems; it's provided only as a simple portable way to implement tests and examples where the main thread sleeps while audio is acquired or played by an asynchronous callback function.

1.6 Host APIs

A Host API acts as a top-level grouping for all of the Devices offered by a single native platform audio API. Each Host API has a unique type identifier, a name, zero or more Devices, and nominated default input and output Devices.

Host APIs are usually referenced by index: an integer of type host-api-index that ranges between zero and `(- (get-host-api-count ) 1)`. You can enumerate all available Host APIs by counting across this range.

You can retrieve the index of the default Host API by calling `get-default-host-api` .

Information about a Host API, such as it's name and default devices, is stored in a `host-api-info` structure. You can retrieve a pointer to a particular Host API's `host-api-info` structure by calling `get-host-api-info` with the Host API's index as a parameter.

Most PortAudio functions reference Host APIs by `host-api-index` indices. Each Host API also has a unique type identifier defined in the host-api-type-id enumeration. You can call `host-api-type-id-to-host-api-index` to retrieve the current host-api-index for a particular host-api-type-id.

1.7 Devices

A Device represents an audio endpoint provided by a particular native audio API. This usually corresponds to a specific input or output port on a hardware audio interface, or to the interface as a whole. Each Host API operates independently, so a single physical audio port may be addressable via different Devices exposed by different Host APIs.

A Device has a name, is associated with a Host API, and has a maximum number of supported input and output channels. PortAudio provides recommended default latency values and a default sample rate for each Device. To obtain more detailed information about device capabilities you can call `is-format-supported` to query whether it is possible to open a Stream using particular Devices, parameters and sample rate.

Although each Device conceptually belongs to a specific Host API, most PortAudio functions and data structures refer to Devices using a global, Host API-independent index of type device-index an integer of that ranges between zero and `(- (get-device-count ) 1)`. The reasons for this are partly historical but it also makes it easy for applications to ignore the Host API abstraction and just work with Devices and Streams.

If you want to enumerate Devices belonging to a particular Host API you can count between 0 and `(- (host-api-info-device-count ) 1)`. You can convert this Host API-specific index value to a global device-index value by calling `host-api-device-index-to-device-index` .

Information about a Device is stored in a PaDeviceInfo structure. You can retrieve a pointer to a Devices's `device-info` structure by calling `get-device-info` with the Device's index as a parameter.

You can retrieve the indices of the global default input and output devices using `get-default-input-device` and `get-default-output-device` . Default Devices for each Host API are stored in the Host API's `host-api-info` structures.

For an example of enumerating devices and printing information about their

capabilities see the `print-devices` function.

1.8 Streams

A Stream represents an active flow of audio data between your application and one or more audio Devices. A Stream operates at a specific sample rate with specific sample formats and buffer sizes.

1.9 Opening and Closing Streams

You call `open-stream` to open a Stream, specifying the Device(s) to use, the number of input and output channels, sample formats, suggested latency values and flags that control dithering, clipping and overflow handling. You specify many of these parameters in two `stream-parameters` structures, one for input and one for output.

Devices may be full duplex (supporting simultaneous input and output) or half duplex (supporting input or output) usually this reflects the structure of the underlying native audio API. When opening a Stream you can specify one full duplex Device for both input and output, or two different Devices for input and output. Some Host APIs only support full-duplex operation with a full-duplex device (e.g. ASIO) but most are able to aggregate two half duplex devices into a full duplex Stream. PortAudio requires that all devices specified in a call to `open-stream` belong to the same Host API.

A successful call to `open-stream` creates a pointer to a `pa-stream` an opaque handle representing the open Stream. All PortAudio API functions that operate on open Streams take a pointer to a `pa-stream` as their first parameter.

PortAudio also provides `open-default-stream` a simpler alternative to `open-stream` which you can use when you want to open the default audio Device(s) with default latency parameters.

You call `close-stream` close a Stream when you've finished using it.

There are two macros to simplify work with stream: `with-audio-stream` and `with-default-audio-stream` . These macros open and start stream at the beginning and stop and close stream at the end. Body is protected by unwind-protect.

1.10 Starting and Stopping Streams

Newly opened Streams are initially stopped. You call `start-stream` to start a Stream. You can stop a running Stream using `stop-stream` or `abort-stream` (the Stop function plays out all internally queued audio data, while Abort tries to stop as quickly as possible). An open Stream can be started and stopped multiple times. You can call `is-stream-stopped` to query whether a Stream is running or stopped.

1.11 The Read/Write I/O Method

PortAudio provides a synchronous read/write interface for acquiring and playing audio.

To write audio data to a Stream call `write-stream` and to read data call `read-stream` . These functions will block if the internal buffers are full, making them safe to call in a tight loop. If you want to avoid blocking you can query the amount of available read or write space using `get-stream-read-available` or `get-stream-write-available` .

For examples of the read/write I/O method see the following examples in the /t directory of the PortAudio distribution: tests.lisp (`portaudio-tests:test-read-write-converted-echo`).

1.12 Retreiving Stream Information

You can retrieve information about an open Stream by calling `get-stream-info` . This returns a `stream-info` structure containing the actual input and output latency and sample rate of the stream. It's possible for these values to be different from the suggested values passed to `open-stream` .

1.13 Error Handling

Most PortAudio functions invokes signal. Possible conditions are described in pa-error enum. Some functions return values greater than or equal to zero for normal results.

PortAudio usually tries to translate error conditions into portable pa-error error codes. However if an unexpected error is encountered the unanticipated-host-error code may be returned. In this case a further mechanism is provided to query for Host API-specific error information. If PortAudio throws unanticipated-host-error you can call `get-last-host-error-info` to retrieve a `host-error-info` structure that provides more information, including the Host API that encountered the error, a native API error code and error text.

1.14 Conditions

There are conditions, that are translated from PaError.

- not-anticipated
- unanticipated-host-error
- invalid-channel-count
- invalid-sample-rate

- invalid-device
- invalid-flag
- sample-format-not-supported
- bad-i-o-device-combination
- insufficient-memory
- buffer-too-big
- buffer-too-small
- null-callback
- bad-stream-ptr
- timed-out
- internal-error
- device-unavailable
- incompatible-host-api-specific-stream-info
- stream-is-stopped
- stream-is-not-stopped
- input-overflowed
- output-underflowed
- host-api-not-found
- invalid-host-api
- can-not-read-from-a-callback-stream
- can-not-write-to-a-callback-stream
- can-not-read-from-an-output-only-stream
- can-not-write-to-an-input-only-stream
- incompatible-stream-host-api
- bad-buffer-ptr

1.15 Bitfields

sample-format

- :float

stream-flags

- :no-flag
- :clip-off
- :dither-off

1.16 Enums

host-api-type-id

- :in-development
- :direct-sound
- :mme
- :asio
- :sound-manager
- :core-audio
- :oss
- :alsa
- :al
- :be-os
- :wdmks
- :jack
- :wasapi
- :audio-science-hpi

1.17 Other functions

`abort-stream` *pa-stream* [*Function*]

Details

Terminates audio processing immediately without waiting for pending buffers to complete.

`close-stream` *pa-stream* [*Function*]

Details

Closes an audio stream. If the audio stream is active it discards any pending buffers as if `abort-stream` had been called.

See also

- abort-stream

`device-info-default-high-input-latency` *instance* [*Function*]

Details

Default latency values for robust non-interactive applications (eg. playing sound files).

`device-info-default-high-output-latency` *instance* [*Function*]

Details

`device-info-default-low-input-latency` *instance* [*Function*]

Details

Default latency values for interactive performance.

`device-info-default-low-output-latency` *instance* [*Function*]

Details

`device-info-default-sample-rate` *instance* [*Function*]

Details

Sample rate

`device-info-host-api` *instance* [*Function*]

Details

note this is a host API index, not a type id.

`device-info-max-input-channels` *instance* [*Function*]

Details

maximum number of input channels

`device-info-max-output-channels` *instance* [*Function*]

Details

maximum number of output channels

`device-info-name` *instance* [*Function*]

Details

device name

`get-default-host-api` [*Function*]

Return Values

A non-negative value ranging from 0 to `(- (get-host-api-count ) 1)` indicating the default host API index or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Retrieve the index of the default host API. The default host API will be the lowest common denominator host API on the current platform and is unlikely to provide the best performance.

See also

- get-host-api-count

`get-default-input-device` [*Function*]

Return Values

The default input device index for the default host API, or raise no-device if no default input device is available or an error was encountered.

Details

Retrieve the index of the default input device. The result can be used in the inputDevice parameter to `open-stream` .

See also

- open-stream

`get-default-output-device` [*Function*]

Return Values

The default output device index for the default host API, or raise no-device if no default output device is available or an error was encountered.

Details

Retrieve the index of the default output device. The result can be used in the outputDevice parameter to `open-stream` .

Note

On the PC, the user can specify a default device by setting an environment variable. For example, to use device #1.

```
set PA_RECOMMENDED_OUTPUT_DEVICE=1
```

The user should first determine the available device ids by using `(print-devices )`.

See also

- open-stream
- print-devices

`get-device-count` [*Function*]

Return Values

A non-negative value indicating the number of available devices or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Retrieve the number of available devices.The number of available devices may be zero.

`get-device-info` *device* [*Function*]

Arguments

device — A valid device index in the range 0 to `(- (get-device-count ) 1)`

Return Values

A object of `device-info` . If the device parameter is out of range the function returns NIL.

Details

Retrieve `device-info` structure containing information about the specified device.

See also

- device-info
- get-device-count

`get-host-api-count` [*Function*]

Return Values

A non-negative value indicating the number of available host APIs or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Retrieve the number of available host APIs. Even if a host API is available it may have no devices available.

`get-host-api-info` *host-api* [*Function*]

Arguments

host-api — A valid host API index ranging from 0 to `(- (get-host-api-count ) 1)`

Return Values

An object of `host-api-info` describing a specific host API. If the hostApi parameter is out of range or an error is encountered, the function returns NIL.

Details

Retrieve a pointer to a structure containing information about a specific host Api.

See also

- get-host-api-count
- host-api-info

`get-sample-size` *format* [*Function*]

Details

Retrieve the size of a given sample format in bytes.The size in bytes of a single sample in the specified format, or paSampleFormatNotSupported if the format is not supported.

`get-stream-info` *pa-stream* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

Return Values

A object of `stream-info` structure. If the stream parameter invalid, or an error is encountered, the function returns NIL.

Details

Retrieve a object of class `stream-info` containing information about the specified stream.

See also

- stream-info
- open-stream

`get-stream-read-available` *pa-stream* [*Function*]

Return Values

Returns a non-negative value representing the maximum number of frames that can be read from the stream without blocking or busy waiting or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Retrieve the number of frames that can be read from the stream without waiting.

`get-stream-time` *pa-stream* [*Function*]

Details

Returns valid time values for the entire life of the stream, from when the stream is opened until it is closed. Starting and stopping the stream does not affect the passage of time returned by get-stream-time.

This time may be used for synchronizing other events to the audio stream, for example synchronizing audio to MIDI.

The stream's current time in seconds, or 0 if an error occurred.

`get-stream-write-available` *pa-stream* [*Function*]

Return Values

A non-negative value representing the maximum number of frames that can be written to the stream without blocking or busy waiting or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Retrieve the number of frames that can be written to the stream without waiting.

`get-version` [*Function*]

Details

Retrieve the release number of the currently running PortAudio build, eg 1900.

`get-version-text` [*Function*]

Details

Retrieve a textual description of the current PortAudio build,eg "PortAudio V19-devel 13 October 2002".

`host-api-device-index-to-device-index` *host-api host-api-device-index* [*Function*]

Arguments

host-api — A valid host API index ranging from 0 to `(- (get-host-api-count ) 1)`

host-api-device-index — A valid per-host device index in the range 0 to `(- (host-api-info-device-count (get-host-api-info host-api)) 1)`

Return Values

A non-negative index ranging from 0 to `(- (get-device-count ) 1)` or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Convert a host-API-specific device index to standard PortAudio device index. This function may be used in conjunction with the deviceCount field of PaHostApiInfo to enumerate all devices for the specified host API.

A invalid-host-api error indicates that the host API index specified by the hostApi parameter is out of range.

A invalid-device error indicates that the host-api-device-index parameter is out of range.

See also

- get-host-api-count

- host-api-info-device-count
- get-host-api-info
- get-device-count

`host-api-info-default-input-device` *instance* [*Function*]

Details

The default input device for this host API. The value will be a device index ranging from 0 to (- (get-device-count) 1), or no-device if no default input device is available.

`host-api-info-default-output-device` *instance* [*Function*]

Details

The default output device for this host API. The value will be a device index ranging from 0 to (- (get-device-count) 1), or paNoDevice if no default output device is available.

`host-api-info-device-count` *instance* [*Function*]

Details

The number of devices belonging to this host API. This field may be used in conjunction with host-api-device-index-to-device-index to enumerate all devices for this host API.

`host-api-info-name` *instance* [*Function*]

Details

A textual description of the host API for display on user interfaces.

`host-api-info-type` *instance* [*Function*]

Details

The well known unique identifier of this host API.

`host-api-type-id-to-host-api-index` *type* [*Function*]

Arguments

type — A unique host API identifier belonging to the PaHostApiTypeId enumeration.

Return Values

A valid host-api-idnex ranging from 0 to `(- (get-host-api-count ) 1)` or, raises an error if PortAudio is not initialized or

Details

Convert a static host API unique identifier, into a runtime host API index.

The host-api-not-found error indicates that the host API specified by the type parameter is not available.

See also

- get-host-api-count

`host-error-info-error-code` *instance* [*Function*]

Details

the error code returned

`host-error-info-error-text` *instance* [*Function*]

Details

a textual description of the error if available, otherwise a zero-length string

---

`host-error-info-host-api-type` *instance* [*Function*]

Details

the host API which returned the error code

`initialize` [*Function*]

Details

Library initialization function - call this before using PortAudio. This function initializes internal data structures and prepares underlying host APIs for use. With the exception of `get-version` , `get-version-text` , and `get-error-text` , this function MUST be called before using any other PortAudio API functions.

If initialize is called multiple times, each successful call must be matched with a corresponding call to `terminate` . Pairs of calls to initialize/`terminate` may overlap, and are not required to be fully nested.

Note that if initialize raises an error, `terminate` should NOT be called.

NIL if successful, otherwise raises an error indicating the cause of failure.

See also

- get-version
- get-version-text
- get-error-text
- terminate

`is-stream-active` *pa-stream* [*Function*]

Return Values

Returns one (1) when the stream is active (ie playing or recording audio), zero (0) when not playing or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Determine whether the stream is active. A stream is active after a successful call to `start-stream` , until it becomes inactive either as a result of a call to `stop-stream` or `abort-stream` . In the latter case, the stream is considered inactive after the last buffer has finished playing.

See also

- start-stream
- stop-stream
- abort-stream

`is-stream-stopped` *pa-stream* [*Function*]

Return Values

Returns one (1) when the stream is stopped, zero (0) when the stream is running or, raises an error if PortAudio is not initialized or an error is encountered.

Details

Determine whether the stream is stopped. A stream is considered to be stopped prior to a successful call to `start-stream` and after a successful call to `stop-stream` or `abort-stream` .

See also

- start-stream
- stop-stream
- abort-stream

`make-stream-parameters` [*Function*]

Details

Make stream-parameters object

`open-default-stream` *num-input num-output sample-format sample-rate frames-per-buffer* [*Function*]

Arguments

num-input-channels — The number of channels of sound that will be returned by `read-stream` . It can range from 1 to the value of max-input-channels in the `device-info` class for the default input device. If 0 the stream is opened as an output-only stream.

num-output-channels — The number of channels of sound to be passed to `write-stream` . It can range from 1 to the value of max-output-channels in the `device-info` class for the default output device. If 0 the stream is opened as an output-only stream.

sample-format — The sample format of both the input and output buffers passed to and from `read-stream` and `write-stream` . sample-format may be any of the formats described by the sample-format enumeration.

sample-rate — Same as `open-stream` parameter of the same name.

frames-per-buffer — Same as `open-stream` parameter of the same name.

Details

A simplified version of `open-stream` that opens the default input and/or output devices.

As for `open-stream`

See also

- open-stream
- read-stream
- device-info
- write-stream

`open-stream` *input-parameters output-parameters sample-rate frames-per-buffer stream-flags* [*Function*]

Arguments

input-parameters — A structure that describes the input parameters used by the opened stream. See `stream-parameters` for a description of these parameters. input-parameters must be NIL for output-only streams.

output-parameters — A structure that describes the output parameters used by the opened stream. See `stream-parameters` for a description of these parameters. output-parameters must be NIL for input-only streams.

sample-rate — The desired sample-rate. For full-duplex streams it is the sample rate for both input and output

frames-per-buffer — Preferred block granularity for a blocking read/write stream.

stream-flags — List of flags which modify the behavior of the streaming process. Some flags may only be relevant to certain buffer formats.

Return Values

Upon success pen-stream returns object of `pa-stream` class. The stream is inactive (stopped). If a call to open-stream fails, an error code is raised and the value of stream is NIL.

Details

Opens a stream for either input, output or both.

See also

- stream-parameters
- pa-stream

`pa-sleep` *msec* [*Function*]

Details

Put the caller to sleep for at least 'msec' milliseconds. This function is provided only as a convenience for authors of portable code (such as the tests and examples in the PortAudio distribution.)

The function may sleep longer than requested so don't rely on this for accurate musical timing.

`pa-stream-frames-per-buffer` *instance* [*Function*]

Details

Frames per buffer for current stream

`pa-stream-input-channels` *instance* [*Function*]

Details

Number of input channels

`pa-stream-input-sample-format` *instance* [*Function*]

Details

value of sample-format for input channel

`pa-stream-output-channels` *instance* [*Function*]

Details

Number of output channels

`pa-stream-output-sample-format` *instance* [*Function*]

Details

value of sample-format for output channel

`print-devices` [*Function*]

Details

List available sound devices, including device information.

`read-stream-into-array` *pa-stream array* [*Function*]

Arguments

pa-stream — A object of stream previously created with `open-stream` .

array — Simple array with has element-type equal to sample-format from `open-stream` . Size of array equal to `(* frames-per-buffer channel-count)`.

Return Values

NIL or, raises input-overflowed if input data was discarded by PortAudio after the previous call and before this call.

Details

Read samples from an input stream. The function doesn't return until the entire buffer has been filled - this may involve waiting for the operating system to supply the data.

See also

- open-stream

`start-stream` *pa-stream* [*Function*]

Details

Commences audio processing.

`stream-info-input-latency` *instance* [*Function*]

Details

The input latency of the stream in seconds. This value provides the most accurate estimate of input latency available to the implementation. It may differ significantly from the suggestedLatency value passed to open-stream. The value of this field will be zero (0.) for output-only streams.

`stream-info-output-latency` *instance* [*Function*]

Details

The output latency of the stream in seconds. This value provides the most accurate estimate of output latency available to the implementation. It may differ significantly from the suggestedLatency value passed to open-stream. The value of this field will be zero (0.) for input-only streams.

`stream-info-sample-rate` *instance* [*Function*]

Details

The sample rate of the stream in Hertz (samples per second). In cases where the hardware sample rate is inaccurate and PortAudio is aware of it, the value of this field may be different from the sample-rate parameter passed to open-stream. If information about the actual hardware sample rate is not available, this field will have the same value as the sample-rate parameter passed to open-stream.

`stream-parameters-channel-count` *instance* [*Function*]

Details

The number of channels of sound to be delivered to the stream callback.

`stream-parameters-device` *instance* [*Function*]

Details

A valid device index in the range 0 to `(- (get-device-count ) 1)` specifying the device to be used. This field must not be set to paNoDevice.

See also

- get-device-count

`stream-parameters-sample-format` *instance* [*Function*]

Details

The sample format of the buffer provided to read-stream or write-stream.

`stream-parameters-suggested-latency` *instance* [*Function*]

Details

The desired latency in seconds. Where practical, implementations should configure their latency based on these parameters, otherwise they may choose the closest viable latency instead. Unless the suggested latency is greater than the absolute upper limit for the device implementations should round the suggestedLatency up to the next practical value - ie to provide an equal or higher latency than suggestedLatency wherever possible.

`terminate` [*Function*]

Return Values

NIL if successful, otherwise raises an error indicating the cause of failure.

Details

Library termination function - call this when finished using PortAudio. This function deallocates all resources allocated by PortAudio since it was initialized by a call to `initialize` . In cases where `initialize` has been called multiple times, each call must be matched with a corresponding call to terminate. The final matching call to terminate will automatically close any PortAudio streams that are still open.

terminate MUST be called before exiting a program which uses PortAudio. Failure to do so may result in serious resource leaks, such as audio devices not being available until the next reboot.

See also

- initialize

1.18 Other macros

`with-audio` &*body* *body* [*Macro*]

Details

Execute body in PortAudio initialize/terminate environment.

`with-audio-stream` (*var* `input-parameters` [*Macro*]
`output-parameters` &`key` `(sample-rate 44100.0d0)`
`(`*frames-per-buffer* `1024)` (*stream-flags* `(list`
`no-flag)))` &*body* *body*

Details

Execute body with opened and started stream VAR and shut down the stream after it is done. It is required use these macro in with-audio or initialize/terminate environment.

`with-default-audio-stream` [*Macro*]
`(`*var* `num-input num-output` &`key (sample-format`
`float)` (*sample-rate* `44100.0d0)` (*frames-per-buffer* `1024))` &*body* *body*

Details

Execute body with opened and started stream VAR and shut down the stream after it is done. It is required use these macro in with-audio or initialize/terminate environment.

1.19 Other classes

`device-info` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

default-high-input-latency — Default latency values for robust non-interactive applications (eg. playing sound files).

default-high-output-latency —

default-low-input-latency — Default latency values for interactive performance.

default-low-output-latency —

default-sample-rate — Sample rate

host-api — note this is a host API index, not a type id.

max-input-channels —

max-output-channels —

name — Device name.

p —

struct-version — Structure version.

Details

A structure providing information and capabilities of PortAudio devices. Devices may support input, output or both input and output.

`host-api-info` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

type — The well known unique identifier of this host API.

default-input-device — The default input device for this host API. The value will be a device index ranging from 0 to (- (get-device-count) 1), or no-device if no default input device is available.

default-output-device — The default output device for this host API. The value will be a device index ranging from 0 to (- (get-device-count) 1), or paNoDevice if no default output device is available.

device-count — The number of devices belonging to this host API. This field may be used in conjunction with host-api-device-index-to-device-index to enumerate all devices for this host API.

name — A textual description of the host API for display on user interfaces.

struct-version — Struct version.

Details

A structure containing information about a particular host API.

`host-error-info` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

error-code — the error code returned

error-text — a textual description of the error if available, otherwise a zero-length string

host-api-type — the host API which returned the error code

Details

Structure used to return information about a host error condition.

`pa-stream` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

frames-per-buffer — Frames per buffer

handle — Foreign pointer to pa-stream

input-channels — Number of input channels

input-sample-format — Format of input samples

output-channels — Number of output channels

output-sample-format — Format of output samples

Details

A single PaStream can provide multiple channels of real-time streaming audio input and output to a client application. A stream provides access to audio hardware represented by one or more devices. Depending on the underlying Host API, it may be possible to open multiple streams using the same device, however this behavior is implementation defined. Portable applications should assume that a device may be simultaneously used by at most one stream.

`stream-info` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

input-latency — The input latency of the stream in seconds. This value provides the most accurate estimate of input latency available to the implementation. It may differ significantly from the suggestedLatency value passed to open-stream. The value of this field will be zero (0.) for output-only streams.

output-latency — The output latency of the stream in seconds. This value provides the most accurate estimate of output latency available to the implementation. It may differ significantly from the suggestedLatency value passed to open-stream. The value of this field will be zero (0.) for input-only streams.

sample-rate — The sample rate of the stream in Hertz (samples per second). In cases where the hardware sample rate is inaccurate and PortAudio is aware of it, the value of this field may be different from the sample-rate parameter passed to open-stream. If information about the actual hardware sample rate is

not available, this field will have the same value as the sample-rate parameter passed to open-stream.

struct-version — Struct version

Details

A structure containing unchanging information about an open stream.

`stream-parameters` [*Class*]

Superclasses

common-lisp:standard-object, sb-pcl::slot-object, common-lisp:t

Documented Subclasses

None

Direct Slots

channel-count — The number of channels of sound to be delivered to the stream callback.

device — A valid device index in the range 0 to (- get-device-count 1) specifying the device to be used. This field must not be set to paNoDevice.

sample-format — The sample format of the buffer provided to read-stream or write-stream.

suggested-latency — The desired latency in seconds. Where practical, implementations should configure their latency based on these parameters, otherwise they may choose the closest viable latency instead. Unless the suggested latency is greater than the absolute upper limit for the device implementations should round the suggestedLatency up to the next practical value - ie to provide an equal or higher latency than suggestedLatency wherever possible.

Details

Parameters for one direction (input or output) of a stream.

Chapter 2

The portaudio-tests package

This package contains tests/examples for PortAudio bindings.

2.1 Other functions

`test-read-write-converted-echo` [*Function*]

Details

Record input into an array; Separate array to channels; Merge channels into array; Play last array.

`test-read-write-echo` [*Function*]

Details

Record input into an array; Playback recorded data.

Index

abort-stream `abort-stream` function, 13
close-stream `close-stream` function, 13
device-info `device-info` class, 31
device-info-default-high-input-latency `device-info-default-high-input-latency` function, 13
device-info-default-high-output-latency `device-info-default-high-output-latency` function, 13
device-info-default-low-input-latency `device-info-default-low-input-latency` function, 14
device-info-default-low-output-latency `device-info-default-low-output-latency` function, 14
device-info-default-sample-rate `device-info-default-sample-rate` function, 14
device-info-host-api `device-info-host-api` function, 14
device-info-max-input-channels `device-info-max-input-channels` function, 14
device-info-max-output-channels `device-info-max-output-channels` function, 14
device-info-name `device-info-name` function, 15
get-default-host-api `get-default-host-api` function, 15
get-default-input-device `get-default-input-device` function, 15
get-default-output-device `get-default-output-device` function, 16
get-device-count `get-device-count` function, 16
get-device-info `get-device-info` function, 17
get-host-api-count `get-host-api-count` function, 17
get-host-api-info `get-host-api-info` function, 17
get-sample-size `get-sample-size` function, 18
get-stream-info `get-stream-info` function, 18
get-stream-read-available `get-stream-read-available` function, 19
get-stream-time `get-stream-time` function, 19
get-stream-write-available `get-stream-write-available` function, 19
get-version `get-version` function, 20
get-version-text `get-version-text` function, 20
host-api-device-index-to-host-api-device-index `host-api-device-index-to-host-api-device-index` function, 20
host-api-info `host-api-info` class, 32
host-api-info-default-input-device `host-api-info-default-input-device` function, 21
host-api-info-default-output-device `host-api-info-default-output-device` function, 21
host-api-info-device-count `host-api-info-device-count` function, 21
host-api-info-name `host-api-info-name` function, 22
host-api-info-type `host-api-info-type` function, 22
host-api-type-id-to-host-api-index `host-api-type-id-to-host-api-index` function, 22
host-error-info `host-error-info` class, 33
host-error-info-error-code `host-error-info-error-code` function, 22
host-error-info-error-text `host-error-info-error-text` function, 23
host-error-info-host-api-type `host-error-info-host-api-type` function, 23
initialize `initialize` function, 23
is-stream-active `is-stream-active` function, 23
is-stream-stopped `is-stream-stopped` function, 24
make-stream-parameters `make-stream-parameters` function, 24
merge-channels-into-array `merge-channels-into-array` function, 4

open-default-stream `open-default-stream` function, 25
open-stream `open-stream` function, 25
pa-sleep `pa-sleep` function, 26
pa-stream `pa-stream` class, 33
pa-stream-frames-per-buffer `pa-stream-frames-per-buffer` function, 26
pa-stream-input-channels `pa-stream-input-channels` function, 27
pa-stream-input-sample-format `pa-stream-input-sample-format` function, 27
pa-stream-output-channels `pa-stream-output-channels` function, 27
pa-stream-output-sample-format `pa-stream-output-sample-format` function, 27
print-devices `print-devices` function, 27
read-stream `read-stream` function, 3
read-stream-into-array `read-stream-into-array` function, 28
separate-array-to-channels `separate-array-to-channels` function, 4
start-stream `start-stream` function, 28
stream-info `stream-info` class, 34
stream-info-input-latency `stream-info-input-latency` function, 28
stream-info-output-latency `stream-info-output-latency` function, 29
stream-info-sample-rate `stream-info-sample-rate` function, 29
stream-parameters `stream-parameters` class, 35
stream-parameters-channel-count `stream-parameters-channel-count` function, 29
stream-parameters-device `stream-parameters-device` function, 29
stream-parameters-sample-format `stream-parameters-sample-format` function, 30
stream-parameters-suggested-latency `stream-parameters-suggested-latency` function, 30
terminate `terminate` function, 30
test-read-write-converted-echo `test-read-write-converted-echo` function, 36
test-read-write-echo `test-read-write-echo` function, 36
with-audio `with-audio` macro, 3, 31
with-audio-stream `with-audio-stream` macro, 31
with-default-audio-stream `with-default-audio-stream` macro, 3, 31
write-stream `write-stream` function, 5