cythonsounddevice

Release 0.0.1

Apr 17, 2020

Contents

CHAPTER 1

cython-sounddevice

1.1 Description

Python bindings for the PortAudio library to interface with audio streams. This project was inspired by python-sounddevice, but uses Cython instead of CFFI.

This allows for use in other Cython projects needing audio I/O without the performance penalty of the switching between Python and C/C++ contexts. All of the necessary classes, functions and data types have shared declarations for this purpose.

1.2 Links

- Documentation
  - https://cython-sounddevice.readthedocs.io/en/latest/
- Source Code
  - https://github.com/nocarryr/cython-soundevice

1.3 Usage

TODO

1.4 Dependencies

- Cython >= 0.29.1
- PortAudio

1.5 Installation

TODO

1.5.1 Linux

```
sudo apt-get install portaudio19-dev
```

1.5.2 Windows

TODO

1.5.3 MacOS

TODO

1.6 License

See the LICENSE file for license information (GPLv3).

1.6.1 Reference

cysounddevice.devices module

PortAudio class

HostApiInfo class

DeviceInfo class

cysounddevice.streams module

Stream class

StreamInfo class

StreamCallback class

C-API

`CallbackUserData`
Container for data used in *`_stream_callback()`*

int **`input_channels`**
Number of input channels

int **`output_channels`**
: Number of output channels

*SampleBuffer** **`in_buffer`**
: Pointer to a `SampleBuffer` to write input data to

*SampleBuffer** **`out_buffer`**
: Pointer to a `SampleBuffer` to read output data from

int **`_stream_callback`**(const void* *in_bfr*, void* *out_bfr*, unsigned long *frame_count*, const PaStreamCallbackTimeInfo* *time_info*, PaStreamCallbackFlags *status_flags*, void* *user_data*)
: Callback function that reads and writes input/output data using the `SampleBuffer` pointers stored in user_data as `CallbackUserData`

cysounddevice.buffer module

StreamBuffer class

StreamInputBuffer class

StreamOutputBuffer class

C-API

`SampleBuffer`
: A buffering structure with preallocated memory for use in `_stream_callback`

BufferItem ***`items`**
: Buffer array of `BufferItem`

Py_ssize_t **`length`**
: Number of `items` to allocate

Py_ssize_t **`itemsize`**
: Size in bytes per sample

Py_ssize_t **`item_length`**
: Number of samples to allocate for each `BufferItem` (block size)

Py_ssize_t **`nchannels`**
: Number of channels

Py_ssize_t **`write_index`**
: Index of the next item to use for writing

Py_ssize_t **`read_index`**
: Index of the next item to use for reading

BLOCK_t **`current_block`**
: The current block of samples

int **`read_available`**
: Number of items available to read from

int **`write_available`**
: Number of items available to write to

`BufferItem`
: A single item used to store data for `SampleBuffer`

SampleTime_s **start_time**
The time of the first sample in the item's buffer, as reported by PortAudio

Py_ssize_t **index**
Index of the item within its parent `SampleBuffer`

Py_ssize_t **length**
Number of samples the item contains

Py_ssize_t **itemsize**
Size in bytes per sample

Py_ssize_t **nchannels**
Number of channels

Py_ssize_t **total_size**
The total size in bytes to allocate ‘‘ length * itemsize * nchannels ‘‘

char ***bfr**
Pointer to the preallocated buffer

*SampleBuffer** **sample_buffer_create** (*SampleTime_s start_time*, Py_ssize_t *length*, Py_ssize_t *nchannels*, Py_ssize_t *itemsize*)
Creates a `SampleBuffer` and child items (`BufferItem`), allocating all required char buffers.

void **sample_buffer_destroy** (*SampleBuffer* bfr*)
Deallocates the given `SampleBuffer` and all of its child items.

int **sample_buffer_write** (*SampleBuffer* bfr*, const void **data*, Py_ssize_t *length*)
Copy the given data to the next available item in the given `SampleBuffer`. If no items are available to write (the buffer is full), no data is copied.

Returns 1 if successful

*SampleTime_s** **sample_buffer_read** (*SampleBuffer* bfr*, char **data*, Py_ssize_t *length*)
Copy data from the next available item into the given buffer.

Returns: A `SampleTime_s` pointer to the `BufferItem.start_time` describing the source timing of the data. If no data is available, returns `NULL`.

*SampleTime_s** **sample_buffer_read_sf32** (*SampleBuffer* bfr*, float[:, :] *data*)
Copy stream data from a `SampleBuffer` into a `float` array

Deinterleaves the stream and casts it to 32-bit float. A typed memoryview may be used.

The sample format must be `paFloat32`.

Returns: A `SampleTime_s` pointer to the `BufferItem.start_time` describing the source timing of the data. If no data is available, returns `NULL`.

cysounddevice.types module

SampleTime class

C-API

`SampleFormat`

PaSampleFormat **pa_ident**

Py_ssize_t **bit_width**

bint **`is_signed`**

bint **`is_float`**

bint **`is_24bit`**

void* **`dtype_ptr`**

`SampleTime_s`

PaTime **`pa_time`**
Time in seconds

PaTime **`time_offset`**
Time offset in seconds

SAMPLE_RATE_t **`sample_rate`**
Sample rate

Py_ssize_t **`block_size`**
Number of samples per block

BLOCK_t **`block`**
Block count

Py_ssize_t **`block_index`**
Index within the block

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index