

LIBFFADO

2.0.0

Generated by Doxygen 1.7.3

Mon Apr 18 2011 14:49:34

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	ffado_device_info Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	6
3.1.2.1	device_spec_strings	6
3.1.2.2	nb_device_spec_strings	6
3.1.2.3	reserved	6
3.2	ffado_options Struct Reference	6
3.2.1	Detailed Description	6
3.2.2	Field Documentation	7
3.2.2.1	nb_buffers	7
3.2.2.2	packetizer_priority	7
3.2.2.3	period_size	7
3.2.2.4	realtime	7
3.2.2.5	reserved	7
3.2.2.6	sample_rate	7
3.2.2.7	slave_mode	7
3.2.2.8	snoop_mode	7
3.2.2.9	verbose	7
4	File Documentation	9
4.1	ffado.h File Reference	9
4.1.1	Define Documentation	11
4.1.1.1	FFADO_IGNORE_CAPTURE	11
4.1.1.2	FFADO_IGNORE_PLAYBACK	11
4.1.1.3	FFADO_MAX_NAME_LEN	11
4.1.1.4	FFADO_MAX_SPECSTRING_LENGTH	11
4.1.1.5	FFADO_MAX_SPECSTRINGS	11
4.1.1.6	FFADO_STREAMING_MAX_URL_LENGTH	11
4.1.2	Typedef Documentation	11
4.1.2.1	ffado_device_info_t	11
4.1.2.2	ffado_device_t	12
4.1.2.3	ffado_handle_t	12

4.1.2.4	<code>ffado_nframes_t</code>	12
4.1.2.5	<code>ffado_options_t</code>	12
4.1.2.6	<code>ffado_sample_t</code>	12
4.1.3	Enumeration Type Documentation	12
4.1.3.1	<code>ffado_direction</code>	12
4.1.3.2	<code>ffado_streaming_audio_datatype</code>	12
4.1.3.3	<code>ffado_streaming_stream_type</code>	12
4.1.3.4	<code>ffado_wait_response</code>	13
4.1.4	Function Documentation	13
4.1.4.1	<code>ffado_get_api_version</code>	13
4.1.4.2	<code>ffado_get_version</code>	13
4.1.4.3	<code>ffado_streaming_capture_stream_onoff</code>	13
4.1.4.4	<code>ffado_streaming_finish</code>	13
4.1.4.5	<code>ffado_streaming_get_audio_datatype</code>	14
4.1.4.6	<code>ffado_streaming_get_capture_stream_name</code>	14
4.1.4.7	<code>ffado_streaming_get_capture_stream_type</code>	14
4.1.4.8	<code>ffado_streaming_get_nb_capture_streams</code>	14
4.1.4.9	<code>ffado_streaming_get_nb_playback_streams</code>	15
4.1.4.10	<code>ffado_streaming_get_playback_stream_name</code>	15
4.1.4.11	<code>ffado_streaming_get_playback_stream_type</code>	15
4.1.4.12	<code>ffado_streaming_init</code>	15
4.1.4.13	<code>ffado_streaming_playback_stream_onoff</code>	16
4.1.4.14	<code>ffado_streaming_prepare</code>	16
4.1.4.15	<code>ffado_streaming_reset</code>	16
4.1.4.16	<code>ffado_streaming_set_audio_datatype</code>	17
4.1.4.17	<code>ffado_streaming_set_capture_stream_buffer</code>	17
4.1.4.18	<code>ffado_streaming_set_playback_stream_buffer</code>	17
4.1.4.19	<code>ffado_streaming_start</code>	18
4.1.4.20	<code>ffado_streaming_stop</code>	18
4.1.4.21	<code>ffado_streaming_transfer_buffers</code>	18
4.1.4.22	<code>ffado_streaming_transfer_capture_buffers</code>	19
4.1.4.23	<code>ffado_streaming_transfer_playback_buffers</code>	19
4.1.4.24	<code>ffado_streaming_wait</code>	20

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

ffado_device_info	5
ffado_options	6

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

ffado.h	9
-----------------------------------	---

Chapter 3

Data Structure Documentation

3.1 `ffado_device_info` Struct Reference

```
#include <ffado.h>
```

Data Fields

- unsigned int `nb_device_spec_strings`
- char ** `device_spec_strings`
- int32_t `reserved` [32]

3.1.1 Detailed Description

This struct serves to define the devices that should be used by the library `device_spec_strings` is an array of pointers that should contain `nb_device_spec_strings` valid pointers to strings.

The spec strings should be null terminated and can be no longer than `FFADO_MAX_SPECSTRINGS`.

```
nb_device_spec_strings < FFADO_MAX_SPECSTRING_LENGTH nb_device_spec_strings >= 0
```

If `nb_device_spec_strings == 0`, all busses are scanned for attached devices, and all found devices that are supported are combined into one large pseudo-device. The device order is defined by the GUID of the device. Devices with lower GUID's will be the first ones.

If multiple device specifications are present, the device order is defined as follows:

- device(s) that correspond to a spec string with a lower index will be added before devices from higher indexes.
- if a spec string results in multiple devices, they are sorted by GUID unless the spec format dictates otherwise.

The actual meaning of the device specification should be one of the following:

- Format 1: "hw:x[,y[,z]]" x = the firewire bus to use ('port' in raw1394 terminology) (mandatory) y = the node id the device currently has (bus resets might change that, but FFADO will track these changes and keep using the device specified on startup) (optional) z = the stream direction to use. 0 => capture (record) channels only 1 => playback channels only other/unspecified => both playback and capture (optional)
- Format 2: the device alias as defined in the ffado config file (UNIMPLEMENTED)

3.1.2 Field Documentation

3.1.2.1 `char** ffado_device_info::device_spec_strings`

3.1.2.2 `unsigned int ffado_device_info::nb_device_spec_strings`

3.1.2.3 `int32_t ffado_device_info::reserved[32]`

The documentation for this struct was generated from the following file:

- [ffado.h](#)

3.2 ffado_options Struct Reference

```
#include <ffado.h>
```

Data Fields

- `int32_t sample_rate`
- `int32_t period_size`
- `int32_t nb_buffers`
- `int32_t realtime`
- `int32_t packetizer_priority`
- `int32_t verbose`
- `int32_t slave_mode`
- `int32_t snoop_mode`
- `int32_t reserved [24]`

3.2.1 Detailed Description

Structure to pass the options to the ffado streaming code.

3.2.2 Field Documentation

- 3.2.2.1 `int32_t ffado_options::nb_buffers`
- 3.2.2.2 `int32_t ffado_options::packetizer_priority`
- 3.2.2.3 `int32_t ffado_options::period_size`
- 3.2.2.4 `int32_t ffado_options::realtime`
- 3.2.2.5 `int32_t ffado_options::reserved[24]`
- 3.2.2.6 `int32_t ffado_options::sample_rate`
- 3.2.2.7 `int32_t ffado_options::slave_mode`
- 3.2.2.8 `int32_t ffado_options::snoop_mode`
- 3.2.2.9 `int32_t ffado_options::verbose`

The documentation for this struct was generated from the following file:

- [ffado.h](#)

Chapter 4

File Documentation

4.1 ffado.h File Reference

```
#include <stdlib.h>
```

Data Structures

- struct [ffado_device_info](#)
- struct [ffado_options](#)

Defines

- #define [FFADO_MAX_NAME_LEN](#) 256
- #define [FFADO_STREAMING_MAX_URL_LENGTH](#) 2048
- #define [FFADO_IGNORE_CAPTURE](#) (1<<0)
- #define [FFADO_IGNORE_PLAYBACK](#) (1<<1)
- #define [FFADO_MAX_SPECSTRING_LENGTH](#) 256
- #define [FFADO_MAX_SPECSTRINGS](#) 64

Typedefs

- typedef struct ffado_handle * [ffado_handle_t](#)
- typedef struct _ffado_device [ffado_device_t](#)
- typedef unsigned int [ffado_sample_t](#)
- typedef unsigned int [ffado_nframes_t](#)
- typedef struct [ffado_device_info](#) [ffado_device_info_t](#)
- typedef struct [ffado_options](#) [ffado_options_t](#)

Enumerations

- enum `ffado_direction` { `FFADO_CAPTURE` = 0, `FFADO_PLAYBACK` = 1 }
- enum `ffado_streaming_stream_type` {
 `ffado_stream_type_invalid` = -1, `ffado_stream_type_unknown` = 0, `ffado_stream_type_audio` = 1, `ffado_stream_type_midi` = 2,
 `ffado_stream_type_control` = 3 }
- enum `ffado_streaming_audio_datatype` { `ffado_audio_datatype_error` = -1, `ffado_audio_datatype_int24` = 0, `ffado_audio_datatype_float` = 1 }
- enum `ffado_wait_response` { `ffado_wait_shutdown` = -3, `ffado_wait_error` = -2, `ffado_wait_xrun` = -1, `ffado_wait_ok` = 0 }

Functions

- const char * `ffado_get_version` ()
- int `ffado_get_api_version` ()
- `ffado_device_t` * `ffado_streaming_init` (`ffado_device_info_t` device_info, `ffado_options_t` options)
- int `ffado_streaming_prepare` (`ffado_device_t` *dev)
- void `ffado_streaming_finish` (`ffado_device_t` *dev)
- int `ffado_streaming_get_nb_capture_streams` (`ffado_device_t` *dev)
- int `ffado_streaming_get_nb_playback_streams` (`ffado_device_t` *dev)
- int `ffado_streaming_get_capture_stream_name` (`ffado_device_t` *dev, int number, char *buffer, size_t buffersize)
- int `ffado_streaming_get_playback_stream_name` (`ffado_device_t` *dev, int number, char *buffer, size_t buffersize)
- `ffado_streaming_stream_type` `ffado_streaming_get_capture_stream_type` (`ffado_device_t` *dev, int number)
- `ffado_streaming_stream_type` `ffado_streaming_get_playback_stream_type` (`ffado_device_t` *dev, int number)
- int `ffado_streaming_set_capture_stream_buffer` (`ffado_device_t` *dev, int number, char *buff)
- int `ffado_streaming_capture_stream_onoff` (`ffado_device_t` *dev, int number, int on)
- int `ffado_streaming_set_playback_stream_buffer` (`ffado_device_t` *dev, int number, char *buff)
- int `ffado_streaming_playback_stream_onoff` (`ffado_device_t` *dev, int number, int on)
- `ffado_streaming_audio_datatype` `ffado_streaming_get_audio_datatype` (`ffado_device_t` *dev)
- int `ffado_streaming_set_audio_datatype` (`ffado_device_t` *dev, `ffado_streaming_audio_datatype` t)
- int `ffado_streaming_start` (`ffado_device_t` *dev)
- int `ffado_streaming_stop` (`ffado_device_t` *dev)
- int `ffado_streaming_reset` (`ffado_device_t` *dev)
- `ffado_wait_response` `ffado_streaming_wait` (`ffado_device_t` *dev)
- int `ffado_streaming_transfer_buffers` (`ffado_device_t` *dev)

- int `ffado_streaming_transfer_playback_buffers` (`ffado_device_t *dev`)
- int `ffado_streaming_transfer_capture_buffers` (`ffado_device_t *dev`)

4.1.1 Define Documentation

4.1.1.1 `#define FFADO_IGNORE_CAPTURE (1<<0)`

4.1.1.2 `#define FFADO_IGNORE_PLAYBACK (1<<1)`

4.1.1.3 `#define FFADO_MAX_NAME_LEN 256`

4.1.1.4 `#define FFADO_MAX_SPECSTRING_LENGTH 256`

4.1.1.5 `#define FFADO_MAX_SPECSTRINGS 64`

4.1.1.6 `#define FFADO_STREAMING_MAX_URL_LENGTH 2048`

4.1.2 Typedef Documentation

4.1.2.1 `typedef struct ffado_device_info ffado_device_info_t`

This struct serves to define the devices that should be used by the library `device_spec_strings` is an array of pointers that should contain `nb_device_spec_strings` valid pointers to strings.

The spec strings should be null terminated and can be no longer than `FFADO_MAX_SPECSTRINGS`.

`nb_device_spec_strings < FFADO_MAX_SPECSTRING_LENGTH` `nb_device_spec_strings >= 0`

If `nb_device_spec_strings == 0`, all busses are scanned for attached devices, and all found devices that are supported are combined into one large pseudo-device. The device order is defined by the GUID of the device. Devices with lower GUID's will be the first ones.

If multiple device specifications are present, the device order is defined as follows:

- device(s) that correspond to a spec string with a lower index will be added before devices from higher indexes.
- if a spec string results in multiple devices, they are sorted by GUID unless the spec format dictates otherwise.

The actual meaning of the device specification should be one of the following:

- Format 1: "hw:x[,y[,z]]" x = the firewire bus to use ('port' in raw1394 terminology) (mandatory) y = the node id the device currently has (bus resets might change that, but FFADO will track these changes and keep using the device specified on startup) (optional) z = the stream direction to use. 0 => capture (record)

channels only 1 => playback channels only other/unspecified => both playback and capture (optional)

- Format 2: the device alias as defined in the ffado config file (UNIMPLEMENTED)

4.1.2.2 `typedef struct _ffado_device ffado_device_t`

4.1.2.3 `typedef struct ffado_handle* ffado_handle_t`

4.1.2.4 `typedef unsigned int ffado_nframes_t`

4.1.2.5 `typedef struct ffado_options ffado_options_t`

Structure to pass the options to the ffado streaming code.

4.1.2.6 `typedef unsigned int ffado_sample_t`

The sample format used by the ffado streaming API

4.1.3 Enumeration Type Documentation

4.1.3.1 `enum ffado_direction`

Enumerator:

FFADO_CAPTURE
FFADO_PLAYBACK

4.1.3.2 `enum ffado_streaming_audio_datatype`

Audio data types known to the API

Enumerator:

ffado_audio_datatype_error
ffado_audio_datatype_int24
ffado_audio_datatype_float

4.1.3.3 `enum ffado_streaming_stream_type`

The types of streams supported by the API

A ffado_audio type stream is a stream that consists of successive samples. The format is a 24bit UINT in host byte order, aligned as the 24LSB's of the 32bit UINT of the read/write buffer. The wait operation looks at this type of streams only.

A `ffado_midi` type stream is a stream of midi bytes. The bytes are 8bit UINT, aligned as the first 8LSB's of the 32bit UINT of the read/write buffer.

A `ffado_control` type stream is a stream that provides control information. The format of this control information is undefined, and the stream should be ignored.

Enumerator:

ffado_stream_type_invalid
ffado_stream_type_unknown
ffado_stream_type_audio
ffado_stream_type_midi
ffado_stream_type_control

4.1.3.4 enum ffado_wait_response

Wait responses

Enumerator:

ffado_wait_shutdown
ffado_wait_error
ffado_wait_xrun
ffado_wait_ok

4.1.4 Function Documentation

4.1.4.1 int ffado_get_api_version ()

4.1.4.2 const char* ffado_get_version ()

4.1.4.3 int ffado_streaming_capture_stream_onoff (ffado_device_t * dev, int number, int on)

4.1.4.4 void ffado_streaming_finish (ffado_device_t * dev)

Finishes the FFADO streaming. Cleans up all internal data structures and terminates connections.

Parameters

<i>dev</i>	the ffado device to be closed.
------------	--------------------------------

4.1.4.5 `ffado_streaming_audio_datatype ffado_streaming_get_audio_datatype (ffado_device_t * dev)`

4.1.4.6 `int ffado_streaming_get_capture_stream_name (ffado_device_t * dev, int number, char * buffer, size_t buffersize)`

Copies the capture channel name into the specified buffer

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number
<i>buffer</i>	the buffer to copy the name into. has to be allocated.
<i>buffersize</i>	the size of the buffer

Returns

the number of characters copied into the buffer

4.1.4.7 `ffado_streaming_stream_type ffado_streaming_get_capture_stream_type (ffado_device_t * dev, int number)`

Returns the type of a capture channel

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number

Returns

the channel type

4.1.4.8 `int ffado_streaming_get_nb_capture_streams (ffado_device_t * dev)`

Returns the amount of capture channels available

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

the number of capture streams present & active on the device. can be 0. returns -1 upon error.

4.1.4.9 int ffado_streaming_get_nb_playback_streams (ffado_device_t * dev)

Returns the amount of playack channels available

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

the number of playback streams present & active on the device. can be 0. returns -1 upon error.

4.1.4.10 int ffado_streaming_get_playback_stream_name (ffado_device_t * dev, int number, char * buffer, size_t buffersize)

Copies the playback channel name into the specified buffer

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number
<i>buffer</i>	the buffer to copy the name into. has to be allocated.
<i>buffersize</i>	the size of the buffer

Returns

the number of characters copied into the buffer

4.1.4.11 ffado_streaming_stream_type ffado_streaming_get_playback_stream_type (ffado_device_t * dev, int number)

Returns the type of a playback channel

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number

Returns

the channel type

4.1.4.12 ffado_device_t* ffado_streaming_init (ffado_device_info_t device_info, ffado_options_t options)

Initializes the streaming from/to a FFADO device. A FFADO device is a virtual device composed of several BeBoB or compatible devices, linked together in one sync domain.

This prepares all IEEE1394 related stuff and sets up all buffering. It elects a sync master if necessary.

Parameters

<i>device_info</i>	provides a way to specify the virtual device
<i>options</i>	options regarding buffers, ieee1394 setup, ...

Returns

Opaque device handle if successful. If this is NULL, the init operation failed.

4.1.4.13 `int ffado_streaming_playback_stream_onoff (ffado_device_t * dev, int number, int on)`

4.1.4.14 `int ffado_streaming_prepare (ffado_device_t * dev)`

preparation should be done after setting all per-stream parameters the way you want them. being buffer data type etc...

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

preparation should be done after setting all per-stream parameters the way you want them. being buffer data type etc...

Parameters

<i>dev</i>	
------------	--

Returns

4.1.4.15 `int ffado_streaming_reset (ffado_device_t * dev)`

Resets the streaming as if it was stopped and restarted. The difference is that the connections are not necessarily broken and restored.

All buffers are reset in the initial state and all data in them is lost.

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

0 on success, -1 on failure.

4.1.4.16 `int ffado_streaming_set_audio_datatype (ffado_device_t * dev,
ffado_streaming_audio_datatype t)`

4.1.4.17 `int ffado_streaming_set_capture_stream_buffer (ffado_device_t * dev, int number,
char * buff)`

Sets the decode buffer for the stream. This allows for zero-copy decoding. The call to `ffado_streaming_transfer_buffers` will decode one period of the stream to this buffer. Make sure it is large enough.

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number
<i>buff</i>	a pointer to the sample buffer, make sure it is large enough i.e. <code>sizeof(your_sample_type)*period_size</code>
<i>t</i>	the type of the buffer. this determines sample type and the decode function used.

Returns

-1 on error, 0 on success

4.1.4.18 `int ffado_streaming_set_playback_stream_buffer (ffado_device_t * dev, int number,
char * buff)`

Sets the encode buffer for the stream. This allows for zero-copy encoding (directly to the events). The call to `ffado_streaming_transfer_buffers` will encode one period of the stream from this buffer to the event buffer.

Parameters

<i>dev</i>	the ffado device
<i>number</i>	the stream number
<i>buff</i>	a pointer to the sample buffer
<i>t</i>	the type of the buffer. this determines sample type and the decode function used.

Returns

-1 on error, 0 on success

4.1.4.19 `int ffado_streaming_start (ffado_device_t * dev)`

Starts the streaming operation. This initiates the connections to the FFADO devices and starts the packet handling thread(s). This has to be called before any I/O can occur.

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

0 on success, -1 on failure.

4.1.4.20 `int ffado_streaming_stop (ffado_device_t * dev)`

Stops the streaming operation. This closes the connections to the FFADO devices and stops the packet handling thread(s).

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

0 on success, -1 on failure.

4.1.4.21 `int ffado_streaming_transfer_buffers (ffado_device_t * dev)`

Transfer & decode the events from the packet buffer to the sample buffers

This should be called after the wait call returns, before reading/writing the sample buffers with `ffado_streaming_[read|write]`.

The purpose is to allow more precise timing information. `ffado_streaming_wait` returns as soon as the period boundary is crossed, and can therefore be used to determine the time instant of this crossing (e.g. jack DLL).

The actual decoding work is done in this function and can therefore be omitted in this timing calculation. Note that you HAVE to call this function in order for the buffers not to overflow, and only call it when `ffado_streaming_wait` doesn't indicate a buffer xrun (xrun handler resets buffer).

If user supplied playback buffers are specified with `ffado_streaming_set_playback_buffers` their contents should be valid before calling this function. If user supplied capture buffers are specified with `ffado_streaming_set_capture_buffers` their contents are updated in this function.

Use either `ffado_streaming_transfer_buffers` to transfer all buffers at once, or use `ffado_streaming_transfer_playback_buffers` and `ffado_streaming_transfer_capture_buffers` to have more control. Don't use both.

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

-1 on error.

4.1.4.22 int ffado_streaming_transfer_capture_buffers (ffado_device_t * *dev*)

Transfer & decode the events from the packet buffer to the sample buffers

This should be called after the wait call returns, before reading the sample buffers with `ffado_streaming_read`.

If user supplied capture buffers are specified with `ffado_streaming_set_capture_buffers` their contents are updated in this function.

Use either `ffado_streaming_transfer_buffers` to transfer all buffers at once, or use `ffado_streaming_transfer_playback_buffers` and `ffado_streaming_transfer_capture_buffers` to have more control. Don't use both.

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

-1 on error.

4.1.4.23 int ffado_streaming_transfer_playback_buffers (ffado_device_t * *dev*)

Transfer & encode the events from the sample buffers to the packet buffer

This should be called after the wait call returns, after writing the sample buffers with `ffado_streaming_write`.

If user supplied playback buffers are specified with `ffado_streaming_set_playback_buffers` their contents should be valid before calling this function.

Use either `ffado_streaming_transfer_buffers` to transfer all buffers at once, or use `ffado_streaming_transfer_playback_buffers` and `ffado_streaming_transfer_capture_buffers` to have more control. Don't use both.

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

-1 on error.

4.1.4.24 ffado_wait_response ffado_streaming_wait (ffado_device_t * *dev*)

Waits until there is at least one period of data available on all capture connections and room for one period of data on all playback connections

Parameters

<i>dev</i>	the ffado device
------------	------------------

Returns

The number of frames ready. -1 when a problem occurred.

Index

device_spec_strings
 ffado_device_info, 6

ffado.h, 9

 ffado_audio_datatype_error, 12

 ffado_audio_datatype_float, 12

 ffado_audio_datatype_int24, 12

 FFADO_CAPTURE, 12

 FFADO_PLAYBACK, 12

 ffado_stream_type_audio, 13

 ffado_stream_type_control, 13

 ffado_stream_type_invalid, 13

 ffado_stream_type_midi, 13

 ffado_stream_type_unknown, 13

 ffado_wait_error, 13

 ffado_wait_ok, 13

 ffado_wait_shutdown, 13

 ffado_wait_xrun, 13

 ffado_device_info_t, 11

 ffado_device_t, 12

 ffado_direction, 12

 ffado_get_api_version, 13

 ffado_get_version, 13

 ffado_handle_t, 12

 FFADO_IGNORE_CAPTURE, 11

 FFADO_IGNORE_PLAYBACK, 11

 FFADO_MAX_NAME_LEN, 11

 FFADO_MAX_SPECSTRING_LENGTH, 11

 FFADO_MAX_SPECSTRINGS, 11

 ffado_nframes_t, 12

 ffado_options_t, 12

 ffado_sample_t, 12

 ffado_streaming_audio_datatype, 12

 ffado_streaming_capture_stream_onoff, 13

 ffado_streaming_finish, 13

 ffado_streaming_get_audio_datatype, 13

 ffado_streaming_get_capture_stream_name, 14

 ffado_streaming_get_capture_stream_type, 14

 ffado_streaming_get_nb_capture_streams, 14

 ffado_streaming_get_nb_playback_streams, 14

 ffado_streaming_get_playback_stream_name, 15

 ffado_streaming_get_playback_stream_type, 15

 ffado_streaming_init, 15

 FFADO_STREAMING_MAX_URL_LENGTH, 11

 ffado_streaming_playback_stream_onoff, 16

 ffado_streaming_prepare, 16

 ffado_streaming_reset, 16

 ffado_streaming_set_audio_datatype, 17

 ffado_streaming_set_capture_stream_buffer, 17

 ffado_streaming_set_playback_stream_buffer, 17

 ffado_streaming_start, 17

 ffado_streaming_stop, 18

 ffado_streaming_stream_type, 12

 ffado_streaming_transfer_buffers, 18

 ffado_streaming_transfer_capture_buffers, 19

 ffado_streaming_transfer_playback_buffers, 19

 ffado_streaming_wait, 19

 ffado_wait_response, 13

ffado_audio_datatype_error
 ffado.h, 12

ffado_audio_datatype_float
 ffado.h, 12

ffado_audio_datatype_int24
 ffado.h, 12

FFADO_CAPTURE
 ffado.h, 12

- FFADO_PLAYBACK
 - ffado.h, [12](#)
- ffado_stream_type_audio
 - ffado.h, [13](#)
- ffado_stream_type_control
 - ffado.h, [13](#)
- ffado_stream_type_invalid
 - ffado.h, [13](#)
- ffado_stream_type_midi
 - ffado.h, [13](#)
- ffado_stream_type_unknown
 - ffado.h, [13](#)
- ffado_wait_error
 - ffado.h, [13](#)
- ffado_wait_ok
 - ffado.h, [13](#)
- ffado_wait_shutdown
 - ffado.h, [13](#)
- ffado_wait_xrun
 - ffado.h, [13](#)
- ffado_device_info, [5](#)
 - device_spec_strings, [6](#)
 - nb_device_spec_strings, [6](#)
 - reserved, [6](#)
- ffado_device_info_t
 - ffado.h, [11](#)
- ffado_device_t
 - ffado.h, [12](#)
- ffado_direction
 - ffado.h, [12](#)
- ffado_get_api_version
 - ffado.h, [13](#)
- ffado_get_version
 - ffado.h, [13](#)
- ffado_handle_t
 - ffado.h, [12](#)
- FFADO_IGNORE_CAPTURE
 - ffado.h, [11](#)
- FFADO_IGNORE_PLAYBACK
 - ffado.h, [11](#)
- FFADO_MAX_NAME_LEN
 - ffado.h, [11](#)
- FFADO_MAX_SPECSTRING_LENGTH
 - ffado.h, [11](#)
- FFADO_MAX_SPECSTRINGS
 - ffado.h, [11](#)
- ffado_nframes_t
 - ffado.h, [12](#)
- ffado_options, [6](#)
 - nb_buffers, [7](#)
 - packetizer_priority, [7](#)
 - period_size, [7](#)
 - realtime, [7](#)
 - reserved, [7](#)
 - sample_rate, [7](#)
 - slave_mode, [7](#)
 - snoop_mode, [7](#)
 - verbose, [7](#)
- ffado_options_t
 - ffado.h, [12](#)
- ffado_sample_t
 - ffado.h, [12](#)
- ffado_streaming_audio_datatype
 - ffado.h, [12](#)
- ffado_streaming_capture_stream_onoff
 - ffado.h, [13](#)
- ffado_streaming_finish
 - ffado.h, [13](#)
- ffado_streaming_get_audio_datatype
 - ffado.h, [13](#)
- ffado_streaming_get_capture_stream_name
 - ffado.h, [14](#)
- ffado_streaming_get_capture_stream_type
 - ffado.h, [14](#)
- ffado_streaming_get_nb_capture_streams
 - ffado.h, [14](#)
- ffado_streaming_get_nb_playback_streams
 - ffado.h, [14](#)
- ffado_streaming_get_playback_stream_name
 - ffado.h, [15](#)
- ffado_streaming_get_playback_stream_type
 - ffado.h, [15](#)
- ffado_streaming_init
 - ffado.h, [15](#)
- FFADO_STREAMING_MAX_URL_LENGTH
 - ffado.h, [11](#)
- ffado_streaming_playback_stream_onoff
 - ffado.h, [16](#)
- ffado_streaming_prepare
 - ffado.h, [16](#)
- ffado_streaming_reset
 - ffado.h, [16](#)
- ffado_streaming_set_audio_datatype
 - ffado.h, [17](#)
- ffado_streaming_set_capture_stream_buffer
 - ffado.h, [17](#)
- ffado_streaming_set_playback_stream_buffer
 - ffado.h, [17](#)
- ffado_streaming_start
 - ffado.h, [17](#)

ffado_streaming_stop
 ffado.h, [18](#)

ffado_streaming_stream_type
 ffado.h, [12](#)

ffado_streaming_transfer_buffers
 ffado.h, [18](#)

ffado_streaming_transfer_capture_buffers
 ffado.h, [19](#)

ffado_streaming_transfer_playback_buffers
 ffado.h, [19](#)

ffado_streaming_wait
 ffado.h, [19](#)

ffado_wait_response
 ffado.h, [13](#)

nb_buffers
 ffado_options, [7](#)

nb_device_spec_strings
 ffado_device_info, [6](#)

packetizer_priority
 ffado_options, [7](#)

period_size
 ffado_options, [7](#)

realtime
 ffado_options, [7](#)

reserved
 ffado_device_info, [6](#)
 ffado_options, [7](#)

sample_rate
 ffado_options, [7](#)

slave_mode
 ffado_options, [7](#)

snoop_mode
 ffado_options, [7](#)

verbose
 ffado_options, [7](#)