# libfishsound Reference Manual

0.5.20

Generated by Doxygen 1.3.4

Tue Oct 21 08:27:01 2003

# Contents

# Chapter 1

# libfishsound Main Page

**Author:**
Conrad Parker

## 1.1 FishSound, the sound of fish!

This is the documentation for the FishSound C API. FishSound provides a simple programming interface for decoding and encoding audio data using Xiph.Org codecs (Vorbis and Speex).

### 1.1.1 API specification

The entire FishSound API is documented in the **fishsound.h** (p. 19) header file.

- **fishsound.h** (p. 19)

### 1.1.2 Library customization

You can build a smaller version of libfishsound to only decode or encode, or and you can choose to disable support for a particular codec.

- **Configuration** (p. 9)

### 1.1.3 Building against libfishsound

- **Building** (p. 14)

## 1.2 Licensing

libfishsound is provided under the following BSD-style open source license:

## 1.3 History and Motivation

libfishsound was designed and developed by Conrad Parker on the weekend of October 18-19 2003. Much of the API design follows the style of `libsndfile`.

# Chapter 2

# libfishsound Module Index

## 2.1 libfishsound Modules

Here is a list of all modules:

# Chapter 3

# libfishsound Data Structure Index

## 3.1 libfishsound Data Structures

Here are the data structures with brief descriptions:

# Chapter 4

# libfishsound File Index

## 4.1 libfishsound File List

Here is a list of all documented files with brief descriptions:

# Chapter 5

# libfishsound Module Documentation

## 5.1 Configuration

### 5.1.1 ./configure

It is possible to customize the functionality of libfishsound by using various ./configure flags when building it from source; for example you can build a smaller version of libfishsound to only decode or encode, or and you can choose to disable support for a particular codec. By default, both decoding and encoding support is built for all codecs found on the system.

For general information about using ./configure, see the file **INSTALL** (p. 11)

#### 5.1.1.1 Removing encoding support

Configuring with –disable-encode will remove all support for encoding:

- All internal encoding related functions will not be built
- The resulting library will not be linked against libvorbisenc
- Any attempt to call fishsound_new() with FISH_SOUND_ENCODE will fail, returning NULL
- Any attempt to call fishsound_encode() will return -1

#### 5.1.1.2 Removing decoding support

Configuring with –disable-decode will remove all support for decoding:

- All internal decoding related functions will not be built
- Any attempt to call fishsound_new() with FISH_SOUND_ENCODE will fail, returning NULL
- Any attempt to call fishsound_decode() will return -1

#### 5.1.1.3 Removing Vorbis support

Configuring with –disable-vorbis will remove all support for Vorbis:

- All internal Vorbis related functions will not be built
- The resulting library will not be linked against libvorbis or libvorbisenc

#### 5.1.1.4 Removing Speex support

Configuring with –disable-speex will remove all support for Speex:

- All internal Speex related functions will not be built
- The resulting library will not be linked against libspeex

#### 5.1.1.5 Configuration summary

Upon successful configuration, you should see something like this:

```
------------------------------------------------------------------------
  libfishsound 0.6.0:  Automatic configuration OK.

  General configuration:

    Experimental code: ........... no
    Decoding support: ............ yes
    Encoding support: ............ yes

  Library configuration (./src/libfishsound):

    Vorbis support: .............. yes
    Speex support: ............... yes

  Example programs (./src/examples):

    Ogg: liboggz example: ........ yes
    PCM: libsndfile1 example: .... yes

  Installation paths:

    libfishsound: ................ /usr/local/lib
    C header files: .............. /usr/local/include/fishsound
    Documentation: ............... /usr/local/share/doc/libfishsound

  Example programs will be built but not installed.
------------------------------------------------------------------------
```

## 5.2 Installation

### 5.2.1 INSTALL

```
Basic Installation
==================

   These are generic installation instructions.

   The `configure' shell script attempts to guess correct values for
various system-dependent variables used during compilation.  It uses
those values to create a `Makefile' in each directory of the package.
It may also create one or more `.h' files containing system-dependent
definitions.  Finally, it creates a shell script `config.status' that
you can run in the future to recreate the current configuration, a file
`config.cache' that saves the results of its tests to speed up
reconfiguring, and a file `config.log' containing compiler output
(useful mainly for debugging `configure').

   If you need to do unusual things to compile the package, please try
to figure out how `configure' could check whether to do them, and mail
diffs or instructions to the address given in the `README' so they can
be considered for the next release.  If at some point `config.cache'
contains results you don't want to keep, you may remove or edit it.

   The file `configure.in' is used to create `configure' by a program
called `autoconf'.  You only need `configure.in' if you want to change
it or regenerate `configure' using a newer version of `autoconf'.

The simplest way to compile this package is:

  1. `cd' to the directory containing the package's source code and type
     `./configure' to configure the package for your system.  If you're
     using `csh' on an old version of System V, you might need to type
     `sh ./configure' instead to prevent `csh' from trying to execute
     `configure' itself.

     Running `configure' takes awhile.  While running, it prints some
     messages telling which features it is checking for.

  2. Type `make' to compile the package.

  3. Optionally, type `make check' to run any self-tests that come with
     the package.

  4. Type `make install' to install the programs and any data files and
     documentation.

  5. You can remove the program binaries and object files from the
     source code directory by typing `make clean'.  To also remove the
     files that `configure' created (so you can compile the package for
     a different kind of computer), type `make distclean'.  There is
     also a `make maintainer-clean' target, but that is intended mainly
     for the package's developers.  If you use it, you may have to get
     all sorts of other programs in order to regenerate files that came
     with the distribution.

Compilers and Options
=====================

   Some systems require unusual options for compilation or linking that
the `configure' script does not know about.  You can give `configure'
initial values for variables by setting them in the environment.  Using
a Bourne-compatible shell, you can do that on the command line like
this:
     CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure
```

```
Or on systems that have the `env' program, you can do it like this:
     env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure

Compiling For Multiple Architectures
====================================

   You can compile the package for more than one kind of computer at the
same time, by placing the object files for each architecture in their
own directory.  To do this, you must use a version of `make' that
supports the `VPATH' variable, such as GNU `make'.  `cd' to the
directory where you want the object files and executables to go and run
the `configure' script.  `configure' automatically checks for the
source code in the directory that `configure' is in and in `..'.

   If you have to use a `make' that does not supports the `VPATH'
variable, you have to compile the package for one architecture at a time
in the source code directory.  After you have installed the package for
one architecture, use `make distclean' before reconfiguring for another
architecture.

Installation Names
==================

   By default, `make install' will install the package's files in
`/usr/local/bin', `/usr/local/man', etc.  You can specify an
installation prefix other than `/usr/local' by giving `configure' the
option `--prefix=PATH'.

   You can specify separate installation prefixes for
architecture-specific files and architecture-independent files.  If you
give `configure' the option `--exec-prefix=PATH', the package will use
PATH as the prefix for installing programs and libraries.
Documentation and other data files will still use the regular prefix.

   In addition, if you use an unusual directory layout you can give
options like `--bindir=PATH' to specify different values for particular
kinds of files.  Run `configure --help' for a list of the directories
you can set and what kinds of files go in them.

   If the package supports it, you can cause programs to be installed
with an extra prefix or suffix on their names by giving `configure' the
option `--program-prefix=PREFIX' or `--program-suffix=SUFFIX'.

Optional Features
=================

   Some packages pay attention to `--enable-FEATURE' options to
`configure', where FEATURE indicates an optional part of the package.
They may also pay attention to `--with-PACKAGE' options, where PACKAGE
is something like `gnu-as' or `x' (for the X Window System).  The
`README' should mention any `--enable-' and `--with-' options that the
package recognizes.

   For packages that use the X Window System, `configure' can usually
find the X include and library files automatically, but if it doesn't,
you can use the `configure' options `--x-includes=DIR' and
`--x-libraries=DIR' to specify their locations.

Specifying the System Type
==========================

   There may be some features `configure' can not figure out
automatically, but needs to determine by the type of host the package
will run on.  Usually `configure' can figure that out, but if it prints
a message saying it can not guess the host type, give it the
`--host=TYPE' option.  TYPE can either be a short name for the system
```

```
type, such as `sun4', or a canonical name with three fields:
     CPU-COMPANY-SYSTEM

See the file `config.sub' for the possible values of each field.  If
`config.sub' isn't included in this package, then this package doesn't
need to know the host type.

   If you are building compiler tools for cross-compiling, you can also
use the `--target=TYPE' option to select the type of system they will
produce code for and the `--build=TYPE' option to select the type of
system on which you are compiling the package.

Sharing Defaults
================

   If you want to set default values for `configure' scripts to share,
you can create a site shell script called `config.site' that gives
default values for variables like `CC', `cache_file', and `prefix'.
`configure' looks for `PREFIX/share/config.site' if it exists, then
`PREFIX/etc/config.site' if it exists.  Or, you can set the
`CONFIG_SITE' environment variable to the location of the site script.
A warning: not all `configure' scripts look for a site script.

Operation Controls
==================

   `configure' recognizes the following options to control how it
operates.

`--cache-file=FILE'
     Use and save the results of the tests in FILE instead of
     `./config.cache'.  Set FILE to `/dev/null' to disable caching, for
     debugging `configure'.

`--help'
     Print a summary of the options to `configure', and exit.

`--quiet'
`--silent'
`-q'
     Do not print messages saying which checks are being made.  To
     suppress all normal output, redirect it to `/dev/null' (any error
     messages will still be shown).

`--srcdir=DIR'
     Look for the package's source code in directory DIR.  Usually
     `configure' can determine that directory automatically.

`--version'
     Print the version of Autoconf used to generate the `configure'
     script, and exit.

`configure' also accepts some other, not widely useful, options.
```

## 5.3 Building against libfishsound

### 5.3.1 Using GNU autoconf

If you are using GNU autoconf, you do not need to call pkg-config directly. Use the following macro to determine if libfishsound is available:

```
PKG_CHECK_MODULES(FISHSOUND, fishsound $>$= 0.6.0,
                  HAVE_FISHSOUND="yes", HAVE_FISHSOUND="no")
if test "x$HAVE_FISHSOUND" = "xyes" ; then
  AC_SUBST(FISHSOUND_CFLAGS)
  AC_SUBST(FISHSOUND_LIBS)
fi
```

If libfishsound is found, HAVE_FISHSOUND will be set to "yes", and the autoconf variables FISHSOUND_CFLAGS and FISHSOUND_LIBS will be set appropriately.

### 5.3.2 Determining compiler options with pkg-config

If you are not using GNU autoconf in your project, you can use the pkg-config tool directly to determine the correct compiler options.

```
FISHSOUND_CFLAGS=‘pkg-config --cflags fishsound‘

FISHSOUND_LIBS=‘pkg-config --libs fishsound‘
```

# Chapter 6

# libfishsound Data Structure Documentation

## 6.1 FishSoundFormat Struct Reference

`#include <fishsound.h>`

### 6.1.1 Detailed Description

Info about a particular sound format.

#### Data Fields

- int **format**

  *FISH_SOUND_VORBIS, FISH_SOUND_SPEEX etc.*

- const char ∗ **name**

  *Printable name.*

- const char ∗ **extension**

  *Commonly used file extension.*

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

- **fishsound.h**

## 6.2 FishSoundInfo Struct Reference

`#include <fishsound.h>`

### 6.2.1 Detailed Description

Info about a particular encoder/decoder instance.

### Data Fields

- int **samplerate**

  *Sample rate of audio data in Hz.*

- int **channels**

  *Count of channels.*

- int **format**

  *FISH_SOUND_VORBIS, FISH_SOUND_SPEEX etc.*

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

- **fishsound.h**

# Chapter 7

# libfishsound File Documentation

## 7.1 constants.h File Reference

### 7.1.1 Detailed Description

Constants used by libfishsound.

### Enumerations

- enum **FishSoundMode** { **FISH_SOUND_DECODE** = 0x10, **FISH_SOUND_-ENCODE** = 0x20 }
- enum **FishSoundFormat** { **FISH_SOUND_UNKNOWN** = 0x00, **FISH_-SOUND_VORBIS** = 0x01, **FISH_SOUND_SPEEX** = 0x02 }
- enum **FishSoundCommand** {

  **FISH_SOUND_COMMAND_NOP** = 0x0000, **FISH_SOUND_GET_INFO** = 0x1000, **FISH_SOUND_GET_DECODE_INTERLEAVE** = 0x2000, **FISH_-SOUND_SET_DECODE_INTERLEAVE** = 0x2001,

  **FISH_SOUND_SET_ENCODE_VBR** = 0x4000, **FISH_SOUND_-COMMAND_MAX** }

### 7.1.2 Enumeration Type Documentation

#### 7.1.2.1 enum FishSoundCommand

**Enumeration values:**

***FISH_SOUND_COMMAND_NOP*** No operation.

***FISH_SOUND_GET_INFO*** Retrieve the **FishSoundInfo**(p. 16).

***FISH_SOUND_GET_DECODE_INTERLEAVE*** Query if decoding should be interleaved.

***FISH_SOUND_SET_DECODE_INTERLEAVE*** Set to 1 to interleave, 0 to non-interleave.

#### 7.1.2.2 enum FishSoundFormat

**Enumeration values:**

| | |
|---|---|
| ***FISH_SOUND_UNKNOWN*** | Unknown. |
| ***FISH_SOUND_VORBIS*** | Vorbis. |
| ***FISH_SOUND_SPEEX*** | Speex. |

#### 7.1.2.3 enum FishSoundMode

**Enumeration values:**

| | |
|---|---|
| ***FISH_SOUND_DECODE*** | Decode. |
| ***FISH_SOUND_ENCODE*** | Encode. |

## 7.2 fishsound.h File Reference

### 7.2.1 Detailed Description

The libfishsound C API.

### 7.2.2 General usage

All access is managed via a FishSound∗ handle. This is instantiated using **fish_sound_-new()**(p. 23) and should be deleted with **fish_sound_delete()**(p. 22) when no longer required. If there is a discontinuity in the input data (eg. after seeking in an input file), call **fish_sound_-reset()**(p. 23) to reset the internal codec state.

### 7.2.3 Decoding

To decode audio data using libfishsound, first create a FishSound∗ object with mode FISH_-SOUND_DECODE. **fish_sound_new()**(p. 23) will return a new FishSound∗ object, initialised for decoding, and the **FishSoundInfo**(p. 16) structure will be cleared.

### 7.2.4 Encoding

To encode audio data using libfishsound, first create a FishSound∗ object with mode FISH_-SOUND_ENCODE, and with a **FishSoundInfo**(p. 16) structure filled in with the required encoding parameters. **fish_sound_new()**(p. 23) will return a new FishSound∗ object initialised for encoding.

`#include <fishsound/constants.h>`

## Data Structures

- struct **FishSoundFormat**

  *Info about a particular sound format.*

- struct **FishSoundInfo**

  *Info about a particular encoder/decoder instance.*

## Typedefs

- typedef void ∗ **FishSound**

  *An opaque handle to a FishSound.*

- typedef int(∗ **FishSoundDecoded** )(**FishSound** ∗fsound, float ∗∗pcm, long frames, void ∗user_data)

  *Signature of a callback for libfishsound to call when it has decoded audio PCM data.*

- typedef int(∗ **FishSoundEncoded** )(**FishSound** ∗fsound, unsigned char ∗buf, long bytes, void ∗user_data)

  *Signature of a callback for libfishsound to call when it has encoded data.*

## Functions

- int **fish_sound_identify** (unsigned char ∗buf, long bytes)

  *Identify a codec based on the first few bytes of the data.*

- **FishSound** ∗ **fish_sound_new** (int mode, **FishSoundInfo** ∗fsinfo)

  *Instantiate a new FishSound∗ handle.*

- int **fish_sound_set_decoded_callback** (**FishSound** ∗fsound, **FishSoundDecoded** decoded, void ∗user_data)

  *Set the callback for libfishsound to call when it has a block of decoded audio ready.*

- int **fish_sound_set_encoded_callback** (**FishSound** ∗fsound, **FishSoundEncoded** encoded, void ∗user_data)

  *Set the callback for libfishsound to call when it has a block of encoded data ready.*

- long **fish_sound_decode** (**FishSound** ∗fsound, unsigned char ∗buf, long bytes)

  *Decode a block of data.*

- long **fish_sound_encode_i** (**FishSound** ∗fsound, float ∗∗pcm, long frames)

  *Encode a block of interleaved audio.*

- long **fish_sound_encode_n** (**FishSound** ∗fsound, float ∗pcm[ ], long frames)

  *Encode a block of non-interleaved audio.*

- int **fish_sound_reset** (**FishSound** ∗fsound)

  *Reset the codec state of a FishSound object.*

- int **fish_sound_delete** (**FishSound** ∗fsound)

  *Delete a FishSound object.*

- int **fish_sound_command** (**FishSound** ∗fsound, int command, void ∗data, int datasize)

  *Command interface.*

### 7.2.5 Typedef Documentation

#### 7.2.5.1 typedef int(∗ FishSoundDecoded)(FishSound ∗ fsound, float ∗∗ pcm, long frames, void ∗ user_data)

Signature of a callback for libfishsound to call when it has decoded audio PCM data.

**Parameters:**

***fsound*** The FishSound∗ handle

***pcm*** The decoded audio

***frames*** The count of frames decoded

***user_data*** Arbitrary user data

**Returns:**

0 to continue, non-zero to stop decoding immediately and return control to the **fish_sound_-decode()**(p. 21) caller

#### 7.2.5.2 typedef int(* FishSoundEncoded)(FishSound * fsound, unsigned char * buf, long bytes, void * user_data)

Signature of a callback for libfishsound to call when it has encoded data.

**Parameters:**

***fsound*** The FishSound* handle

***buf*** The encoded data

***bytes*** The count of bytes encoded

***user_data*** Arbitrary user data

**Returns:**

0 to continue, non-zero to stop encoding immediately and return control to the fish_sound_-encode() caller

### 7.2.6 Function Documentation

#### 7.2.6.1 int fish_sound_command (FishSound * *fsound*, int *command*, void * *data*, int *datasize*)

Command interface.

**Parameters:**

***fsound*** A FishSound* handle

***command*** The command action

***data*** Command data

***datasize*** Size of the data in bytes

**Returns:**

0 on success, -1 on failure

#### 7.2.6.2 long fish_sound_decode (FishSound * *fsound*, unsigned char * *buf*, long *bytes*)

Decode a block of data.

**Parameters:**

***fsound*** A FishSound* handle (created with mode FISH_SOUND_DECODE)

***buf*** A buffer of data

***bytes*** A count of bytes to decode (ie. the length of buf)

**Returns:**

The number of bytes consumed

#### 7.2.6.3 int fish_sound_delete (FishSound ∗ *fsound*)

Delete a FishSound object.

**Parameters:**

*fsound* A FishSound∗ handle

**Returns:**

0 on success, -1 on failure

#### 7.2.6.4 long fish_sound_encode_i (FishSound ∗ *fsound*, float ∗∗ *pcm*, long *frames*)

Encode a block of interleaved audio.

**Parameters:**

*fsound* A FishSound∗ handle (created with mode FISH_SOUND_ENCODE)

*pcm* A block of audio data

*frames* A count of frames to encode

**Returns:**

The number of frames encoded

#### 7.2.6.5 long fish_sound_encode_n (FishSound ∗ *fsound*, float ∗ *pcm*[ ], long *frames*)

Encode a block of non-interleaved audio.

**Parameters:**

*fsound* A FishSound∗ handle (created with mode FISH_SOUND_ENCODE)

*pcm* An array of pointers to audio data, one block per channel

*frames* A count of frames to encode

**Returns:**

The number of frames encoded

#### 7.2.6.6 int fish_sound_identify (unsigned char ∗ *buf*, long *bytes*)

Identify a codec based on the first few bytes of the data.

**Parameters:**

*buf* A pointer to the first few bytes of the data

*bytes* The count of bytes available at buf

**Returns:**

FISH_SOUND_VORBIS, FISH_SOUND_SPEEX etc. or FISH_SOUND_UNKNOWN

**Note:**

You should pass at least 8 bytes of data to this function :)

#### 7.2.6.7 FishSound∗ fish_sound_new (int *mode*, FishSoundInfo ∗ *fsinfo*)

Instantiate a new FishSound∗ handle.

**Parameters:**

*mode* FISH_SOUND_DECODE or FISH_SOUND_ENCODE

*fsinfo*

**Returns:**

A new FishSound∗ handle, or NULL on error

#### 7.2.6.8 int fish_sound_reset (FishSound ∗ *fsound*)

Reset the codec state of a FishSound object.

**Parameters:**

*fsound* A FishSound∗ handle

**Returns:**

0 on success, -1 on failure

#### 7.2.6.9 int fish_sound_set_decoded_callback (FishSound ∗ *fsound*, FishSoundDecoded *decoded*, void ∗ *user_data*)

Set the callback for libfishsound to call when it has a block of decoded audio ready.

**Parameters:**

*fsound* A FishSound∗ handle (created with mode FISH_SOUND_DECODE)

*decoded* The callback to call

*user_data* Arbitrary user data to pass to the callback

**Returns:**

0 on success, -1 on failure

#### 7.2.6.10 int fish_sound_set_encoded_callback (FishSound ∗ *fsound*, FishSoundEncoded *encoded*, void ∗ *user_data*)

Set the callback for libfishsound to call when it has a block of encoded data ready.

**Parameters:**

*fsound* A FishSound∗ handle (created with mode FISH_SOUND_ENCODE)

*encoded* The callback to call

*user_data* Arbitrary user data to pass to the callback

**Returns:**

0 on success, -1 on failure

# Index