TFM

0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

Chapter 6

Class Documentation

6.1 AS_EFFECT Class Reference

`#include <Compressor.h>`

Public Member Functions

- **Compressor** ()
- void **apply** (float ∗input, float ∗output, int samples, **SACBitstream::ChannelType::channeltype** channel)
- std::vector< std::vector< double > > **plot** (std::string chart)
- void **update** ()
- **Equalizer** ()
- void **apply** (float ∗input, float ∗output, int samples, **SACBitstream::ChannelType::channeltype** channel)
- std::vector< std::vector< double > > **plot** (std::string chart)
- void **peakingFilter** (float ∗input, float ∗output, int samples, double f_0, double **gain**, double Q, int **order**)
- void **lowShelfFilter** (float ∗input, float ∗output, int samples, double f_0, double **gain**, int **order**)
- void **highShelfFilter** (float ∗input, float ∗output, int samples, double f_0, double **gain**, int **order**)
- void **filter** (float ∗ **x**, float ∗ **y**, int samples, float ∗a, float ∗b, int **order**)
- **Reverb** ()
- void **apply** (float ∗input, float ∗output, int samples, **SACBitstream::ChannelType::channeltype** channel)
- std::vector< std::vector< double > > **plot** (std::string chart)
- void **schroederfilter** (float ∗input, float ∗output, int samples, bool addition, float **gain**, float g, int delay)
- void **schroederdiffusionfilter** (float ∗input, float ∗output, int samples, bool addition, float **gain**, float g, int delay)
- void **feedforwardfilter** (float ∗input, float ∗output, int samples, bool addition, float **gain**, float original, int delay)
- void **lowpassfeedbackfilter** (float ∗input, float ∗output, int samples, bool addition, float **gain**, float rs, float d, int delay)
- void **combfilter** (float ∗input, float ∗output, int samples, bool addition, float ∗a, float ∗b, int **order**, float a_delay, float b_delay, int delay)

Private Types

- enum **compressortype** { **downward**, **upward** }
- enum **compressorfunction** { **compression**, **expansion** }

Private Member Functions

- double **gain** (double inputlevel)

Private Attributes

- Compressor::compressortype **type**
- Compressor::compressorfunction **function**
- double **threshold**
- double **ratio**
- double **frequency** [**bands**]
- float **x** [**order**+1]
- float **y** [**order**+1]
- int **pointer** [**maxfilters**]
- int **filterindex**

Static Private Attributes

- static const int **bands** = 10

6.1.1 Member Enumeration Documentation

6.1.1.1 compressorfunction

`enum` **AS_EFFECT::compressorfunction** `[private]`

Enumerator

compression	effect is a compressor
expansion	effect is a expander

6.1.1.2 compressortype

`enum` **AS_EFFECT::compressortype** `[private]`

Enumerator

downward	downward compression (reduces loud sounds over a certain threshold while quiet sounds remain unaffected)
upward	upward compression (increases the loudness of sounds below a certain threshold while leaving louder sounds unaffected)

6.1.2 Member Function Documentation

6.1.2.1 apply() [1/3]

```
void AS_EFFECT::apply (
            float * input,
            float * output,
            int samples,
             SACBitstream::ChannelType::channeltype channel )
```

6.1.2.2 apply() [2/3]

```
void AS_EFFECT::apply (
            float * input,
            float * output,
            int samples,
             SACBitstream::ChannelType::channeltype channel )
```

6.1.2.3 apply() [3/3]

```
void AS_EFFECT::apply (
            float * input,
            float * output,
            int samples,
             SACBitstream::ChannelType::channeltype channel )
```

6.1.2.4 combfilter()

```
void AS_EFFECT::combfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float * a,
            float * b,
            int order,
            float a_delay,
            float b_delay,
            int delay )
```

6.1.2.5 Compressor()

```
AS_EFFECT::Compressor ( )
```

6.1.2.6 Equalizer()

```
AS_EFFECT::Equalizer ( )
```

6.1.2.7 feedforwardfilter()

```
void AS_EFFECT::feedforwardfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float gain,
            float original,
            int delay )
```

6.1.2.8 filter()

```
void AS_EFFECT::filter (
            float * x,
            float * y,
            int samples,
            float * a,
            float * b,
            int order )
```

6.1.2.9 gain()

```
double AS_EFFECT::gain (
            double inputlevel )  [private]
```

6.1.2.10 highShelfFilter()

```
void AS_EFFECT::highShelfFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            int order )
```

6.1.2.11 lowpassfeedbackfilter()

```
void AS_EFFECT::lowpassfeedbackfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float gain,
            float rs,
            float d,
            int delay )
```

6.1.2.12 lowShelfFilter()

```
void AS_EFFECT::lowShelfFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            int order )
```

6.1.2.13 peakingFilter()

```
void AS_EFFECT::peakingFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            double Q,
            int order )
```

6.1.2.14 plot() [1/3]

```
std::vector<std::vector<double> > AS_EFFECT::plot (
            std::string chart )
```

6.1.2.15 plot() [2/3]

```
std::vector<std::vector<double> > AS_EFFECT::plot (
            std::string chart )
```

6.1.2.16 plot() [3/3]

```
std::vector<std::vector<double> > AS_EFFECT::plot (
            std::string chart )
```

6.1.2.17 Reverb()

```
AS_EFFECT::Reverb ( )
```

6.1.2.18 schroederdiffusionfilter()

```
void AS_EFFECT::schroederdiffusionfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float gain,
            float g,
            int delay )
```

6.1.2.19 schroederfilter()

```
void AS_EFFECT::schroederfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float gain,
            float g,
            int delay )
```

6.1.2.20 update()

```
void AS_EFFECT::update ( )
```

6.1.3 Member Data Documentation

6.1.3.1 bands

```
const int AS_EFFECT::bands = 10  [static], [private]
```

number of equalizer bands

6.1.3.2 filterindex

```
int AS_EFFECT::filterindex  [private]
```

filter counter in each iteration

6.1.3.3 frequency

```
double AS_EFFECT::frequency[ bands]  [private]
```

Initial value:

```
= {31.25,
                                62.5,
                                125.0,
                                250.0,
                                500.0,
                                1000.0,
                                2000.0,
                                4000.0,
                                8000.0,
                                16000.0}
```

center frequencies [Hz]

6.1.3.4 function

```
Compressor::compressorfunction AS_EFFECT::function  [private]
```

compressor function

6.1.3.5 pointer

```
int AS_EFFECT::pointer[ maxfilters]  [private]
```

pointer to the last element in memory

6.1.3.6 ratio

```
double AS_EFFECT::ratio  [private]
```

compression ratio

6.1.3.7 threshold

```
double AS_EFFECT::threshold  [private]
```

input level threshold for compressing [dB]

6.1.3.8 type

```
Compressor::compressortype AS_EFFECT::type  [private]
```

compressor type

6.1.3.9 x

```
float AS_EFFECT::x  [private]
```

filter buffer of input samples

6.1.3.10 y

```
float AS_EFFECT::y  [private]
```

filter buffer of output samples

The documentation for this class was generated from the following files:

- src/effects/ **Compressor.h**
- src/effects/ **Equalizer.h**
- src/effects/ **Reverb.h**

6.2 AudioInfo Class Reference

Audio object info dialog class.

```
#include <AudioInfo.h>
```

Inheritance diagram for AudioInfo:

Public Member Functions

- **AudioInfo** (QWidget ∗parent=0)

  AudioInfo *(*p. **??***) constructor.*
- ∼**AudioInfo** ()

  AudioInfo *(*p. **??***) destructor.*
- void **setFile** (**WAVFile** ∗ **file**)

  It sets a audio file.

Private Attributes

- Ui::AudioInfo ∗ **ui**
- **WAVFile** ∗ **file**

6.2.1 Detailed Description

Audio object info dialog class.

Author

Andrés González Fornell

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AudioInfo()

```
AudioInfo::AudioInfo (
            QWidget * parent = 0 )
```

AudioInfo (p. **??**) constructor.

Parameters

parent	user interface parent object

6.2.2.2 ∼AudioInfo()

```
AudioInfo::~AudioInfo ( )
```

AudioInfo (p. **??**) destructor.

6.2.3 Member Function Documentation

6.2.3.1 setFile()

```
void AudioInfo::setFile (
              WAVFile * file )
```

It sets a audio file.

Parameters

file	audio file object

6.2.4 Member Data Documentation

6.2.4.1 file

```
WAVFile* AudioInfo::file  [private]
```

audio file object

6.2.4.2 ui

```
Ui::AudioInfo* AudioInfo::ui  [private]
```

user interface object

The documentation for this class was generated from the following files:

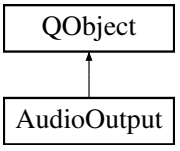
- src/interface/ **AudioInfo.h**
- src/interface/ **AudioInfo.cpp**

6.3 AudioOutput Class Reference

Audio output class.

```
#include <AudioOutput.h>
```

Inheritance diagram for AudioOutput:



Public Slots

User interface slots

They are called when a user interface element is being changed.

- void **setDevice** (int index)
  It selects an output device.

Public Member Functions

- **AudioOutput** (QComboBox * **selector**, int **fs**, int **samplesize**)
  AudioOuput constructor.
- ~**AudioOutput** ()
  AudioOutput *(p. **??**) destructor.*
- void **start** ()
  It resumes audio output playback.
- void **stop** ()
  It stops audio output playback.
- void **setFormat** (int **fs**, int **samplesize**)
  It sets signal sampling frequency.
- void **setDevices** ()
  It sets all available audio output devices.
- void **setDevices** (QList< QAudioDeviceInfo > **devices**)
  It sets a list of audio devices.
- void **setVolume** (float **volume**)
  It sets audio output volume level.

Public Attributes

- **OutputDevice** * **outputdevice**
- int **fs**
- int **samplesize**
- float **volume**

Private Member Functions

- void **initialize** ()
  It initializes output audio device.

Private Attributes

- QList< QAudioDeviceInfo > **devices**
- QAudioDeviceInfo **device**
- QAudioOutput * **audiooutput**
- QAudioFormat **format**
- QComboBox * **selector**

6.3.1 Detailed Description

Audio output class.

Author

Andrés González Fornell

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AudioOutput()

```
AudioOutput::AudioOutput (
            QComboBox * selector,
            int fs,
            int samplesize )
```

AudioOuput constructor.

Parameters

selector	user interface combo box to select audio device
fs	signal sampling frequency
samplesize	audio sample size [bits]

6.3.2.2 ~AudioOutput()

```
AudioOutput::~AudioOutput ( )
```

AudioOutput (p. **??**) destructor.

6.3.3 Member Function Documentation

6.3.3.1 initialize()

```
void AudioOutput::initialize ( )  [private]
```

It initializes output audio device.

6.3.3.2 setDevice

```
void AudioOutput::setDevice (
            int index )  [slot]
```

It selects an output device.

Parameters

index	device index

6.3.3.3 setDevices() [1/2]

```
void AudioOutput::setDevices ( )
```

It sets all available audio output devices.

6.3.3.4 setDevices() [2/2]

```
void AudioOutput::setDevices (
            QList< QAudioDeviceInfo > devices )
```

It sets a list of audio devices.

Parameters

devices	

6.3.3.5 setFormat()

```
void AudioOutput::setFormat (
            int fs,
            int samplesize )
```

It sets signal sampling frequency.

Parameters

fs	signal sampling frequency.
samplesize	signal sample size

6.3.3.6 setVolume()

```
void AudioOutput::setVolume (
            float volume )
```

It sets audio output volume level.

Parameters

volume	real number from 0 to 1

6.3.3.7 start()

```
void AudioOutput::start ( )
```

It resumes audio output playback.

6.3.3.8 stop()

```
void AudioOutput::stop ( )
```

It stops audio output playback.

6.3.4 Member Data Documentation

6.3.4.1 audiooutput

```
QAudioOutput* AudioOutput::audiooutput  [private]
```

audio output object

6.3.4.2 device

```
QAudioDeviceInfo AudioOutput::device  [private]
```

current system audio output device

6.3.4.3 devices

```
QList<QAudioDeviceInfo> AudioOutput::devices  [private]
```

list of system audio output devices

6.3.4.4 format

```
QAudioFormat AudioOutput::format  [private]
```

output audio format object

6.3.4.5 fs

```
int AudioOutput::fs
```

signal sampling frequency [Hz]

6.3.4.6 outputdevice

```
OutputDevice* AudioOutput::outputdevice
```

audio output QIODevice class object to control audio output device functions

6.3.4.7 samplesize

```
int AudioOutput::samplesize
```

audio sample size [bits]

6.3.4.8 selector

```
QComboBox* AudioOutput::selector  [private]
```

user interface combo box to select audio device

6.3.4.9 volume

```
float AudioOutput::volume
```

audio output volume

The documentation for this class was generated from the following files:

- src/interface/ **AudioOutput.h**
- src/interface/ **AudioOutput.cpp**

6.4 AudioSignal Class Reference

TODO **AudioSignal.cpp** (p. **??**) description.

```
#include <AudioSignal.h>
```

Public Member Functions

- **AudioSignal** (int **fs**)

  AudioSignal *(*p. **??***) constructor (empty signal vector).*
- **AudioSignal** (std::vector< float > **signal**, int **fs**)

  AudioSignal *(*p. **??***) constructor.*
- ~**AudioSignal** ()

  AudioSignal *(*p. **??***) destructor.*
- float **operator[ ]** (int index)

  It gets a sample from the selected index.
- **AudioSignal** **getSample** (int start, int end)

  It gets samples from a specific range.
- void **setSample** (int index, float sample)

  It sets a sample in the selected index.
- void **addSample** (float sample)

  It adds a sample to the end of the signal.
- void **deleteSample** (int index)

  It deletes a sample at a selected position.
- void **deleteSample** (int start, int end)

  It deletes a range of samples.
- std::vector< float > **getSignal** ()

  It gets the entire signal.
- void **setSignal** (std::vector< float > **signal**)

  It sets the entire signal.
- std::vector< float > **getTimes** ()

  It gets time [s] axis as a vector beggining at time t = 0 s.
- std::vector< float > **getTimes** (float initialtime)

  It gets time [s] axis as a vector beggining at a specific initial time.
- std::vector< float > **getSpectrum** ()

  It gets the signal spectral density.
- std::vector< float > **getSpectrum** (int bands)

  It gets the signal spectral density.
- std::vector< float > **getFrequencies** ()

  It gets frequencies [Hz] axis as a vector.
- std::vector< float > **getFrequencies** (int bands)

  It gets frequencies [Hz] axis as a vector.
- void **clear** ()

  It removes all samples from the signal.

Public Attributes

- int **size**
- int **fs**

Static Public Attributes

- static const unsigned int **maxsamples** = 0xFFFFFF

Private Attributes

- std::vector< float > **signal**

6.4.1 Detailed Description

TODO **AudioSignal.cpp** (p. **??**) description.

Author

Andrés González Fornell

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AudioSignal() [1/2]

```
AudioSignal::AudioSignal (
            int fs )
```

AudioSignal (p. **??**) constructor (empty signal vector).

Parameters

fs	signal sampling frequency [Hz]

6.4.2.2 AudioSignal() [2/2]

```
AudioSignal::AudioSignal (
            std::vector< float > signal,
            int fs )
```

AudioSignal (p. **??**) constructor.

Parameters

signal	vector of signal samples
fs	signal sampling frequency [Hz]

6.4.2.3 ~AudioSignal()

```
AudioSignal::~AudioSignal ( )
```

AudioSignal (p. **??**) destructor.

6.4.3 Member Function Documentation

6.4.3.1 addSample()

```
void AudioSignal::addSample (
            float sample )
```

It adds a sample to the end of the signal.

Parameters

sample	

6.4.3.2 clear()

```
void AudioSignal::clear ( )
```

It removes all samples from the signal.

6.4.3.3 deleteSample() [1/2]

```
void AudioSignal::deleteSample (
            int index )
```

It deletes a sample at a selected position.

Parameters

index	sample position index

6.4.3.4 deleteSample() [2/2]

```
void AudioSignal::deleteSample (
            int start,
            int end )
```

It deletes a range of samples.

Parameters

start	first index of the range (included)
end	last index of the range (included)

6.4.3.5 getFrequencies() [1/2]

```
std::vector< float > AudioSignal::getFrequencies ( )
```

It gets frequencies [Hz] axis as a vector.

Returns

frequencies vector

6.4.3.6 getFrequencies() [2/2]

```
std::vector< float > AudioSignal::getFrequencies (
            int bands )
```

It gets frequencies [Hz] axis as a vector.

Parameters

bands	number of frequency bands of the signal spectral density (if higher number than available has been requested, it returns as the highest number of frequency as possible)

Returns

frequencies vector

6.4.3.7 getSample()

```
AudioSignal AudioSignal::getSample (
            int start,
            int end )
```

It gets samples from a specific range.

Parameters

start	first index of the range (included)
end	last index of the range (included)

Returns

subsignal object

6.4.3.8 getSignal()

```
std::vector< float > AudioSignal::getSignal ( )
```

It gets the entire signal.

Returns

signal

6.4.3.9 getSpectrum() [1/2]

```
std::vector< float > AudioSignal::getSpectrum ( )
```

It gets the signal spectral density.

Returns

signal spectral density

6.4.3.10 getSpectrum() [2/2]

```
std::vector< float > AudioSignal::getSpectrum (
            int bands )
```

It gets the signal spectral density.

Parameters

bands	number of frequency bands of the signal spectral density (if higher number than available has been requested, it returns as the highest number of frequency as possible)

Returns

signal spectral density

6.4.3.11 getTimes() [1/2]

```
std::vector< float > AudioSignal::getTimes ( )
```

It gets time [s] axis as a vector beggining at time t = 0 s.

Returns

time vector

6.4.3.12 getTimes() [2/2]

```
std::vector< float > AudioSignal::getTimes (
            float initialtime )
```

It gets time [s] axis as a vector beggining at a specific initial time.

Parameters

initialtime	initial time [s]

Returns

time vector

6.4.3.13 operator[]()

```
float AudioSignal::operator[] (
            int index )
```

It gets a sample from the selected index.

Parameters

index	

Returns

sample

6.4.3.14 setSample()

```
void AudioSignal::setSample (
            int index,
            float sample )
```

It sets a sample in the selected index.

Parameters

index	
sample	

6.4.3.15 setSignal()

```
void AudioSignal::setSignal (
            std::vector< float > signal )
```

It sets the entire signal.

Parameters

signal	

6.4.4 Member Data Documentation

6.4.4.1 fs

```
int AudioSignal::fs
```

signal sampling frequency [Hz]

6.4.4.2 maxsamples

```
const unsigned int AudioSignal::maxsamples = 0xFFFFFF  [static]
```

maximum number of samples

6.4.4.3 signal

```
std::vector<float> AudioSignal::signal  [private]
```

signal data vector

6.4.4.4 size

```
int AudioSignal::size
```

number of samples

The documentation for this class was generated from the following files:

- src/process/ **AudioSignal.h**
- src/process/ **AudioSignal.cpp**

6.5 AudioStream Class Reference

Audio objects for the SAOC interface.

```
#include <AudioObject.h>
```

Classes

- struct **SignalRange**

  index range.
- struct **TimeSlot**

  It indicates time slot of the available signal.

Public Member Functions

- **AudioStream** ()
- **AudioStream** (int **fs**)

  AudioStream *(*p. **??***) constructor.*
- ∼**AudioStream** ()

  AudioStream *(*p. **??***) destructor.*
- void **push** (float sample)

  It adds a new sample to the end of the stream.
- float **pop** ()

  It gets the first sample and deletes it from the stream.
- std::vector< float > **getSamples** ()
- void **setSample** (int time, float sample)

  It sets a sample to a selected index (it replaces the previous sample at the same selected index).
- float **getSample** (int time)
- bool **isAvailable** (int time)

  It checks if the selected index is available in the stream.
- **AudioStream** (int **fs**)
- ∼**AudioStream** ()
- int **getfs** ()

  It gets the signal sampling frequency.
- void **setfs** (int value)

  It sets the signal sampling frequency.
- void **push** (float sample)

- void **push** (**AudioSignal** sample)

  It adds several new samples to the end of the stream.
- float **pop** ()
- **AudioSignal pop** (int n)

  It gets the first n samples and deletes them from the stream.
- float **operator[ ]** (int index)

  It gets a sample from a selected index (the sample is not deleted).
- void **setSample** (int index, float sample)
- int **size** ()

  It gets the number of samples.
- bool **isAvailable** (int index)

Public Attributes

- **TimeSlot timestamp**
- int **fs**
- **SignalRange range**
- **AudioSignal signal**

Private Attributes

- std::vector< float > **samples**

6.5.1 Detailed Description

Audio objects for the SAOC interface.

Class for audio streams. This is the interface between the user interface and the coding and audio processing.

Author

Andrés González Fornell

6.5.2 Constructor & Destructor Documentation

6.5.2.1 AudioStream() [1/3]

```
AudioStream::AudioStream ( )
```

6.5.2.2 AudioStream() [2/3]

```
AudioStream::AudioStream (
            int fs )
```

AudioStream (p. **??**) constructor.

Parameters

fs	signal sampling frequency [Hz]

6.5.2.3 ~AudioStream() [1/2]

```
AudioStream::~AudioStream ( )
```

AudioStream (p. **??**) destructor.

6.5.2.4 AudioStream() [3/3]

```
AudioStream::AudioStream (
            int fs )
```

6.5.2.5 ~AudioStream() [2/2]

```
AudioStream::~AudioStream ( )
```

6.5.3 Member Function Documentation

6.5.3.1 getfs()

```
int AudioStream::getfs ( )
```

It gets the signal sampling frequency.

Returns

signal sampling frequency

6.5.3.2 getSample()

```
float AudioStream::getSample (
            int time )
```

6.5.3.3 getSamples()

```
std::vector<float> AudioStream::getSamples ( )
```

6.5.3.4 isAvailable() [1/2]

```
bool AudioStream::isAvailable (
            int index )
```

It checks if the selected index is available in the stream.

Parameters

index	

Returns

true if it is available

6.5.3.5 isAvailable() [2/2]

```
bool AudioStream::isAvailable (
            int index )
```

6.5.3.6 operator[]()

```
float AudioStream::operator[] (
            int index )
```

It gets a sample from a selected index (the sample is not deleted).

Parameters

index	

Returns

sample

6.5.3.7 pop() [1/3]

```
float AudioStream::pop ( )
```

It gets the first sample and deletes it from the stream.

Returns

first sample

6.5.3.8 pop() [2/3]

```
float AudioStream::pop ( )
```

6.5.3.9 pop() [3/3]

```
AudioSignal AudioStream::pop (
            int n )
```

It gets the first n samples and deletes them from the stream.

Parameters

n	number of samples to pop from the stream

Returns

first n samples

6.5.3.10 push() [1/3]

```
void AudioStream::push (
            float sample )
```

It adds a new sample to the end of the stream.

Parameters

sample	

6.5.3.11 push() [2/3]

```
void AudioStream::push (
            float sample )
```

6.5.3.12 push() [3/3]

```
void AudioStream::push (
             AudioSignal samples )
```

It adds several new samples to the end of the stream.

Parameters

samples	

6.5.3.13 setfs()

```
void AudioStream::setfs (
            int fs )
```

It sets the signal sampling frequency.

Parameters

fs	signal sampling frequency

6.5.3.14 setSample() [1/2]

```
void AudioStream::setSample (
            int index,
            float sample )
```

It sets a sample to a selected index (it replaces the previous sample at the same selected index).

Parameters

index	
sample	

6.5.3.15 setSample() [2/2]

```
void AudioStream::setSample (
            int index,
            float sample )
```

6.5.3.16 size()

```
int AudioStream::size ( )
```

It gets the number of samples.

Returns

number of samples of audio stream

6.5.4 Member Data Documentation

6.5.4.1 fs

```
int AudioStream::fs
```

signal sampling frequency [Hz]

audio sampling frequency [Hz]

6.5.4.2 range

```
SignalRange AudioStream::range
```

index range of the audio stream

6.5.4.3 samples

```
std::vector<float> AudioStream::samples  [private]
```

available data from the audio object

6.5.4.4 signal

```
AudioSignal AudioStream::signal
```

signal object

6.5.4.5 timestamp

```
TimeSlot AudioStream::timestamp
```

corresponding timestamp for the available data

The documentation for this class was generated from the following files:

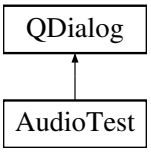
- src/interface/ **AudioObject.h**
- src/process/ **AudioStream.h**
- src/process/ **AudioStream.cpp**

6.6 AudioTest Class Reference

Audio output test class.

`#include <AudioOutput.h>`

Inheritance diagram for AudioTest:



Public Member Functions

- **AudioTest** (QWidget *parent=0)
  AudioTest *(*p. **??***) constructor.*
- ~**AudioTest** ()
  AudioTest *(*p. **??***) destructor.*

Private Slots

- void **play** ()
  It plays audio test.

User interface slots

They are called when a user interface element is being changed.

- void **setDevice** (int index)
  It sets a device.
- void **setSampleFormat** (int index)
  It sets a sample format.
- void **setFS** (int index)
  It sets a sampling frequency.
- void **setFrequency** (double frequency)
  It sets a tone frequency.
- void **setAmplitude** (int amplitude)
  It sets an amplitude.

Private Member Functions

- int **getSampleSize** ()

  It gets the sample size from the selected sample format.
- int **getFS** ()

  It gets the selected sampling frequency.
- void **start** ()

  It starts playing the test.

Private Attributes

- Ui::AudioTest ∗ **ui**
- **AudioOutput** ∗ **audiooutput**
- QTimer ∗ **clock**
- const double **period** = 1

6.6.1 Detailed Description

Audio output test class.

Author

Andrés González Fornell

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AudioTest()

```
AudioTest::AudioTest (
            QWidget * parent = 0 )
```

AudioTest (p. **??**) constructor.

Parameters

parent	window parent

6.6.2.2 ∼AudioTest()

```
AudioTest::~AudioTest ( )
```

AudioTest (p. **??**) destructor.

6.6.3 Member Function Documentation

6.6.3.1 getFS()

```
int AudioTest::getFS ( )  [private]
```

It gets the selected sampling frequency.

Returns

sampling frequency [Hz]

6.6.3.2 getSampleSize()

```
int AudioTest::getSampleSize ( )  [private]
```

It gets the sample size from the selected sample format.

Returns

sample size [bits]

6.6.3.3 play

```
void AudioTest::play ( )  [private], [slot]
```

It plays audio test.

6.6.3.4 setAmplitude

```
void AudioTest::setAmplitude (
            int amplitude )  [private], [slot]
```

It sets an amplitude.

Parameters

amplitude	amplitude value

6.6.3.5 setDevice

```
void AudioTest::setDevice (
            int index )  [private], [slot]
```

It sets a device.

Parameters

index	device index from the combo box

6.6.3.6 setFrequency

```
void AudioTest::setFrequency (
            double frequency )  [private], [slot]
```

It sets a tone frequency.

Parameters

frequency	frequency value

6.6.3.7 setFS

```
void AudioTest::setFS (
            int index )  [private], [slot]
```

It sets a sampling frequency.

Parameters

index	sampling frequency index

6.6.3.8 setSampleFormat

```
void AudioTest::setSampleFormat (
            int index )  [private], [slot]
```

It sets a sample format.

Parameters

index	sample format index from the combo box

6.6.3.9 start()

```
void AudioTest::start ( )  [private]
```

It starts playing the test.

6.6.4 Member Data Documentation

6.6.4.1 audiooutput

```
AudioOutput* AudioTest::audiooutput  [private]
```

audio output object

6.6.4.2 clock

```
QTimer* AudioTest::clock  [private]
```

test clock to send test signal periodically

6.6.4.3 period

```
const double AudioTest::period = 1  [private]
```

clock period to send new data [s]

6.6.4.4 ui

```
Ui::AudioTest* AudioTest::ui  [private]
```

user interface object

The documentation for this class was generated from the following files:

- src/interface/ **AudioOutput.h**
- src/interface/ **AudioOutput.cpp**

6.7 BinauralQuality Struct Reference

SAC decoder parameter binaural quality.

`#include <SACEffects.h>`

Public Types

- enum **binauralquality** { **parametric** = 0, **filtering** = 1 }

6.7.1 Detailed Description

SAC decoder parameter binaural quality.

6.7.2 Member Enumeration Documentation

6.7.2.1 binauralquality

`enum` **BinauralQuality::binauralquality**

Enumerator

parametric	
filtering	

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

- src/interface/ **SACEffects.h**

6.8 Channel Class Reference

Single-object class from channels list.

`#include <ChannelsList.h>`

Public Member Functions

- **Channel** (QLayout *framework, std::string **prefix**, int **index**, bool **isoutput**)
  Channels constructor.
- ~**Channel** ()
  Channels desctructor.

- int **getIndex** ()
  It gets the channel index.
- void **setIndex** (int **index**)
- void **setLabel** (std::string **label**)
  It sets a label to the channel name, i.e., group box title and label text.
- void **setVolume** (int **volume**)
  It sets the channel volume level.
- void **mute** (bool state)
  It mutes channel.
- void **bypass** (bool state)
  It sets channel to bypass effects.

Public Attributes

- int **index**
- std::string **name**
- double **volume**
- bool **muted**
- bool **bypassed**
- **AudioOutput** * **audiooutput**

- QGroupBox * **groupbox**
  user interface elements
- QLineEdit * **label**
- QSlider * **volumeslider**
- QCheckBox * **mutecheckbox**
- QCheckBox * **bypasscheckbox**
- QComboBox * **deviceselector**

Private Attributes

- std::string **prefix**
- bool **isoutput**

6.8.1 Detailed Description

Single-object class from channels list.

Author

Andrés González Fornell

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Channel()

```
Channel::Channel (
            QLayout * framework,
            std::string prefix,
            int index,
            bool isoutput )
```

Channels constructor.

Parameters

framework	channel user interface framework
prefix	prefix of objects name of channel user interface
index	channel index
isoutput	true to create device selector to send audio to the system audio output devices

6.8.2.2 ~Channel()

```
Channel::~Channel ( )
```

Channels desctructor.

6.8.3 Member Function Documentation

6.8.3.1 bypass()

```
void Channel::bypass (
            bool state )
```

It sets channel to bypass effects.

Parameters

state	true to bypass effects and false to apply them

6.8.3.2 getIndex()

```
int Channel::getIndex ( )
```

It gets the channel index.

Returns

index

6.8.3.3 mute()

```
void Channel::mute (
            bool state )
```

It mutes channel.

Parameters

state	true to mute the channel and false to unmuted

6.8.3.4 setIndex()

```
void Channel::setIndex (
            int index )
```

6.8.3.5 setLabel()

```
void Channel::setLabel (
            std::string label )
```

It sets a label to the channel name, i.e., group box title and label text.

Parameters

label	

6.8.3.6 setVolume()

```
void Channel::setVolume (
            int volume )
```

It sets the channel volume level.

Parameters

volume	integer number from 0 to 100

6.8.4 Member Data Documentation

6.8.4.1 audiooutput

```
AudioOutput* Channel::audiooutput
```

system audio output devices object

6.8.4.2 bypasscheckbox

```
QCheckBox* Channel::bypasscheckbox
```

checkbox object to bypass effect

6.8.4.3 bypassed

```
bool Channel::bypassed
```

it tells channel to bypass effects or apply them

6.8.4.4 deviceselector

```
QComboBox* Channel::deviceselector
```

audio output device selector object

6.8.4.5 groupbox

```
QGroupBox* Channel::groupbox
```

user interface elements

channel group box

6.8.4.6 index

```
int Channel::index
```

channel index

6.8.4.7 isoutput

```
bool Channel::isoutput  [private]
```

true to show audio output device selector (in case of sending channel to speakers or other audio output system device)

6.8.4.8 label

```
QLineEdit* Channel::label
```

field to change the channel label

6.8.4.9 mutecheckbox

```
QCheckBox* Channel::mutecheckbox
```

muted checkbox object

6.8.4.10 muted

```
bool Channel::muted
```

it indicates if channel is muted

6.8.4.11 name

```
std::string Channel::name
```

channel name

6.8.4.12 prefix

```
std::string Channel::prefix  [private]
```

user interface prefix

6.8.4.13 volume

```
double Channel::volume
```

current audio volume level

6.8.4.14 volumeslider

```
QSlider* Channel::volumeslider
```

volume level slider

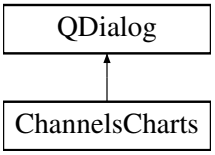
The documentation for this class was generated from the following files:

- src/interface/ **ChannelsList.h**
- src/interface/ **ChannelsList.cpp**

6.9 ChannelsCharts Class Reference

`#include <ChannelsList.h>`

Inheritance diagram for ChannelsCharts:



Public Member Functions

- **ChannelsCharts** (float ** **input**, float ** **output**, **ChannelsList** * **input_channels**, **ChannelsList** * **output_channels**, int **samples**, QWidget *parent=0)

  ChannelsCharts *(*p. **??***) constructor.*
- ~**ChannelsCharts** ()

  ChannelsCharts *(*p. **??***) destructor.*

Private Slots

- void **setTimeCursor** (int sample)

  It sets the start time cursor.
- int **getTimeCursor** ()

  It gets the start time cursor.
- void **setScope** (int time)

  It sets the charts scope.
- int **getScope** ()

  It gets the charts scope.
- void **plot** ()

  It plots the current input and ouput signals.

Private Member Functions

- void **updateSelectors** ()

  It loads current channels names on chart selectors.

Private Attributes

- Ui::ChannelsCharts * **ui**
- float ** **input**
- float ** **output**
- **ChannelsList** * **input_channels**
- **ChannelsList** * **output_channels**
- int **samples**
- **Chart2D** * **input_chart**
- **Chart2D** * **output_chart**

6.9.1 Constructor & Destructor Documentation

6.9.1.1 ChannelsCharts()

```
ChannelsCharts::ChannelsCharts (
            float ** input,
            float ** output,
             ChannelsList * input_channels,
             ChannelsList * output_channels,
            int samples,
            QWidget * parent = 0 )
```

ChannelsCharts (p. **??**) constructor.

Parameters

input	input signal pointer
output	output signal pointer
input_channels	input channels object
output_channels	output channels object
samples	number of samples each channel
parent	user inteface parent object

6.9.1.2 ~ChannelsCharts()

```
ChannelsCharts::~ChannelsCharts ( )
```

ChannelsCharts (p. **??**) destructor.

6.9.2 Member Function Documentation

6.9.2.1 getScope

```
int ChannelsCharts::getScope ( )  [private], [slot]
```

It gets the charts scope.

Returns

number of sample of charts scope

6.9.2.2 getTimeCursor

```
int ChannelsCharts::getTimeCursor ( )  [private], [slot]
```

It gets the start time cursor.

Returns

first sample of the chart

6.9.2.3 plot

```
void ChannelsCharts::plot ( )  [private], [slot]
```

It plots the current input and ouput signals.

6.9.2.4 setScope

```
void ChannelsCharts::setScope (
            int time )  [private], [slot]
```

It sets the charts scope.

Parameters

time	charts scope [ms]

6.9.2.5 setTimeCursor

```
void ChannelsCharts::setTimeCursor (
            int sample )  [private], [slot]
```

It sets the start time cursor.

Parameters

sample	first sample of the chart

6.9.2.6 updateSelectors()

```
void ChannelsCharts::updateSelectors ( )  [private]
```

It loads current channels names on chart selectors.

6.9.3 Member Data Documentation

6.9.3.1 input

```
float** ChannelsCharts::input  [private]
```

input signal pointer

6.9.3.2 input_channels

```
ChannelsList* ChannelsCharts::input_channels  [private]
```

input channels object

6.9.3.3 input_chart

```
Chart2D* ChannelsCharts::input_chart  [private]
```

input chart object

6.9.3.4 output

```
float** ChannelsCharts::output  [private]
```

output signal pointer

6.9.3.5 output_channels

```
ChannelsList* ChannelsCharts::output_channels  [private]
```

output channels object

6.9.3.6 output_chart

```
Chart2D* ChannelsCharts::output_chart  [private]
```

output chart object

6.9.3.7 samples

```
int ChannelsCharts::samples  [private]
```

number of samples each channel

6.9.3.8 ui

```
Ui::ChannelsCharts* ChannelsCharts::ui  [private]
```

user interface object

The documentation for this class was generated from the following files:

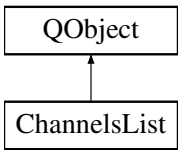
- src/interface/ **ChannelsList.h**
- src/interface/ **ChannelsList.cpp**

6.10 ChannelsList Class Reference

Channels list class. It shows information about channels signals.

```
#include <ChannelsList.h>
```

Inheritance diagram for ChannelsList:



Signals

- void **namechanged** (QString, int)

Public Member Functions

- **ChannelsList** (QWidget * **framework**, int number, bool **showdevices**)
  ChannelsList *(*p. **??***) constructor.*
- ~**ChannelsList** ()
  ChannelsList *(*p. **??***) destructor.*
- **Channel** * **getChannel** (int index)
  It gets a channel.
- void **deleteChannel** (int index)
  It deletes a channel.
- int **getSize** ()
  It gets the number of channels.
- void **setSize** (int size)
  It sets a number of channels up.
- std::vector< std::string > **getNames** ()
  It gets all channels names.

Static Public Attributes

- static int **fs**
- static int **samplesize**

Private Slots

Channels interface slots

User interface control functions of channels list.

- void **setLabel** (QString label)
  Slot for setting the channel label.
- void **setLabel** (QString label, int index)
  Slot for setting the channel label.
- void **setVolume** (int volume)
  Slots for setting the channel level.
- void **mute** (bool state)
  Slots when muted checkbox has been changed.
- void **bypass** (bool state)
  Slots when muted checkbox has been changed.
- void **setDevice** (int device)
  Slots for setting an audio output device.

Private Member Functions

- int **getIndex** (QObject ∗element)
  It gets the channel index of a user interface element.

Private Attributes

- std::vector< **Channel** ∗ > **channels**
- std::string **prefix**
- bool **showdevices**
- QWidget ∗ **framework**
- QLayout ∗ **layout**

6.10.1 Detailed Description

Channels list class. It shows information about channels signals.

Author

Andrés González Fornell

6.10.2 Constructor & Destructor Documentation

6.10.2.1 ChannelsList()

```
ChannelsList::ChannelsList (
            QWidget * framework,
            int number,
            bool showdevices )
```

ChannelsList (p. **??**) constructor.

Parameters

framework	user interface framework of channels list
number	number of channels
showdevices	true to create device selector to send audio to the system audio output devices

6.10.2.2 ~ChannelsList()

```
ChannelsList::~ChannelsList ( )
```

ChannelsList (p. **??**) destructor.

6.10.3 Member Function Documentation

6.10.3.1 bypass

```
void ChannelsList::bypass (
            bool state )  [private], [slot]
```

Slots when muted checkbox has been changed.

Parameters

state	current checkbox state

6.10.3.2 deleteChannel()

```
void ChannelsList::deleteChannel (
            int index )
```

It deletes a channel.

Parameters

index	channel index

6.10.3.3 getChannel()

```
Channel * ChannelsList::getChannel (
            int index )
```

It gets a channel.

Parameters

index	channel index

Returns

channel pointer

6.10.3.4 getIndex()

```
int ChannelsList::getIndex (
            QObject * element )  [private]
```

It gets the channel index of a user interface element.

Parameters

element	user interface element

Returns

index

6.10.3.5 getNames()

```
std::vector< std::string > ChannelsList::getNames ( )
```

It gets all channels names.

Returns

list of channels names

6.10.3.6 getSize()

```
int ChannelsList::getSize ( )
```

It gets the number of channels.

Returns

number of channels

6.10.3.7 mute

```
void ChannelsList::mute (
            bool state )  [private], [slot]
```

Slots when muted checkbox has been changed.

Parameters

state	current checkbox state

6.10.3.8 namechanged

```
void ChannelsList::namechanged (
            QString ,
            int  )  [signal]
```

6.10.3.9 setDevice

```
void ChannelsList::setDevice (
            int device )  [private], [slot]
```

Slots for setting an audio output device.

Parameters

device	device index

6.10.3.10 setLabel [1/2]

```
void ChannelsList::setLabel (
            QString label )  [private], [slot]
```

Slot for setting the channel label.

Parameters

label	

6.10.3.11 setLabel [2/2]

```
void ChannelsList::setLabel (
            QString label,
            int index )  [private], [slot]
```

Slot for setting the channel label.

Parameters

label	
index	channel index

6.10.3.12 setSize()

```
void ChannelsList::setSize (
            int size )
```

It sets a number of channels up.

Parameters

size	number of channels

6.10.3.13 setVolume

```
void ChannelsList::setVolume (
            int volume )  [private], [slot]
```

Slots for setting the channel level.

Parameters

volume	integer number from 0 to 100

6.10.4 Member Data Documentation

6.10.4.1 channels

```
std::vector< Channel *> ChannelsList::channels  [private]
```

list of channels

6.10.4.2 framework

```
QWidget* ChannelsList::framework  [private]
```

user interface framework of channels list

6.10.4.3 fs

```
int ChannelsList::fs  [static]
```

signal sampling frequency

6.10.4.4 layout

```
QLayout* ChannelsList::layout  [private]
```

user interface layout of channels list

6.10.4.5 prefix

```
std::string ChannelsList::prefix  [private]
```

user interface prefix

6.10.4.6 samplesize

```
int ChannelsList::samplesize  [static]
```

signal sample size

6.10.4.7 showdevices

```
bool ChannelsList::showdevices  [private]
```

true to show audio output device selector (in case of sending channels to speakers or other audio output system devices)

The documentation for this class was generated from the following files:

- src/interface/ **ChannelsList.h**
- src/interface/ **ChannelsList.cpp**
- src/interface/ **SACEffects.cpp**

6.11 SACBitstream::ChannelType Struct Reference

It specifies the channel type.

```
#include <SACBitstream.h>
```

Public Types

- enum **channeltype** {
  L = 0x0, **Lc** = 0x1, **Ls** = 0x2, **Lsr** = 0x3,
  R = 0x4, **Rc** = 0x5, **Rs** = 0x6, **Rsr** = 0x7,
  C = 0x8, **LFE** = 0x9 }

6.11.1 Detailed Description

It specifies the channel type.

6.11.2 Member Enumeration Documentation

6.11.2.1 channeltype

```
enum SACBitstream::ChannelType::channeltype
```

Enumerator

L	left front channel
Lc	left front center channel
Ls	left surround channel
Lsr	rear surround left channel
R	left front channel
Rc	left front center channel
Rs	left surround channel
Rsr	rear surround left channel
C	center front channel
LFE	low frequency enhancement channel

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

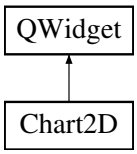
- src/sac/ **SACBitstream.h**

6.12 Chart2D Class Reference

Class for plotting two-dimensional charts.

`#include <Chart2D.h>`

Inheritance diagram for Chart2D:



Classes

- struct **ChartOptions**

  It defines some features of the chart.

Public Member Functions

- **Chart2D** (QWidget ∗framework)

  Chart constructor.
- **Chart2D** (QWidget ∗framework, double range[2][2], std::string title, std::string **xlabel**, std::string **ylabel**, int options)

  Chart constructor.
- ∼**Chart2D** ()

  Chart destructor.
- void **setPoints** (QVector< QPointF > points)

  It sets the points to the chart serie.
- QVector< QPointF > **getPoints** ()

  It gets the points from the chart serie.
- void **setRange** (double range[2][2])

  It sets the axis range.
- void **setTitle** (std::string title)

  It sets chart title.
- void **setOptions** (int options)

  It sets chart options.
- void **clear** ()

  It clears the chart.

Public Attributes

- std::string **xlabel**
- std::string **ylabel**

Private Attributes

- QChart * **chart**
- QChartView * **view**
- QLineSeries * **series**

6.12.1 Detailed Description

Class for plotting two-dimensional charts.

Author

Andrés González Fornell

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Chart2D() [1/2]

```
Chart2D::Chart2D (
            QWidget * framework )
```

Chart constructor.

Parameters

framework	user interface framework of chart

6.12.2.2 Chart2D() [2/2]

```
Chart2D::Chart2D (
            QWidget * framework,
            double range[2][2],
            std::string title,
            std::string xlabel,
            std::string ylabel,
            int options )
```

Chart constructor.

Parameters

framework	user interface framework of chart
range	axes range matrix (range[0][0] = x_min, range[0][1] = x_max, range[1][0] = y_min, range[1][1] = y_max)
title	chart title (it will be impress on the chart)

Parameters

xlabel	label for horizontal (x) axis
ylabel	label for vertical (y) axis
options	**ChartOptions** (p. **??**)

6.12.2.3 ~Chart2D()

```
Chart2D::~Chart2D ( )
```

Chart destructor.

6.12.3 Member Function Documentation

6.12.3.1 clear()

```
void Chart2D::clear ( )
```

It clears the chart.

6.12.3.2 getPoints()

```
QVector< QPointF > Chart2D::getPoints ( )
```

It gets the points from the chart serie.

Returns

points

6.12.3.3 setOptions()

```
void Chart2D::setOptions (
            int options )
```

It sets chart options.

Parameters

options	

6.12.3.4 setPoints()

```
void Chart2D::setPoints (
            QVector< QPointF > points )
```

It sets the points to the chart serie.

Parameters

points	

6.12.3.5 setRange()

```
void Chart2D::setRange (
            double range[2][2] )
```

It sets the axis range.

Parameters

range	axis range matrix (range[0][0] = x_min, range[0][1] = x_max, range[1][0] = y_min, range[1][1] = y_max)

6.12.3.6 setTitle()

```
void Chart2D::setTitle (
            std::string title )
```

It sets chart title.

Parameters

title	

6.12.4 Member Data Documentation

6.12.4.1 chart

```
QChart* Chart2D::chart  [private]
```

chart object

6.12.4.2 series

```
QLineSeries* Chart2D::series  [private]
```

series object from chart

6.12.4.3 view

```
QChartView* Chart2D::view  [private]
```

chart view object from chart

6.12.4.4 xlabel

```
std::string Chart2D::xlabel
```

horizontal (x) axis label

6.12.4.5 ylabel

```
std::string Chart2D::ylabel
```

vertical (y) axis label

The documentation for this class was generated from the following files:

- src/interface/ **Chart2D.h**
- src/interface/ **Chart2D.cpp**

6.13 Chart2D::ChartOptions Struct Reference

It defines some features of the chart.

```
#include <Chart2D.h>
```

Public Types

- enum **Options** {
  logX = 0x00001, **logY** = 0x00010, **labelX** = 0x00100, **labelY** = 0x01000,
  legend = 0x10000 }

6.13.1 Detailed Description

It defines some features of the chart.

6.13.2 Member Enumeration Documentation

6.13.2.1 Options

enum **Chart2D::ChartOptions::Options**

Enumerator

logX	it configures the x axis as logarithm scale
logY	it configures the y axis as logarithm scale
labelX	it shows the x axis description on the chart
labelY	it shows the y axis description on the chart
legend	it shows the legend on the chart

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

- src/interface/ **Chart2D.h**

6.14 Compressor Class Reference

Audio compressor.

`#include <Compressor.h>`

6.14.1 Detailed Description

Audio compressor.

Author

Andrés González Fornell

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

- src/effects/ **Compressor.h**

6.15 DecodingType Struct Reference

SAC decoder parameter decoding type.

`#include <SACEffects.h>`

Public Types

- enum **decodingtype** { **low** = 0, **high** = 1 }

6.15.1 Detailed Description

SAC decoder parameter decoding type.

6.15.2 Member Enumeration Documentation

6.15.2.1 decodingtype

`enum` **DecodingType::decodingtype**

Enumerator

low	
high	

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

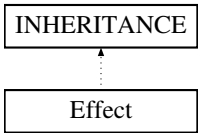
- src/interface/ **SACEffects.h**

6.16 Effect Class Reference

Effect (p. **??**) class. It contains (by inheritance) all effects classes.

`#include <Effect.h>`

Inheritance diagram for Effect:



Public Types

- enum **effectID** { **LIST** }

Public Member Functions

- **Effect** (**Effect::effectID** **effect**, int **fs**)

  Effect *(*p. **??***) constructor.*
- **Effect** (**Effect::effectID** **effect**, std::map< std::string, std::string > params, int **fs**)

  Effect *(*p. **??***) constructor.*
- ~**Effect** ()

  Effect *(*p. **??***) destructor.*
- void **setParams** (std::map< std::string, std::string > params)

  It sets params.
- bool **apply** (float *input, float *output, int samples, **SACBitstream::ChannelType::channeltype** channel)

  It applies the selected effect to the input and sets the result into output variable.
- std::vector< std::vector< double > > **plot** (std::string chart)

  It sends some values to user interface charts.

Static Public Member Functions

- static std::map< **Effect::effectID**, std::string > **getEffects** ()

  It gets the list of available effects.
- static **Effect::effectID** **getEffect** (std::string effectname)

  It get effects type from the effect name.
- static std::map< std::string, std::string > **getParams** (std::string configuration)

  It get params from a effect configuration file (.fx) text.
- static std::vector< bool > **getChannels** (std::string configuration, int size)

  It get channels vector from a effect configuration file (.fx) text.
- static std::vector< double > **getLevels** (std::string configuration, int size)

  It get levels vector from a effect configuration file (.fx) text.
- static std::string **getTag** (std::string configuration, std::string tag)

  It extracts the value in a tag from a effect configuration file (.fx) text.
- static std::map< std::string, std::string > **getTagMap** (std::string configuration, std::string tag)

  It extracts the map of values in a map-structured tag from a effect configuration file (.fx) text.

Public Attributes

- std::pair< **Effect::effectID**, std::string > **effect**

6.16.1 Detailed Description

Effect (p. **??**) class. It contains (by inheritance) all effects classes.

Author

Andrés González Fornell

6.16.2 Member Enumeration Documentation

6.16.2.1 effectID

`enum` **`Effect::effectID`**

Enumerator

LIST	

6.16.3 Constructor & Destructor Documentation

6.16.3.1 Effect() [1/2]

```
Effect::Effect (
             Effect::effectID effect,
            int fs )
```

Effect (p. **??**) constructor.

Parameters

effect	effect ID
fs	signal sampling frequency

6.16.3.2 Effect() [2/2]

```
Effect::Effect (
             Effect::effectID effect,
            std::map< std::string, std::string > params,
            int fs )
```

Effect (p. **??**) constructor.

Parameters

effect	effect ID
params	map of effect parameters
fs	signal sampling frequency

6.16.3.3 ∼Effect()

```
Effect::∼Effect ( )
```

Effect (p. **??**) destructor.

6.16.4 Member Function Documentation

6.16.4.1 apply()

```
bool Effect::apply (
            float * input,
            float * output,
            int samples,
             SACBitstream::ChannelType::channeltype channel )
```

It applies the selected effect to the input and sets the result into output variable.

Parameters

input	input data pointer
output	output data pointer
samples	number of samples
channel	type of channel

Returns

true if it was successful

6.16.4.2 getChannels()

```
std::vector< bool > Effect::getChannels (
            std::string configuration,
            int size )  [static]
```

It get channels vector from a effect configuration file (.fx) text.

Parameters

configuration	contained text of a effect configuration file (.fx)
size	number of channels

Returns

channels boolean vector to select channels when applying effects

6.16.4.3 getEffect()

```
Effect::effectID Effect::getEffect (
            std::string effectname )  [static]
```

It get effects type from the effect name.

Parameters

effectname	effect name string

Returns

effect type **effectID** (p. **??**)

6.16.4.4 getEffects()

```
std::map< Effect::effectID, std::string > Effect::getEffects ( ) [static]
```

It gets the list of available effects.

Returns

map of available effects

6.16.4.5 getLevels()

```
std::vector< double > Effect::getLevels (
            std::string configuration,
            int size ) [static]
```

It get levels vector from a effect configuration file (.fx) text.

Parameters

configuration	contained text of a effect configuration file (.fx)
size	number of channels

Returns

levels vector of input channels before applying effects

6.16.4.6 getParams()

```
std::map< std::string, std::string > Effect::getParams (
            std::string configuration ) [static]
```

It get params from a effect configuration file (.fx) text.

Parameters

configuration	contained text of a effect configuration file (.fx)

Returns

parameters map variable valid to apply effects

6.16.4.7 getTag()

```
std::string Effect::getTag (
            std::string configuration,
            std::string tag )  [static]
```

It extracts the value in a tag from a effect configuration file (.fx) text.

Parameters

configuration	contained text of a effect configuration file (.fx)
tag	tag name of the requested field

Returns

contained value in the tag

6.16.4.8 getTagMap()

```
std::map< std::string, std::string > Effect::getTagMap (
            std::string configuration,
            std::string tag )  [static]
```

It extracts the map of values in a map-structured tag from a effect configuration file (.fx) text.

Parameters

configuration	contained text of a effect configuration file (.fx)
tag	tag name of the requested field

Returns

contained map of values in the tag

6.16.4.9 plot()

```
std::vector< std::vector< double > > Effect::plot (
            std::string chart )
```

It sends some values to user interface charts.

Parameters

chart	chart id

Returns

array of values as values[axis][sample] axis: 0 = x (horizontal) and 1 = y (vertical)

6.16.4.10 setParams()

```
void Effect::setParams (
            std::map< std::string, std::string > params )
```

It sets params.

Parameters

params	

6.16.5 Member Data Documentation

6.16.5.1 effect

```
std::pair< Effect::effectID, std::string> Effect::effect
```

selected effect name and id

The documentation for this class was generated from the following files:

- src/effects/ **Effect.h**
- src/effects/ **Effect.cpp**

6.17 EffectBase Class Reference

Effect (p. **??**) base class.

```
#include <EffectBase.h>
```

Public Member Functions

- **EffectBase** ()

  EffectBase *(*p. **??***) constructor.*

Static Public Member Functions

- static int **getInt** (std::string param)

  It parse a parameter value to double.
- static double **getDouble** (std::string param)

  It parse a parameter value to integer.
- static std::string **getString** (std::string param)

  It parse a parameter value to string.
- static bool **getBool** (std::string param)

  It parse a parameter value to bool.

Static Public Attributes

- static int **fs**
- static std::map< std::string, std::string > **params**

6.17.1 Detailed Description

Effect (p. **??**) base class.

Author

Andrés González Fornell

6.17.2 Constructor & Destructor Documentation

6.17.2.1 EffectBase()

```
EffectBase::EffectBase ( )
```

EffectBase (p. **??**) constructor.

6.17.3 Member Function Documentation

6.17.3.1 getBool()

```
bool EffectBase::getBool (
            std::string param )  [static]
```

It parse a parameter value to bool.

Parameters

param	parameter value

Returns

boolean value (false by default)

6.17.3.2 getDouble()

```
double EffectBase::getDouble (
            std::string param )  [static]
```

It parse a parameter value to integer.

Parameters

param	parameter value

Returns

value

6.17.3.3 getInt()

```
int EffectBase::getInt (
            std::string param )  [static]
```

It parse a parameter value to double.

Parameters

param	parameter value

Returns

value as integer

6.17.3.4 getString()

```
std::string EffectBase::getString (
            std::string param )  [static]
```

It parse a parameter value to string.

Parameters

param	parameter value

Returns

value

6.17.4 Member Data Documentation

6.17.4.1 fs

```
int EffectBase::fs  [static]
```

signal sampling frequency [Hz]

6.17.4.2 params

```
std::map< std::string, std::string > EffectBase::params  [static]
```

string of effect parameters

The documentation for this class was generated from the following files:

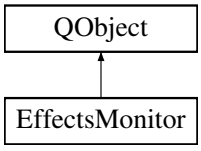
- src/effects/ **EffectBase.h**
- src/effects/ **Effect.cpp**

6.18 EffectsMonitor Class Reference

Class for managing effects parameters.

```
#include <EffectsMonitor.h>
```

Inheritance diagram for EffectsMonitor:



Public Slots

Parameters slots

User interface functions for effect parameters control.

- void **updateParameter** (int value)

  Slot for updating parameters parameters of type int when one of them is changed.
- void **updateParameter** (double value)

  Slot for updating parameters parameters of type double when one of them is changed.
- void **updateParameter** (QString value)

  Slot for updating parameters of type string when one of them is changed.
- void **updateParameter** (bool value)

  Slot for updating parameters parameters of type bool and enum when one of them is changed.

Public Member Functions

- **EffectsMonitor** (QWidget ∗ **framework**)

  EffectsMonitor *(*p. **??***) constructor.*
- **EffectsMonitor** (QWidget ∗ **framework**, **Effect** ∗ **effect**)

  EffectsMonitor *(*p. **??***) constructor.*
- ∼**EffectsMonitor** ()

  EffectsMonitor *(*p. **??***) destructor.*
- void **setEffect** (**Effect** ∗ **effect**)

  It selects an effect.
- void **clear** ()

  It clears the user interface framework.
- void **setParameter** (std::string key, std::string value)

  It sets a parameter from the parameter user interface object.
- void **plotChart** ()

  It plots every chart on the effects monitor.

Public Attributes

- **Effect** ∗ **effect**
- std::map< **Effect::effectID**, std::string > **effects**
- std::map< **Effect::effectID**, std::string > **files**
- std::map< std::string, std::string > **parameters**
- std::map< std::string, **Chart2D** ∗ > **charts**

Private Member Functions

- void **loadFiles** ()

  It loads all imported effects template files from the effects folder as application resource in order to be used by the running program.
- void **loadTemplate** ()

  It loads a user interface template from a xml file and loads it as an effect monitor interface.
- std::string **getAttribute** (std::string element, std::string attribute)

  It gets an attribute from a xml line (correponded to a xml element).
- void **loadField** (std::string element)

  It loads a field from a xml line (corresponded to an element).

Private Attributes

- QWidget * **framework**
- QFormLayout * **layout**

6.18.1 Detailed Description

Class for managing effects parameters.

Author

Andrés González Fornell

6.18.2 Constructor & Destructor Documentation

6.18.2.1 EffectsMonitor() [1/2]

```
EffectsMonitor::EffectsMonitor (
            QWidget * framework )
```

EffectsMonitor (p. **??**) constructor.

Parameters

framework	user interface framework

6.18.2.2 EffectsMonitor() [2/2]

```
EffectsMonitor::EffectsMonitor (
            QWidget * framework,
             Effect * effect )
```

EffectsMonitor (p. **??**) constructor.

Parameters

framework	user interface framework
effect	selected effect to be load

6.18.2.3 ~EffectsMonitor()

```
EffectsMonitor::~EffectsMonitor ( )
```

EffectsMonitor (p. **??**) destructor.

6.18.3 Member Function Documentation

6.18.3.1 clear()

```
void EffectsMonitor::clear ( )
```

It clears the user interface framework.

6.18.3.2 getAttribute()

```
std::string EffectsMonitor::getAttribute (
            std::string element,
            std::string attribute )  [private]
```

It gets an attribute from a xml line (correponded to a xml element).

Parameters

element	string of the xml line (full element)
attribute	name of the attribute

Returns

value of the attribute

6.18.3.3 loadField()

```
void EffectsMonitor::loadField (
            std::string element )  [private]
```

It loads a field from a xml line (corresponded to an element).

Parameters

element	string of the xml line (full element)

6.18.3.4 loadFiles()

```
void EffectsMonitor::loadFiles ( )  [private]
```

It loads all imported effects template files from the effects folder as application resource in order to be used by the running program.

6.18.3.5 loadTemplate()

```
void EffectsMonitor::loadTemplate ( )  [private]
```

It loads a user interface template from a xml file and loads it as an effect monitor interface.

6.18.3.6 plotChart()

```
void EffectsMonitor::plotChart ( )
```

It plots every chart on the effects monitor.

6.18.3.7 setEffect()

```
void EffectsMonitor::setEffect (
              Effect * effect )
```

It selects an effect.

Parameters

effect	selected effect

6.18.3.8 setParameter()

```
void EffectsMonitor::setParameter (
             std::string parameter,
             std::string value )
```

It sets a parameter from the parameter user interface object.

Parameters

parameter	parameter name
value	new parameter value

6.18.3.9 updateParameter [1/4]

```
void EffectsMonitor::updateParameter (
            int value )  [slot]
```

Slot for updating parameters parameters of type int when one of them is changed.

Parameters

value	changed value

6.18.3.10 updateParameter [2/4]

```
void EffectsMonitor::updateParameter (
            double value )  [slot]
```

Slot for updating parameters parameters of type double when one of them is changed.

Parameters

value	changed value

6.18.3.11 updateParameter [3/4]

```
void EffectsMonitor::updateParameter (
            QString value )  [slot]
```

Slot for updating parameters of type string when one of them is changed.

Parameters

value	changed value

6.18.3.12 updateParameter [4/4]

```
void EffectsMonitor::updateParameter (
            bool value )  [slot]
```

Slot for updating parameters parameters of type bool and enum when one of them is changed.

Parameters

value	changed value

6.18.4 Member Data Documentation

6.18.4.1 charts

```
std::map<std::string,  Chart2D *> EffectsMonitor::charts
```

list of charts of effect monitoring

6.18.4.2 effect

```
 Effect* EffectsMonitor::effect
```

pointer to current selected effect

6.18.4.3 effects

```
std::map< Effect::effectID, std::string> EffectsMonitor::effects
```

list of all available effects

6.18.4.4 files

```
std::map< Effect::effectID, std::string> EffectsMonitor::files
```

list of all available effects template files

6.18.4.5 framework

```
QWidget* EffectsMonitor::framework  [private]
```

effects monitor framework

6.18.4.6 layout

```
QFormLayout* EffectsMonitor::layout  [private]
```

form layout of effect parameters

6.18.4.7 parameters

```
std::map<std::string, std::string> EffectsMonitor::parameters
```

list of the current effect parameters and their values

The documentation for this class was generated from the following files:

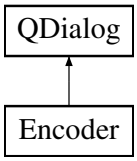
- src/interface/ **EffectsMonitor.h**
- src/interface/ **EffectsMonitor.cpp**

6.19 Encoder Class Reference

Encoder (p. **??**) window interface.

```
#include <Encoder.h>
```

Inheritance diagram for Encoder:



Public Member Functions

- **Encoder** (QWidget *parent=0)
  Encoder *(p. **??**) constructor.*
- ~**Encoder** ()
  Encoder *(p. **??**) destructor.*
- void **setInput** (std::string filename)
  It sets the input audio file.
- void **setOutput** (std::string filename)
  It sets the output audio file.
- void **setTree** (int **tree**)
  It sets a tree configuration.

Public Attributes

- int **fs**
- **WAVFile** * **input**
- **WAVFile** * **output**
- **File** * **bitstream**

Private Slots

Coder interface slots

User interface control functions of coder.

- void **setBitstream** ()
  It sets the bitstream file name.
- void **setBuried** (bool state)
  It sets the buried parameter.
- void **toggleTree** ()
  It performs change action when tree configuration is changed.
- void **load** ()
  It loads an input audio file.
- void **reset** ()
  It resets all encoding parameters, including output file.
- void **apply** ()
  It applies the codification.
- void **cancel** ()
  It closes the coder dialog without no consequences.
- void **submit** ()
  It loads the output to decoder and closes coder dialog.

Private Attributes

- Ui::Encoder * **ui**
- int **tree**

6.19.1 Detailed Description

Encoder (p. **??**) window interface.

Author

Andrés González Fornell

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Encoder()

```
Encoder::Encoder (
            QWidget * parent = 0 )
```

Encoder (p. **??**) constructor.

6.19.2.2 ~Encoder()

```
Encoder::~Encoder ( )
```

Encoder (p. **??**) destructor.

6.19.3 Member Function Documentation

6.19.3.1 apply

```
void Encoder::apply ( )  [private], [slot]
```

It applies the codification.

6.19.3.2 cancel

```
void Encoder::cancel ( )  [private], [slot]
```

It closes the coder dialog without no consequences.

6.19.3.3 load

```
void Encoder::load ( )  [private], [slot]
```

It loads an input audio file.

6.19.3.4 reset

```
void Encoder::reset ( )  [private], [slot]
```

It resets all encoding parameters, including output file.

6.19.3.5 setBitstream

```
void Encoder::setBitstream ( )  [private], [slot]
```

It sets the bitstream file name.

6.19.3.6 setBuried

```
void Encoder::setBuried (
            bool state )  [private], [slot]
```

It sets the buried parameter.

Parameters

state	true if it is buried

6.19.3.7 setInput()

```
void Encoder::setInput (
            std::string filename )
```

It sets the input audio file.

Parameters

filename	file path

6.19.3.8 setOutput()

```
void Encoder::setOutput (
            std::string filename )
```

It sets the output audio file.

Parameters

filename	file path

6.19.3.9 setTree()

```
void Encoder::setTree (
            int tree )
```

It sets a tree configuration.

Parameters

tree	configuration index

6.19.3.10 submit

```
void Encoder::submit ( )  [private], [slot]
```

It loads the output to decoder and closes coder dialog.

6.19.3.11 toggleTree

```
void Encoder::toggleTree ( )  [private], [slot]
```

It performs change action when tree configuration is changed.

6.19.4 Member Data Documentation

6.19.4.1 bitstream

```
File* Encoder::bitstream
```

output bit stream file object

6.19.4.2 fs

```
int Encoder::fs
```

signal sampling frequency [Hz]

6.19.4.3 input

```
WAVFile* Encoder::input
```

input file object

6.19.4.4 output

```
WAVFile* Encoder::output
```

output file object

6.19.4.5 tree

```
int Encoder::tree  [private]
```

SAC encoder parameter

6.19.4.6 ui

```
Ui::Encoder* Encoder::ui  [private]
```

user interface object

The documentation for this class was generated from the following files:

- src/interface/ **Encoder.h**
- src/interface/ **Encoder.cpp**

6.20 File::Endianess Struct Reference

```
#include <File.h>
```

Public Types

- enum **endianess** { **littleendian**, **bigendian** }

6.20.1 Member Enumeration Documentation

6.20.1.1 endianess

```
enum File::Endianess::endianess
```

Enumerator

littleendian	little endian
bigendian	big endian

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

- src/process/ **File.h**

6.21 Equalizer Class Reference

Audio compressor.

```
#include <Equalizer.h>
```

6.21.1 Detailed Description

Audio compressor.

Author

Andrés González Fornell

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

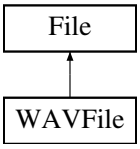
- src/effects/ **Equalizer.h**

6.22 File Class Reference

Audio file class.

`#include <File.h>`

Inheritance diagram for File:



Classes

- struct **Endianess**

Endianess

Endianess (p. **??**) type.

- std::fstream * **file**
- std::string **filename**
- bool **writepermission**
- int **cursor**
- **File** (bool **writepermission**)

  File *(p.* **??***) constructor.*
- **File** (std::string **filename**, bool **writepermission**)

  File *(p.* **??***) constructor.*
- ~**File** ()

  File *(p.* **??***) destructor.*
- void **setFilename** (std::string **filename**)

  It sets the file path name.
- std::string **getFilename** ()

It gets the file path name.

- void **setCursor** (int **cursor**)

  It sets the file reading cursor to keep on reading from another position.
- int **getCursor** ()

  It gets the current file reading cursor.
- int **size** ()

  It gets the total file size.
- bool **exists** ()

  It indicates if the file object exists.
- char * **read** (int length)

  It reads data from the file.
- void **write** (const char *data, int length)

  It writes data on the file.
- std::string **readText** (int length)

  It reads text data from the file.
- void **writeText** (std::string data)

  It writes text data on the file.
- unsigned **readNumber** (int length, **Endianess::endianess** endianess)

  It reads a data number from the file.
- void **writeNumber** (unsigned int data, int length, **Endianess::endianess** endianess)

  It writes a data number on the file.

6.22.1 Detailed Description

Audio file class.

Author

Andrés González Fornell

6.22.2 Constructor & Destructor Documentation

6.22.2.1 File() [1/2]

```
File::File (
            bool writepermission )
```

File (p. **??**) constructor.

6.22.2.2 File() [2/2]

```
File::File (
            std::string filename,
            bool writepermission )
```

File (p. **??**) constructor.

Parameters

filename	file path
writepermission	file write permission (true if it is allowed)

6.22.2.3 ~File()

```
File::~File ( )
```

File (p. **??**) destructor.

6.22.3 Member Function Documentation

6.22.3.1 exists()

```
bool File::exists ( )
```

It indicates if the file object exists.

Returns

true if the file object exists

6.22.3.2 getCursor()

```
int File::getCursor ( )
```

It gets the current file reading cursor.

Returns

cursor [Bytes] from the beginning of the file

6.22.3.3 getFilename()

```
std::string File::getFilename ( )
```

It gets the file path name.

Returns

file path name

6.22.3.4 read()

```
char * File::read (
            int length )
```

It reads data from the file.

Parameters

length	data length [Bytes]

Returns

data pointer

6.22.3.5 readNumber()

```
unsigned int File::readNumber (
            int length,
             Endianess::endianess endianess )
```

It reads a data number from the file.

Parameters

length	data length [Bytes]
endianess	data order (big endian or little endian)

Returns

value of data number

6.22.3.6 readText()

```
std::string File::readText (
            int length )
```

It reads text data from the file.

Parameters

length	data length [Bytes] (if length = 0 function returns all available data from the file)

Returns

string of data

6.22.3.7 setCursor()

```
void File::setCursor (
            int cursor )
```

It sets the file reading cursor to keep on reading from another position.

Parameters

cursor	new cursor position [Bytes] from the beginning of the file

6.22.3.8 setFilename()

```
void File::setFilename (
            std::string filename )
```

It sets the file path name.

Parameters

filename	file path name

6.22.3.9 size()

```
int File::size ( )
```

It gets the total file size.

Returns

file size [Bytes]

6.22.3.10 write()

```
void File::write (
            const char * data,
            int length )
```

It writes data on the file.

Parameters

data	data pointer
length	data length [Bytes]

6.22.3.11 writeNumber()

```
void File::writeNumber (
            unsigned int value,
            int length,
             Endianess::endianess endianess )
```

It writes a data number on the file.

Parameters

value	value of data number
length	data length [Bytes]
endianess	data order (big endian or little endian)

6.22.3.12 writeText()

```
void File::writeText (
            std::string data )
```

It writes text data on the file.

Parameters

data	string of data

6.22.4 Member Data Documentation

6.22.4.1 cursor

```
int File::cursor  [private]
```

file reading cursor [Bytes]

6.22.4.2 file

```
std::fstream* File::file  [private]
```

file object

6.22.4.3 filename

```
std::string File::filename  [private]
```

file path name

6.22.4.4 writepermission

```
bool File::writepermission  [private]
```

write file permission (true if it is allowed)

The documentation for this class was generated from the following files:

- src/process/ **File.h**
- src/process/ **File.cpp**

6.23 WAVFile::Header Struct Reference

Audio file header struct.

```
#include <File.h>
```

Public Member Functions

- int **size** ()

Public Attributes

Chunk header

It indicates the audio format (wave).

- std::string **chunkID**
- unsigned int **chunksize**
- std::string **format**

Subshunk 1 header

It describes the format of the sound information in the data sub-chunk.

- std::string **subchunk1ID**
- unsigned int **subchunk1size**
- unsigned int **audioformat**
- unsigned int **numchannels**
- unsigned int **samplerate**
- unsigned int **byterate**
- unsigned int **blockalign**
- unsigned int **bitspersample**

Subshunk 2 header

It indicates the size of the sound information.

- std::string **subchunk2ID**
- unsigned int **subchunk2size**

6.23.1 Detailed Description

Audio file header struct.

6.23.2 Member Function Documentation

6.23.2.1 size()

```
int WAVFile::Header::size ( )  [inline]
```

6.23.3 Member Data Documentation

6.23.3.1 audioformat

```
unsigned int WAVFile::Header::audioformat
```

PCM = 1 (linear quantization) values others than 1 indicate some form of compression

6.23.3.2 bitspersample

```
unsigned int WAVFile::Header::bitspersample
```

number of bits per sample

6.23.3.3 blockalign

```
unsigned int WAVFile::Header::blockalign
```

number of bytes for one sample including all channels (= numchannels $*$ bitspersample/8)

6.23.3.4 byterate

```
unsigned int WAVFile::Header::byterate
```

byte rate (= samplerate $*$ numchannels $*$ bitspersample/8)

6.23.3.5 chunkID

```
std::string WAVFile::Header::chunkID
```

it contains the letters "RIFF" in ASCII form

6.23.3.6 chunksize

```
unsigned int WAVFile::Header::chunksize
```

size of the entire file in bytes minus 8 bytes for the two fields not included in this count (ChunkID and ChunkSize)

6.23.3.7 format

```
std::string WAVFile::Header::format
```

it contains the letters "WAVE"

6.23.3.8 numchannels

```
unsigned int WAVFile::Header::numchannels
```

number of channels

6.23.3.9 samplerate

```
unsigned int WAVFile::Header::samplerate
```

sample rate

6.23.3.10 subchunk1ID

```
std::string WAVFile::Header::subchunk1ID
```

it contains the letters "fmt "

6.23.3.11 subchunk1size

```
unsigned int WAVFile::Header::subchunk1size
```

size of the rest of the subchunk (16 for PCM)

6.23.3.12 subchunk2ID

```
std::string WAVFile::Header::subchunk2ID
```

it contains the letters "data

6.23.3.13 subchunk2size

```
unsigned int WAVFile::Header::subchunk2size
```

size of ther rest of the subchunk (it is the size of the data)

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

- src/process/ **File.h**

6.24 HRTFModel Struct Reference

SAC decoder parameter HRTF model.

```
#include <SACEffects.h>
```

Public Types

- enum **hrtfmodel** { **kemar** = 0, **vast** = 1, **mps_vt** = 2 }

6.24.1 Detailed Description

SAC decoder parameter HRTF model.

6.24.2 Member Enumeration Documentation

6.24.2.1 hrtfmodel

```
enum HRTFModel::hrtfmodel
```

Enumerator

kemar	
vast	
mps_vt	

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

- src/interface/ **SACEffects.h**

6.25 LogType Struct Reference

`#include <Logger.h>`

Public Types

- enum **logtype** {
  info, **warning**, **error**, **progress**,
  interaction }

6.25.1 Member Enumeration Documentation

6.25.1.1 logtype

`enum` **`LogType::logtype`**

Enumerator

info	The message is not important, just some information for the user
warning	The message is a warning
error	The message comes from an bad execution (do not confuse with execution or compilation errors)
progress	Information about the current steps in the running execution
interaction	Information about an user interaction

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

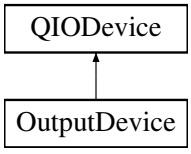
- src/tools/ **Logger.h**

6.26 OutputDevice Class Reference

Audio output device class (QIODevice extension).

`#include <AudioOutput.h>`

Inheritance diagram for OutputDevice:



Public Member Functions

- **OutputDevice** (QAudioFormat **format**)

  OutputDevice *(*p. **??***) constructor.*
- ∼**OutputDevice** ()

  OutputDevice *(*p. **??***) destructor.*
- void **send** (float ∗signal, int samples)

  It sends an audio signal to the buffer to be sent to the audio output device.
- qint64 **readData** (char ∗data, qint64 length)

  It gets data from the audio output device.
- qint64 **writeData** (const char ∗data, qint64 length)

  It gets written data from the audio input device (not used).
- qint64 **bytesAvailable** () const

  It gets available bytes to be read by the audio output device.
- void **test** (double **amplitude**, double frequency, float duration)

  It plays an audio test by generating a tone.
- void **clear** ()

  It clears output buffer.

Public Attributes

- char ∗ **buffer**
- int **cursor_read**
- int **cursor_write**
- int **buffersize**

Private Attributes

- QAudioFormat **format**
- long int **amplitude**
- double **offset**

6.26.1 Detailed Description

Audio output device class (QIODevice extension).

Author

Andrés González Fornell

6.26.2 Constructor & Destructor Documentation

6.26.2.1 OutputDevice()

```
OutputDevice::OutputDevice (
            QAudioFormat format )
```

OutputDevice (p. **??**) constructor.

Parameters

format	audio format object

6.26.2.2 ~OutputDevice()

```
OutputDevice::~OutputDevice ( )
```

OutputDevice (p. **??**) destructor.

6.26.3 Member Function Documentation

6.26.3.1 bytesAvailable()

```
qint64 OutputDevice::bytesAvailable ( ) const
```

It gets available bytes to be read by the audio output device.

Returns

6.26.3.2 clear()

```
void OutputDevice::clear ( )
```

It clears output buffer.

6.26.3.3 readData()

```
qint64 OutputDevice::readData (
            char * data,
            qint64 length )
```

It gets data from the audio output device.

Parameters

data	data pointer
length	data length

Returns

6.26.3.4 send()

```
void OutputDevice::send (
            float * signal,
            int samples )
```

It sends an audio signal to the buffer to be sent to the audio output device.

Parameters

signal	audio signal pointer
samples	number of samples

6.26.3.5 test()

```
void OutputDevice::test (
            double amplitude,
            double frequency,
            float duration )
```

It plays an audio test by generating a tone.

Parameters

amplitude	tone amplitude (from 0 to 1)
frequency	tone frequency [Hz]
duration	test duration [s]

6.26.3.6 writeData()

```
qint64 OutputDevice::writeData (
            const char * data,
            qint64 length )
```

It gets written data from the audio input device (not used).

Parameters

data	data pointer
length	data length

Returns

6.26.4 Member Data Documentation

6.26.4.1 amplitude

```
long int OutputDevice::amplitude  [private]
```

audio signal max amplitude

6.26.4.2 buffer

```
char* OutputDevice::buffer
```

audio output data buffer

6.26.4.3 buffersize

```
int OutputDevice::buffersize
```

total size of buffer [Bytes]

6.26.4.4 cursor_read

```
int OutputDevice::cursor_read
```

cursor of read audio output data in buffer

6.26.4.5 cursor_write

```
int OutputDevice::cursor_write
```

cursor of pendient audio output data in buffer

6.26.4.6 format

```
QAudioFormat OutputDevice::format  [private]
```

audio format object

6.26.4.7 offset

```
double OutputDevice::offset  [private]
```

audio signal offset (used only for unsigned format)

The documentation for this class was generated from the following files:

- src/interface/ **AudioOutput.h**
- src/interface/ **AudioOutput.cpp**

6.27 ProcessManager Class Reference

Process manager class. It contains all functions to perform the signal treatment process.

```
#include <ProcessManager.h>
```

Public Member Functions

- **ProcessManager** (int **chunksize**)

  ProcessManager *(*p. **??***) constructor.*
- ∼**ProcessManager** ()

  ProcessManager *(*p. **??***) destructor.*
- bool **setInput** (std::string filename)

  It sets input variable from the existing input file.
- bool **setOutput** (std::string filename)

  It sets an output file from the existing output variable.
- bool **decode** (std::string **input**, std::string **bitstream**, std::string **output**, int upmixtype, int decodingtype, int binauralquality, int hrtfmodel)

  It performs the SAC encoder.
- bool **applyEffect** (**Effect** ∗effect, std::vector< bool > **channels**, std::vector< double > levels)

  It applys the selected effect to the input stream.
- void **clear** ()

  It clears all variables and resets the process.

Public Attributes

- int **fs**
- float ∗∗ **input**
- float ∗∗ **output**
- int **channels**
- int **samples**
- int **cursor**
- int **total**

Private Attributes

- **SACBitstream** ∗ **bitstream**
- **WAVFile** ∗ **inputfile**
- **WAVFile** ∗ **outputfile**
- bool **allocated**
- int **chunksize**

6.27.1 Detailed Description

Process manager class. It contains all functions to perform the signal treatment process.

Author

Andrés González Fornell

6.27.2 Constructor & Destructor Documentation

6.27.2.1 ProcessManager()

```
ProcessManager::ProcessManager (
            int chunksize )
```

ProcessManager (p. **??**) constructor.

Parameters

chunksize	number of samples in a chunk to apply effect step by step (if 0 then chunk size is the number of samples and effect is applied at once)

6.27.2.2 ∼ProcessManager()

```
ProcessManager::∼ProcessManager ( )
```

ProcessManager (p. **??**) destructor.

6.27.3 Member Function Documentation

6.27.3.1 applyEffect()

```
bool ProcessManager::applyEffect (
              Effect * effect,
             std::vector< bool > channels,
             std::vector< double > levels )
```

It applys the selected effect to the input stream.

Parameters

effect	effect object (it includes all parameters)
channels	boolean vector where true means to apply effect to that channel
levels	vector of input levels (>=0) for each channel

Returns

true if it was successful

6.27.3.2 clear()

```
void ProcessManager::clear ( )
```

It clears all variables and resets the process.

6.27.3.3 decode()

```
bool ProcessManager::decode (
             std::string input,
             std::string bitstream,
             std::string output,
             int decodingtype,
             int upmixtype,
             int binauralquality,
             int hrtfmodel )
```

It performs the SAC encoder.

Parameters

input	filename of the multichannel input audio file
output	filename of the downmix output audio file (it will be automatically created)
bitstream	filename of the bitstream output file or "buried" (it will be automatically created)
upmixtype	upmix type 0: normal 1: blind 2: binaural 3: stereo
decodingtype	decoding type 0: low 1: high
binauralquality	binaural upmix quality 0: parametric 1: filtering
hrtfmodel	HRTF model 0: kemar 1: vast 2: mps_vt

Returns

true if it was successful

6.27.3.4 setInput()

```
bool ProcessManager::setInput (
            std::string filename )
```

It sets input variable from the existing input file.

Parameters

filename	audio input file name

Returns

true if it was successful

6.27.3.5 setOutput()

```
bool ProcessManager::setOutput (
            std::string filename )
```

It sets an output file from the existing output variable.

Parameters

filename	audio output file name

Returns

true if it was successful

6.27.4 Member Data Documentation

6.27.4.1 allocated

```
bool ProcessManager::allocated  [private]
```

true if input and output signals variables are currently allocated

6.27.4.2 bitstream

```
SACBitstream* ProcessManager::bitstream  [private]
```

bitstream object

6.27.4.3 channels

```
int ProcessManager::channels
```

number of channels

6.27.4.4 chunksize

```
int ProcessManager::chunksize  [private]
```

number of samples in a chunk

6.27.4.5 cursor

```
int ProcessManager::cursor
```

pointer to current sample index when executing real time process

6.27.4.6 fs

```
int ProcessManager::fs
```

signal sampling frequency

6.27.4.7 input

```
float** ProcessManager::input
```

vector of input channels stream (sample = input[channel][sample index])

6.27.4.8 inputfile

```
WAVFile* ProcessManager::inputfile  [private]
```

audio input file object

6.27.4.9 output

```
float** ProcessManager::output
```

vector of input channels stream (sample = output[channel][sample index])

6.27.4.10 outputfile

```
WAVFile* ProcessManager::outputfile  [private]
```

audio output file object

6.27.4.11 samples

```
int ProcessManager::samples
```

number of samples in each channel

6.27.4.12 total

```
int ProcessManager::total
```

number of available output samples

The documentation for this class was generated from the following files:

- src/process/ **ProcessManager.h**
- src/process/ **ProcessManager.cpp**

6.28 Reverb Class Reference

Audio reverb effect.

```
#include <Reverb.h>
```

6.28.1 Detailed Description

Audio reverb effect.

Author

Andrés González Fornell

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

- src/effects/ **Reverb.h**

6.29 SACBitstream Class Reference

SAC bitstream class.

```
#include <SACBitstream.h>
```

Classes

- struct **ChannelType**
  It specifies the channel type.

Public Member Functions

- **SACBitstream** (std::string filename)
  Bitstream constructor.
- ~**SACBitstream** ()
  Bitstream destructor.
- long **getVariable** (int position, int length)
  It gets the value of a bitstream variable.
- void **load** ()
  It loads variables from bitstream file.

Public Attributes

- int **fs**
- **ChannelType::channeltype** * **channel**
- double **gain_surround**
- double **gain_LFE**
- double **gain_downmix**

Private Attributes

- **File** * **bitstream**

6.29.1 Detailed Description

SAC bitstream class.

Author

Andrés González Fornell

6.29.2 Constructor & Destructor Documentation

6.29.2.1 SACBitstream()

```
SACBitstream::SACBitstream (
            std::string filename )
```

Bitstream constructor.

6.29.2.2 ~SACBitstream()

```
SACBitstream::~SACBitstream ( )
```

Bitstream destructor.

6.29.3 Member Function Documentation

6.29.3.1 getVariable()

```
long int SACBitstream::getVariable (
            int position,
            int length )
```

It gets the value of a bitstream variable.

Parameters

position	position in bits
length	number of bits

Returns

value of the bitstream variable

6.29.3.2 load()

```
void SACBitstream::load ( )
```

It loads variables from bitstream file.

6.29.4 Member Data Documentation

6.29.4.1 bitstream

```
File* SACBitstream::bitstream  [private]
```

6.29.4.2 channel

```
ChannelType::channeltype* SACBitstream::channel
```

channels order

6.29.4.3 fs

```
int SACBitstream::fs
```

signal sampling frequencye

6.29.4.4 gain_downmix

```
double SACBitstream::gain_downmix
```

gain of downmix

6.29.4.5 gain_LFE

```
double SACBitstream::gain_LFE
```

downmix of LFE channels

6.29.4.6 gain_surround

```
double SACBitstream::gain_surround
```

downmix of surround channels

The documentation for this class was generated from the following files:

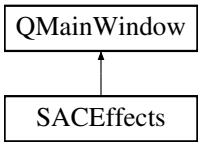
- src/sac/ **SACBitstream.h**
- src/sac/ **SACBitstream.cpp**

6.30 SACEffects Class Reference

SACEffects (p. **??**) window interface.

```
#include <SACEffects.h>
```

Inheritance diagram for SACEffects:



Public Member Functions

- **SACEffects** (QWidget *framework=0)

  SACEffects *(*p. **??***) constructor.*
- ~**SACEffects** ()

  SACEffects *(*p. **??***) destructor.*
- void **play** ()

  It starts playing input.
- void **pause** ()

  It pauses input playback.
- void **reset** ()

  It resets all decoding parameters, including input file.
- void **updateControls** ()

  It updates enability of user interface controls according to the current parameters state.
- void **setEffect** (**Effect::effectID effect**)

  It sets an effect for the effect monitor.
- void **setSource** (std::string filename)

  It sets the source audio file.
- void **setBitstream** (std::string filename)

  It sets the bitstream audio file.
- void **setInput** (std::string filename)

  It sets the input audio file.
- void **setFormat** (int **fs**, int samplesize)

  It sets audio output format.
- void **setDuration** (QLabel *label, double duration)

  It sets a duration indicator text on an user interface label object.
- void **getDuration** (QLabel label)
- void **setUpmixType** (**UpmixType::upmixtype upmixtype**)

  It sets SAC parameter upmix type.
- void **setDecodingType** (**DecodingType::decodingtype decodingtype**)

  It sets SAC parameter decoding type.
- void **setBinauralQuality** (**BinauralQuality::binauralquality binauralquality**)

  It sets SAC parameter binaural quality.
- void **setHRTFModel** (**HRTFModel::hrtfmodel hrtfmodel**)

  It sets SAC parameter HRTF model.

Public Attributes

- const int **fs** = 44100

Private Slots

- void **stop** ()

  It stops input playback.
- void **applyEffect** ()

  It applies the selected effect to the input signal.
- void **sendOutput** ()

  It sends the current output signal to the output device and plots on the output chart.

Audio menu bar interface slots

User interface functions for audio control.

- void **load** ()
  It loads a file as decoder input.
- void **exportOutput** ()
  It exports generated output singal as audio file.
- void **openChannelsCharts** ()
  It opens audio file info dialog.
- void **test** ()
  It plays a test to a selected output audio device.
- void **encode** ()
  It encodes a new audio through Coder.
- void **decode** ()
  It decodes the audio input.

SACEffects menu bar interface slots

User interface control functions for SAC decoder parameters.

- void **setBuried** (bool state)
  It sets bitstream as buried or not.
- void **toggleUpmixType** (QAction ∗item)
  Slot action for menu upmix type items.
- void **toggleDecodingType** (QAction ∗item)
  Slot action for menu decoding type items.
- void **toggleBinauralQuality** (QAction ∗item)
  Slot action for menu binaural quality items.
- void **toggleHRTFModel** (QAction ∗item)
  Slot action for menu HRTF model items.
- void **toggleEffect** ()
  Slot action for menu effect items.

Private Attributes

- **ProcessManager** ∗ **process**
- QTimer ∗ **clock**
- Ui::SACEffects ∗ **ui**
- QStatusBar ∗ **status**
- **WAVFile** ∗ **source**
- **File** ∗ **bitstream**
- **WAVFile** ∗ **input**
- **Effect** ∗ **effect**
- **ChannelsList** ∗ **channels_input**
- **ChannelsList** ∗ **channels_output**
- **EffectsMonitor** ∗ **effectsmonitor**
- int **chunksize**
- bool **muted**
- bool **buried**
- **DecodingType::decodingtype** **decodingtype**
- **UpmixType::upmixtype** **upmixtype**
- **BinauralQuality::binauralquality** **binauralquality**
- **HRTFModel::hrtfmodel** **hrtfmodel**

Playback controls slots

User interface functions for audio playback control.

- void **setTimer** ()

  It sets audio playback time from the cursor.
- void **setPlayback** (bool state)

  It controls the input and output playback.
- void **openInfo** ()

  It opens audio file info dialog.
- void **setTimer** (QTime time)

  It sets audio playback time from a specific time selected by user.

6.30.1 Detailed Description

SACEffects (p. **??**) window interface.

Author

Andrés González Fornell

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SACEffects()

```
SACEffects::SACEffects (
            QWidget * framework = 0 )  [explicit]
```

SACEffects (p. **??**) constructor.

Parameters

framework	**SACEffects** (p. **??**) user interface object

6.30.2.2 ∼SACEffects()

```
SACEffects::∼SACEffects ( )
```

SACEffects (p. **??**) destructor.

6.30.3 Member Function Documentation

6.30.3.1 applyEffect

```
void SACEffects::applyEffect ( )  [private], [slot]
```

It applies the selected effect to the input signal.

6.30.3.2 decode

```
void SACEffects::decode ( )  [private], [slot]
```

It decodes the audio input.

6.30.3.3 encode

```
void SACEffects::encode ( )  [private], [slot]
```

It encodes a new audio through Coder.

6.30.3.4 exportOutput

```
void SACEffects::exportOutput ( )  [private], [slot]
```

It exports generated output singal as audio file.

6.30.3.5 getDuration()

```
void SACEffects::getDuration (
            QLabel label )
```

6.30.3.6 load

```
void SACEffects::load ( )  [private], [slot]
```

It loads a file as decoder input.

6.30.3.7 openChannelsCharts

```
void SACEffects::openChannelsCharts ( )  [private], [slot]
```

It opens audio file info dialog.

6.30.3.8 openInfo

```
void SACEffects::openInfo ( )  [private], [slot]
```

It opens audio file info dialog.

6.30.3.9 pause()

```
void SACEffects::pause ( )
```

It pauses input playback.

6.30.3.10 play()

```
void SACEffects::play ( )
```

It starts playing input.

6.30.3.11 reset()

```
void SACEffects::reset ( )
```

It resets all decoding parameters, including input file.

6.30.3.12 sendOutput

```
void SACEffects::sendOutput ( )  [private], [slot]
```

It sends the current output signal to the output device and plots on the output chart.

6.30.3.13 setBinauralQuality()

```
void SACEffects::setBinauralQuality (
            BinauralQuality::binauralquality binauralquality )
```

It sets SAC parameter binaural quality.

Parameters

binauralquality	binaural quality

6.30.3.14 setBitstream()

```
void SACEffects::setBitstream (
            std::string filename )
```

It sets the bitstream audio file.

Parameters

filename	file path

6.30.3.15 setBuried

```
void SACEffects::setBuried (
            bool state )  [private], [slot]
```

It sets bitstream as buried or not.

Parameters

state	true if bitstream is buried

6.30.3.16 setDecodingType()

```
void SACEffects::setDecodingType (
            DecodingType::decodingtype decodingtype )
```

It sets SAC parameter decoding type.

Parameters

decodingtype	decoding type

6.30.3.17 setDuration()

```
void SACEffects::setDuration (
            QLabel * label,
            double duration )
```

It sets a duration indicator text on an user interface label object.

Parameters

label	user interface object where to indicate duration
duration	input audio file duration [s]

6.30.3.18 setEffect()

```
void SACEffects::setEffect (
              Effect::effectID effect )
```

It sets an effect for the effect monitor.

Parameters

effect	selected effect

6.30.3.19 setFormat()

```
void SACEffects::setFormat (
            int fs,
            int samplesize )
```

It sets audio output format.

Parameters

fs	signal sampling frequency
samplesize	signal sample size

6.30.3.20 setHRTFModel()

```
void SACEffects::setHRTFModel (
              HRTFModel::hrtfmodel hrtfmodel )
```

It sets SAC parameter HRTF model.

Parameters

hrtfmodel	HRTF model

6.30.3.21 setInput()

```
void SACEffects::setInput (
            std::string filename )
```

It sets the input audio file.

Parameters

filename	file path

6.30.3.22 setPlayback

```
void SACEffects::setPlayback (
            bool state )  [private], [slot]
```

It controls the input and output playback.

Parameters

state	true to play and false to pause

6.30.3.23 setSource()

```
void SACEffects::setSource (
            std::string filename )
```

It sets the source audio file.

Parameters

filename	file path

6.30.3.24 setTimer() [1/2]

```
void SACEffects::setTimer ( )  [private]
```

It sets audio playback time from the cursor.

6.30.3.25 setTimer [2/2]

```
void SACEffects::setTimer (
            QTime time )  [private], [slot]
```

It sets audio playback time from a specific time selected by user.

Parameters

time	

6.30.3.26 setUpmixType()

```
void SACEffects::setUpmixType (
              UpmixType::upmixtype upmixtype )
```

It sets SAC parameter upmix type.

Parameters

upmixtype	upmix type

6.30.3.27 stop

```
void SACEffects::stop ( )  [private], [slot]
```

It stops input playback.

6.30.3.28 test

```
void SACEffects::test ( )  [private], [slot]
```

It plays a test to a selected output audio device.

6.30.3.29 toggleBinauralQuality

```
void SACEffects::toggleBinauralQuality (
            QAction * item )  [private], [slot]
```

Slot action for menu binaural quality items.

Parameters

item	selected item

6.30.3.30 toggleDecodingType

```
void SACEffects::toggleDecodingType (
            QAction * item )  [private], [slot]
```

Slot action for menu decoding type items.

Parameters

item	selected item

6.30.3.31 toggleEffect

```
void SACEffects::toggleEffect ( )  [private], [slot]
```

Slot action for menu effect items.

6.30.3.32 toggleHRTFModel

```
void SACEffects::toggleHRTFModel (
            QAction * item )  [private], [slot]
```

Slot action for menu HRTF model items.

Parameters

item	selected item

6.30.3.33 toggleUpmixType

```
void SACEffects::toggleUpmixType (
            QAction * item )  [private], [slot]
```

Slot action for menu upmix type items.

Parameters

item	selected item

6.30.3.34 updateControls()

```
void SACEffects::updateControls ( )
```

It updates enability of user interface controls according to the current parameters state.

6.30.4 Member Data Documentation

6.30.4.1 binauralquality

```
BinauralQuality::binauralquality SACEffects::binauralquality  [private]
```

SAC decoder parameter

6.30.4.2 bitstream

```
File* SACEffects::bitstream  [private]
```

encoded bit stream file object

6.30.4.3 buried

```
bool SACEffects::buried  [private]
```

SAC decoder parameter

6.30.4.4 channels_input

```
ChannelsList* SACEffects::channels_input  [private]
```

input channels list

6.30.4.5 channels_output

```
ChannelsList* SACEffects::channels_output  [private]
```

output channels list

6.30.4.6 chunksize

```
int SACEffects::chunksize  [private]
```

number of samples in a chunk

6.30.4.7 clock

```
QTimer* SACEffects::clock  [private]
```

application clock to call effect in proccess periodically

6.30.4.8 decodingtype

```
DecodingType::decodingtype SACEffects::decodingtype  [private]
```

SAC decoder parameter

6.30.4.9 effect

```
Effect* SACEffects::effect  [private]
```

effect object

6.30.4.10 effectsmonitor

```
EffectsMonitor* SACEffects::effectsmonitor  [private]
```

effects monitor object

6.30.4.11 fs

```
const int SACEffects::fs = 44100
```

signal sampling frequency [Hz]

6.30.4.12 hrtfmodel

```
HRTFModel::hrtfmodel SACEffects::hrtfmodel  [private]
```

SAC decoder parameter

6.30.4.13 input

```
WAVFile* SACEffects::input  [private]
```

decoded input file object

6.30.4.14 muted

```
bool SACEffects::muted  [private]
```

it indicates if output playback is muted (true) or not (false)

6.30.4.15 process

```
ProcessManager* SACEffects::process  [private]
```

process manager object

6.30.4.16 source

```
WAVFile* SACEffects::source  [private]
```

encoded source file object

6.30.4.17 status

```
QStatusBar* SACEffects::status  [private]
```

user interface status bar object

6.30.4.18 ui

```
Ui::SACEffects* SACEffects::ui  [private]
```

user interface object

6.30.4.19 upmixtype

```
UpmixType::upmixtype SACEffects::upmixtype  [private]
```

SAC decoder parameter

The documentation for this class was generated from the following files:

- src/interface/ **SACEffects.h**
- src/interface/ **SACEffects.cpp**

6.31 AudioStream::SignalRange Struct Reference

index range.

`#include <AudioStream.h>`

Public Attributes

- int **start**
- int **end**

6.31.1 Detailed Description

index range.

6.31.2 Member Data Documentation

6.31.2.1 end

`int AudioStream::SignalRange::end`

end index of the audio stream

6.31.2.2 start

`int AudioStream::SignalRange::start`

start index of the audio stream

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

- src/process/ **AudioStream.h**

6.32 AudioStream::TimeSlot Struct Reference

It indicates time slot of the available signal.

`#include <AudioObject.h>`

Public Attributes

- int **start**
- int **end**

6.32.1 Detailed Description

It indicates time slot of the available signal.

6.32.2 Member Data Documentation

6.32.2.1 end

```
int AudioStream::TimeSlot::end
```

end time

6.32.2.2 start

```
int AudioStream::TimeSlot::start
```

start time

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

- src/interface/ **AudioObject.h**

6.33 UpmixType Struct Reference

SAC decoder parameter upmix type.

```
#include <SACEffects.h>
```

Public Types

- enum **upmixtype** { **normal** = 0, **blind** = 1, **binaural** = 2, **stereo** = 3 }

6.33.1 Detailed Description

SAC decoder parameter upmix type.

6.33.2 Member Enumeration Documentation

6.33.2.1 upmixtype

```
enum UpmixType::upmixtype
```

Enumerator

normal	
blind	
binaural	
stereo	

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

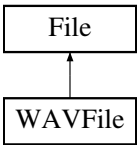
- src/interface/ **SACEffects.h**

6.34 WAVFile Class Reference

Audio file as WAV format class.

`#include <File.h>`

Inheritance diagram for WAVFile:



Classes

- struct **Header**

  Audio file header struct.

Public Member Functions

- **WAVFile** (bool **writepermission**)

  WAVFile *(*p. **??***) constructor.*
- **WAVFile** (std::string **filename**, bool **writepermission**)

  WAVFile *(*p. **??***) constructor.*
- **WAVFile** (std::string **filename**, int channels, int **fs**, int sampleformat)

  WAVFile *(*p. **??***) constructor. Write file is allowed.*
- ∼**WAVFile** ()

  WAVFile *(*p. **??***) destructor.*
- void **setCursor** (int **cursor**)

  It sets the signal reading cursor to keep on reading from another position.
- int **getCursor** ()

  It gets the current signal reading cursor.
- int **samples** ()

  It gets the number of audio samples.
- void **readHeader** ()

  It reads the file header and sets the format header into the audio file object.
- void **writeHeader** ()

  It writes the header on the file from the audio file object header.
- float ∗∗ **readSamples** (int **samples**)

  It reads an array of samples from the audio file.
- void **writeSamples** (float ∗∗array, int **samples**)

  It writes an array of samples on the audio file.

Public Attributes

- **Header header**
- double **duration**

6.34.1 Detailed Description

Audio file as WAV format class.

Author

Andrés González Fornell

6.34.2 Constructor & Destructor Documentation

6.34.2.1 WAVFile() [1/3]

```
WAVFile::WAVFile (
            bool writepermission )
```

WAVFile (p. **??**) constructor.

Parameters

writepermission	file write permission (true if it is allowed)

6.34.2.2 WAVFile() [2/3]

```
WAVFile::WAVFile (
            std::string filename,
            bool writepermission )
```

WAVFile (p. **??**) constructor.

Parameters

filename	file path
writepermission	file write permission (true if it is allowed)

6.34.2.3 WAVFile() [3/3]

```
WAVFile::WAVFile (
            std::string filename,
            int channels,
            int fs,
            int sampleformat )
```

WAVFile (p. **??**) constructor. Write file is allowed.

Parameters

filename	file path
channels	number of channels
fs	signal sample rate
sampleformat	number of bits of a sample

6.34.2.4 ~WAVFile()

```
WAVFile::~WAVFile ( )
```

WAVFile (p. **??**) destructor.

6.34.3 Member Function Documentation

6.34.3.1 getCursor()

```
int WAVFile::getCursor ( )
```

It gets the current signal reading cursor.

Returns

cursor [Bytes] from the beginning of the signal (instead of the file)

6.34.3.2 readHeader()

```
void WAVFile::readHeader ( )
```

It reads the file header and sets the format header into the audio file object.

6.34.3.3 readSamples()

```
float ** WAVFile::readSamples (
            int samples )
```

It reads an array of samples from the audio file.

Parameters

samples	number of samples

Returns

two dimensional array ([channel][sample]) of samples (from -1 to 1)

6.34.3.4 samples()

```
int WAVFile::samples ( )
```

It gets the number of audio samples.

Returns

number of audio samples

6.34.3.5 setCursor()

```
void WAVFile::setCursor (
            int cursor )
```

It sets the signal reading cursor to keep on reading from another position.

Parameters

cursor	new cursor position in samples (instead of bytes) from the beginning of the signal (instead of the file)

6.34.3.6 writeHeader()

```
void WAVFile::writeHeader ( )
```

It writes the header on the file from the audio file object header.

6.34.3.7 writeSamples()

```
void WAVFile::writeSamples (
            float ** array,
            int samples )
```

It writes an array of samples on the audio file.

Parameters

array	two dimensional array ([channel][sample]) of samples (from -1 to 1)
samples	number of samples

6.34.4 Member Data Documentation

6.34.4.1 duration

```
double WAVFile::duration
```

audio file duration [s]

6.34.4.2 header

```
Header WAVFile::header
```

audio file header

The documentation for this class was generated from the following files:

- src/process/ **File.h**
- src/process/ **File.cpp**

Chapter 7

File Documentation

7.1 src/effects/Compressor.cpp File Reference

```
#include "Compressor.h"
```

7.2 src/effects/Compressor.h File Reference

```
#include "EffectBase.h"
```

Classes

- class **AS_EFFECT**

7.3 src/effects/Effect.cpp File Reference

```
#include "Effect.h"
#include "EffectBase.h"
```

Macros

- #define **EFFECT**(ID, NAME) ID,
- #define **EFFECT**(ID, NAME) NAME,
- #define **EFFECT**(ID, NAME) NAME,

7.3.1 Macro Definition Documentation

7.3.1.1 EFFECT [1/3]

```
#define EFFECT(
               ID,
               NAME ) ID,
```

7.3.1.2 EFFECT [2/3]

```
#define EFFECT(
               ID,
               NAME ) NAME,
```

7.3.1.3 EFFECT [3/3]

```
#define EFFECT(
               ID,
               NAME ) NAME,
```

7.4 src/effects/Effect.h File Reference

```
#include "Compressor.h"
#include "Equalizer.h"
#include "Reverb.h"
```

Classes

- class **Effect**

  Effect (p. **??**) *class. It contains (by inheritance) all effects classes.*

Macros

- #define **LIST**
- #define **INHERITANCE**
- #define **EFFECT**(ID, NAME) ID,

7.4.1 Macro Definition Documentation

7.4.1.1 EFFECT

```
#define EFFECT(
               ID,
               NAME ) ID,
```

7.4.1.2 INHERITANCE

```
#define INHERITANCE
```

Value:

```
public Compressor, \
    public Equalizer, \
    public Reverb
```

7.4.1.3 LIST

```
#define LIST
```

Value:

```
EFFECT(Compressor, "Compressor") \
    EFFECT(Equalizer, "Equalizer") \
    EFFECT(Reverb, "Reverb")
```

7.5 src/effects/EffectBase.h File Reference

```
#include "stdlib.h"
#include "cmath"
#include "vector"
#include "map"
#include "../sac/SACBitstream.h"
#include "../tools/Logger.h"
```

Classes

- class **EffectBase**

  Effect *(*p. **??***) base class.*

Macros

- #define **AS_EFFECT** : public **EffectBase**
- #define **AS_EFFECT_CONSTRUCTOR** : **EffectBase::EffectBase**()

7.5.1 Macro Definition Documentation

7.5.1.1 AS_EFFECT

```
#define AS_EFFECT : public EffectBase
```

7.5.1.2 AS_EFFECT_CONSTRUCTOR

```
#define AS_EFFECT_CONSTRUCTOR : EffectBase::EffectBase()
```

7.6 src/effects/Equalizer.cpp File Reference

```
#include "Equalizer.h"
```

7.7 src/effects/Equalizer.h File Reference

```
#include "EffectBase.h"
```

Classes

- class **AS_EFFECT**

Variables

- const int **order** = 2

7.7.1 Variable Documentation

7.7.1.1 order

```
const int order = 2
```

7.8 src/effects/Reverb.cpp File Reference

```
#include "Reverb.h"
```

7.9 src/effects/Reverb.h File Reference

```
#include "EffectBase.h"
```

Classes

- class **AS_EFFECT**

Variables

- const int **maxdelay** = 5899
- const int **maxfilters** = 12

7.9.1 Variable Documentation

7.9.1.1 maxdelay

```
const int maxdelay = 5899
```

7.9.1.2 maxfilters

```
const int maxfilters = 12
```

7.10 src/interface/AudioInfo.cpp File Reference

```
#include "AudioInfo.h"
#include "ui_AudioInfo.h"
```

7.11 src/interface/AudioInfo.h File Reference

```
#include "stdlib.h"
#include "QDialog"
#include "QTableView"
#include "QVector"
#include "../process/File.h"
#include "../tools/Logger.h"
```

Classes

- class **AudioInfo**

  Audio object info dialog class.

Namespaces

- **Ui**

7.12 src/interface/AudioObject.h File Reference

```
#include "vector"
#include "../tools/Logger.h"
```

Classes

- class **AudioStream**

  Audio objects for the SAOC interface.
- struct **AudioStream::TimeSlot**

  It indicates time slot of the available signal.

7.13 src/interface/AudioOutput.cpp File Reference

```
#include "AudioOutput.h"
#include "ui_AudioTest.h"
```

7.14 src/interface/AudioOutput.h File Reference

```
#include <cmath>
#include "QtMath"
#include "stdlib.h"
#include "QIODevice"
#include "QDialog"
#include "QTimer"
#include "QtEndian"
#include "QList"
#include "QAudioDeviceInfo"
#include "QAudioOutput"
#include "QComboBox"
#include "../process/AudioSignal.h"
#include "../tools/Logger.h"
```

Classes

- class **OutputDevice**

  Audio output device class (QIODevice extension).
- class **AudioOutput**

  Audio output class.
- class **AudioTest**

  Audio output test class.

Namespaces

- **Ui**

7.15 src/interface/ChannelsList.cpp File Reference

```
#include "ChannelsList.h"
#include "ui_ChannelsCharts.h"
```

7.16 src/interface/ChannelsList.h File Reference

```
#include "stdlib.h"
#include "vector"
#include "cmath"
#include "QDialog"
#include "QObject"
#include "QWidget"
#include "QLayout"
#include "QGroupBox"
#include "QLabel"
#include "QLineEdit"
#include "QCheckBox"
```

```
#include "AudioOutput.h"
#include "Chart2D.h"
#include "../process/AudioSignal.h"
#include "../process/AudioStream.h"
#include "../tools/Logger.h"
```

Classes

- class **Channel**

  Single-object class from channels list.
- class **ChannelsList**

  Channels list class. It shows information about channels signals.
- class **ChannelsCharts**

Namespaces

- **Ui**

7.17 src/interface/Chart2D.cpp File Reference

```
#include "Chart2D.h"
```

7.18 src/interface/Chart2D.h File Reference

```
#include "cmath"
#include "stdlib.h"
#include "QWidget"
#include "QObject"
#include "QtCharts"
#include "QPointF"
#include "../tools/Logger.h"
```

Classes

- class **Chart2D**

  Class for plotting two-dimensional charts.
- struct **Chart2D::ChartOptions**

  It defines some features of the chart.

7.19 src/interface/EffectsMonitor.cpp File Reference

```
#include "EffectsMonitor.h"
```

Variables

- const std::string **folder** = ":/imports/effects/"
- const std::string **prefix** = "effect_params_"

7.19.1 Variable Documentation

7.19.1.1 folder

```
const std::string folder = ":/imports/effects/"
```

7.19.1.2 prefix

```
const std::string prefix = "effect_params_"
```

7.20 src/interface/EffectsMonitor.h File Reference

```
#include "stdlib.h"
#include "vector"
#include "map"
#include "QObject"
#include "QDir"
#include "QFile"
#include "QTextStream"
#include "QWidget"
#include "QLayout"
#include "QFormLayout"
#include "QLabel"
#include "QSpinBox"
#include "QLineEdit"
#include "QCheckBox"
#include "QGroupBox"
#include "QRadioButton"
#include "Chart2D.h"
#include "../effects/Effect.h"
#include "../process/File.h"
#include "../tools/Logger.h"
```

Classes

- class **EffectsMonitor**

  Class for managing effects parameters.

7.21 src/interface/Encoder.cpp File Reference

```
#include "Encoder.h"
```

7.22 src/interface/Encoder.h File Reference

```
#include "stdlib.h"
#include "cmath"
#include "QApplication"
#include "QMainWindow"
#include "QFileDialog"
#include "QPushButton"
#include "ui_Encoder.h"
#include "../process/File.h"
#include "../tools/Logger.h"
#include "../sac/sac_encoder.h"
```

Classes

- class **Encoder**

  Encoder *(*p. **??***) window interface.*

Namespaces

- **Ui**

7.23 src/interface/main.cpp File Reference

```
#include "QApplication"
#include "QMainWindow"
#include "Encoder.h"
#include "SACEffects.h"
#include "../tools/Logger.h"
```

Functions

main.cpp

Main function

Author

Andrés González Fornell

- int **main** (int argc, char ∗argv[ ])

  Main function.

7.23.1 Function Documentation

7.23.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Main function.

7.24 src/main.cpp File Reference

```
#include "cstring"
#include "process/ProcessManager.h"
#include "effects/Effect.h"
#include "tools/Logger.h"
```

TFM.cpp

Test Unit file.

Author

Andrés González Fornell

- const int **fs** = 44100
- void **printHelp** (std::string app)
  It prints help.
- int **main** (int argc, char ∗argv[ ])

7.24.1 Function Documentation

7.24.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

7.24.1.2 printHelp()

```
void printHelp (
            std::string app )
```

It prints help.

Parameters

app	application name

7.24.2 Variable Documentation

7.24.2.1 fs

```
const int fs = 44100
```

7.25 src/interface/SACEffects.cpp File Reference

```
#include "SACEffects.h"
#include "ui_SACEffects.h"
```

7.26 src/interface/SACEffects.h File Reference

```
#include "iostream"
#include "stdlib.h"
#include "map"
#include "QApplication"
#include "QMainWindow"
#include "QTimer"
#include "Encoder.h"
#include "ChannelsList.h"
#include "EffectsMonitor.h"
#include "AudioOutput.h"
#include "AudioInfo.h"
#include "Chart2D.h"
#include "../process/ProcessManager.h"
#include "../process/File.h"
#include "../tools/Logger.h"
```

Classes

- struct **UpmixType**

  SAC decoder parameter upmix type.
- struct **DecodingType**

  SAC decoder parameter decoding type.
- struct **BinauralQuality**

  SAC decoder parameter binaural quality.
- struct **HRTFModel**

  SAC decoder parameter HRTF model.
- class **SACEffects**

  SACEffects *(*p. **??***) window interface.*

Namespaces

- **Ui**

7.27 src/process/AudioSignal.cpp File Reference

```
#include "AudioSignal.h"
```

Macros

Constants references for complex numbers

It can be used to reference real and imaginary parts of a complex number

- #define **real** 0
- #define **imag** 1

7.27.1 Macro Definition Documentation

7.27.1.1 imag

```
#define imag 1
```

7.27.1.2 real

```
#define real 0
```

7.28 src/process/AudioSignal.h File Reference

```
#include "vector"
#include "cmath"
#include "fftw3.h"
#include "../tools/Logger.h"
```

Classes

- class **AudioSignal**

  *TODO **AudioSignal.cpp** (*p. **??**) *description.*

7.29 src/process/AudioStream.cpp File Reference

```
#include "AudioStream.h"
```

7.30 src/process/AudioStream.h File Reference

```
#include "cmath"
#include "stdlib.h"
#include "vector"
#include "../process/AudioSignal.h"
#include "../tools/Logger.h"
```

Classes

- class **AudioStream**

  Audio objects for the SAOC interface.
- struct **AudioStream::SignalRange**

  index range.

7.31 src/process/File.cpp File Reference

```
#include "File.h"
```

7.32 src/process/File.h File Reference

```
#include "stdlib.h"
#include <cmath>
#include "fstream"
#include "sys/stat.h"
#include "../tools/Logger.h"
```

Classes

- class **File**

  Audio file class.
- struct **File::Endianess**
- class **WAVFile**

  Audio file as WAV format class.
- struct **WAVFile::Header**

  Audio file header struct.

7.33 src/process/ProcessManager.cpp File Reference

```
#include "ProcessManager.h"
```

7.34 src/process/ProcessManager.h File Reference

```
#include "stdlib.h"
#include "AudioStream.h"
#include "File.h"
#include "../effects/Effect.h"
#include "../tools/Logger.h"
#include "../sac/SACBitstream.h"
#include "../sac/sac_decoder.h"
```

Classes

- class **ProcessManager**

  Process manager class. It contains all functions to perform the signal treatment process.

7.35 src/sac/sac_decoder.c File Reference

```
#include "sac_decoder.h"
#include "spatial_frontend.h"
```

Functions

- void **myexit** (char ∗s)

  It adds a char string to the char string variable error to s.
- char ∗ **sac_decode** (const char ∗input_filename, const char ∗output_filename, const char ∗bitstream_↩ filename, double **fs**, int upmixtype, int decodingtype, int binauralquality, int hrtfmodel)

  It performs the SAC decoder.
- **if** (input_samples !=NULL)
- **if** (input_interleaved !=NULL)
- **for** (int channel=0;channel< input_channels;channel++)
- **SpatialDecClose** (decoder)
- **if** (bitstream_type==BS_FILE)
- **AFclose** (input)
- **if** (strcmp(**error**,"")==0)

Variables

- char ∗ **error** = ""
- **else**

7.35.1 Function Documentation

7.35.1.1 AFclose()

```
AFclose (
            input  )
```

7.35.1.2 for()

```
for ( )
```

7.35.1.3 if() [1/4]

```
if (
            input_samples !    = NULL )
```

7.35.1.4 if() [2/4]

```
if (
            input_interleaved !    = NULL )
```

7.35.1.5 if() [3/4]

```
if (
            bitstream_type  = = BS_FILE )
```

7.35.1.6 if() [4/4]

```
if (
            strcmp( error,"")  =0 )
```

7.35.1.7 myexit()

```
void myexit (
            char * s )
```

It adds a char string to the char string variable error to s.

Parameters

s	

7.35.1.8 sac_decode()

```
char* sac_decode (
            const char * input_filename,
            const char * output_filename,
            const char * bitstream_filename,
            double fs,
            int upmixtype,
            int decodingtype,
            int binauralquality,
            int hrtfmodel )
```

It performs the SAC decoder.

Parameters

input_filename	filename of the downmix input audio file
output_filename	filename of the multichannel output audio file (it will be automatically created)
bitstream_filename	filename of the bitstream file (not present in blind upmix case)
fs	audio sampling frequency [Hz]
upmixtype	upmix type 0: normal 1: blind 2: binaural 3: stereo
decodingtype	decoding type 0: low 1: high
binauralquality	binaural upmix quality 0: parametric 1: filtering
hrtfmodel	HRTF model 0: kemar 1: vast 2: mps_vt

Returns

error message (NULL if the decoding has succeded)

7.35.1.9 SpatialDecClose()

```
SpatialDecClose (
            decoder  )
```

7.35.2 Variable Documentation

7.35.2.1 else

```
else
```

Initial value:

```
{
        return error
```

7.35.2.2 error

```
char* error = ""
```

7.36 src/sac/sac_decoder.h File Reference

Functions

- char * **sac_decode** (const char *input_filename, const char *output_filename, const char *bitstream_↩ filename, double **fs**, int upmixtype, int decodingtype, int binauralquality, int hrtfmodel)

  It performs the SAC decoder.
- void **myexit** (char *s)

  It adds a char string to the char string variable error to s.

7.36.1 Function Documentation

7.36.1.1 myexit()

```
void myexit (
            char * s )
```

It adds a char string to the char string variable error to s.

Parameters

s	

7.36.1.2 sac_decode()

```
char* sac_decode (
            const char * input_filename,
```

```
const char * output_filename,
const char * bitstream_filename,
double fs,
int upmixtype,
int decodingtype,
int binauralquality,
int hrtfmodel )
```

It performs the SAC decoder.

Parameters

input_filename	filename of the downmix input audio file
output_filename	filename of the multichannel output audio file (it will be automatically created)
bitstream_filename	filename of the bitstream file (not present in blind upmix case)
fs	audio sampling frequency [Hz]
upmixtype	upmix type 0: normal 1: blind 2: binaural 3: stereo
decodingtype	decoding type 0: low 1: high
binauralquality	binaural upmix quality 0: parametric 1: filtering
hrtfmodel	HRTF model 0: kemar 1: vast 2: mps_vt

Returns

error message (NULL if the decoding has succeded)

7.37 src/sac/sac_encoder.c File Reference

```
#include "sac_encoder.h"
#include "string.h"
#include "libtsp.h"
#include "sac_enc.h"
#include "bitstream.h"
#include "sac_bd_embedder.h"
```

Functions

- char * **sac_encode** (const char *input_filename, const char *output_filename, const char *bitstream_↩ filename, double **fs**, int tree, int timeslots)

  It performs the SAC encoder.

7.37.1 Function Documentation

7.37.1.1 sac_encode()

```
char* sac_encode (
            const char * input_filename,
            const char * output_filename,
            const char * bitstream_filename,
            double fs,
            int tree,
            int timeslots )
```

It performs the SAC encoder.

Parameters

input_filename	filename of the multichannel input audio file
output_filename	filename of the downmix output audio file (it will be automatically created)
bitstream_filename	filename of the bitstream output file or "buried" (it will be automatically created)
fs	audio sampling frequency [Hz]
tree	tree config: 5151 (mono), 5152 (mono), 525 (stereo) (5151 by default)
timeslots	times slots: 16 or 32 (32 by default)

Returns

error message (NULL if the encoding has succeded)

7.38 src/sac/sac_encoder.h File Reference

```
#include "math.h"
#include "stdlib.h"
```

Functions

- char * **sac_encode** (const char *input_filename, const char *output_filename, const char *bitstream_↩ filename, double **fs**, int tree, int timeslots)

  It performs the SAC encoder.

7.38.1 Function Documentation

7.38.1.1 sac_encode()

```
char* sac_encode (
            const char * input_filename,
            const char * output_filename,
            const char * bitstream_filename,
            double fs,
            int tree,
            int timeslots )
```

It performs the SAC encoder.

Parameters

input_filename	filename of the multichannel input audio file
output_filename	filename of the downmix output audio file (it will be automatically created)
bitstream_filename	filename of the bitstream output file or "buried" (it will be automatically created)
fs	audio sampling frequency [Hz]
tree	tree config: 5151 (mono), 5152 (mono), 525 (stereo) (5151 by default)
timeslots	times slots: 16 or 32 (32 by default)

Returns

error message (NULL if the encoding has succeded)

7.39 src/sac/SACBitstream.cpp File Reference

```
#include "SACBitstream.h"
```

7.40 src/sac/SACBitstream.h File Reference

```
#include "stdlib.h"
#include "../process/File.h"
#include "../tools/Logger.h"
```

Classes

- class **SACBitstream**

  SAC bitstream class.
- struct **SACBitstream::ChannelType**

  It specifies the channel type.

7.41 src/tools/Logger.cpp File Reference

Functions to create log messages on console.

```
#include "Logger.h"
```

Functions

- std::string **tab** (std::string content, const int tab_max)

  It returns the string tab code to align log messages.
- void **consolelog** (std::string source, **LogType::logtype** logtype, std::string message)

  Log a message on console.

Variables

Font styles

ANSI code for some font styles for log messages usage.

- const std::string **reset** = "\033[0m"
- const std::string **bold** = "\033[1m"
- const std::string **italic** = "\033[3m"
- const std::string **black** = "\033[30m"
- const std::string **red** = "\033[31m"
- const std::string **green** = "\033[32m"
- const std::string **yellow** = "\033[33m"
- const std::string **blue** = "\033[34m"
- const std::string **magenta** = "\033[35m"
- const std::string **cyan** = "\033[36m"
- const std::string **white** = "\033[37m"

7.41.1 Detailed Description

Functions to create log messages on console.

Author

Andrés González Fornell

7.41.2 Function Documentation

7.41.2.1 consolelog()

```
void consolelog (
            std::string source,
             LogType::logtype logtype,
            std::string message )
```

Log a message on console.

Parameters

source	origin class/method/file where the message was logged
logtype	type of message
message	message

Returns

void

7.41.2.2 tab()

```
std::string tab (
            std::string content,
            const int tab_max )
```

It returns the string tab code to align log messages.

Parameters

content	Content of the tabulation
tab_max	Maximum number of tabulations

7.41.3 Variable Documentation

7.41.3.1 black

```
const std::string black = "\033[30m"
```

7.41.3.2 blue

```
const std::string blue = "\033[34m"
```

7.41.3.3 bold

```
const std::string bold = "\033[1m"
```

7.41.3.4 cyan

```
const std::string cyan = "\033[36m"
```

7.41.3.5 green

```
const std::string green = "\033[32m"
```

7.41.3.6 italic

```
const std::string italic = "\033[3m"
```

7.41.3.7 magenta

```
const std::string magenta = "\033[35m"
```

7.41.3.8 red

```
const std::string red = "\033[31m"
```

7.41.3.9 reset

```
const std::string reset = "\033[0m"
```

7.41.3.10 white

```
const std::string white = "\033[37m"
```

7.41.3.11 yellow

```
const std::string yellow = "\033[33m"
```

7.42 src/tools/Logger.h File Reference

```
#include "iostream"
#include "stdlib.h"
```

Classes

- struct **LogType**

Functions

LogType

Logger message type.

- void **consolelog** (std::string, **LogType::logtype** logtype, std::string)
  Log a message on console.

7.42.1 Function Documentation

7.42.1.1 consolelog()

```
void consolelog (
            std::string source,
             LogType::logtype logtype,
            std::string message )
```

Log a message on console.

Parameters

source	origin class/method/file where the message was logged
logtype	type of message
message	message

Returns

void