

Ambiente de Teste para Filtros Adaptativos

Generated by Doxygen 1.8.6

Wed Jan 29 2014 15:50:25

Contents

1	Main Page	1
2	Todo List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Class Documentation	11
6.1	Signal::DFTDriver Class Reference	11
6.1.1	Detailed Description	12
6.1.2	Member Enumeration Documentation	13
6.1.2.1	dir_t	13
6.1.3	Constructor & Destructor Documentation	13
6.1.3.1	DFTDriver	13
6.1.4	Member Function Documentation	13
6.1.4.1	br	13
6.1.4.2	initialize_costbl	13
6.1.4.3	initialize_sintbl	14
6.1.4.4	operator()	14
6.1.4.5	Wim	14
6.1.4.6	Wre	15
6.1.5	Member Data Documentation	15
6.1.5.1	bits	15
6.1.5.2	costbl	15
6.1.5.3	sintbl	16
6.1.5.4	tblbits	16
6.1.5.5	tblsize	16

6.2	FileError Class Reference	17
6.2.1	Detailed Description	19
6.2.2	Constructor & Destructor Documentation	19
6.2.2.1	FileError	19
6.2.2.2	~FileError	19
6.2.3	Member Function Documentation	19
6.2.3.1	what	19
6.2.4	Member Data Documentation	19
6.2.4.1	filename	19
6.2.4.2	msg	20
6.3	Signal Class Reference	20
6.3.1	Detailed Description	22
6.3.2	Member Typedef Documentation	22
6.3.2.1	container_t	22
6.3.2.2	index_t	22
6.3.2.3	sample_t	22
6.3.3	Member Enumeration Documentation	22
6.3.3.1	delay_t	22
6.3.4	Constructor & Destructor Documentation	23
6.3.4.1	Signal	23
6.3.4.2	Signal	23
6.3.4.3	Signal	23
6.3.4.4	~Signal	23
6.3.5	Member Function Documentation	23
6.3.5.1	array	23
6.3.5.2	delay	24
6.3.5.3	filter	24
6.3.5.4	gain	24
6.3.5.5	l_inf_norm	25
6.3.5.6	normalize	25
6.3.5.7	operator+=	25
6.3.5.8	operator[]	25
6.3.5.9	operator[]	26
6.3.5.10	play	26
6.3.5.11	samplerate	26
6.3.5.12	samples	26
6.3.5.13	set_samplerate	27
6.3.5.14	set_size	27
6.3.6	Member Data Documentation	27
6.3.6.1	counter	27

6.3.6.2	data	28
6.3.6.3	dft	28
6.3.6.4	srate	28
7	File Documentation	29
7.1	main.cpp File Reference	29
7.1.1	Detailed Description	30
7.1.2	Macro Definition Documentation	30
7.1.2.1	ATFA_DIR	30
7.1.3	Function Documentation	30
7.1.3.1	main	30
7.1.3.2	portaudio_end	31
7.1.3.3	portaudio_init	31
7.1.4	Variable Documentation	31
7.1.4.1	static_dirname	31
7.1.4.2	static_filename	31
7.1.4.3	static_projdirname	31
7.2	README.md File Reference	31
7.3	Signal.cpp File Reference	32
7.3.1	Detailed Description	32
7.3.2	Function Documentation	32
7.3.2.1	callback	32
7.4	Signal.h File Reference	33
7.4.1	Detailed Description	34
7.4.2	Function Documentation	34
7.4.2.1	operator+	34
7.4.3	Variable Documentation	34
7.4.3.1	NULL	34
Index		36

Chapter 1

Main Page

Projeto Final de Graduação

Para compilar:

```
[ pf/ ]$ cd build
[ pf/build/ ]$ ./config.sh
[ pf/build/ ]$ cd release
[ pf/build/release/ ]$ make
```

O executável será colocado no diretório `build/release`.

Para gerar um executável do *Debug Build*, basta substituir o `cd release` por `cd debug`. O novo diretório do executável será `build/debug`.

Manual em PDF [aqui](#).

Chapter 2

Todo List

Class **Signal**

Implement "stream" signals, to provide real-time processing.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Signal::DFTDriver	11
std::exception	
std::runtime_error	
FileError	17
Signal	20

Chapter 4

Class Index

4.1 Class List

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

Signal::DFTDriver	A class for providing discrete Fourier transform capabilities	11
FileError	A runtime exception while trying to process a file	17
Signal	A time- or frequency-domain signal	20

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

main.cpp	29
Signal.cpp	32
Signal.h	33

Chapter 6

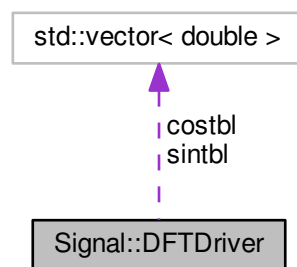
Class Documentation

6.1 Signal::DFTDriver Class Reference

A class for providing discrete Fourier transform capabilities.

```
#include <Signal.h>
```

Collaboration diagram for Signal::DFTDriver:



Public Types

- enum `dir_t` { `DIRECT`, `INVERSE` }

Public Member Functions

- `DFTDriver ()`
Constructor for an object that computes DFTs.
- void `operator() (container_t &re, container_t &im, dir_t direction=DIRECT)`
Used to perform the actual computation of the DFT.

Static Public Member Functions

- static `std::vector< double > initialize_costbl ()`

Initializes the table of cosines.

- static std::vector< double > [initialize_sintbl](#) ()

Initializes the table of sines.

Static Public Attributes

- static const unsigned [tblbits](#) = 14
Number of bits for the index of the table of sines and cosines.
- static const size_t [tblsize](#) = 16384
Number of entries in the tables of sines and cosines.

Private Member Functions

- double [Wre](#) (unsigned k)
Easy access to the table of cosines.
- double [Wim](#) (unsigned k)
Easy access to the table of sines.

Static Private Member Functions

- template<typename T >
static T [br](#) (T x, int [bits](#))
Bit-reverse.

Private Attributes

- unsigned [bits](#)
Number of bits for the current FFT computation.

Static Private Attributes

- static const std::vector< double > [sintbl](#)
Table of sines.
- static const std::vector< double > [costbl](#)
Table of cosines.

6.1.1 Detailed Description

A class for providing discrete Fourier transform capabilities.

This class implements the radix-2 FFT algorithm used in the [Signal::filter\(\)](#) method.

Usage:

```
Signal::DFTDriver dft;
Signal::container_t real, imag;
// initialize the real and imaginary parts of a complex time-domain
// signal
dft(real, imag); // performs in-place FFT
// now, work with the real and imaginary parts of the
// frequency-domain version of the signal
dft(real, imag, Signal::DFTDriver::INVERSE); // inverse in-place fft
// now, we can work again with the time-domain complex signal
```

Definition at line 259 of file [Signal.h](#).

6.1.2 Member Enumeration Documentation

6.1.2.1 enum Signal::DFTDriver::dir_t

This is a type for specifying whether we should perform a direct or inverse FFT.

Enumerator

DIRECT Perform direct FFT.

INVERSE Perform inverse FFT.

Definition at line 404 of file Signal.h.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 Signal::DFTDriver::DFTDriver () [inline]

Constructor for an object that computes DFTs.

Does nothing at all.

Definition at line 440 of file Signal.h.

6.1.4 Member Function Documentation

6.1.4.1 template<typename T > static T Signal::DFTDriver::br (T x, int bits) [inline], [static], [private]

Bit-reverse.

Returns the bit-reversed version of the parameter *x*. Assumes *x* is *bits*-bit wide, and ignore any bits with more significance than that.

This function assumes that the number of bits in one `char` is 8, and that bitshifting is zero-padded, and not circular.

Template Parameters

<i>T</i>	The type of the parameter <i>x</i> . It must be an unsigned integer type.
----------	--

Parameters

<i>in</i>	<i>x</i>	The <i>bits</i> -bit unsigned integer to be bit-reversed.
<i>in</i>	<i>bits</i>	The number of bits of the integer <i>x</i> .

Returns

the unsigned integer *x*, bit-reversed.

Definition at line 285 of file Signal.h.

Referenced by operator()().

6.1.4.2 static std::vector<double> Signal::DFTDriver::initialize_costbl () [inline], [static]

Initializes the table of cosines.

Computes a table of cosines that will be handed to the `costbl` member.

See Also

[initialize_sintbl](#)
[costbl](#)

Definition at line 417 of file Signal.h.

References [tblsize](#).

6.1.4.3 `static std::vector<double> Signal::DFTDriver::initialize_sintbl () [inline], [static]`

Initializes the table of sines.

See Also

[initialize_costbl](#)
[sintbl](#)

Definition at line 429 of file Signal.h.

References [tblsize](#).

6.1.4.4 `void Signal::DFTDriver::operator() (container_t & re, container_t & im, dir_t direction = DIRECT)`

Used to perform the actual computation of the DFT.

Implements the radix-2 time-decimation FFT algorithm. The computation happens in-place, which means that the *re* and *im* parameters are substituted by their new versions.

Of course, the *re* and *im* vectors must be of the same size. This size must be a power of two not greater than [tblsize](#).

Exceptions

<code>std::runtime_error</code>	if any of the above conditions aren't met.
---------------------------------	--

Parameters

<i>in, out</i>	<i>re</i>	Real part of the complex signal on which the FFT will act.
<i>in, out</i>	<i>im</i>	Imaginary part.
<i>in</i>	<i>direction</i>	Whether this is a direct or inverse DFT.

Definition at line 428 of file Signal.cpp.

References [bits](#), [br\(\)](#), [DIRECT](#), [INVERSE](#), [tblsize](#), [Wim\(\)](#), and [Wre\(\)](#).

6.1.4.5 `double Signal::DFTDriver::Wim (unsigned k) [inline], [private]`

Easy access to the table of sines.

Parameters

<i>in</i>	<i>k</i>	Same as in Wre .
-----------	----------	----------------------------------

Returns

$$\sin\left(\tau \cdot k / 2^{\text{bits}}\right)$$

See Also

[sintbl](#)
[Wre](#)

Definition at line 378 of file Signal.h.

References [bits](#), [sintbl](#), and [tblbits](#).

Referenced by [operator\(\)\(\)](#).

6.1.4.6 double Signal::DFTDriver::Wre (unsigned *k*) [inline],[private]

Easy access to the table of cosines.

This function is aware of the number of bits of the current FFT, and makes it easy to get the cosine of $\tau \cdot k / 2^{\text{bits}}$, using the pre-computed table of cosines.

Parameters

<i>in</i>	<i>k</i>	An integer in the range $0, 2^{\text{bits}}$.
-----------	----------	--

Returns

$\cos\left(\tau \cdot k / 2^{\text{bits}}\right)$, where τ is shorthand for 2π .

See Also

[costbl](#)
[Wim](#)

Definition at line 366 of file Signal.h.

References [bits](#), [costbl](#), and [tblbits](#).

Referenced by [operator\(\)\(\)](#).

6.1.5 Member Data Documentation

6.1.5.1 unsigned Signal::DFTDriver::bits [private]

Number of bits for the current FFT computation.

Always assume this is uninitialized, and all methods that use it should initialize it themselves.

Definition at line 266 of file Signal.h.

Referenced by [operator\(\)\(\)](#), [Wim\(\)](#), and [Wre\(\)](#).

6.1.5.2 const std::vector< double > Signal::DFTDriver::costbl [static],[private]

Initial value:

=
[Signal::DFTDriver::initialize_costbl\(\)](#)

Table of cosines.

See Also

[sintbl](#)

Definition at line 461 of file Signal.h.

Referenced by Wre().

6.1.5.3 `const std::vector< double > Signal::DFTDriver::sintbl` `[static], [private]`

Initial value:

```
=
    Signal::DFTDriver::initialize_sintbl()
```

Table of sines.

Holds the sines of $\tau \cdot k / \text{tblsize}$, for k in the range $[0, \text{tblsize}]$. Here, τ is shorthand for 2π .

See Also

[costbl](#)

Definition at line 455 of file Signal.h.

Referenced by Wim().

6.1.5.4 `const unsigned Signal::DFTDriver::tblbits = 14` `[static]`

Number of bits for the index of the table of sines and cosines.

We won't be able to perform an N -bit dft if $N > \text{tblbits}$, so this should be big. Also, this **must** be equal to $\log_2(\text{tblsize})$, but there's nothing in the source code that enforces it.

See Also

[tblsize](#)

Definition at line 391 of file Signal.h.

Referenced by Wim(), and Wre().

6.1.5.5 `const size_t Signal::DFTDriver::tblsize = 16384` `[static]`

Number of entries in the tables of sines and cosines.

This **must** be equal to 2^{tblbits} , but there's nothing in the source code that enforces it.

See Also

[tblbits](#)

Definition at line 400 of file Signal.h.

Referenced by Signal::filter(), initialize_costbl(), initialize_sintbl(), and operator()().

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

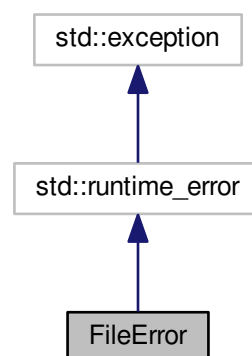
- [Signal.h \(v0.1-10-gf33dea9\)](#)
- [main.cpp \(v0.1-10-gf33dea9\)](#)
- [Signal.cpp \(v0.1-10-gf33dea9\)](#)

6.2 FileError Class Reference

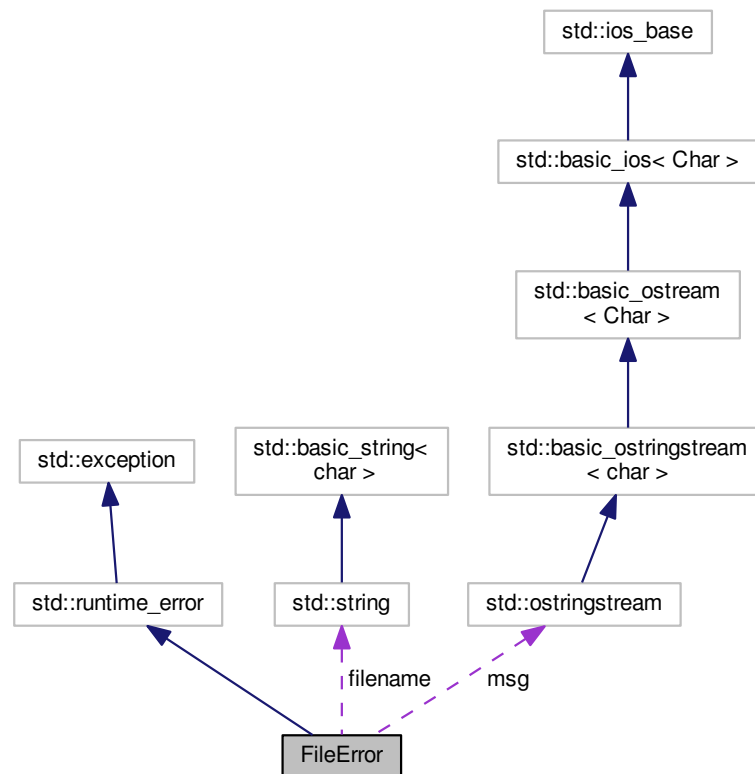
A runtime exception while trying to process a file.

```
#include <Signal.h>
```

Inheritance diagram for FileError:



Collaboration diagram for FileError:



Public Member Functions

- `FileError` (const std::string &fn)
Constructs the exception object from the filename.
- `~FileError` () throw ()
Destructor that does nothing.
- virtual const char * `what` () const throw ()
Gives a description for the error.

Private Attributes

- const std::string `filename`
The name of the file that caused the error.

Static Private Attributes

- static std::ostringstream `msg`
The message that will be displayed if we don't catch the exception.

6.2.1 Detailed Description

A runtime exception while trying to process a file.

Thrown when we cannot read a file, for some reason.

Usage:

```
if (error occurred) throw FileError("badfile.wav");
```

Or:

```
std::string filename;
std::cin >> filename;
// ...
if (error occurred) throw FileError(filename);
```

Definition at line 63 of file Signal.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 FileError::FileError (const std::string & fn) [inline]

Constructs the exception object from the filename.

Parameters

<i>in</i>	<i>fn</i>	A <code>std::string</code> that holds the filename.
-----------	-----------	---

Definition at line 81 of file Signal.h.

6.2.2.2 FileError::~FileError () throw [inline]

Destructor that does nothing.

Needed to prevent the `looser throw specifier error because`, `std::runtime_error::~~runtime_error()` is declared as `throw()`

Definition at line 89 of file Signal.h.

6.2.3 Member Function Documentation

6.2.3.1 virtual const char* FileError::what () const throw [inline], [virtual]

Gives a description for the error.

Updates the `msg` static member with the error message, and returns it as a C string.

Definition at line 96 of file Signal.h.

References `filename`, and `msg`.

6.2.4 Member Data Documentation

6.2.4.1 const std::string FileError::filename [private]

The name of the file that caused the error.

Definition at line 74 of file Signal.h.

Referenced by `what()`.

6.2.4.2 `std::ostringstream FileError::msg` `[static], [private]`

The message that will be displayed if we don't catch the exception.

Must be static, so that we can modify it inside the `what()` `const` function, and read it after the temporary object has been destroyed.

Definition at line 71 of file `Signal.h`.

Referenced by `what()`.

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

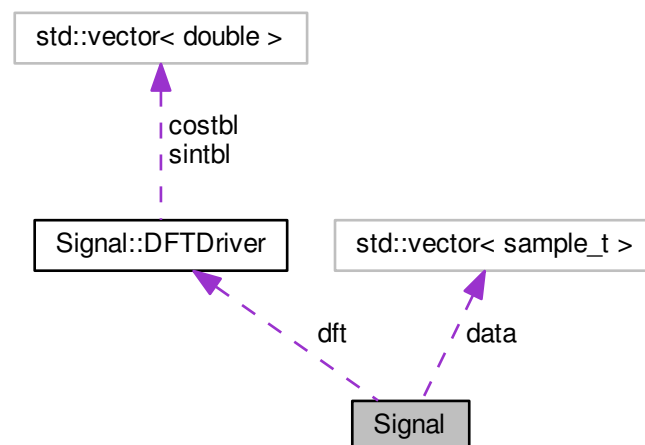
- [Signal.h \(v0.1-10-gf33dea9\)](#)
- [main.cpp \(v0.1-10-gf33dea9\)](#)

6.3 Signal Class Reference

A time- or frequency-domain signal.

```
#include <Signal.h>
```

Collaboration diagram for `Signal`:



Classes

- class [DFTDriver](#)
A class for providing discrete Fourier transform capabilities.

Public Types

- enum [delay_t](#) { `MS`, `SAMPLE` }
This is a type for specifying whether a time interval is given in milliseconds or in samples.
- typedef float [sample_t](#)
The type for holding each signal sample.

- typedef unsigned long [index_t](#)
The type for holding each signal sample index.
- typedef std::vector< [sample_t](#) > [container_t](#)
The type for holding the whole vector of signal samples.

Public Member Functions

- [Signal](#) ()
Constructs an empty signal.
- [Signal](#) (const std::string &filename)
Constructs a signal from an audio file.
- [Signal](#) (const [Signal](#) &other)
Copy-constructor. Constructs a signal as a copy of another.
- [~Signal](#) ()
Frees memory used.
- const [sample_t](#) * [array](#) () const
Returns a pointer to the first sample.
- [index_t](#) [samples](#) () const
Number of samples.
- int [samplerate](#) () const
Sample rate in samples per second.
- [sample_t](#) & [operator\[\]](#) ([index_t](#) index)
Returns a sample.
- const [sample_t](#) & [operator\[\]](#) ([index_t](#) index) const
Returns a "read-only" sample.
- void [set_size](#) ([index_t](#) n)
Changes the number of samples.
- void [set_samplerate](#) (int sr)
Re-samples the signal.
- void [delay](#) ([delay_t](#) t, unsigned long d)
Delays the signal in time.
- void [gain](#) (double g)
Applies gain g to the signal.
- [sample_t](#) [l_inf_norm](#) ()
Gets the ℓ^∞ -norm of the signal.
- void [normalize](#) ()
Normalize the signal according to its ℓ^∞ -norm.
- [Signal](#) & [operator+=](#) ([Signal](#) other)
Adds the other signal to the caller.
- void [filter](#) ([Signal](#) imp_resp)
Convolve the signal with an impulse response.
- void [play](#) (bool sleep=true)
Makes PortAudio playback the audio signal.

Public Attributes

- [index_t](#) [counter](#)
general-purpose variable for external use.

Static Public Attributes

- static [DFTDriver dft](#)
Single instance of the [DFTDriver](#) class.

Private Attributes

- [container_t data](#)
Holds the signal samples.
- int [srate](#)
Signal sample rate in Hertz.

6.3.1 Detailed Description

A time- or frequency-domain signal.

Holds data and provides routines for dealing with time-domain and frequency-domain signals. Currently, all Signals are an array of single-precision floating-point samples. Signals are aware of their sample rates.

Todo Implement "stream" signals, to provide real-time processing.

Definition at line 114 of file Signal.h.

6.3.2 Member Typedef Documentation

6.3.2.1 typedef std::vector<sample_t> Signal::container_t

The type for holding the whole vector of signal samples.

Definition at line 125 of file Signal.h.

6.3.2.2 typedef unsigned long Signal::index_t

The type for holding each signal sample index.

Definition at line 122 of file Signal.h.

6.3.2.3 typedef float Signal::sample_t

The type for holding each signal sample.

Definition at line 119 of file Signal.h.

6.3.3 Member Enumeration Documentation

6.3.3.1 enum Signal::delay_t

This is a type for specifying whether a time interval is given in milliseconds or in samples.

Enumerator

MS Time interval given in milliseconds.

SAMPLE Time interval given in samples.

Definition at line 129 of file Signal.h.

6.3.4 Constructor & Destructor Documentation

6.3.4.1 `Signal::Signal ()` `[inline]`

Constructs an empty signal.

Initializes the signal with no meta-data and no samples. The user needs to specify the sample rate and create samples before using the signal.

Definition at line 139 of file `Signal.h`.

6.3.4.2 `Signal::Signal (const std::string & filename)`

Constructs a signal from an audio file.

Constructs a signal getting the signal data from an audio file. This is done using the `libsndfile` library. The filetypes supported are listed [here](#). WAV is supported, but MP3 is not.

If the given file is stereo, or otherwise multi-channel, just the first channel will be read. (On stereo audio files, this is the left channel.)

The sample rate is extracted from the file's meta-data info.

Parameters

<code>in</code>	<code>filename</code>	Audio file name.
-----------------	-----------------------	------------------

Exceptions

FileError	if file opening/reading fails.
---------------------------	--------------------------------

Definition at line 45 of file `Signal.cpp`.

References `data`, `samples()`, `set_size()`, and `srate`.

6.3.4.3 `Signal::Signal (const Signal & other)` `[inline]`

Copy-constructor. Constructs a signal as a copy of another.

Constructs a signal as a copy of another one. If this signal is not empty, we destroy it.

Parameters

<code>in</code>	<code>other</code>	The signal to be copied from.
-----------------	--------------------	-------------------------------

Definition at line 151 of file `Signal.h`.

6.3.4.4 `Signal::~~Signal ()` `[inline]`

Frees memory used.

Free the pointer to the array of samples.

Definition at line 158 of file `Signal.h`.

6.3.5 Member Function Documentation

6.3.5.1 `const sample_t* Signal::array () const` `[inline]`

Returns a pointer to the first sample.

Sometimes needed for performance reasons. Shouldn't be used to modify the samples.

Returns

a pointer to the first element of a contiguous region of memory that holds the samples.

Definition at line 168 of file Signal.h.

References data.

6.3.5.2 void Signal::delay (delay_t *t*, unsigned long *d*)

Delays the signal in time.

Adds zeroed samples at the beginning of the signal.

If we try to delay a signal by milliseconds, but the signal has no associated sample rate, a warning is emitted, and nothing is done. No exception is thrown.

Parameters

in	<i>t</i>	A delay type element.
in	<i>d</i>	The time interval to be delayed, given in the units specified by <i>t</i> .

Definition at line 88 of file Signal.cpp.

References data, MS, samples(), set_size(), and srate.

6.3.5.3 void Signal::filter (Signal *imp_resp*)

Convolve the signal with an impulse response.

Convolve the signal with the given finite impulse response (FIR).

The algorithm used is the "overlap-and-add", and we use the FFT implemented in the [DFTDriver](#) class to compute each step. We try to do it using the least possible number of DFTs.

Parameters

in	<i>imp_resp</i>	The filter impulse response to be convolved with.
----	-----------------	---

See Also

DFTDriver::operator()

Definition at line 115 of file Signal.cpp.

References data, dft, Signal::DFTDriver::INVERSE, samples(), set_samplerate(), set_size(), srate, and Signal::DFTDriver::tblsize.

6.3.5.4 void Signal::gain (double *g*)

Applies gain *g* to the signal.

This can be useful, for example, to make sure that the signal is within the $[-1, 1]$ range.

Parameters

in	<i>g</i>	The signal gain to be applied.
----	----------	--------------------------------

Definition at line 394 of file Signal.cpp.

References data.

Referenced by normalize().

6.3.5.5 Signal::sample_t Signal::l_inf_norm ()

Gets the ℓ^∞ -norm of the signal.

Take the signal's infinity-norm, which is the maximum absolute value of all the samples of the signal.

Returns

the ℓ^∞ -norm of the signal.

Definition at line 406 of file Signal.cpp.

References data.

Referenced by normalize().

6.3.5.6 void Signal::normalize () [inline]

Normalize the signal according to its ℓ^∞ -norm.

Divide the signal by a constant so that the maximum absolute value of its samples is 1.

Definition at line 231 of file Signal.h.

References gain(), and l_inf_norm().

6.3.5.7 Signal & Signal::operator+=(Signal other)

Adds the *other* signal to the caller.

First, we re-sample *other* into a new temporary signal. Then we increase the caller's size if needed, and finally add the signals sample-by-sample.

Parameters

in	<i>other</i>	The signal to be added to the caller.
----	--------------	---------------------------------------

Returns

a reference to this signal, already added to the *other*.

Definition at line 379 of file Signal.cpp.

References data, samples(), set_samplerate(), set_size(), and srate.

6.3.5.8 sample_t& Signal::operator[] (index_t index) [inline]

Returns a sample.

Gets a sample of the signal. For performance reasons, this method does not check that the given index is valid.

Parameters

in	<i>index</i>	The index of the desired sample. Signal indexes are zero-based.
----	--------------	---

Returns

a reference to the sample.

Definition at line 195 of file Signal.h.

References data.

6.3.5.9 `const sample_t& Signal::operator[] (index_t index) const` `[inline]`

Returns a "read-only" sample.

Just like the "read-write" version, but returns a const reference to a sample.

Parameters

<code>in</code>	<code>index</code>	The index of the desired sample. Signal indexes are zero-based.
-----------------	--------------------	---

Returns

a const reference to the sample.

Definition at line 207 of file `Signal.h`.

References data.

6.3.5.10 `void Signal::play (bool sleep = true)`

Makes PortAudio playback the audio signal.

Creates a PortAudio stream for audio playback of the signal content. If `sleep` is `true`, we wait for the playback to end before returning. (If it's false, the function returns, while playback goes on in the background.)

Parameters

<code>in</code>	<code>sleep</code>	If set to true, the method will only return when the playback ends (that is, when the end of the signal is reached). Otherwise, it returns immediately, and the playback goes on in the background.
-----------------	--------------------	---

Exceptions

<code>std::runtime_error</code>	if any of the PortAudio steps fail (check the source code)
---------------------------------	--

See Also

[callback](#)

Definition at line 298 of file `Signal.cpp`.

References `callback()`, `counter`, `samples()`, and `srate`.

6.3.5.11 `int Signal::samplerate () const` `[inline]`

Sample rate in samples per second.

Returns

the number of samples per second that should be used when playing back the signal.

Definition at line 181 of file `Signal.h`.

References `srate`.

6.3.5.12 `index_t Signal::samples () const` `[inline]`

Number of samples.

Returns

the number of elements inside the vector of samples.

Definition at line 174 of file Signal.h.

References data.

Referenced by delay(), filter(), operator+=(), play(), set_samplerate(), and Signal().

6.3.5.13 void Signal::set_samplerate (int sr)

Re-samples the signal.

Changes the sample rate of the signal. The way it is done, this is equivalent to reconstructing the time-domain signal by linear interpolation, and then re-sampling the continuous-time reconstructed signal at the new sample rate.

Parameters

<code>in</code>	<code>sr</code>	The new sample rate in Hertz.
-----------------	-----------------	-------------------------------

See Also

[srate](#)

Definition at line 356 of file Signal.cpp.

References data, samples(), and srate.

Referenced by filter(), and operator+=().

6.3.5.14 void Signal::set_size (index_t n) [inline]

Changes the number of samples.

Changes the signal length. Allocates more space if we are growing the signal, and deletes the last samples if we are shrinking it. Also initializes any new samples to zero.

Parameters

<code>in</code>	<code>n</code>	The desired signal length.
-----------------	----------------	----------------------------

See Also

`container_t::resize()`

Definition at line 219 of file Signal.h.

References data.

Referenced by delay(), filter(), operator+=(), and Signal().

6.3.6 Member Data Documentation**6.3.6.1 index_t Signal::counter**

general-purpose variable for external use.

Definition at line 183 of file Signal.h.

Referenced by play().

6.3.6.2 `container_t Signal::data` `[private]`

Holds the signal samples.

Definition at line 468 of file `Signal.h`.

Referenced by `array()`, `delay()`, `filter()`, `gain()`, `l_inf_norm()`, `operator+=()`, `operator[]()`, `samples()`, `set_samplerate()`, `set_size()`, and `Signal()`.

6.3.6.3 `Signal::DFTDriver Signal::dft` `[static]`

Single instance of the `DFTDriver` class.

Definition at line 465 of file `Signal.h`.

Referenced by `filter()`.

6.3.6.4 `int Signal::srate` `[private]`

Signal sample rate in Hertz.

Definition at line 469 of file `Signal.h`.

Referenced by `delay()`, `filter()`, `operator+=()`, `play()`, `samplerate()`, `set_samplerate()`, and `Signal()`.

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

- [Signal.h \(v0.1-10-gf33dea9\)](#)
- [main.cpp \(v0.1-10-gf33dea9\)](#)
- [Signal.cpp \(v0.1-10-gf33dea9\)](#)

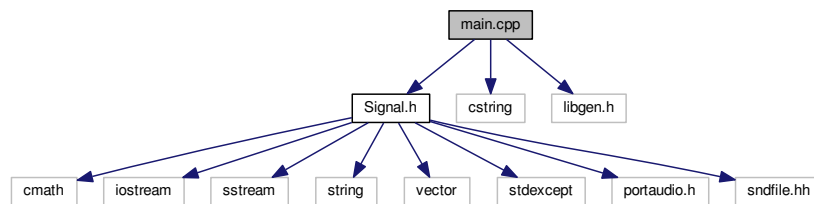
Chapter 7

File Documentation

7.1 main.cpp File Reference

```
#include "Signal.h"  
#include <cstring>  
#include <libgen.h>
```

Include dependency graph for main.cpp:



Macros

- `#define ATFA_DIR`
Macro for getting the path to the project directory from cmake.

Functions

- `void portaudio_init (bool list_devices=false)`
Initialize PortAudio.
- `void portaudio_end ()`
Close PortAudio.
- `int main (int argc, char *argv[])`
main() function.

Variables

- `char static_filename [] = __FILE__`
- `char static_dirname [] = __FILE__`
- `char static_projdirname [] = __FILE__`

7.1.1 Detailed Description

Holds the `main()` function and other routines.

Author

Pedro Angelo Medeiros Fonini

Definition in file `main.cpp`.

7.1.2 Macro Definition Documentation

7.1.2.1 `#define ATFA_DIR`

Value:

```
(static_cast<const char *>( \
    std::strcpy(static_dirname, dirname(static_filename)), \
    std::strcpy(static_filename, __FILE__), \
    std::strcpy(static_projdirname, dirname(static_dirname)), \
    static_projdirname \
))
```

Macro for getting the path to the project directory from cmake.

Should be passed from `CMakeLists.txt`, but if it's not, we try to deduce it from the `__FILE__` macro

Definition at line 35 of file `main.cpp`.

7.1.3 Function Documentation

7.1.3.1 `int main (int argc, char * argv[])`

`main()` function.

No command-line parameters.

This function:

1. Prints version info
2. Creates two `Signals`, `sound_me` and `sound_other` from two input files.
3. Delays the second.
4. Creates an impulse response.
5. Filters `sound_me` according to the impulse response and adds it to the second, delayed.
6. Initializes a PortAudio session, plays the resulting sound, and closes PortAudio.

Parameters

in	<i>argc</i>	argument count (unused)
in	<i>argv</i>	argument values (unused)

Returns

0 if no errors

Definition at line 136 of file `main.cpp`.

7.1.3.2 void portaudio_end ()

Close PortAudio.

Ends a PortAudio session.

Exceptions

<code>std::runtime_error</code>	if PortAudio closing fails.
---------------------------------	-----------------------------

See Also

`portaudio_init()`

Definition at line 108 of file main.cpp.

7.1.3.3 void portaudio_init (bool *list_devices* = false)

Initialize PortAudio.

Initializes a PortAudio session. Also prints out a list of available devices that PortAudio sees., if requested.

Parameters

<code>in</code>	<code>list_devices</code>	Whether or not to print the device list.
-----------------	---------------------------	--

Exceptions

<code>std::runtime_error</code>	if PortAudio initialization fails.
---------------------------------	------------------------------------

See Also

`portaudio_end()`

Definition at line 66 of file main.cpp.

7.1.4 Variable Documentation

7.1.4.1 char static_dirname[] = __FILE__

Definition at line 27 of file main.cpp.

7.1.4.2 char static_filename[] = __FILE__

Definition at line 26 of file main.cpp.

7.1.4.3 char static_projdirname[] = __FILE__

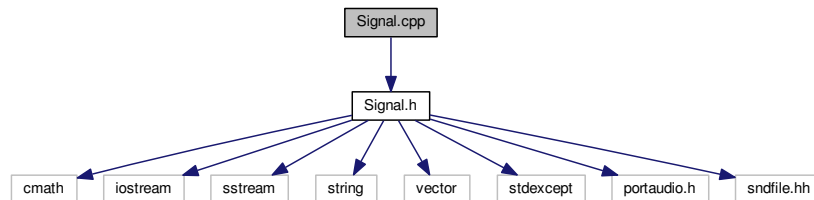
Definition at line 28 of file main.cpp.

7.2 README.md File Reference

7.3 Signal.cpp File Reference

```
#include "Signal.h"
```

Include dependency graph for Signal.cpp:



Functions

- static int [callback](#) (const void *in_buf, void *out_buf, unsigned long frames_per_buf, const PaStreamCallbackTimeInfo *time_info, PaStreamCallbackFlags status_flags, void *user_data)
PortAudio callback function.

7.3.1 Detailed Description

Holds the implementation of the [Signal](#) class.

Author

Pedro Angelo Medeiros Fonini

Definition in file [Signal.cpp](#).

7.3.2 Function Documentation

7.3.2.1 static int [callback](#) (const void * *in_buf*, void * *out_buf*, unsigned long *frames_per_buf*, const PaStreamCallbackTimeInfo * *time_info*, PaStreamCallbackFlags *status_flags*, void * *user_data*) [static]

PortAudio callback function.

The PortAudio library implements the stream playback using *callback* functions. These functions get called at interrupt time whenever PortAudio needs a new buffer of samples to pass to the hardware. Callback functions should not take long to return; in particular, they should **not** throw or catch exceptions, or do I/O.

This callback function just reads a given [Signal](#) and passes its samples to PortAudio.

Parameters

in	<i>in_buf</i>	Pointer to a buffer of samples retrieved from an input audio device. This parameter is unused because we're not reading from any device.
out	<i>out_buf</i>	Pointer to a buffer where the callback function will store samples to be given to an output audio device.

in	<i>frames_per_buf</i>	Number of samples we will store in the buffer. This is actually the number of <i>frames</i> , which in turn is equal to number of samples because we're working with mono-channel signals.
in	<i>time_info</i>	PortAudio time information. (unused)
in	<i>status_flags</i>	PortAudio status flags. (unused)
in, out	<i>user_data</i>	Pointer to an arbitrary data-holder passed to the stream open function. We use this to get the signal samples, and to keep track of where in the signal we are (using the Signal::counter auxiliary member).

See Also[Signal::play](#)

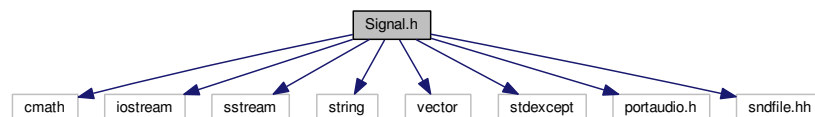
Definition at line 258 of file Signal.cpp.

Referenced by [Signal::play\(\)](#).

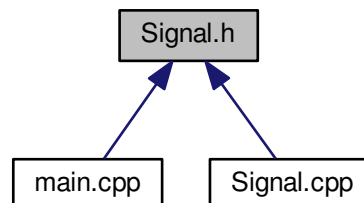
7.4 Signal.h File Reference

```
#include <cmath>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
#include <stdexcept>
#include <portaudio.h>
#include <sndfile.hh>
```

Include dependency graph for Signal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [FileError](#)
A runtime exception while trying to process a file.
- class [Signal](#)
A time- or frequency-domain signal.
- class [Signal::DFTDriver](#)
A class for providing discrete Fourier transform capabilities.

Functions

- [Signal operator+](#) ([Signal](#) lhs, const [Signal](#) &rhs)
Adds two signals.

Variables

- static const void *const [NULL](#)
Null pointer.

7.4.1 Detailed Description

Holds the interface to the [Signal](#) class.

Author

Pedro Angelo Medeiros Fonini

Definition in file [Signal.h](#).

7.4.2 Function Documentation

7.4.2.1 [Signal operator+](#) ([Signal](#) lhs, const [Signal](#) & rhs) [inline]

Adds two signals.

See Also

[Signal::operator+=](#)

Definition at line 477 of file [Signal.h](#).

7.4.3 Variable Documentation

7.4.3.1 `const void* const NULL` [static]

Initial value:

```
= ((void *)0)
```

```
static const double TAU = 6.283185307179586477
```


Null pointer.

Replaces the standard null pointer in case it's not defined.

Definition at line 38 of file Signal.h.

Index

~FileError
 FileError, [19](#)
~Signal
 Signal, [23](#)

ATFA_DIR
 main.cpp, [30](#)
array
 Signal, [23](#)

bits
 Signal::DFTDriver, [15](#)
br
 Signal::DFTDriver, [13](#)

callback
 Signal.cpp, [32](#)
container_t
 Signal, [22](#)
costbl
 Signal::DFTDriver, [15](#)
counter
 Signal, [27](#)

DIRECT
 Signal::DFTDriver, [13](#)
DFTDriver
 Signal::DFTDriver, [13](#)
data
 Signal, [27](#)
delay
 Signal, [24](#)
delay_t
 Signal, [22](#)
dft
 Signal, [28](#)
dir_t
 Signal::DFTDriver, [13](#)

FileError, [17](#)
 ~FileError, [19](#)
 FileError, [19](#)
 FileError, [19](#)
 filename, [19](#)
 msg, [19](#)
 what, [19](#)
filename
 FileError, [19](#)
filter
 Signal, [24](#)

gain
 Signal, [24](#)

INVERSE
 Signal::DFTDriver, [13](#)
index_t
 Signal, [22](#)
initialize_costbl
 Signal::DFTDriver, [13](#)
initialize_sintbl
 Signal::DFTDriver, [14](#)

l_inf_norm
 Signal, [24](#)

MS
 Signal, [22](#)
main
 main.cpp, [30](#)
main.cpp
 ATFA_DIR, [30](#)
 main, [30](#)
 portaudio_end, [30](#)
 portaudio_init, [31](#)
 static_dirname, [31](#)
 static_filename, [31](#)
 static_projdirname, [31](#)
main.cpp(v0.1-10-gf33dea9), [29](#)
msg
 FileError, [19](#)

NULL
 Signal.h, [34](#)
normalize
 Signal, [25](#)

operator()
 Signal::DFTDriver, [14](#)
operator+
 Signal.h, [34](#)
operator+=
 Signal, [25](#)

play
 Signal, [26](#)
portaudio_end
 main.cpp, [30](#)
portaudio_init
 main.cpp, [31](#)

README.md(v0.1-10-gf33dea9), [31](#)

- SAMPLE
 - Signal, [22](#)
- sample_t
 - Signal, [22](#)
- samplerate
 - Signal, [26](#)
- samples
 - Signal, [26](#)
- set_samplerate
 - Signal, [27](#)
- set_size
 - Signal, [27](#)
- Signal, [20](#)
 - ~Signal, [23](#)
 - array, [23](#)
 - container_t, [22](#)
 - counter, [27](#)
 - data, [27](#)
 - delay, [24](#)
 - delay_t, [22](#)
 - dft, [28](#)
 - filter, [24](#)
 - gain, [24](#)
 - index_t, [22](#)
 - l_inf_norm, [24](#)
 - MS, [22](#)
 - normalize, [25](#)
 - operator+=, [25](#)
 - play, [26](#)
 - SAMPLE, [22](#)
 - sample_t, [22](#)
 - samplerate, [26](#)
 - samples, [26](#)
 - set_samplerate, [27](#)
 - set_size, [27](#)
 - Signal, [23](#)
 - srate, [28](#)
- Signal.cpp
 - callback, [32](#)
- Signal.cpp(v0.1-10-gf33dea9), [32](#)
- Signal.h
 - NULL, [34](#)
 - operator+, [34](#)
- Signal.h(v0.1-10-gf33dea9), [33](#)
- Signal::DFTDriver
 - DIRECT, [13](#)
 - INVERSE, [13](#)
- Signal::DFTDriver, [11](#)
 - bits, [15](#)
 - br, [13](#)
 - costbl, [15](#)
 - DFTDriver, [13](#)
 - dir_t, [13](#)
 - initialize_costbl, [13](#)
 - initialize_sintbl, [14](#)
 - operator(), [14](#)
 - sintbl, [16](#)
 - tblbits, [16](#)
 - tblsize, [16](#)
 - Wim, [14](#)
 - Wre, [15](#)
- sintbl
 - Signal::DFTDriver, [16](#)
- srate
 - Signal, [28](#)
- static_dirname
 - main.cpp, [31](#)
- static_filename
 - main.cpp, [31](#)
- static_projdirname
 - main.cpp, [31](#)
- tblbits
 - Signal::DFTDriver, [16](#)
- tblsize
 - Signal::DFTDriver, [16](#)
- what
 - FileError, [19](#)
- Wim
 - Signal::DFTDriver, [14](#)
- Wre
 - Signal::DFTDriver, [15](#)