

## Ambiente de Teste para Filtros Adaptativos

Generated by Doxygen 1.8.6

Tue Jan 21 2014 14:12:15



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
6.1	Signal::DFTDriver Class Reference . . . . .	11
6.1.1	Detailed Description . . . . .	12
6.1.2	Member Enumeration Documentation . . . . .	12
6.1.2.1	dir_t . . . . .	12
6.1.3	Constructor & Destructor Documentation . . . . .	12
6.1.3.1	DFTDriver . . . . .	12
6.1.4	Member Function Documentation . . . . .	13
6.1.4.1	br . . . . .	13
6.1.4.2	operator() . . . . .	13
6.1.4.3	Wim . . . . .	14
6.1.4.4	Wre . . . . .	14
6.1.5	Member Data Documentation . . . . .	14
6.1.5.1	bits . . . . .	15
6.1.5.2	costbl . . . . .	15
6.1.5.3	sintbl . . . . .	15
6.1.5.4	tblbits . . . . .	15
6.1.5.5	tblsize . . . . .	15
6.2	FileError Class Reference . . . . .	16
6.2.1	Detailed Description . . . . .	18

6.2.2	Constructor & Destructor Documentation . . . . .	18
6.2.2.1	FileError . . . . .	18
6.2.2.2	~FileError . . . . .	18
6.2.3	Member Function Documentation . . . . .	18
6.2.3.1	what . . . . .	18
6.2.4	Member Data Documentation . . . . .	18
6.2.4.1	filename . . . . .	18
6.2.4.2	msg . . . . .	19
6.3	Signal Class Reference . . . . .	19
6.3.1	Detailed Description . . . . .	21
6.3.2	Member Typedef Documentation . . . . .	21
6.3.2.1	container_t . . . . .	21
6.3.2.2	index_t . . . . .	21
6.3.2.3	sample_t . . . . .	21
6.3.3	Member Enumeration Documentation . . . . .	21
6.3.3.1	delay_t . . . . .	21
6.3.4	Constructor & Destructor Documentation . . . . .	22
6.3.4.1	Signal . . . . .	22
6.3.4.2	Signal . . . . .	22
6.3.4.3	Signal . . . . .	22
6.3.4.4	~Signal . . . . .	22
6.3.5	Member Function Documentation . . . . .	22
6.3.5.1	array . . . . .	22
6.3.5.2	delay . . . . .	23
6.3.5.3	filter . . . . .	23
6.3.5.4	gain . . . . .	23
6.3.5.5	l_inf_norm . . . . .	24
6.3.5.6	normalize . . . . .	24
6.3.5.7	operator+= . . . . .	24
6.3.5.8	operator[] . . . . .	24
6.3.5.9	operator[] . . . . .	25
6.3.5.10	play . . . . .	25
6.3.5.11	samplerate . . . . .	25
6.3.5.12	samples . . . . .	26
6.3.5.13	set_samplerate . . . . .	26
6.3.5.14	set_size . . . . .	26
6.3.6	Member Data Documentation . . . . .	27
6.3.6.1	counter . . . . .	27
6.3.6.2	data . . . . .	27
6.3.6.3	dft . . . . .	27

6.3.6.4	srate	27
<b>7</b>	<b>File Documentation</b>	<b>29</b>
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	playsig	31
7.1.3.3	portaudio_end	31
7.1.3.4	portaudio_init	31
7.1.4	Variable Documentation	32
7.1.4.1	static_dirname	32
7.1.4.2	static_filename	32
7.1.4.3	static_projdirname	32
7.2	README.md File Reference	32
7.3	Signal.cpp File Reference	32
7.3.1	Detailed Description	32
7.3.2	Function Documentation	33
7.3.2.1	callback	33
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	35
7.4.3.1	NULL	35
	<b>Index</b>	<b>36</b>



# Chapter 1

## Main Page

Projeto Final de Graduação

Para compilar:

```
cd build      # ../pf/build
./config.sh
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`.





## Chapter 2

# Todo List

### Class [Signal](#)

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

### Member [Signal::DFTDriver::DFTDriver \(\)](#)

make the table be inside a `std::vector`, instead of a built-in array, so that we can use the solution provided [here](#).  
An alternative is documented [here](#).

this static member should be initialized only once, even though multiple instances of the `DFTDriver` might be created (that is, static members shouldn't be initialized in the constructor). (perhaps we could have a private `bool` telling wether it's been already initialized)



## 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 . . . . .	16
Signal . . . . .	19



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">Signal::DFTDriver</a>	A class for providing discrete Fourier transform capabilities . . . . .	<a href="#">11</a>
<a href="#">FileError</a>	A runtime exception while trying to process a file . . . . .	<a href="#">16</a>
<a href="#">Signal</a>	A time- or frequency-domain signal . . . . .	<a href="#">19</a>



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">main.cpp</a>	.....	29
<a href="#">Signal.cpp</a>	.....	32
<a href="#">Signal.h</a>	.....	33





## Chapter 6

# Class Documentation

### 6.1 Signal::DFTDriver Class Reference

A class for providing discrete Fourier transform capabilities.

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

#### 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 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 double [sintbl](#) [[tblsize](#)]

*Table of sines.*

- static double [costbl](#) [[tblsize](#)]

*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.

Computes and initializes the table of sines and cosines. After this initialization, the entries of the table shouldn't be modified.

**Todo** make the table be inside a `std::vector`, instead of a built-in array, so that we can use the solution provided [here](#). An alternative is documented [here](#).

**Todo** this static member should be initialized only once, even though multiple instances of the [DFTDriver](#) might be created (that is, static members shouldn't be initialized in the constructor). (perhaps we could have a private bool telling wether it's been already initialized)

#### See Also

[sintbl](#)  
[costbl](#)

Definition at line 430 of file `Signal.h`.

References `costbl`, `sintbl`, and `tblsize`.

### 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 <b>must</b> 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** `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 compelx signal on which the FFT will act.
----------------	-----------	--

<i>in, out</i>	<i>im</i>	Imaginary part.
<i>in</i>	<i>direction</i>	Wether this is a direct or inverse DFT.

Definition at line 428 of file Signal.cpp.

References `bits`, `br()`, `DIRECT`, `INVERSE`, `tblsize`, `Wim()`, and `Wre()`.

#### 6.1.4.3 `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 <a href="#">Wre</a> .
-----------	----------	----------------------------------

##### 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()()`.

#### 6.1.4.4 `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

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

##### 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 double Signal::DFTDriver::costbl [static], [private]

Table of cosines.

See Also

[sintbl](#)

Definition at line 456 of file Signal.h.

Referenced by DFTDriver(), and Wre().

#### 6.1.5.3 double Signal::DFTDriver::sintbl [static], [private]

Table of sines.

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

See Also

[costbl](#)

Definition at line 450 of file Signal.h.

Referenced by DFTDriver(), and 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  $2^{\text{tblbits}}$ , but there's nothing in the source code that enforces it.

#### See Also

[tblbits](#)

Definition at line 400 of file `Signal.h`.

Referenced by `DFTDriver()`, `Signal::filter()`, and `operator()()`.

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

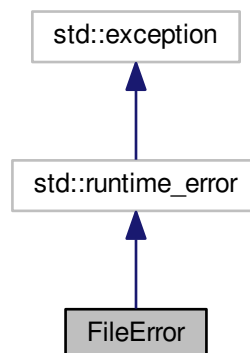
- [Signal.h \(aaa.versao.teste.1-46-g96cebde\)](#)
- [main.cpp \(aaa.versao.teste.1-46-g96cebde\)](#)
- [Signal.cpp \(aaa.versao.teste.1-46-g96cebde\)](#)

## 6.2 FileError Class Reference

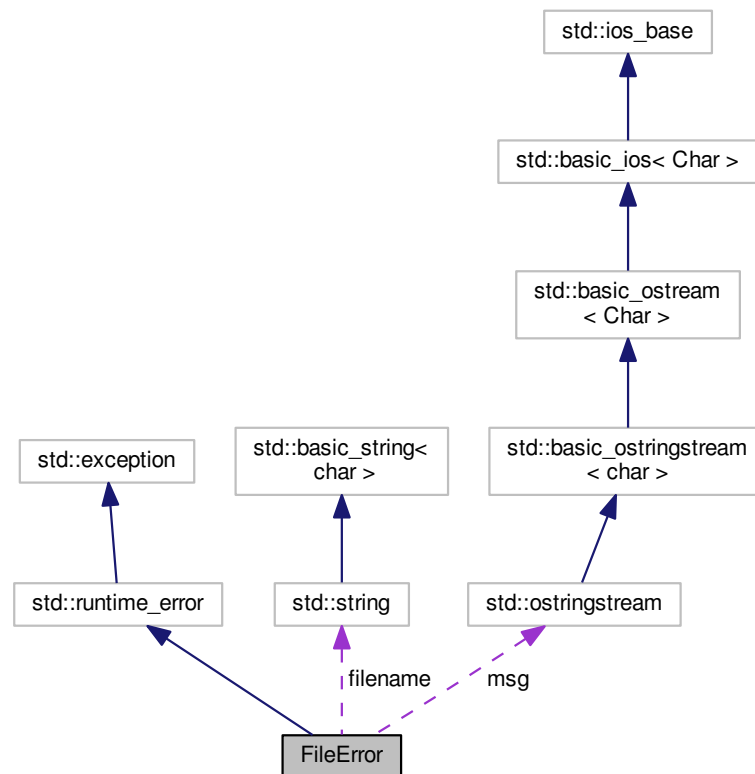
A runtime exception while trying to process a file.

```
#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::ostream 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:

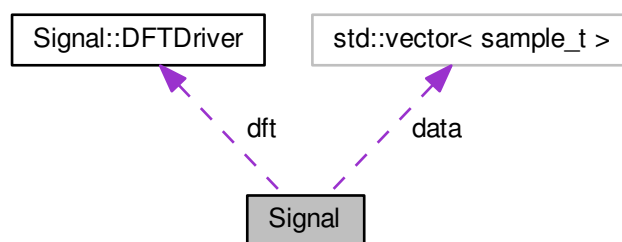
- [Signal.h](#) ([aaa.versao.teste.1-46-g96cebde](#))
- [main.cpp](#) ([aaa.versao.teste.1-46-g96cebde](#))

## 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  $\ell^\infty$ -norm of the signal.*
- void [normalize](#) ()  
*Normalize the signal according to its  $\ell^\infty$ -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

<a href="#">FileError</a>	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**

<i>in</i>	<i>t</i>	A delay type element.
<i>in</i>	<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.

Referenced by main().

**6.3.5.3 void Signal::filter ( Signal *imp\_resp* )**

Convolves the signal with an impulse response.

Convolves 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**

<i>in</i>	<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.

Referenced by main().

**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**

<i>in</i>	<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  $\ell^\infty$ -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  $\ell^\infty$ -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  $\ell^\infty$ -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().

Referenced by main().

#### 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

<i>in</i>	<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

<code>in</code>	<code>index</code>	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`.

Referenced by `playsig()`.

#### 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()`, `main()`, `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 463 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 460 of file `Signal.h`.

Referenced by `filter()`.

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

Signal sample rate in Hertz.

Definition at line 464 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 \(aaa.versao.teste.1-46-g96cebde\)](#)
- [main.cpp \(aaa.versao.teste.1-46-g96cebde\)](#)
- [Signal.cpp \(aaa.versao.teste.1-46-g96cebde\)](#)



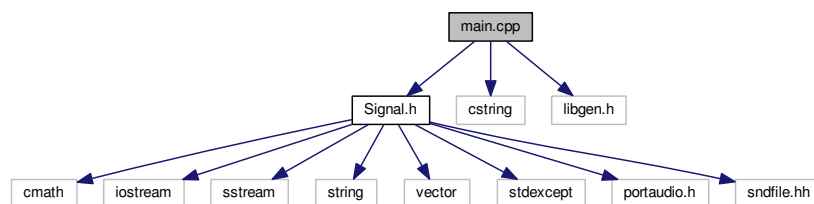
## Chapter 7

# 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.*
- void `playsig` (Signal s)  
*Playback signal.*
- 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`.

Referenced by `main()`.

### 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 144 of file main.cpp.

References ATFA\_DIR, Signal::delay(), Signal::filter(), Signal::MS, Signal::normalize(), playsig(), and Signal::set\_size().

## 7.1.3.2 void playsig ( Signal s )

Playback signal.

Wrapper function for [Signal::play](#) that handles PortAudio.

Definition at line 119 of file main.cpp.

References Signal::play(), portaudio\_end(), and portaudio\_init().

Referenced by main().

## 7.1.3.3 void portaudio\_end ( )

Close PortAudio.

Ends a PortAudio session.

## Exceptions

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

## See Also

[portaudio\\_init\(\)](#)

Definition at line 106 of file main.cpp.

Referenced by playsig().

## 7.1.3.4 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

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

## Exceptions

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

## See Also

[portaudio\\_end\(\)](#)

Definition at line 64 of file main.cpp.

Referenced by playsig().

### 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__`

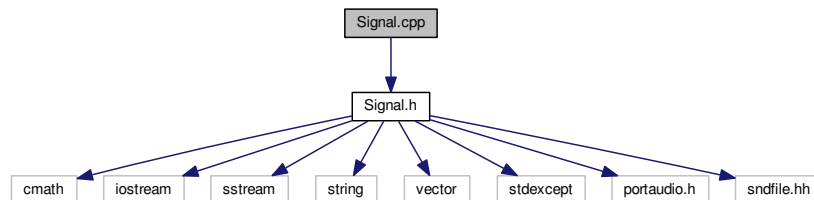
Definition at line 28 of file main.cpp.

## 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 <a href="#">Signal::counter</a> auxiliary member).

#### See Also

[Signal::play](#)

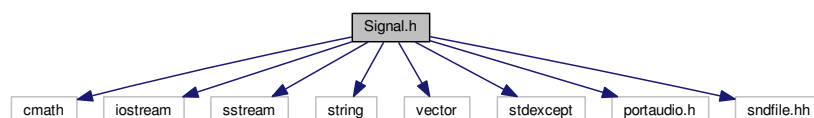
Definition at line 258 of file `Signal.cpp`.

Referenced by `Signal::play()`.

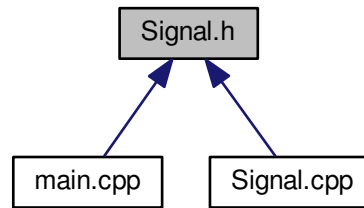
## 7.4 Signal.h File Reference

```
#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 472 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, [18](#)
- ~Signal
  - Signal, [22](#)
- ATFA\_DIR
  - main.cpp, [30](#)
- array
  - Signal, [22](#)
- bits
  - Signal::DFTDriver, [14](#)
- br
  - Signal::DFTDriver, [13](#)
- callback
  - Signal.cpp, [33](#)
- container\_t
  - Signal, [21](#)
- costbl
  - Signal::DFTDriver, [15](#)
- counter
  - Signal, [27](#)
- DIRECT
  - Signal::DFTDriver, [12](#)
- DFTDriver
  - Signal::DFTDriver, [12](#)
- data
  - Signal, [27](#)
- delay
  - Signal, [23](#)
- delay\_t
  - Signal, [21](#)
- dft
  - Signal, [27](#)
- dir\_t
  - Signal::DFTDriver, [12](#)
- FileError, [16](#)
  - ~FileError, [18](#)
  - FileError, [18](#)
  - FileError, [18](#)
  - filename, [18](#)
  - msg, [18](#)
  - what, [18](#)
- filename
  - FileError, [18](#)
- filter
  - Signal, [23](#)
- gain
  - Signal, [23](#)
- INVERSE
  - Signal::DFTDriver, [12](#)
- index\_t
  - Signal, [21](#)
- l\_inf\_norm
  - Signal, [24](#)
- MS
  - Signal, [21](#)
- main
  - main.cpp, [30](#)
- main.cpp
  - ATFA\_DIR, [30](#)
  - main, [30](#)
  - playsig, [31](#)
  - portaudio\_end, [31](#)
  - portaudio\_init, [31](#)
  - static\_dirname, [32](#)
  - static\_filename, [32](#)
  - static\_projdirname, [32](#)
- main.cpp(aaa.versao.teste.1-46-g96cebde), [29](#)
- msg
  - FileError, [18](#)
- NULL
  - Signal.h, [35](#)
- normalize
  - Signal, [24](#)
- operator()
  - Signal::DFTDriver, [13](#)
- operator+
  - Signal.h, [34](#)
- operator+=
  - Signal, [24](#)
- play
  - Signal, [25](#)
- playsig
  - main.cpp, [31](#)
- portaudio\_end
  - main.cpp, [31](#)
- portaudio\_init
  - main.cpp, [31](#)
- README.md(aaa.versao.teste.1-46-g96cebde), [32](#)

SAMPLE  
    Signal, 21  
sample\_t  
    Signal, 21  
samplerate  
    Signal, 25  
samples  
    Signal, 26  
set\_samplerate  
    Signal, 26  
set\_size  
    Signal, 26  
Signal, 19  
    ~Signal, 22  
    array, 22  
    container\_t, 21  
    counter, 27  
    data, 27  
    delay, 23  
    delay\_t, 21  
    dft, 27  
    filter, 23  
    gain, 23  
    index\_t, 21  
    l\_inf\_norm, 24  
    MS, 21  
    normalize, 24  
    operator+=, 24  
    play, 25  
    SAMPLE, 21  
    sample\_t, 21  
    samplerate, 25  
    samples, 26  
    set\_samplerate, 26  
    set\_size, 26  
    Signal, 22  
    srate, 27  
Signal.cpp  
    callback, 33  
Signal.cpp(aaa.versao.teste.1-46-g96cebde), 32  
Signal.h  
    NULL, 35  
    operator+, 34  
Signal.h(aaa.versao.teste.1-46-g96cebde), 33  
Signal::DFTDriver  
    DIRECT, 12  
    INVERSE, 12  
Signal::DFTDriver, 11  
    bits, 14  
    br, 13  
    costbl, 15  
    DFTDriver, 12  
    dir\_t, 12  
    operator(), 13  
    sintbl, 15  
    tblbits, 15  
    tblsize, 15  
    Wim, 14  
    Wre, 14  
sintbl  
    Signal::DFTDriver, 15  
srate  
    Signal, 27  
static\_dirname  
    main.cpp, 32  
static\_filename  
    main.cpp, 32  
static\_projdirname  
    main.cpp, 32  
tblbits  
    Signal::DFTDriver, 15  
tblsize  
    Signal::DFTDriver, 15  
what  
    FileError, 18  
Wim  
    Signal::DFTDriver, 14  
Wre  
    Signal::DFTDriver, 14