OSP

2018C

Generated by Doxygen 1.8.15

Chapter 1

README

Chapter 2

README

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

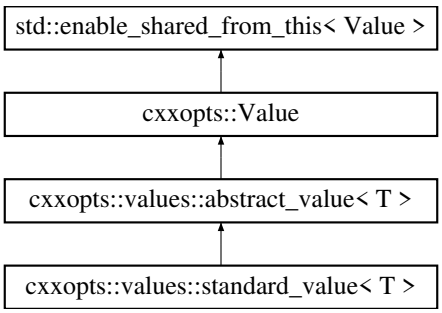
Here is a list of all documented files with brief descriptions:

Chapter 6

Class Documentation

6.1 cxxopts::values::abstract_value< T > Class Template Reference

Inheritance diagram for cxxopts::values::abstract_value< T >:



Public Member Functions

- **abstract_value** (T *t)
- **abstract_value** (const abstract_value &rhs)
- void **parse** (const std::string &text) const
- bool **is_container** () const
- void **parse** () const
- bool **has_default** () const
- bool **has_implicit** () const
- std::shared_ptr< Value > **default_value** (const std::string &value)
- std::shared_ptr< Value > **implicit_value** (const std::string &value)
- std::string **get_default_value** () const
- std::string **get_implicit_value** () const
- bool **is_boolean** () const
- const T & **get** () const

Protected Attributes

- std::shared_ptr< T > **m_result**
- T * **m_store**
- bool **m_default** = false
- bool **m_implicit** = false
- std::string **m_default_value**
- std::string **m_implicit_value**

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

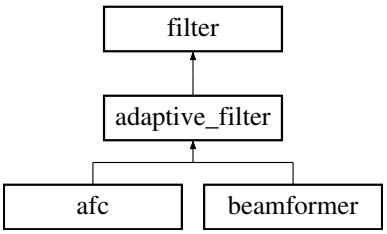
- OSP/include/cxxopts.hpp

6.2 adaptive_filter Class Reference

Adaptive Filter Class.

`#include <adaptive_filter.hpp>`

Inheritance diagram for adaptive_filter:



Public Member Functions

- adaptive_filter (float *adaptive_filter_taps, size_t adaptive_filter_tap_len, size_t max_frame_size, int adaptation_type, float mu, float delta, float rho, float alpha, float beta, float p, float c, float power_estimate)

  Adaptive filter constructor.
- ~adaptive_filter ()

  Adaptive filter destructor.
- int update_taps (float *u_ref, float *e_ref, size_t ref_size)

  To update the taps of this adaptive filter based on the reference signals u_ref and e_ref.
- size_t get_max_frame_size ()

  Getting the maximum frame size.
- void get_params (float &mu, float &rho, float &delta, float &alpha, float &beta, float &p, float &c, int &adaptation_type)

  Getting all parameters from this adaptive filter.
- void set_params (float mu, float rho, float delta, float alpha, float beta, float p, float c, int adaptation_type)

  Setting all parameters from this adaptive filter.

Protected Member Functions

- int get_adaptation_type ()

  A function to get the adaptation type.
- void get_step_size_weights_IPNLMS (float *taps, float *step_size_weights, float alpha, float beta, float delta, size_t tap_len)

  A function computing the step size control matrix for IPNLMS-l_0.
- void get_step_size_weights_SLMS (float *taps, float *step_size_weights, float p, float c, size_t tap_len)

  A function computing the step size control matrix for SLMS.

6.2.1 Detailed Description

Adaptive Filter Class.

This adaptive filter class implements several popular LMS-based algorithms including Modified LMS [Greenberg, 1998], IPNLMS-l_0 [Paleologu et al., 2010] and SLMS [Lee et al., 2017].

6.2.2 Constructor & Destructor Documentation

6.2.2.1 adaptive_filter()

```
adaptive_filter::adaptive_filter (
            float * adaptive_filter_taps,
            size_t adaptive_filter_tap_len,
            size_t max_frame_size,
            int adaptation_type,
            float mu,
            float delta,
            float rho,
            float alpha,
            float beta,
            float p,
            float c,
            float power_estimate )  [explicit]
```

Adaptive filter constructor.

Parameters

in	*adaptive_filter_taps*	The initial filter taps for adaptive filter
in	*adaptive_filter_tap_len*	The number of filter taps of adaptive filter
in	*max_frame_size*	The maximum processing frame size in adaptive filter
in	*adaptation_type*	-1: 0 y_hat, 0: stop adaptation, 1: Modified LMS, 2: IPNLMS-l_0, 3: SLMS
in	*mu*	The gradient descent step size (learning rate) for LMS-based algorithms
in	*delta*	A small positive number to prevent dividing zero
in	*rho*	The forgetting factor for power estimate
in	*alpha*	A number between -1 to 1 for different degrees of sparsity in IPNLMS-l_0
in	*beta*	A number between 0 to 500 for different degrees of sparsity in IPNLMS-l_0
in	*p*	A number between 0 to 2 for fitting different degrees of sparsity in SLMS
in	*c*	A small positive number for preventing stagnation in SLMS
in	*power_estimate*	An initial power estimate for adaptation

6.2.3 Member Function Documentation

6.2.3.1 get_adaptation_type()

```
int adaptive_filter::get_adaptation_type ( )  [protected]
```

A function to get the adaptation type.

Returns

Adaptation type

6.2.3.2 get_max_frame_size()

```
size_t adaptive_filter::get_max_frame_size ( )
```

Getting the maximum frame size.

Returns

Maximum frame size

6.2.3.3 get_params()

```
void adaptive_filter::get_params (
            float & mu,
            float & rho,
            float & delta,
            float & alpha,
            float & beta,
            float & p,
            float & c,
            int & adaptation_type )
```

Getting all parameters from this adaptive filter.

Parameters

out	*mu*	The gradient descent step size (learning rate) for LMS-based algorithms
out	*rho*	The forgetting factor for power estimate
out	*delta*	A small positive number to prevent dividing zero
out	*alpha*	A number between -1 to 1 for different degrees of sparsity in IPNLMS-l_0
out	*beta*	A number between 0 to 500 for different degrees of sparsity in IPNLMS-l_0
out	*p*	A number between 0 to 2 for fitting different degrees of sparsity in SLMS
out	*c*	A small positive number for preventing stagnation in SLMS
out	*adaptation_type*	-1: 0 y_hat, 0: stop adaptation, 1: Modified LMS, 2: IPNLMS-l_0, 3: SLMS

6.2.3.4 get_step_size_weights_IPNLMS()

```
void adaptive_filter::get_step_size_weights_IPNLMS (
            float * taps,
            float * step_size_weights,
            float alpha,
            float beta,
            float delta,
            size_t tap_len )  [protected]
```

A function computing the step size control matrix for IPNLMS-l_0.

Parameters

in	*taps*	The current filter taps of the adaptive filter
out	*step_size_weights*	The step size control matrix (it is an 1-D array due to the diagonal matrix)
in	*alpha*	A number between -1 to 1 for different degrees of sparsity in IPNLMS-l_0
in	*beta*	A number between 0 to 500 for different degrees of sparsity in IPNLMS-l_0
in	*delta*	A small positive number to prevent dividing zero
in	*tap_len*	The number of taps of the adaptive filter

6.2.3.5 get_step_size_weights_SLMS()

```
void adaptive_filter::get_step_size_weights_SLMS (
            float * taps,
            float * step_size_weights,
            float p,
            float c,
            size_t tap_len )  [protected]
```

A function computing the step size control matrix for SLMS.

Parameters

in	*taps*	The current filter taps of the adaptive filter
out	*step_size_weights*	The step size control matrix (it is an 1-D array due to the diagonal matrix)
in	*p*	A number between 0 to 2 for fitting different degrees of sparsity in SLMS
in	*c*	A small positive number for preventing stagnation in SLMS
in	*tap_len*	The number of taps of the adaptive filter

6.2.3.6 set_params()

```
void adaptive_filter::set_params (
            float mu,
```

```
            float rho,
            float delta,
            float alpha,
            float beta,
            float p,
            float c,
            int adaptation_type )
```

Setting all parameters from this adaptive filter.

Parameters

in	*mu*	The gradient descent step size (learning rate) for LMS-based algorithms
in	*rho*	The forgetting factor for power estimate
in	*delta*	A small positive number to prevent dividing zero
in	*alpha*	A number between -1 to 1 for different degrees of sparsity in IPNLMS-l_0
in	*beta*	A number between 0 to 500 for different degrees of sparsity in IPNLMS-l_0
in	*p*	A number between 0 to 2 for fitting different degrees of sparsity in SLMS
in	*c*	A small positive number for preventing stagnation in SLMS
in	*adaptation_type*	-1: 0 y_hat, 0: stop adaptation, 1: Modified LMS, 2: IPNLMS-l_0, 3: SLMS

6.2.3.7 update_taps()

```
int adaptive_filter::update_taps (
            float * u_ref,
            float * e_ref,
            size_t ref_size )
```

To update the taps of this adaptive filter based on the reference signals u_ref and e_ref.

Parameters

in	*u_ref*	A reference input signal for adaptation
in	*e_ref*	A reference error signal for adaptation
in	*ref_size*	The size of each reference signal (u_ref and e_ref have the same size)

Returns

A flag indicating the success of adaptation

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

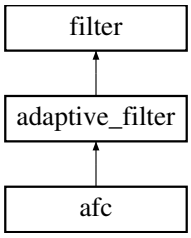
- libosp/OSP/adaptive_filter/adaptive_filter.hpp
- libosp/OSP/adaptive_filter/adaptive_filter.cpp

6.3 afc Class Reference

Adaptive Feedback Cancellation (AFC) Class.

`#include <afc.hpp>`

Inheritance diagram for afc:



Public Member Functions

- afc (float *bandlimited_filter_taps, size_t bandlimited_filter_tap_len, float *prefilter_taps, size_t prefilter_tap↩ _len, float *adaptive_filter_taps, size_t adaptive_filter_tap_len, size_t max_frame_size, int adaptation_type, float mu, float delta, float rho, float alpha, float beta, float p, float c, float power_estimate, size_t delay_len)

  AFC constructor.
- ~afc ()

  AFC destructor.
- int get_y_hat (float *y_hat, float *e, float *s, size_t ref_size)

  Getting y_hat signal (an estimated feedback signal)
- void get_delay (size_t &delay_len)

  Getting the length of delay line in samples.
- int set_delay (size_t delay_len)

  Setting the length of delay line in samples.

Additional Inherited Members

6.3.1 Detailed Description

Adaptive Feedback Cancellation (AFC) Class.

Under the FXLMS framework, this AFC class utilizes an adaptive filter to estimate the feedback signal, namely, y_hat.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 afc()

```
afc::afc (
            float * bandlimited_filter_taps,
            size_t bandlimited_filter_tap_len,
            float * prefilter_taps,
            size_t prefilter_tap_len,
            float * adaptive_filter_taps,
            size_t adaptive_filter_tap_len,
            size_t max_frame_size,
            int adaptation_type,
            float mu,
            float delta,
            float rho,
            float alpha,
            float beta,
            float p,
            float c,
            float power_estimate,
            size_t delay_len )  [explicit]
```

AFC constructor.

Parameters

in	*bandlimited_filter_taps*	The filter taps for bandlimited filter in AFC
in	*bandlimited_filter_tap_len*	The number of taps of bandlimited filter in AFC
in	*prefilter_taps*	The filter taps for whitening filter in AFC
in	*prefilter_tap_len*	The number of taps of prefilter in AFC
in	*adaptive_filter_taps*	The initial filter taps for adaptive filter in AFC
in	*adaptive_filter_tap_len*	The number of filter taps of adaptive filter in AFC
in	*max_frame_size*	The maximum processing frame size in adaptive filter
in	*adaptation_type*	The adaptation type for adaptive filter
in	*mu*	A parameter for adaptive filter
in	*delta*	A parameter for adaptive filter
in	*rho*	A parameter for adaptive filter
in	*alpha*	A parameter for adaptive filter
in	*beta*	A parameter for adaptive filter
in	*p*	A parameter for adaptive filter
in	*c*	A parameter for adaptive filter
in	*power_estimate*	A parameter for adaptive filter
in	*delay_len*	The number of delay in samples

See also

adaptive_filter

6.3.3 Member Function Documentation

6.3.3.1 get_delay()

```
void afc::get_delay (
            size_t & delay_len )
```

Getting the length of delay line in samples.

Parameters

out	*delay_len*	The number of delay in samples

6.3.3.2 get_y_hat()

```
int afc::get_y_hat (
            float * y_hat,
            float * e,
            float * s,
            size_t ref_size )
```

Getting y_hat signal (an estimated feedback signal)

Parameters

out	*y_hat*	An estimated feedback signal
in	*e*	An error signal for AFC (the output of hearing aid processing)
in	*s*	An input signal for AFC (the input of hearing aid processing)
in	*ref_size*	The size of each signal (e and s have the same size)

Returns

A flag indicating the success of getting correct y_hat according to the adaptation type

6.3.3.3 set_delay()

```
int afc::set_delay (
            size_t delay_len )
```

Setting the length of delay line in samples.

Parameters

in	*delay_len*	The number of delay in samples

Returns

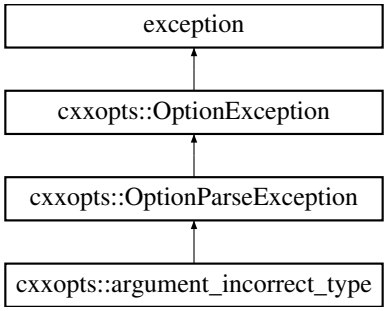
A flag indicating the success of setting delay_len

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

- libosp/OSP/afc/afc.hpp
- libosp/OSP/afc/afc.cpp

6.4 cxxopts::argument_incorrect_type Class Reference

Inheritance diagram for cxxopts::argument_incorrect_type:



Public Member Functions

- **argument_incorrect_type** (const std::string &arg)

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

- OSP/include/cxxopts.hpp

6.5 array_file Class Reference

Array File Class.

`#include <array_file.hpp>`

Public Member Functions

- array_file (std::string path)
  Array file constructor.
- ~array_file ()
  Array file destructor.
- size_t get_len ()
  Getting the length of the array.
- float * get_ptr ()
  Getting the pointer which points to the array.

6.5.1 Detailed Description

Array File Class.

Reading a binary file into an array in single-precision floating-point format

6.5.2 Constructor & Destructor Documentation

6.5.2.1 array_file()

```
array_file::array_file (
            std::string path )
```

Array file constructor.

Parameters

path	The path of the binary file

6.5.3 Member Function Documentation

6.5.3.1 get_len()

```
size_t array_file::get_len ( )
```

Getting the length of the array.

Returns

The length of the array

6.5.3.2 get_ptr()

```
float * array_file::get_ptr ( )
```

Getting the pointer which points to the array.

Returns

The pointer which points to the array

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

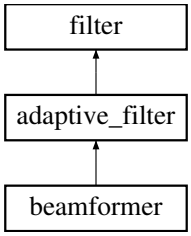
- libosp/OSP/array_file/array_file.hpp
- libosp/OSP/array_file/array_file.cpp

6.6 beamformer Class Reference

Beamformer Class.

`#include <beamformer.hpp>`

Inheritance diagram for beamformer:



Public Member Functions

- beamformer (size_t delay_len, float *adaptive_filter_taps, size_t adaptive_filter_tap_len, size_t max_frame_size, int adaptation_type, float mu, float delta, float rho, float alpha, float beta, float p, float c, float power_estimate)

  Beamformer constructor.
- ~beamformer ()

  Beamformer destructor.
- int get_e (float *e, const float *x_l, const float *x_r, size_t ref_size)

  Getting e signal (the output signal of this beamformer)

Additional Inherited Members

6.6.1 Detailed Description

Beamformer Class.

This beamformer class implements the generalized sidelobe canceller (GSC) using SLMS [Lee et al., IHCON 2018].

6.6.2 Constructor & Destructor Documentation

6.6.2.1 beamformer()

```
beamformer::beamformer (
            size_t delay_len,
            float * adaptive_filter_taps,
            size_t adaptive_filter_tap_len,
            size_t max_frame_size,
            int adaptation_type,
            float mu,
            float delta,
            float rho,
            float alpha,
            float beta,
            float p,
            float c,
            float power_estimate )  [explicit]
```

Beamformer constructor.

Parameters

in	*delay_len*	The length of delay line in samples for beamformer
in	*adaptive_filter_taps*	The initial filter taps for adaptive filter in beamformer
in	*adaptive_filter_tap_len*	The number of filter taps of adaptive filter in beamformer
in	*max_frame_size*	The maximum processing frame size in adaptive filter
in	*adaptation_type*	The adaptation type for adaptive filter
in	*mu*	A parameter for adaptive filter
in	*delta*	A parameter for adaptive filter
in	*rho*	A parameter for adaptive filter
in	*alpha*	A parameter for adaptive filter
in	*beta*	A parameter for adaptive filter
in	*p*	A parameter for adaptive filter
in	*c*	A parameter for adaptive filter
in	*power_estimate*	A parameter for adaptive filter

See also

adaptive_filter

6.6.3 Member Function Documentation

6.6.3.1 get_e()

```
int beamformer::get_e (
            float * e,
            const float * x_l,
            const float * x_r,
            size_t ref_size )
```

Getting e signal (the output signal of this beamformer)

Parameters

out	*e*	The output signal of this beamformer
in	*x_l*	The input signal from the left channel
in	*x_r*	the input signal from the right channel
in	*ref_size*	The size of each input signal (x_l and x_r have the same size)

Returns

A flag indicating the success of adaptation in adaptive filter

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

- libosp/OSP/beamformer/beamformer.hpp
- libosp/OSP/beamformer/beamformer.cpp

6.7 circular_buffer Class Reference

Circular Buffer Class.

`#include <circular_buffer.hpp>`

Public Member Functions

- circular_buffer (size_t size, float reset)

  Circular buffer constructor.
- ~circular_buffer ()

  Default destructor.
- void set (const float *item, size_t buf_size)

  This is the set command for the circular buffer.
- void get (float *data, size_t buf_size)

  This is the get function for the circular buffer.
- void reset ()

  This is the reset command for circular buffer. It resets all of the values in the buffer to the default value the user entered in the constructor.
- size_t size () const

  Function to get the size of the buffer.

Public Attributes

- std::mutex **mutex_**
- float * **buf_**
- size_t **head_**
- size_t **size_**
- size_t **mask_**
- float **reset_**

6.7.1 Detailed Description

Circular Buffer Class.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 circular_buffer()

```
circular_buffer::circular_buffer (
            size_t size,
            float reset ) [explicit]
```

Circular buffer constructor.

Parameters

size	The maximum size you would want your circular buffer to be
reset	The value you want to reset all of the values in the circular buffer to

6.7.3 Member Function Documentation

6.7.3.1 get()

```
void circular_buffer::get (
            float * data,
            size_t buf_size )
```

This is the get function for the circular buffer.

Parameters

data	A buffer to put your data in.
buf_size	The amount of data you want from the circular buffer

6.7.3.2 set()

```
void circular_buffer::set (
            const float * item,
            size_t buf_size )
```

This is the set command for the circular buffer.

Parameters

item	The buffer of data you want to put in the circular buffer
buf_size	The size of the buffer.

6.7.3.3 size()

```
size_t circular_buffer::size ( ) const
```

Function to get the size of the buffer.

Returns

The size of the circular buffer which will be a power of 2

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

- libosp/OSP/circular_buffer/circular_buffer.hpp
- libosp/OSP/circular_buffer/circular_buffer.cpp

6.8 control_t Struct Reference

Public Attributes

- int **input_device** = -1
- int **output_device** = -1
- bool **endloop** = false
- bool **multithread** = D_MULTI
- int **samp_freq** = D_SAMP_FREQ
- size_t **buf_size** = D_BUF

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

- OSP/include/control_param.h

6.9 ews_connection< P, M, T > Class Template Reference

Public Member Functions

- **ews_connection** (P *parser, M *mha, Poco::UInt16 socket)

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

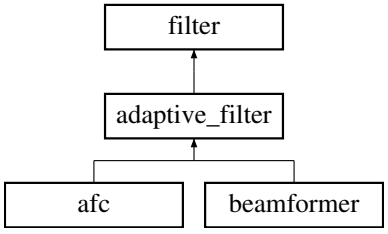
- OSP/include/ews_connect.hpp

6.10 filter Class Reference

Filter Class.

`#include <filter.hpp>`

Inheritance diagram for filter:



Public Member Functions

- filter (float *taps, size_t tap_size, circular_buffer *cir_buf, size_t max_buf_size)

  Filter constructor.
- ~filter ()

  Filter destructor.
- int set_taps (const float *taps, size_t buf_size)

  Setting the filter taps.
- int get_taps (float *taps, size_t buf_size)

  Getting the filter taps.
- void cirfir (float *data_in, float *data_out, size_t num_samp)

  Getting the output of this FIR filter by performing frame-based convolution.
- size_t get_size ()

  Getting the number of taps of this FIR filter.
- void cirfir (float *data_out, size_t num_samp)

  Frame-based convolution for FIR filtering.

6.10.1 Detailed Description

Filter Class.

This filter class implements the FIR filter

6.10.2 Constructor & Destructor Documentation

6.10.2.1 filter()

```
filter::filter (
            float * taps,
            size_t tap_size,
            circular_buffer * cir_buf,
            size_t max_buf_size )  [explicit]
```

Filter constructor.

Parameters

in	*taps*	The filter taps for this FIR filter
in	*tap_size*	The number of taps of this FIR filter
in	*cir_buf*	The circular buffer for this FIR filter to perform frame-based convolution
in	*max_buf_size*	The maximum size of circular buffer you need to specify if there is no circular buffer given in cir_buf

6.10.3 Member Function Documentation

6.10.3.1 cirfir() [1/2]

```
void filter::cirfir (
            float * data_in,
            float * data_out,
            size_t num_samp )
```

Getting the output of this FIR filter by performing frame-based convolution.

Parameters

in	*data_in*	The input signal
out	*data_out*	The output signal
	num_samp	The size of input and output signal (data_in and data_out should have the same size)

6.10.3.2 cirfir() [2/2]

```
void filter::cirfir (
            float * data_out,
            size_t num_samp )
```

Frame-based convolution for FIR filtering.

Parameters

out	*data_out*	The output signal
in	*num_samp*	The size of input and output signal

6.10.3.3 get_size()

```
size_t filter::get_size ( )
```

Getting the number of taps of this FIR filter.

Returns

The number of taps of this FIR filter

6.10.3.4 get_taps()

```
int filter::get_taps (
            float * taps,
            size_t buf_size )
```

Getting the filter taps.

Parameters

out	*taps*	The filter taps (1-D array)
in	*buf_size*	The size of the filter taps (this should be the same as tap_size passed in constructor)

Returns

A flag indicating the success of getting the filter taps

6.10.3.5 set_taps()

```
int filter::set_taps (
            const float * taps,
            size_t buf_size )
```

Setting the filter taps.

Parameters

in	*taps*	The filter taps (an 1-D array)
in	*buf_size*	The size of the filter taps (this should be the same as tap_size passed in constructor)

Returns

A flag indicating the success of setting the filter taps

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

- libosp/OSP/filter/filter.hpp
- libosp/OSP/filter/filter.cpp

6.11 cxxopts::HelpGroupDetails Struct Reference

Public Attributes

- std::string **name**
- std::string **description**
- std::vector< HelpOptionDetails > **options**

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

- OSP/include/cxxopts.hpp

6.12 cxxopts::HelpOptionDetails Struct Reference

Public Attributes

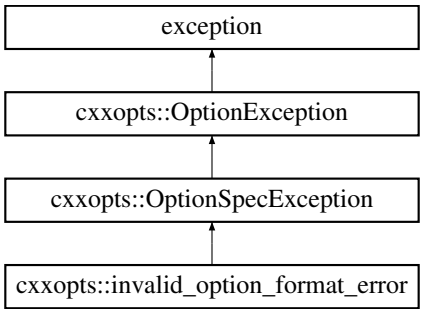
- std::string **s**
- std::string **l**
- String **desc**
- bool **has_default**
- std::string **default_value**
- bool **has_implicit**
- std::string **implicit_value**
- std::string **arg_help**
- bool **is_container**
- bool **is_boolean**

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

- OSP/include/cxxopts.hpp

6.13 cxxopts::invalid_option_format_error Class Reference

Inheritance diagram for cxxopts::invalid_option_format_error:



Public Member Functions

- **invalid_option_format_error** (const std::string &format)

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

- OSP/include/cxxopts.hpp

6.14 cxxopts::KeyValue Class Reference

Public Member Functions

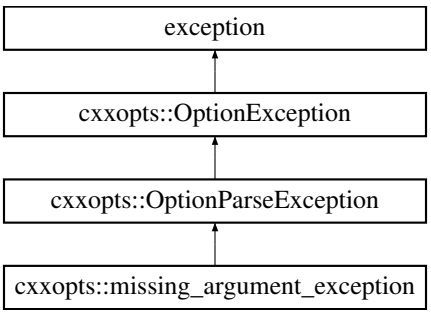
- **KeyValue** (std::string key_, std::string value_)
- const std::string & **key** () const
- const std::string **value** () const
- template<typename T >
  T **as** () const

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

- OSP/include/cxxopts.hpp

6.15 cxxopts::missing_argument_exception Class Reference

Inheritance diagram for cxxopts::missing_argument_exception:



Public Member Functions

- **missing_argument_exception** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.16 noise_management Class Reference

Noise Management Class.

`#include <noise_management.hpp>`

Public Member Functions

- noise_management (int ntype, int stype, float sparam, float fsamp)

  Noise management constructor.
- ~noise_management ()

  Noise management destructor.
- void set_param (int ntype, int stype, float sparam)

  Setting all parameters in noise management.
- void get_param (int &ntype, int &stype, float &sparam)

  Getting all parameters in noise management.
- void speech_enhancement (float *data_in, size_t in_len, float *data_out)

  A function to perform speech enhancement.

6.16.1 Detailed Description

Noise Management Class.

Speech enhancement using peak and valley detection, noise estimation and spectral subtraction

6.16.2 Constructor & Destructor Documentation

6.16.2.1 noise_management()

```
noise_management::noise_management (
            int ntype,
            int stype,
            float sparam,
            float fsamp )  [explicit]
```

Noise management constructor.

Parameters

in	*ntype*	The type of noise estimation, 1: using limits on change (ref: Arslan et al.), 2: using the weighted averaging of Hirsch and Ehrlicher, 3: using MCRA of Cohen and Berdugo
in	*stype*	The type of spectral subtraction, 0: normal, 1: oversubtraction
in	*sparam*	A parameter for spectral subtraction
in	*fsamp*	The sampling rate

6.16.3 Member Function Documentation

6.16.3.1 get_param()

```
void noise_management::get_param (
            int & ntype,
            int & stype,
            float & sparam )
```

Getting all parameters in noise management.

Parameters

in	*ntype*	See constructor
in	*stype*	See constructor
in	*sparam*	See constructor

6.16.3.2 set_param()

```
void noise_management::set_param (
            int ntype,
            int stype,
            float sparam )
```

Setting all parameters in noise management.

Parameters

in	*ntype*	See constructor
in	*stype*	See constructor
in	*sparam*	See constructor

6.16.3.3 speech_enhancement()

```
void noise_management::speech_enhancement (
            float * data_in,
            size_t in_len,
            float * data_out )
```

A function to perform speech enhancement.

Parameters

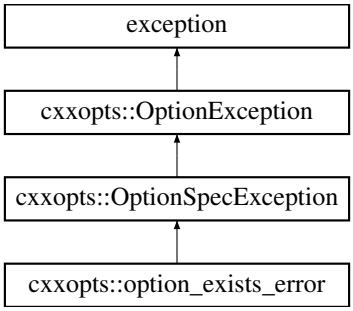
in	*data_in*	The input signal
in	*in_len*	Length of the input signal
out	*data_out*	The output signal, i.e., the enhanced speech signal

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

- libosp/OSP/subband/noise_management.hpp
- libosp/OSP/subband/noise_management.cpp

6.17 cxxopts::option_exists_error Class Reference

Inheritance diagram for cxxopts::option_exists_error:



Public Member Functions

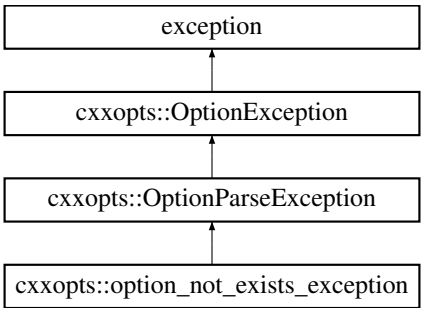
- **option_exists_error** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.18 cxxopts::option_not_exists_exception Class Reference

Inheritance diagram for cxxopts::option_not_exists_exception:



Public Member Functions

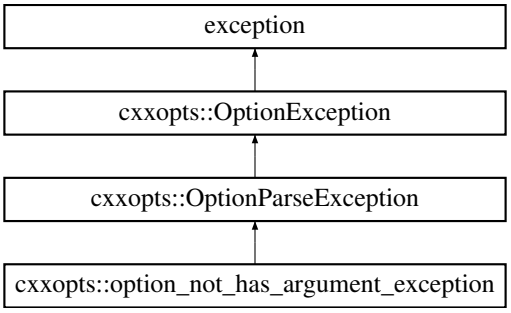
- **option_not_exists_exception** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.19 cxxopts::option_not_has_argument_exception Class Reference

Inheritance diagram for cxxopts::option_not_has_argument_exception:



Public Member Functions

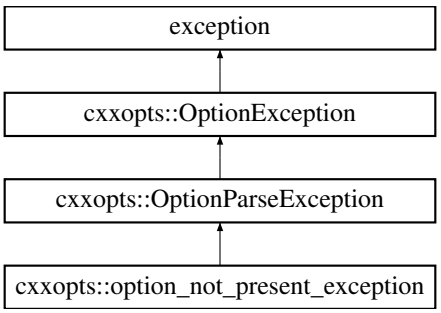
- **option_not_has_argument_exception** (const std::string &option, const std::string &arg)

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

- OSP/include/cxxopts.hpp

6.20 cxxopts::option_not_present_exception Class Reference

Inheritance diagram for cxxopts::option_not_present_exception:



Public Member Functions

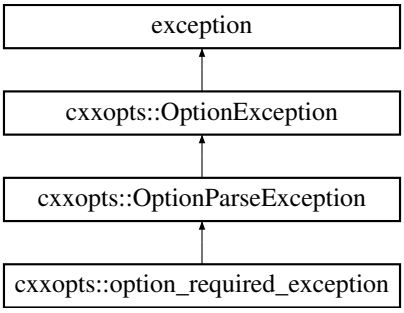
- **option_not_present_exception** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.21 cxxopts::option_required_exception Class Reference

Inheritance diagram for cxxopts::option_required_exception:



Public Member Functions

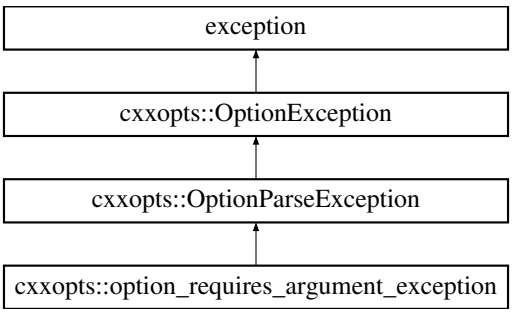
- **option_required_exception** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.22 cxxopts::option_requires_argument_exception Class Reference

Inheritance diagram for cxxopts::option_requires_argument_exception:



Public Member Functions

- **option_requires_argument_exception** (const std::string &option)

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

- OSP/include/cxxopts.hpp

6.23 cxxopts::OptionAdder Class Reference

Public Member Functions

- **OptionAdder** (Options &options, std::string group)
- OptionAdder & **operator()** (const std::string &opts, const std::string &desc, std::shared_ptr< const Value > value=::cxxopts::value< bool >(), std::string arg_help="")

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

- OSP/include/cxxopts.hpp

6.24 cxxopts::OptionDetails Class Reference

Public Member Functions

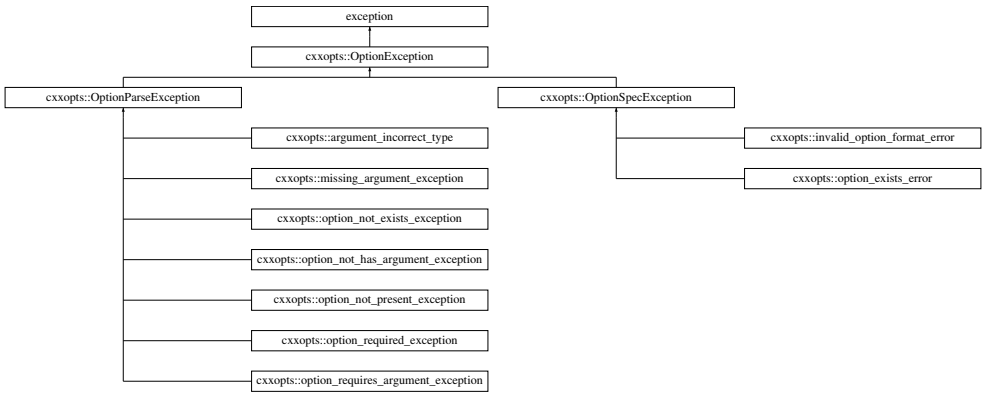
- **OptionDetails** (const std::string &short_, const std::string &long_, const String &desc, std::shared_ptr< const Value > val)
- **OptionDetails** (const OptionDetails &rhs)
- **OptionDetails** (OptionDetails &&rhs)=default
- const String & **description** () const
- const Value & **value** () const
- std::shared_ptr< Value > **make_storage** () const
- const std::string & **short_name** () const
- const std::string & **long_name** () const

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

- OSP/include/cxxopts.hpp

6.25 cxxopts::OptionException Class Reference

Inheritance diagram for cxxopts::OptionException:



Public Member Functions

- **OptionException** (const std::string &message)
- virtual const char * **what** () const noexcept

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

- OSP/include/cxxopts.hpp

6.26 cxxopts::OptionParseException Class Reference

Inheritance diagram for cxxopts::OptionParseException:

Public Member Functions

- **OptionParseException** (const std::string &message)

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

- OSP/include/cxxopts.hpp

6.27 cxxopts::Options Class Reference

Public Member Functions

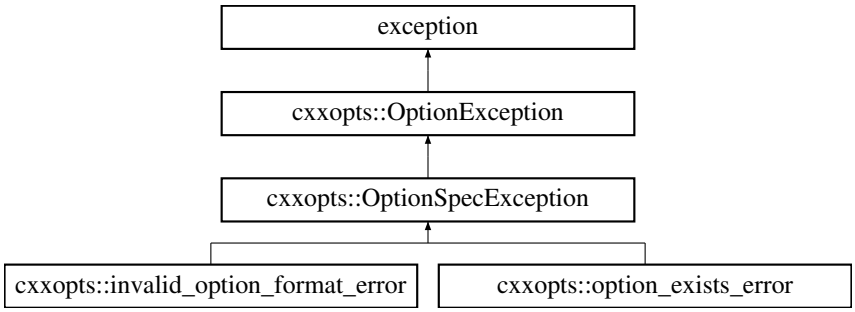
- **Options** (std::string program, std::string help_string="")
- Options & **positional_help** (std::string help_text)
- Options & **custom_help** (std::string help_text)
- Options & **show_positional_help** ()
- Options & **allow_unrecognised_options** ()
- ParseResult **parse** (int &argc, char **&argv)
- OptionAdder **add_options** (std::string group="")
- void **add_option** (const std::string &group, const std::string &s, const std::string &l, std::string desc, std::shared_ptr< const Value > value, std::string arg_help)
- void **parse_positional** (std::string option)
- void **parse_positional** (std::vector< std::string > options)
- void **parse_positional** (std::initializer_list< std::string > options)
- template<typename Iterator >
  void **parse_positional** (Iterator begin, Iterator end)
- std::string **help** (const std::vector< std::string > &groups={""}) const
- const std::vector< std::string > **groups** () const
- const HelpGroupDetails & **group_help** (const std::string &group) const

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

- OSP/include/cxxopts.hpp

6.28 cxxopts::OptionSpecException Class Reference

Inheritance diagram for cxxopts::OptionSpecException:



Public Member Functions

- **OptionSpecException** (const std::string &message)

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

- OSP/include/cxxopts.hpp

6.29 cxxopts::OptionValue Class Reference

Public Member Functions

- void **parse** (std::shared_ptr< const OptionDetails > details, const std::string &text)
- void **parse_default** (std::shared_ptr< const OptionDetails > details)
- size_t **count** () const
- template<typename T >
  const T & **as** () const

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

- OSP/include/cxxopts.hpp

6.30 osp_parser Class Reference

Public Member Functions

- void **parse** (int argc, char *argv[], osp_user_data *user_data)
- void **parse** (int argc, char *argv[], osp_user_data *user_data, controls *main_controls, int initial=0)

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

- OSP/include/osp_parser.hpp

6.31 osp_process Class Reference

OSP Process Class.

`#include <osp_process.hpp>`

Public Member Functions

- osp_process (float samp_freq, size_t max_buffer_in, osp_user_data *user_data, bool multithread)
  OSP process constructor.
- ~osp_process ()
  OSP process destructor.
- void process (float **in, float **out, size_t buf_size)
- void set_params (osp_user_data *user_data)
  Setting all parameters for MHA.
- void get_params (osp_user_data *user_data)
  Getting all parameters for MHA.
- void process_channels (int channel)
  A function to perform MHA processing on the input signal for all the channels.
- void start (int channel, float *in)
  A function which takes the input signal and starts the MHA processing.
- void join (int channel, float *out)
  A function which returns the output signal after the MHA processing.

6.31.1 Detailed Description

OSP Process Class.

This class groups all of the different OSP library calls in one place. This is more of an example way of using the different parts of the OSP libraries to implement a hearing aid algorithm including adaptive feedback cancellation (AFC), wide dynamic range compression (WDRC) and speech enhancement. A wrapper class to initialize all the modules of a master hearing aid (MHA).

6.31.2 Constructor & Destructor Documentation

6.31.2.1 osp_process()

```
osp_process::osp_process (
            float samp_freq,
            size_t max_buffer_in,
            osp_user_data * user_data,
            bool multithread )  [inline], [explicit]
```

OSP process constructor.

Parameters

in	*samp_freq*	The sampling rate of the input signal
in	*max_buffer_in*	The max buffer size of the input signal
in	*user_data*	A general data structure shared between client and C/C++ application
in	*multithread*	A flag enabling the multi-threading feature

See also

osp_user_data_t

6.31.3 Member Function Documentation

6.31.3.1 get_params()

```
void osp_process::get_params (
            osp_user_data * user_data )  [inline]
```

Getting all parameters for MHA.

Parameters

in	*user_data*	A general data structure shared between client and C/C++ application

See also

osp_user_data_t

6.31.3.2 join()

```
void osp_process::join (
            int channel,
            float * out )  [inline]
```

A function which returns the output signal after the MHA processing.

Parameters

in	*channel*	The channel number
out	*out*	The output signal for this channel

6.31.3.3 process()

```
void osp_process::process (
            float ** in,
            float ** out,
            size_t buf_size )  [inline]
```

@breif A function to amplify the input signal for all the channels

Parameters

in	*in*	A pointer to the multi-channel input signal
out	*out*	A pointer to the multi-channel output signal
in	*buf_size*	The length of the input signal which is given for this processing, i.e., the frame length

6.31.3.4 process_channels()

```
void osp_process::process_channels (
            int channel )  [inline]
```

A function to perform MHA processing on the input signal for all the channels.

Parameters

in	*channel*	The channel number

Gain

AFC begin

AFC end

6.31.3.5 set_params()

```
void osp_process::set_params (
            osp_user_data * user_data )  [inline]
```

Setting all parameters for MHA.

Parameters

in	*user_data*	A general data structure shared between client and C/C++ application

See also

osp_user_data_t

6.31.3.6 start()

```
void osp_process::start (
            int channel,
            float * in )  [inline]
```

A function which takes the input signal and starts the MHA processing.

Parameters

in	*channel*	The channel number
in	*in*	The input signal for this channel

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

- OSP/include/osp_process.hpp

6.32 osp_user_data_t Struct Reference

```
#include <osp_param.h>
```

Public Member Functions

- template<class Archive >
  void serialize (Archive &archive)

  SLMS.

Public Attributes

- int en_ha = D_EN_HA

  No operation. The audio is passed from input to output in the audio callback.
- int rear_mics = D_REAR_MIC

  Read mics on/off.
- float **gain** = D_ATTENUATION
- std::vector< float > g50 = std::vector<float>(NUM_BANDS, D_G50)

  The gain values at 50 dB SPL input level.
- std::vector< float > g80 = std::vector<float>(NUM_BANDS,D_G80)

  The gain values at 80 dB SPL input level.
- std::vector< float > knee_low = std::vector<float>(NUM_BANDS,D_KNEE_LOW)

  Lower kneepoints for all bands.
- std::vector< float > knee_high = std::vector<float>(NUM_BANDS,D_KNEE_HIGH)

  Upper kneepoints for all bands.
- std::vector< float > attack = std::vector<float>(NUM_BANDS,D_ATTACK_TIME)

  Attack time for WDRC for all bands.
- std::vector< float > release = std::vector<float>(NUM_BANDS,D_RELEASE_TIME)

  Release time for WDRC for all bands.
- float mpo = D_MPO

MPO for Max power limit for WDRC.

- int noise_estimation_type = D_NOISE_ESTIMATION

  Choose type of Noise estimation technique.
- int **spectral_type** = D_SPECTRAL_TYPE
- float spectral_subtraction = D_SPECTRAL_SUB

  Spectral subtraction Param.
- int afc = D_AFC

  AFC Type -1: return y_hat=0, 0: stop adaptation, 1: FXLMS, 2: PMLMS, 3: SLMS.
- size_t **afc_delay** = AFC_DELAY
- float **afc_mu** = AFC_MU
- float afc_rho = AFC_RHO

  step size
- float afc_power_estimate = AFC_PE

  forgetting factor
- float afc_delta = AFC_DELTA

  power estimate
- float afc_alpha = AFC_ALPHA

  IPNLMS.
- float afc_beta = AFC_BETA

  IPNLMS.
- float afc_p = AFC_P

  IPNLMS.
- float afc_c = AFC_C

  SLMS.

6.32.1 Detailed Description

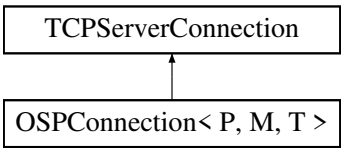
Please note that any variables added to this structure must have the same name in the parser.

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

- OSP/include/osp_param.h

6.33 OSPConnection< P, M, T > Class Template Reference

Inheritance diagram for OSPConnection< P, M, T >:



Public Member Functions

- **OSPConnection** (const Poco::Net::StreamSocket &s, P *parser, M *mha)
- void **run** ()
- int **osp_json_parser** (std::string json_string, int channel)

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

- OSP/include/ews_connect.hpp

6.34 cxxopts::ParseResult Class Reference

Public Member Functions

- **ParseResult** (const std::unordered_map< std::string, std::shared_ptr< OptionDetails >> &, std::vector< std::string >, bool allow_unrecognised, int &, char **&)
- size_t **count** (const std::string &o) const
- const OptionValue & **operator[ ]** (const std::string &option) const
- const std::vector< KeyValue > & **arguments** () const

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

- OSP/include/cxxopts.hpp

6.35 peak_detect Class Reference

Peak Detector Class.

`#include <peak_detect.hpp>`

Public Member Functions

- peak_detect (float fsamp, float attack_time, float release_time)
  Peak detector constructor.
- ~peak_detect ()
  Peak detector destructor.
- void set_param (float attack_time, float release_time)
  Setting the parameters for peak detector (to have alpha and beta)
- void get_param (float &attack_time, float &release_time)
  Getting the parameters from peak detector (in terms of attach time and release time)
- void get_spl (float *data_in, size_t in_len, float *pdb_out)
  Getting the output from the peak detector in SPL.

6.35.1 Detailed Description

Peak Detector Class.

This peak detector implements the algorithm according to Eq. (8.1) in [James M. Kates, Digital hearing aids, Plural publishing, 2008].

6.35.2 Constructor & Destructor Documentation

6.35.2.1 peak_detect()

```
peak_detect::peak_detect (
            float fsamp,
            float attack_time,
            float release_time )  [explicit]
```

Peak detector constructor.

Parameters

in	*fsamp*	The sampling rate of the system
in	*attack_time*	Attack time in milliseconds
in	*release_time*	Release time in milliseconds

6.35.3 Member Function Documentation

6.35.3.1 get_param()

```
void peak_detect::get_param (
            float & attack_time,
            float & release_time )
```

Getting the parameters from peak detector (in terms of attach time and release time)

Parameters

out	*attack_time*	attack_time Attack time in milliseconds
out	*release_time*	release_time Release time in milliseconds

6.35.3.2 get_spl()

```
void peak_detect::get_spl (
            float * data_in,
            size_t in_len,
            float * pdb_out )
```

Getting the output from the peak detector in SPL.

Parameters

in	*data_in*	The input signal
in	*in_len*	The size of the input signal
out	*pdb_out*	The output of peak detector in SPL

6.35.3.3 set_param()

```
void peak_detect::set_param (
            float attack_time,
            float release_time )
```

Setting the parameters for peak detector (to have alpha and beta)

Parameters

in	*attack_time*	Attack time in milliseconds
in	*release_time*	Release time in milliseconds

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

- libosp/OSP/subband/peak_detect.hpp
- libosp/OSP/subband/peak_detect.cpp

6.36 portaudio_wrapper Class Reference

Public Member Functions

- portaudio_wrapper (int in_device, int in_num_channel, int out_device, int out_num_channels, PaStream↩ Callback callback, void ∗userData, int sample_rate, unsigned long frames_per_buffer)
- portaudio_wrapper (int in_num_channel, int out_num_channels, PaStreamCallback callback, void ∗userData, int sample_rate, unsigned long frames_per_buffer)
- ∼portaudio_wrapper ()
- int init_stream (int in_device, int in_num_channel, int out_device, int out_num_channels, PaStreamCallback callback, void ∗userData)
- int start_stream ()
- int stop_stream ()

6.36.1 Constructor & Destructor Documentation

6.36.1.1 portaudio_wrapper() [1/2]

```
portaudio_wrapper::portaudio_wrapper (
            int in_device,
            int in_num_channel,
            int out_device,
            int out_num_channels,
            PaStreamCallback callback,
            void * userData,
            int sample_rate,
            unsigned long frames_per_buffer )
```

Constructor for when the input and output devices are manually entered

Parameters

in_device	
in_num_channel	
out_device	
out_num_channels	
callback	
userData	
sample_rate	
frames_per_buffer	

6.36.1.2 portaudio_wrapper() [2/2]

```
portaudio_wrapper::portaudio_wrapper (
            int in_num_channel,
            int out_num_channels,
            PaStreamCallback callback,
            void * userData,
            int sample_rate,
            unsigned long frames_per_buffer )
```

Constructor for when the input and output devices are the default devices

Parameters

in_num_channel	
out_num_channels	
callback	
userData	
sample_rate	
frames_per_buffer	

6.36.1.3 ~portaudio_wrapper()

```
portaudio_wrapper::~portaudio_wrapper ( )
```

Destructor

6.36.2 Member Function Documentation

6.36.2.1 init_stream()

```
int portaudio_wrapper::init_stream (
            int in_device,
            int in_num_channel,
            int out_device,
            int out_num_channels,
            PaStreamCallback callback,
            void * userData )
```

Initialize the stream

Parameters

in_device	
in_num_channel	
out_device	
out_num_channels	
callback	
userData	

Returns

0 if successful

6.36.2.2 start_stream()

```
int portaudio_wrapper::start_stream ( )
```

Start the stream

Returns

0 if successful

6.36.2.3 stop_stream()

```
int portaudio_wrapper::stop_stream ( )
```

Stop the stream

Returns

0 if successful

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

- OSP/include/portaudio_wrapper.h
- OSP/src/portaudio_wrapper.cpp

6.37 resample Class Reference

Resample Class.

```
#include <resample.hpp>
```

Public Member Functions

- resample (float ∗taps, size_t tap_size, size_t max_in_buf_size, int interp_factor, int deci_factor)

  Resample constructor.
- ∼resample ()

  Resample destructor.
- void resamp (float ∗data_in, size_t in_size, float ∗data_out, size_t ∗out_size)

  Getting the resampled signal.

6.37.1 Detailed Description

Resample Class.

Resampling class implements L/M-fold resampling

6.37.2 Constructor & Destructor Documentation

6.37.2.1 resample()

```
resample::resample (
            float * taps,
            size_t tap_size,
            size_t max_in_buf_size,
            int interp_factor,
            int deci_factor )  [explicit]
```

Resample constructor.

Parameters

in	*taps*	The filter taps of the lowpass filter (to reject images and prevent aliasing)
in	*tap_size*	The number of taps of the lowpass filter
in	*max_in_buf_size*	The maximum input buffer size
in	*interp_factor*	The interpolation factor L (to implement L-fold expander)
in	*deci_factor*	The decimation factor M (to implement M-fold decimator)

6.37.3 Member Function Documentation

6.37.3.1 resamp()

```
void resample::resamp (
            float * data_in,
```

```
size_t in_size,
float * data_out,
size_t * out_size )
```

Getting the resampled signal.

Parameters

in	*data_in*	The signal in original sampling rate
in	*in_size*	The size of the original signal
out	*data_out*	The resampled signal
out	*out_size*	The size of the resampled signal

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

- libosp/OSP/resample/resample.hpp
- libosp/OSP/resample/resample.cpp

6.38 rk_sema Struct Reference

Public Attributes

- sem_t **sem**

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

- OSP/include/sema.hpp

6.39 cxxopts::values::detail::SignedCheck< T, B > Struct Template Reference

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

- OSP/include/cxxopts.hpp

6.40 cxxopts::values::detail::SignedCheck< T, false > Struct Template Reference

Public Member Functions

- template<typename U >
  void **operator()** (bool, U, const std::string &)

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

- OSP/include/cxxopts.hpp

6.41 cxxopts::values::detail::SignedCheck< T, true > Struct Template Reference

Public Member Functions

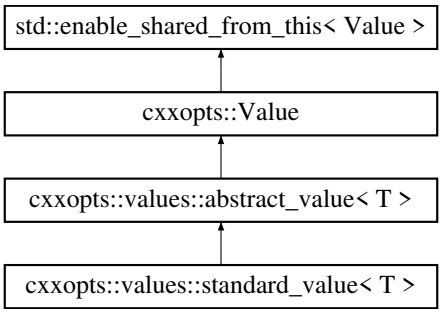
- template<typename U >
  void **operator()** (bool negative, U u, const std::string &text)

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

- OSP/include/cxxopts.hpp

6.42 cxxopts::values::standard_value< T > Class Template Reference

Inheritance diagram for cxxopts::values::standard_value< T >:



Public Member Functions

- std::shared_ptr< Value > **clone** () const

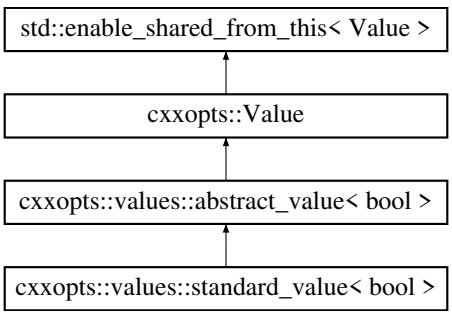
Additional Inherited Members

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

- OSP/include/cxxopts.hpp

6.43 cxxopts::values::standard_value< bool > Class Template Reference

Inheritance diagram for cxxopts::values::standard_value< bool >:



Public Member Functions

- **standard_value** (bool *b)
- std::shared_ptr< Value > **clone** () const

Additional Inherited Members

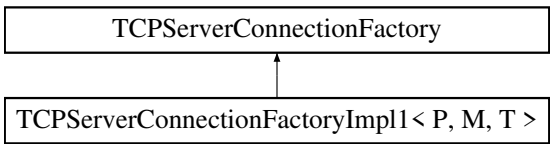
The documentation for this class was generated from the following file:

- OSP/include/cxxopts.hpp

6.44 TCPServerConnectionFactoryImpl1< P, M, T > Class Template Reference

`#include <ews_connect.hpp>`

Inheritance diagram for TCPServerConnectionFactoryImpl1< P, M, T >:



Public Member Functions

- **TCPServerConnectionFactoryImpl1** (P *parser, M *mha)
- Poco::Net::TCPServerConnection * **createConnection** (const Poco::Net::StreamSocket &socket)

6.44.1 Detailed Description

template<class P, class M, class T>
class TCPServerConnectionFactoryImpl1< P, M, T >

This template provides a basic implementation of TCPServerConnectionFactory.

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

- OSP/include/ews_connect.hpp

6.45 cxxopts::values::type_is_container< T > Struct Template Reference

Static Public Attributes

- static constexpr bool **value** = false

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

- OSP/include/cxxopts.hpp

6.46 cxxopts::values::type_is_container< std::vector< T > > Struct Template Reference

Static Public Attributes

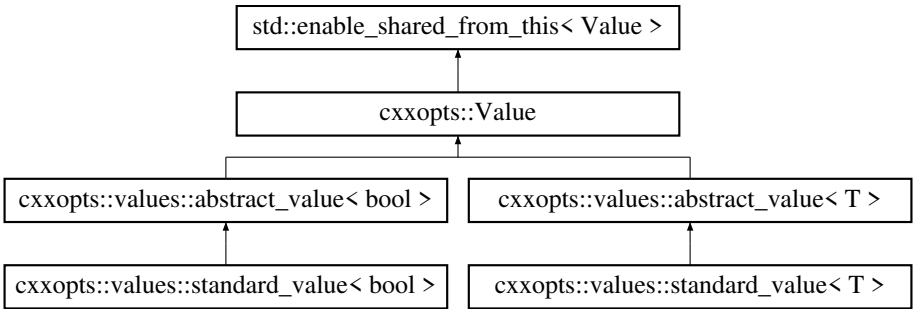
- static constexpr bool **value** = true

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

- OSP/include/cxxopts.hpp

6.47 cxxopts::Value Class Reference

Inheritance diagram for cxxopts::Value:



Public Member Functions

- virtual std::shared_ptr< Value > **clone** () const =0
- virtual void **parse** (const std::string &text) const =0
- virtual void **parse** () const =0
- virtual bool **has_default** () const =0
- virtual bool **is_container** () const =0
- virtual bool **has_implicit** () const =0
- virtual std::string **get_default_value** () const =0
- virtual std::string **get_implicit_value** () const =0
- virtual std::shared_ptr< Value > **default_value** (const std::string &value)=0
- virtual std::shared_ptr< Value > **implicit_value** (const std::string &value)=0
- virtual bool **is_boolean** () const =0

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

- OSP/include/cxxopts.hpp

6.48 wdrc Class Reference

Wide Dynamic Range Compression (WDRC) Class.

`#include <wdrc.hpp>`

Public Member Functions

- wdrc (float gain50, float gain80, float knee_low, float mpo_limit)
  wdrc constructor
- ~wdrc ()
  wdrc destructor
- void set_param (float gain50, float gain80, float knee_low, float mpo_limit)
  Setting WDRC parameters.
- void get_param (float &gain50, float &gain80, float &knee_low, float &mpo_limit)
  Getting WDRC parameters.
- void process (float *input, float *pdb, size_t in_len, float *output)
  Perform WDRC.

6.48.1 Detailed Description

Wide Dynamic Range Compression (WDRC) Class.

Applying WDRC to a subband signal from an analysis filterbank

6.48.2 Constructor & Destructor Documentation

6.48.2.1 wdrc()

```
wdrc::wdrc (
            float gain50,
            float gain80,
            float knee_low,
            float mpo_limit )  [explicit]
```

wdrc constructor

Parameters

in	*gain50*	Gain at 50 dB SPL of input level
in	*gain80*	Gain at 80 dB SPL of input level
in	*knee_low*	Lower knee-point
in	*mpo_limit*	Maximum power output (MPO)

6.48.3 Member Function Documentation

6.48.3.1 get_param()

```
void wdrc::get_param (
            float & gain50,
            float & gain80,
            float & knee_low,
            float & mpo_limit )
```

Getting WDRC parameters.

Parameters

out	*gain50*	Gain at 50 dB SPL of input level
out	*gain80*	Gain at 80 dB SPL of input level
out	*knee_low*	Lower knee-point
out	*mpo_limit*	MPO

6.48.3.2 process()

```
void wdrc::process (
            float * input,
            float * pdb,
            size_t in_len,
            float * output )
```

Perform WDRC.

The peak detector output in dB SPL is needed as one of the inputs. The gain at 50 and 80 dB SPL is specified for the frequency sub-band, along with the lower and upper kneepoints in dB SPL. The compressor is linear below the lower kneepoint and applies compression limiting above the upper kneepoint

Parameters

in	*input*	The input signal (1-D array)
in	*pdb*	The output from the peak detector in SPL, i.e., the output from get_spl member function in peak_detect class
in	*in_len*	Length of the input signal
out	*output*	Pointer to a signal (1-D array) where the compressed output of the subband signal will be written, i.e., the output of WDRC

See also

peak_detect

6.48.3.3 set_param()

```
void wdrc::set_param (
            float gain50,
```

```
            float gain80,
            float knee_low,
            float mpo_limit )
```

Setting WDRC parameters.

Parameters

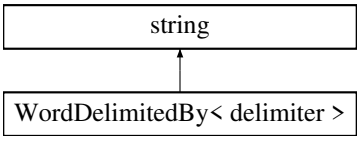
in	*gain50*	Gain at 50 dB SPL of input level
in	*gain80*	Gain at 80 dB SPL of input level
in	*knee_low*	Lower knee-point
in	*mpo_limit*	MPO

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

- libosp/OSP/subband/wdrc.hpp
- libosp/OSP/subband/wdrc.cpp

6.49 WordDelimitedBy< delimiter > Class Template Reference

Inheritance diagram for WordDelimitedBy< delimiter >:



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

- OSP/include/cxxopts.hpp

Chapter 7

File Documentation

7.1 libosp/OSP/array_utilities/array_utilities.hpp File Reference

```
#include <cstddef>
```

Functions

- void array_flip (float *arr, size_t len)
  Function to reverse an array.
- float array_sum (const float *arr, size_t len)
  Function to calculate the sum of an array.
- float array_dot_product (const float *in1, const float *in2, size_t len)
  Function to calculate the dot-product of two 1-D vectors/arrays.
- void array_right_shift (float *arr, size_t len)
  Function to right shift an array by one place. Left most value will be replaced by zero.
- void array_multiply_const (float *arr, float constant, size_t len)
  Function to multiply each element of an array by a scalar constant.
- void array_add_const (float *arr, float constant, size_t len)
  Function to add a scalar constant to each element of an array.
- void array_add_array (float *in1, const float *in2, size_t len)
  Function to do element wise addition of two arrays.
- void array_subtract_array (float *in1, const float *in2, size_t len)
  Function to do element wise subtraction of two arrays.
- void array_element_multiply_array (float *in1, const float *in2, size_t len)
  Function to do element wise multiplication of two arrays.
- void array_element_divide_array (float *in1, const float *in2, size_t len)
  Function to do element wise division of two arrays.
- float array_min (const float *arr, size_t len)
  Function to return the minimum of the elements of an array.
- float array_mean (float *arr, size_t len)
  Function to calculate the mean of the elements of an array.
- void array_square (const float *in, float *out, size_t len)
  Function to populate the output array with square of the elements of an input array.
- float array_mean_square (const float *arr, size_t len)
  Function to calculate the mean square of the elements of an array.
- void array_print (const char *str, float *arr, size_t len)
  Function to print an array for debugging.

7.1.1 Function Documentation

7.1.1.1 array_add_array()

```
void array_add_array (
            float * in1,
            const float * in2,
            size_t len )
```

Function to do element wise addition of two arrays.

Parameters

in1	Pointer to the first array
in2	Pointer to the second array
len	Length of the arrays

Warning

Assumes both the arrays are of same length and takes only one length parameter

7.1.1.2 array_add_const()

```
void array_add_const (
            float * arr,
            float constant,
            size_t len )
```

Function to add a scalar constant to each element of an array.

Parameters

arr	Pointer to the array
constant	The constant scalar adder
len	Length of the array

7.1.1.3 array_dot_product()

```
float array_dot_product (
            const float * in1,
            const float * in2,
            size_t len )
```

Function to calculate the dot-product of two 1-D vectors/arrays.

Parameters

in1	Pointer to the first vector
in2	Pointer to the second vector
len	Length of the vectors

Returns

Dot product (inner product) of the two vectors

Warning

Assumes both the vectors are of same length and takes only one length parameter

7.1.1.4 array_element_divide_array()

```
void array_element_divide_array (
            float * in1,
            const float * in2,
            size_t len )
```

Function to do element wise division of two arrays.

Parameters

in1	Pointer to the first array
in2	Pointer to the second array
len	Length of the arrays

Warning

Assumes both the arrays are of same length and takes only one length parameter

7.1.1.5 array_element_multiply_array()

```
void array_element_multiply_array (
            float * in1,
            const float * in2,
            size_t len )
```

Function to do element wise multiplication of two arrays.

Parameters

in1	Pointer to the first array
in2	Pointer to the second array
len	Length of the arrays

Warning

Assumes both the arrays are of same length and takes only one length parameter

7.1.1.6 array_flip()

```
void array_flip (
            float * arr,
            size_t len )
```

Function to reverse an array.

Parameters

arr	Pointer to the array
len	Length of the array

7.1.1.7 array_mean()

```
float array_mean (
            float * arr,
            size_t len )
```

Function to calculate the mean of the elements of an array.

Parameters

arr	Pointer to the array
len	Length of the array

Returns

Mean of the array elements

7.1.1.8 array_mean_square()

```
float array_mean_square (
            const float * arr,
            size_t len )
```

Function to calculate the mean square of the elements of an array.

Parameters

arr	Pointer to the array
len	Length of the array

Returns

Mean square of the array elements

7.1.1.9 array_min()

```
float array_min (
            const float * arr,
            size_t len )
```

Function to return the minimum of the elements of an array.

Parameters

arr	Pointer to the array
len	Length of the array

Returns

Minimum of the array elements

7.1.1.10 array_multiply_const()

```
void array_multiply_const (
            float * arr,
            float constant,
            size_t len )
```

Function to multiply each element of an array by a scalar constant.

Parameters

arr	Pointer to the array
constant	The constant scalar multiplier
len	Length of the array

7.1.1.11 array_print()

```
void array_print (
            const char * str,
            float * arr,
            size_t len )
```

Function to print an array for debugging.

Parameters

str	String to use for debugging
arr	Pointer to the array
len	Length of the array

7.1.1.12 array_right_shift()

```
void array_right_shift (
            float * arr,
            size_t len )
```

Function to right shift an array by one place. Left most value will be replaced by zero.

Parameters

arr	Pointer to the array
len	Length of the array

7.1.1.13 array_square()

```
void array_square (
            const float * in,
            float * out,
            size_t len )
```

Function to populate the output array with square of the elements of an input array.

Parameters

in	Pointer to the input array
out	Pointer to the output array
len	Length of the arrays

Warning

Assumes that output array already has memory allocated to it

7.1.1.14 array_subtract_array()

```
void array_subtract_array (
            float * in1,
            const float * in2,
            size_t len )
```

Function to do element wise subtraction of two arrays.

Parameters

in1	Pointer to the first array
in2	Pointer to the second array
len	Length of the arrays

Warning

Assumes both the arrays are of same length and takes only one length parameter

7.1.1.15 array_sum()

```
float array_sum (
            const float * arr,
            size_t len )
```

Function to calculate the sum of an array.

Parameters

arr	Pointer to the array
len	Length of the array

Returns

Sum of the array