The Open Master Hearing Aid (openMHA)

4.8.0

Plugin Developers' Manual

The Open Master Hearing Aid (openMHA) – Plugin Developers' Manual

HörTech gGmbH
Marie-Curie-Str. 2
D–26129 Oldenburg

LICENSE AGREEMENT

Contents

1 Overview

The HörTech Open Master Hearing Aid (openMHA), is a development and evaluation software platform that is able to execute hearing aid signal processing in real-time on standard computing hardware with a low delay between sound input and output.

1.1 Structure

The openMHA can be split into four major components :

- **The openMHA command line application (MHA)** (p. 34)
- **Signal processing plugins** (p. 7)
- Audio input-output (IO) plugins (see **io_file_t** (p. 365), **MHAIOJack** (p. 98), **io_parser_t** (p. 372), **io_tcp_parser_t** (p. 378))
- **The openMHA toolbox library** (p. 35)

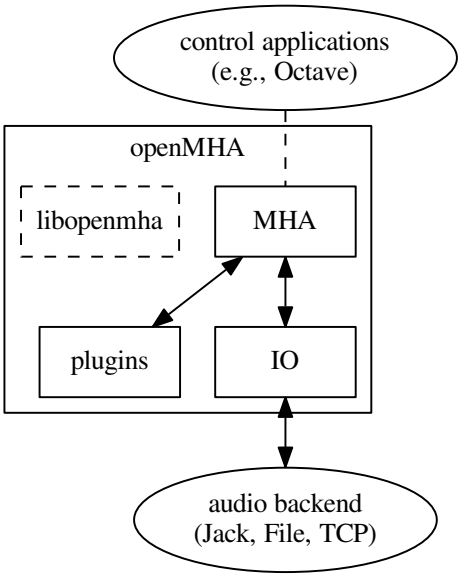


Figure 1 openMHA structure

The openMHA command line application (MHA) (p. 34) acts as a plugin host. It can load signal processing plugins as well as audio input-output (IO) plugins. Additionally, it provides the command line configuration interface and a TCP/IP based configuration interface. Several IO plugins exist: For real-time signal processing, commonly the openMHA **MHAIOJack** (p. 98) plugin (see plugins' manual) is used, which provides an interface to the Jack Audio Connection Kit (JACK). Other IO plugins provide audio file access or TCP/IP-based processing.

openMHA plugins (p. 7) provide the audio signal processing capabilities and audio signal handling. Typically, one openMHA plugin implements one specific algorithm. The complete virtual hearing aid signal processing can be achieved by a combination of several openMHA plugins.

1.2 Platform Services and Conventions

The openMHA platform offers some services and conventions to algorithms implemented in plugins, that make it especially well suited to develop hearing aid algorithms, while still supporting general-purpose signal processing.

1.2.1 Audio Signal Domains

As in most other plugin hosts, the audio signal in the openMHA is processed in audio chunks. However, plugins are not restricted to propagate audio signal as blocks of audio samples in the time domain another option is to propagate the audio signal in the short time Fourier transform (STFT) domain, i.e. as spectra of blocks of audio signal, so that not every plugin has to perform its own STFT analysis and synthesis. Since STFT analysis and re-synthesis of acceptable audio quality always introduces an algorithmic delay, sharing STFT data is a necessity for a hearing aid signal processing platform, because the overall delay of the complete processing has to be as short as possible.

Similar to some other platforms, the openMHA allows also arbitrary data to be exchanged between plugins through a mechanism called **algorithm communication variables** (p. 28) or short "AC vars". This mechanism is commonly used to share data such as filter coefficients or filter states.

1.2.2 Real-Time Safe Complex Configuration Changes

Hearing aid algorithms in the openMHA can export configuration settings that may be changed by the user at run time.

To ensure real-time safe signal processing, the audio processing will normally be done in a signal processing thread with real-time priority, while user interaction with configuration parameters would be performed in a configuration thread with normal priority, so that the audio processing does not get interrupted by configuration tasks. Two types of problems may occur when the user is changing parameters in such a setup:

- The change of a simple parameter exposed to the user may cause an involved recalculation of internal runtime parameters that the algorithm actually uses in processing. The duration required to perform this recalculation may be a significant portion of (or take even longer than) the time available to process one block of audio signal. In hearing aid usage, it is not acceptable to halt audio processing for the duration that the recalculation may require.
- If the user needs to change multiple parameters to reach a desired configuration state of an algorithm from the original configuration state, then it may not be acceptable that processing is performed while some of the parameters have already been changed while others still retain their original values. It is also not acceptable to interrupt signal processing until all pending configuration changes have been performed.

The openMHA provides a mechanism in its toolbox library to enable real-time safe configuration changes in openMHA plugins:

Basically, existing runtime configurations are used in the processing thread until the work of creating an updated runtime configuration has been completed in the configuration thread.

In hearing aids, it is more acceptable to continue to use an outdated configuration for a few more milliseconds than blocking all processing.

The openMHA toolbox library provides an easy-to-use mechanism to integrate real-time safe runtime configuration updates into every plugin.

1.2.3 Plugins can Themselves Host Other Plugins

An openMHA plugin can itself act as a plugin host. This allows to combine analysis and re-synthesis methods in a single plugin. We call plugins that can themselves load other plugins “bridge plugins" in the openMHA.

When such a bridge plugin is then called by the openMHA to process one block of signal, it will first perform its analysis, then invoke (as a function call) the signal processing in the loaded plugin to process the block of signal in the analysis domain, wait to receive a processed block of signal in the analysis domain back from the loaded plugin when the signal processing function call to that plugin returns, then perform the re-synthesis transform, and finally return the block of processed signal in the original domain back to the caller of the bridge plugin.

1.2.4 Central Calibration

The purpose of hearing aid signal processing is to enhance the sound for hearing impaired listeners. Hearing impairment generally means that people suffering from it have increased hearing thresholds, i.e. soft sounds that are audible for normal hearing listeners may be imperceptible for hearing impaired listeners. To provide accurate signal enhancement for hearing impaired people, hearing aid signal processing algorithms have to be able to determine the absolute physical sound pressure level corresponding to a digital signal given to any openM↩ HA plugin for processing. Inside the openMHA, we achieve this with the following convention: The single-precision floating point time-domain sound signal samples, that are processed inside the openMHA plugins in blocks of short durations, have the physical pressure unit Pascal ($1\mathrm{Pa} = 1\mathrm{N}/\mathrm{m}^2$). With this convention in place, all plugins can determine the absolute physical sound pressure level from the sound samples that they process. A derived convention is employed in the spectral domain for STFT signals. Due to the dependency of the calibration on the hardware used, it is the responsibility of the user of the openMHA to perform calibration measurements and adapt the openMHA settings to make sure that this calibration convention is met. We provide the plugin `transducers` which can be configured to perform the necessary signal adjustments.

2 Todo List

Class AuditoryProfile::profile_t (p. 226)

Give more documentation; implement all parts of the auditory profile.

Class mha_stash_environment_variable_t (p. 466)

Move to collection of unit-test support classes when we have one.

Class mhaconfig_t (p. 504)

Add information on number of bands and on center frequencies, or replace by **mha_audio_descriptor_t** (p. 427).

Class MHAFilter::filter_t (p. 531)

Implement a more robust filter form.

Member MHAFilter::polyphase_resampling_t::now_index (p. 563)

Index into what? What is the meaning of now?

Class MHAPlugin::plugin_t< runtime_cfg_t > (p. 728)

Describe all services provided by this class, so that the reason why it is recommended that all plugins use this class as their base is evident. Document all relevant methods and fields.

3 Module Documentation

3.1 Concept of Variables and Data Exchange in the openMHA

Accessibility of configuration variables and data exchange between plugins (processing blocks) are an important issue in the openMHA.

Accessibility of configuration variables and data exchange between plugins (processing blocks) are an important issue in the openMHA.

In general, variable types in the openMHA are distinguished by their different access methods. The variable types in the openMHA are:

- **Configuration variables** : Read and write accesses are possible through the openMHA configuration language interface. Configuration variables are implemented as C++ classes with a public data member of the underlying C type. Configuration variables can be read and modified from “outside" using the configuration language. The plugin which provides the configuration variable can use the exposed data member directly. All accesses through the openMHA configuration language are checked for data type, valid range, and access restrictions.
- **Monitor variables** : Read access is possible through the openMHA configuration language. Write access is only possible from the C++ code. Internally, monitor variables have a similar C++ class interface as configuration variables.

- **AC variables** (**algorithm communication variables** (p. 28)): Any C or C++ data structure can be shared within an openMHA chain. Access management and name space is realised in openMHA chain plugin ('mhachain'). AC variables are not available to the openMHA configuration language interface, although a read-only converter plugin `acmon` is available.

- **Runtime configuration** : Algorithms usually derive more parameters (runtime configuration) from the openMHA configuration language variables. When a configuration variable changes through configuration language write access, then the runtime configuration has to be recomputed. Plugin developers are encouraged to encapsulate the runtime configuration in a C++ class, which recomputes the runtime configuration from configuration variables in the constructor. The openMHA supports lock-free and thread-safe replacement of the runtime configuration instance (see **example5.cpp** (p. 21) and references therein).

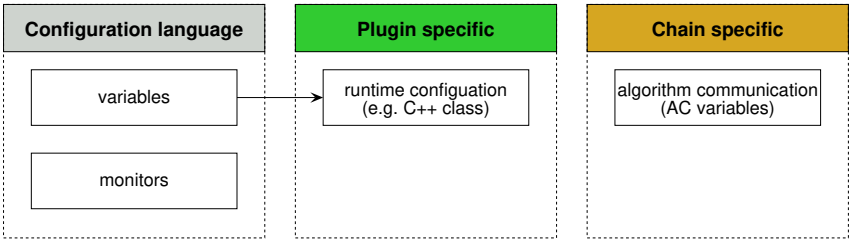


Figure 2 Variable types in the openMHA

The C++ data types are shown in the figure below. These variables can be accessed via the openMHA host application using the openMHA configuration language. For more details see 'Application engineers' manual'.

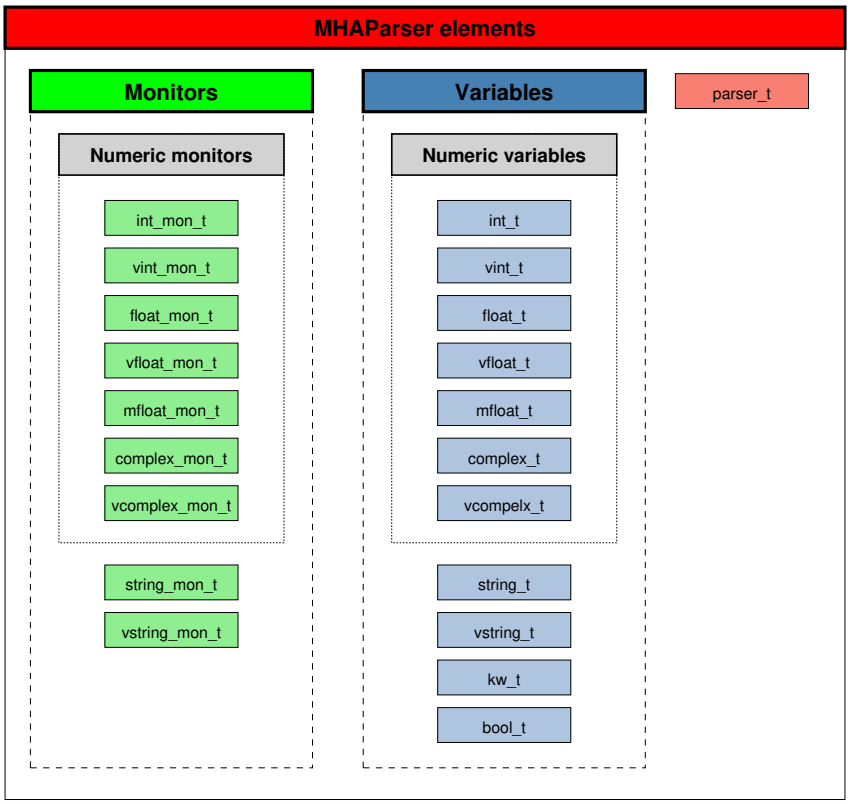


Figure 3 MHAParser elements

3.2 The openMHA Plugins (programming interface)

An openMHA plugin is the signal processing unit, usually an algorithm.

Classes

- class **MHAPlugin::plugin_t**< **runtime_cfg_t** >
  The template class for C++ openMHA plugins.

Macros

- #define **MHAPLUGIN_CALLBACKS_PREFIX**(prefix, classname, indom, outdom)
  C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_CALLBACKS**(plugname, classname, indom, outdom) **MHAPLUGIN_CALLBACKS_PREFIX**(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)
  C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION**(plugname, cat, doc) **MHAPLUGIN_DOCUMENTATION_PREFIX**(MHA_STATIC_ ## plugname ## _,cat,doc)
  Wrapper macro for the plugin documentation interface.

3.2.1 Detailed Description

An openMHA plugin is the signal processing unit, usually an algorithm.

openMHA plugins can be combined into processing chains. One of the configured chains can be selected for output which allows direct comparison of single algorithms or complex signal processing configurations. Algorithms within one chain can communicate with each other by sharing some of their variables, see section **Communication between algorithms** (p. 28).

The openMHA plugins can use the openMHA configuration language for their configuration. If they do so, the configuration can be changed through the framework even at run time. A description of this language can be found in section **The openMHA configuration language** (p. 34). If the algorithms should make use of the openMHA configuration language, they need to be written in C++ rather than pure C.

In the openMHA package a set of example plugins is included. These examples are the base of a step by step tutorial on how to write an openMHA plugin. See section **Writing openMHA Plugins. A step-by-step tutorial** (p. 11) for details.

openMHA plugins communicate with the openMHA using a simple C language interface. This way it is easy to mix plugins compiled with different C++ compilers. For convenience, we provide C++ classes which can be connected to the C++ interface. We strongly recommend the usage of these C++ wrappers. They include out-of-the box support exporting variables to the configuration interface and for thread safe configuration update.

The openMHA C++ plugin interface consists of a few number of method prototypes:

The output domain (spectrum or waveform) of an openMHA plugin will typically be the same as the input domain:

- **mha_wave_t** (p. 496) * process(**mha_wave_t** (p. 496) *): pure waveform processing
- **mha_spec_t** (p. 465) * process(**mha_spec_t** (p. 465) *): pure spectral processing

But it is also possible to implement domain transformations (from the time domain into spectrum or vice versa). The corresponding method signatures are:

- **mha_spec_t** (p. 465) * process(**mha_wave_t** (p. 496) *): Domain transformation from waveform to spectrum
- **mha_wave_t** (p. 496) * process(**mha_spec_t** (p. 465) *): Domain transformation from spectrum to waveform

For preparation and release of a plugin, the methods

- void prepare(**mhaconfig_t** (p. 504) &) and
- void release(void)

have to be implemented. The openMHA will call the `process()` method only ater the prepare method has returned and before `release()` is invoked. It is guarantteed by the openMHA framework that signal processing is performed only between calls of `prepare()` and `release()`. Each call of `prepare()` is followed by a call of `release()` (after some optional signal processing).

For configuration purposes, the plugin class has to export a method called `parse()` which implements the openMHA configuration language. We strongly recommend that you do not implement this method yourself, but by inheriting from the class **MHAParser::parser_t** (p. 688) from the openMHA toolbox, directly or indirectly (inheriting from a class that itself inherits from **MHAParser::parser_t** (p. 688)).

3.2.2 Connecting the C++ class with the C Interface

A C++ class which provides the appropriate methods can be used as an openMHA Plugin by connecting it to the C interface using the **MHAPLUGIN_CALLBACKS** (p. 10) macro.

The openMHA Toolbox library provides a base class **MHAPlugin::plugin_t** (p. 728)<T> (a template class) which can be used as the base class for a plugin class. This base class implements some necessary features for openMHA plugin developers like integration into the openMHA configuration language environment (it inherits from **MHAParser::parser_t** (p. 688)) and thread-safe runtime configuration update.

3.2.3 Error reporting

When your plugin detects a situation that it cannot handle, like input signal of the wrong signal domain at preparation time, unsupported number of input channels at preparation time, unsupported combinations of values in the plugin's variables during configuration, it should throw a C++ exception. The exception should be of type MHAError. Exceptions of this type are caught by the **MHAPLUGIN_CALLBACKS** (p. 10) macro for further error Reporting.

Throwing exceptions in response to unsupported configuration changes does not stop the signal processing. The openMHA configuration language parser will restore the previous value of that variable and report an error to the configurator, while the signal processing continues. Throwing exceptions from the signal processing thread will terminate the signal processing. Therefore, you should generally avoid throwing exceptions from the process method. Only do this if you detected a defect in your plugin, and then you should include enough information in the error message to be able to fix the defect.

3.2.4 Contents of the openMHA Plugin programming interface

3.2.5 Macro Definition Documentation

3.2.5.1 #define MHAPLUGIN_CALLBACKS_PREFIX(*prefix,* *classname,* *indom,* *outdom*)

C++ wrapper macro for the plugin interface.

Parameters

classname	The name of the plugin class
indom	Input domain (`wave` or `spec`)
outdom	Output domain (`wave` or `spec`)

This macro defines all required openMHA Plugin interface functions and passes calls of these functions to the corresponding member functions of the class `'classname'`. The parameters `'indom'` and `'outdom'` specify the input and output domain of the processing method. The `MHAInit()` and `MHADestroy()` functions will create or destroy an instance of the class. The approriate member functions have to be defined in the class. It is suggested to make usage of the **MHAPlugin::plugin_t** (p. 728) template class. Exceptions of type **MHA_Error** (p. 445) are caught and transformed into apropriate error codes with their corresponding error messages.

3.2.5.2 #define MHAPLUGIN_CALLBACKS(*plugname,* *classname,* *indom,* *outdom*) MHAPLUGIN_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)

C++ wrapper macro for the plugin interface.

Parameters

plugname	The file name of the plugin without the .so or .dll extension
classname	The name of the plugin class
indom	Input domain (`wave` or `spec`)
outdom	Output domain (`wave` or `spec`)

This macro defines all required openMHA Plugin interface functions and passes calls of these functions to the corresponding member functions of the class ‘classname'. The parameters ‘indom' and ‘outdom' specify the input and output domain of the processing method. The MHA↩ Init() and MHADestroy() functions will create or destroy an instance of the class. The approriate member functions have to be defined in the class. It is suggested to make usage of the **MH↩ APlugin::plugin_t** (p. 728) template class. Exceptions of type **MHA_Error** (p. 445) are caught and transformed into apropriate error codes with their corresponding error messages.

3.2.5.3 #define MHAPLUGIN_DOCUMENTATION(*plugname,* *cat,* *doc*) MHAPLUGIN_DOCUMENTATION_PREFIX(MHA_STATIC_ ## plugname ## _,cat,doc)

Wrapper macro for the plugin documentation interface.

Parameters

plugin	The file name of the plugin without the .so or .dll extension
cat	Space separated list of categories to which belong the plugin (as const char∗)
doc	Documentation of the plugin (as const char∗)

This macro defines the openMHA Plugin interface function for the documentation. The categories can be any space seperated list of category names. An empty string will categorize the plugin in the category 'other'.

The documentation should contain a description of the plugin including a description of the underlying models, and a paragraph containing hints for usage. The text should be LaTeX compatible (e.g., avoid or quote underscores in the text part); equations should be formatted as LaTeX.

3.3 Writing openMHA Plugins. A step-by-step tutorial

A step-by-step tutorial on writing openMHA plugins.

A step-by-step tutorial on writing openMHA plugins.

openMHA contains a small number of example plugins as C++ source code. They are meant to help developers in understanding the concepts of openMHA plugin programming starting from the simplest example and increasing in complexity. This tutorial explains the basic parts of the example files.

3.3.1 example1.cpp

The example plugin file **example1.cpp** (p. 976) demonstrates the easiest way to implement an openMHA Plugin. It attenuates the sound signal in the first channel by multiplying the sound samples with a factor. The plugin class **MHAPlugin::plugin_t** (p. 728) exports several methods, but only two of them need a non-empty implementation: `prepare()` method is a pure virtual function and `process()` is called when signal processing starts.

```
#include "mha_plugin.hh"

class example1_t : public MHAPlugin::plugin_t<int> {
public:
  example1_t(algo_comm_t & ac,
             const std::string & chain_name,
             const std::string & algo_name)
      : MHAPlugin::plugin_t<int>("",ac)
  {/* Do nothing in constructor */}

  void release(void)
  {/* Do nothing in release */}
```

Every plugin implementation should include the '**mha_plugin.hh** (p. 1022)' header file. C++ helper classes for plugin development are declared in this header file, and most header files needed for plugin development are included by **mha_plugin.hh** (p. 1022).

The class plugin1_t inherits from the class **MHAPlugin::plugin_t** (p. 728), which then inherits from **MHAParser::parser_t** (p. 688) – the configuration language interface in the method "parse". Our plugin class therefore exports the working "parse" method inherited from **MHAParser::parser_t** (p. 688), and the plugin is visible in the openMHA configuration tree.

The constructor has to accept 3 parameters of correct types. In this simple example, we do not make use of them.

The `release()` method is used to free resources after signal processing. In this simple example, we do not allocate resources, so there is no need to free them.

3.3.1.1 The prepare method

```
  void prepare(mhaconfig_t & signal_info)
  {
    if (signal_info.domain != MHA_WAVEFORM)
      throw MHA_Error(__FILE__, __LINE__,
                      "This plugin can only process waveform signals.");
    if (signal_info.channels < 1)
      throw MHA_Error(__FILE__,__LINE__,
                      "This plugin requires at least one input channel.");
  }
```

Parameters

signal_info	Contains information about the input signal's parameters, see **mhaconfig_t** (p. 504).

The `prepare()` method of the plugin is called before the signal processing starts, when the input signal parameters like domain, number of channels, frames per block, and sampling rate are known. The `prepare()` method can check these values and raise an exception if the plugin cannot cope with them, as is done here. The plugin can also change these values if the signal processing performed in the plugin results in an output signal with different parameters. This plugin does not change the signal's parameters, therefore they are not modified here.

3.3.1.2 The signal processing method

```
  mha_wave_t * process(mha_wave_t * signal)
  {
    unsigned int channel = 0; // channels and frames counting starts with 0
    float factor = 0.1f;
    unsigned int frame;

    // Scale channel number "channel" by "factor":
    for(frame = 0; frame < signal->num_frames; frame++) {
      // Waveform channels are stored interleaved.
      signal->buf[signal->num_channels * frame + channel] *= factor;
    }
    // Algorithms may process data in-place and return the input signal
    // structure as their output signal:
    return signal;
  }
};
```

Parameters

signal	Pointer to the input signal structure **mha_wave_t** (p. 496).

Returns

Pointer to the output signal structure. The input signal structure may be reused if the signal has the same domain and dimensions.

The plugin works with time domain input signal (indicated by the data type **mha_wave_t** (p. 496) of the process method's parameter). It scales the first channel by a factor of 0.1. The output signal reuses the structure that previously contained the input signal (in-place processing).

3.3.1.3 Connecting the C++ class with the C plugin interface

Plugins have to export C functions as their interface (to avoid C++ name-mangling issues and other incompatibilities when mixing plugins compiled with different C++ compilers).

```
MHAPLUGIN_CALLBACKS(example1,example1_t,wave,wave)
```

This macro takes care of accessing the C++ class from the C functions required as the plugin's interface. It implements the C funtions and calls the corresponding C++ instance methods. Plugin classes should be derived from the template class **MHAPlugin::plugin_t** (p. 728) to be compatible with the C interface wrapper.

This macro also catches C++ exceptions of type **MHA_Error** (p. 445), when raised in the methods of the plugin class, and reports the error using an error flag as the return value of the underlying C function. It is therefore important to note that only C++ exceptions of type **MHA_Error** (p. 445) may be raised by your plugin. If your code uses different Exception classes, you will have to catch them yourself before control leaves your plugin class, and maybe report the error by throwing an instance of **MHA_Error** (p. 445). This is important, because: (1) C++ exceptions cannot cross the plugin interface, which is in C, and (2) there is no error handling code for your exception classes in the openMHA framework anyways.

3.3.2 example2.cpp

This is another simple example of openMHA plugin written in C++. This plugin also scales one channel of the input signal, working in the time domain. The scale factor and which channel to scale (index number) are made accessible to the configuration language.

The algorithm is again implemented as a C++ class.

```
class example2_t : public MHAPlugin::plugin_t<int> {
    MHAParser::int_t scale_ch;
    MHAParser::float_t factor;
public:
    example2_t(algo_comm_t & ac,
               const std::string & chain_name,
               const std::string & algo_name);

    void prepare(mhaconfig_t & signal_info);

    void release(void);

    mha_wave_t * process(mha_wave_t * signal);
};
```

Parameters

scale_ch	– the channel number to be scaled
factor	– the scale factor of the scaling.

This class again inherits from the template class **MHAPlugin::plugin_t** (p. 728) for intergration with the openMHA configuration language. The two data members serve as externally visible configuration variables. All methods of this class have a non-empty implementation.

3.3.2.1 Constructor

```
example2_t::example2_t(algo_comm_t & ac,
                       const std::string & chain_name,
```

```
                        const std::string & algo_name)
    : MHAPlugin::plugin_t<int>("This plugin multiplies the sound signal"
                               " in one audio channel by a factor",ac),
      scale_ch("Index of audio channel to scale. Indices start from 0.",
               "0",
               "[0,["),
      factor("The scaling factor that is applied to the selected channel.",
             "0.1",
             "[0,[")
{
    insert_item("channel", &scale_ch);
    insert_item("factor", &factor);
}
```

The constructor invokes the superclass constructor with a string parameter. This string parameter serves as the help text that describes the functionality of the plugin. The constructor registers configuration variables with the openMHA configuration tree and sets their default values and permitted ranges. The minimum permitted value for both variables is zero, and there is no maximum limit (apart from the limitations of the underlying C data type). The configuration variables have to be registered with the parser node instance using the **MHAParser::parser_t::insert_item** (p. 690) method.

3.3.2.2 The prepare method

```
void example2_t::prepare(mhaconfig_t & signal_info)
{
    if (signal_info.domain != MHA_WAVEFORM)
        throw MHA_Error(__FILE__, __LINE__,
                        "This plugin can only process waveform signals.");
    // The user may have configured scale_ch before prepare is called.
    // Check that the configured channel is present in the input signal.
    if (signal_info.channels <= unsigned(scale_ch.data))
        throw MHA_Error(__FILE__,__LINE__,
                        "This plugin requires at least %d input channels.",
                        scale_ch.data + 1);
    // Adjust the range of the channel configuration variable so that it
    // cannot be set to an out-of-range value during processing.
    using MHAParser::StrCnv::val2str;
    scale_ch.set_range("[0," + val2str(int(signal_info.channels)) + "[");
}
```

Parameters

signal_info	– contains information about the input signal's parameters, see **mhaconfig_t** (p. 504).

The user may have changed the configuration variables before preparing the openMHA plugin. A consequence of this is that it is not sufficient any more to check if the input signal has at least 1 audio channel.

Instead, this prepare method checks that the input signal has enough channels so that the current value of `scale_ch.data` is a valid channel index, i.e. $0 \leq$ `scale_ch.data` $<$ `signal_info.channels`. The prepare method does not have to check that $0 \leq$ `scale_ch.data`, since this is guaranteed by the valid range setting of the configuration variable.

The prepare method then modifies the valid range of the `scale_ch` variable, it modifies the upper bound so that the user cannot set the variable to a channel index higher than the available channels. Setting the range is done using a string parameter. The prepare method contatenates a string of the form "[0,n[". n is the number of channels in the input signal, and is used here as an exclusive upper boundary. To convert the number of channels into a string, a helper function for string conversion from the openMHA Toolbox is used. This function is overloaded and works for several data types.

It is safe to assume that the value of configuration variables does not change while the prepare method executes, since openMHA preparation is triggered from a configuration language command, and the openMHA configuration language parser is busy and cannot accept other commands until all openMHA plugins are prepared (or one of them stops the process by raising an exception). As we will see later in this tutorial, the same assumption cannot be made for the process method.

3.3.2.3 The release method

```
void example2_t::release(void)
{
    scale_ch.set_range("[0,[");
}
```

The release method should undo the state changes that were performed by the prepare method. In this example, the prepare method has reduced the valid range of the `scale_ch`, so that only valid channels could be selected during signal processing.

The release method reverts this change by setting the valid range back to its original value, "[0,[".

3.3.2.4 The signal processing method

```
mha_wave_t * example2_t::process(mha_wave_t * signal)
{
    unsigned int frame;
    for(frame = 0; frame < signal->num_frames; frame++)
        value(signal,frame,scale_ch.data) *= factor.data;
    return signal;
}
```

The processing function uses the current values of the configuration variables to scale every frame in the selected audio channel.

Note that the value of each configuration variable can change while the processing method executes, since the process method usually executes in a different thread than the configuration interface.

For this simple plugin, this is not a problem, but for more advanced plugins, it has to be taken into consideration. The next section takes a closer look at the problem.

Consistency

Assume that one thread reads the value stored in a variable while another thread writes a new value to that variable concurrently. In this case, you may have a consistency problem. You would perhaps expect that the value retrieved from the variable either (a) the old value, or (b) the new value, but not (c) something else. Yet generally case (c) is a possibility.

Fortunately, for some data types on PC systems, case (c) cannot happen. These are 32bit wide data types with a 4-byte alignment. Therefore, the values in **MHAParser::int_t** (p. 664) and **MHAParser::float_t** (p. 659) are always consistent, but this is not the case for vectors, strings, or complex values. With these, you can get a mixture of the bit patterns of old and new values, or you can even cause a memory access violation in case a vector or string grows and has to be reallocated to a different memory address.

There is also a consistency problem if you take the combination of two "safe" datatypes. The openMHA provides a mechanism that can cope with these types of problems. This thread-safe runtime configuration update mechanism is introduced in example 5.

3.3.3 example3.cpp

This example introduces the openMHA Event mechanism. Plugins that provide configuration variable can receive a callback from the parser base class when a configuration variable is accessed through the configuration language interface.

The third example performes the same processing as before, but now only even channel indices are permitted when selecting the audio channel to scale. This restriction cannot be ensured by setting the range of the channel index configuration variable. Instead, the event mechanism of openMHA configuration variables is used. Configuration variables emit 4 different events, and your plugin can connect callback methods that are called when the events are triggered. These events are:

writeaccess

- triggered on write access to a configuration variable.

valuechanged

- triggered when write access to a configuration variable actually changes the value of this variable.

readaccess

- triggered after the value of the configuration variable has been read.

prereadaccess

- triggered before the value of a configuration variable is read, i.e. the value of the requested variable can be changed by the callback to implement computation on demand.

All of these callbacks are executed in the configuration thread. Therefore, the callback implementation does not have to be realtime-safe. No other updates of configuration language variables through the configuration language can happen in parallel, but your processing method can execute in parallel and may change values.

3.3.3.1 Data member declarations

```
class example3_t : public MHAPlugin::plugin_t<int> {
    MHAParser::int_t scale_ch;
    MHAParser::float_t factor;
    MHAParser::int_mon_t prepared;

    MHAEvents::patchbay_t<example3_t> patchbay;
```

This plugin exposes another configuration variable, "`prepared`", that keeps track of the prepared state of the plugin. This is a read-only (monitor) integer variable, i.e. its value can only be changed by your plugin's C++ code. When using the configuration language interface, the value of this variable can only be read, but not changed.

The patchbay member is an instance of a connector class that connects event sources with callbacks.

3.3.3.2 Method declarations

```
    /* Callbacks triggered by Events */
    void on_scale_ch_writeaccess();
    void on_scale_ch_valuechanged();
    void on_scale_ch_readaccess();
    void on_prereadaccess();
public:
    example3_t(algo_comm_t & ac,
               const std::string & chain_name,
               const std::string & algo_name);

    void prepare(mhaconfig_t & signal_info);

    void release(void);

    mha_wave_t * process(mha_wave_t * signal);
};
```

This plugin exposes 4 callback methods that are triggered by events. Multiple events (from the same or different configuration variables) can be connected to the same callback method, if desired.

This example plugin uses the `valuechanged` event to check that the `scale_ch` configuration variable is only set to valid values.

The other callbacks only cause log messages to stdout, but the comments in the logging callbacks give a hint when listening on the events would be useful.

3.3.3.3 Example 3 constructor

```
example3_t::example3_t(algo_comm_t & ac,
                       const std::string & chain_name,
                       const std::string & algo_name)
    : MHAPlugin::plugin_t<int>("This plugin multiplies the sound signal"
                               " in one audio channel by a factor",ac),
      scale_ch("Index of audio channel to scale. Indices start from 0."
               " Only channels with even indices may be scaled.",
               "0",
               "[0,["),
      factor("The scaling factor that is applied to the selected channel.",
             "0.1",
             "[0,["),
      prepared("State of this plugin: 0 = unprepared, 1 = prepared")
{
    insert_item("channel", &scale_ch);
    insert_item("factor", &factor);
    prepared.data = 0;
    insert_item("prepared", &prepared);

    patchbay.connect(&scale_ch.writeaccess, this,
                     &example3_t::on_scale_ch_writeaccess);
    patchbay.connect(&scale_ch.valuechanged, this,
                     &example3_t::on_scale_ch_valuechanged);
    patchbay.connect(&scale_ch.readaccess, this,
                     &example3_t::on_scale_ch_readaccess);
    patchbay.connect(&scale_ch.prereadaccess, this,
                     &example3_t::on_prereadaccess);
    patchbay.connect(&factor.prereadaccess, this,
                     &example3_t::on_prereadaccess);
    patchbay.connect(&prepared.prereadaccess, this,
                     &example3_t::on_prereadaccess);
}
```

The constructor of monitor variables does not take a parameter for setting the initial value. The single parameter here is the help text describing the contents of the read-only variable. If the initial value should differ from 0, then the `.data` member of the configuration variable has to be set to the initial value in the plugin constructor's body explicitly, as is done here for demonstration although the initial value of this monitor variable is 0.

Events and callback methods are then connected using the patchbay member variable.

3.3.3.4 The prepare method

```
void example3_t::prepare(mhaconfig_t & signal_info)
{
    if (signal_info.domain != MHA_WAVEFORM)
        throw MHA_Error(__FILE__, __LINE__,
                        "This plugin can only process waveform signals.");
    // The user may have configured scale_ch before prepare is called.
    // Check that the configured channel is present in the input signal.
    if (signal_info.channels <= unsigned(scale_ch.data))
        throw MHA_Error(__FILE__,__LINE__,
                        "This plugin requires at least %d input channels.",
                        scale_ch.data + 1);
    // bookkeeping
    prepared.data = 1;
}
```

The prepare method checks wether the current setting of the scale_ch variable is possible with the input signal dimension. It does not adjust the range of the variable, since the range alone is not sufficient to ensure all future settings are also valid: The scale channel index has to be even.

3.3.3.5 The release method

```
void example3_t::release(void)
{
  prepared.data = 0;
}
```

The release method is needed for tracking the prepared state only in this example.

3.3.3.6 The signal processing method

```
mha_wave_t * example3_t::process(mha_wave_t * signal)
{
    unsigned int frame;
    for(frame = 0; frame < signal->num_frames; frame++)
        value(signal,frame,scale_ch.data) *= factor.data;
    return signal;
}
```

The signal processing member function is the same as in example 2.

3.3.3.7 The callback methods

```
void example3_t::on_scale_ch_writeaccess()
{
    printf("Write access: Attempt to set scale_ch=%d.\n", scale_ch.data);
    // Can be used to track any writeaccess to the configuration, even
    // if it does not change the value.  E.g. setting the name of the
    // sound file in a string configuration variable can cause a sound
    // file player plugin to start playing the sound file from the
    // beginning.
}
void example3_t::on_scale_ch_valuechanged()
{
    if (scale_ch.data & 1)
        throw MHA_Error(__FILE__,__LINE__,
                        "Attempt to set scale_ch to non-even value %d",
                        scale_ch.data);
    // Can be used to recompute a runtime configuration only if some
    // configuration variable actually changed.
}
void example3_t::on_scale_ch_readaccess()
{
    printf("scale_ch has been read.\n");
    // A configuration variable used as an accumulator can be reset
    // after it has been read.
}
void example3_t::on_prereadaccess()
{
    printf("A configuration language variable is about to be read.\n");
    // Can be used to compute the value on demand.
}

MHAPLUGIN_CALLBACKS(example3,example3_t,wave,wave)
```

When the `writeaccess` or `valuechanged` callbacks throw an MHAError exception, then the change made to the value of the configuration variable is reverted.

If multiple event sources are connected to a single callback method, then it is not possible to determine which event has caused the callback to execute. Often, this information is not crucial, i.e. when the answer to a change of any variable in a set of variables is the same, e.g. the recomputation of a new runtime configuration that takes all variables of this set as input.

3.3.4 example4.cpp

This plugin is the same as example 3 except that it works on the spectral domain (STFT).

3.3.4.1 The Prepare method

```
void example4_t::prepare(mhaconfig_t & signal_info)
{
    if (signal_info.domain != MHA_SPECTRUM)
        throw MHA_Error(__FILE__, __LINE__,
                        "This plugin can only process spectrum signals.");
    // The user may have configured scale_ch before prepare is called.
    // Check that the configured channel is present in the input signal.
    if (signal_info.channels <= unsigned(scale_ch.data))
        throw MHA_Error(__FILE__,__LINE__,
                        "This plugin requires at least %d input channels.",
                        scale_ch.data + 1);
    // bookkeeping
    prepared.data = 1;
}
```

The prepare method now checks that the signal domain is MHA_SPECTRUM.

3.3.4.2 The signal processing method

```
mha_spec_t * example4_t::process(mha_spec_t * signal)
{
    unsigned int bin;
    // spectral signal is stored non-interleaved.
    mha_complex_t * channeldata =
        signal->buf + signal->num_frames * scale_ch.data;
    for(bin = 0; bin < signal->num_frames; bin++)
        channeldata[bin] *= factor.data;
    return signal;
}
```

The signal processing member function works on the spectral signal instead of the wave signal as before.

The **mha_spec_t** (p. 465) instance stores the complex (**mha_complex_t** (p. 431)) spectral signal for positive frequences only (since the waveform signal is always real). The num_frames member of **mha_spec_t** (p. 465) actually denotes the number of STFT bins.

Please note that different from **mha_wave_t** (p. 496), a multichannel signal in **mha_spec_t** (p. 465) is stored non-interleaved in the signal buffer.

Some arithmetic operations are defined on struct **mha_complex_t** (p. 431) to facilitate efficient complex computations. The $*=$ operator used here (defined for real and for complex arguments) is one of them.

3.3.4.3 Connecting the C++ class with the C plugin interface

```
MHAPLUGIN_CALLBACKS(example4,example4_t,spec,spec)
```

When connecting a class that performs spectral processing with the C interface, use `spec` instead of `wave` as the domain indicator.

3.3.5 example5.cpp

Many algorithms use complex operations to transform the user space variables into run time configurations. If this takes a noticeable time (e.g. more than 100-500 μ sec), the update of the runtime configuration can not take place in the real time processing thread. Furthermore, the parallel access to complex structures may cause unpredictable results if variables are read while only parts of them are written to memory (cf. section **Consistency** (p. 16)). To handle these situations, a special C++ template class **MHAPlugin::plugin_t** (p. 728) was designed. This class helps keeping all access to the configuration language variables in the **configuration** thread rather than in the **processing** thread.

The runtime configuration class **example5_t** (p. 315) is the parameter of the template class **MHAPlugin::plugin_t** (p. 728). Its constructor converts the user variables into a runtime configuration. Because the constructor executes in the configuration thread, there is no harm if the constructor takes a long time. All other member functions and data members of the runtime configurations are accessed only from the signal processing thread (real-time thread).

```
class example5_t {
public:
    example5_t(unsigned int,unsigned int,mha_real_t);
    mha_spec_t* process(mha_spec_t*);
private:
    unsigned int channel;
    mha_real_t scale;
};
```

The plugin interface class inherits from the plugin template class **MHAPlugin::plugin_t** (p. 728), parameterised by the runtime configuration. Configuration changes (write access to the variables) will emit a write access event of the changed variables. These events can be connected to member functions of the interface class by the help of a **MHAEvents::patchbay_t** (p. 512) instance.

```
class plugin_interface_t : public MHAPlugin::plugin_t<example5_t> {
public:
    plugin_interface_t(const algo_comm_t&,const std::string&,const std::string&);
    mha_spec_t* process(mha_spec_t*);
    void prepare(mhaconfig_t&);
private:
    void update_cfg();
    /* integer variable of MHA-parser: */
    MHAParser::int_t scale_ch;
    /* float variable of MHA-parser: */
    MHAParser::float_t factor;
    /* patch bay for connecting configuration parser
       events with local member functions: */
    MHAEvents::patchbay_t<plugin_interface_t> patchbay;
};
```

The constructor of the runtime configuration analyses and validates the user variables. If the configuration is invalid, an exception of type **MHA_Error** (p. 445) is thrown. This will cause the openMHA configuration language command which caused the change to fail: The modified configuration language variable is then reset to its original value, and the error message will contain the message string of the **MHA_Error** (p. 445) exception.

```
example5_t::example5_t(unsigned int ichannel,
                       unsigned int numchannels,
                       mha_real_t iscale)
    : channel(ichannel),scale(iscale)
{
    if( channel >= numchannels )
        throw MHA_Error(__FILE__,__LINE__,
                        "Invalid channel number %d (only %d channels configured).",
                        channel,numchannels);
}
```

In this example, the run time configuration class **example5_t** (p. 315) has a signal processing member function. In this function, the selected channel is scaled by the given scaling factor.

```
mha_spec_t* example5_t::process(mha_spec_t* spec)
{
    /* Scale channel number "scale_ch" by "factor": */
    for(unsigned int fr = 0; fr < spec->num_frames; fr++){
        spec->buf[fr + channel * spec->num_frames].re *= scale;
        spec->buf[fr + channel * spec->num_frames].im *= scale;
    }
    return spec;
}
```

The constructor of the example plugin class is similar to the previous examples. A callback triggered on write access to the variables is registered using the **MHAEvents::patchbay_t** (p. 512) instance.

```
plugin_interface_t::plugin_interface_t(
    const algo_comm_t& iac,
    const std::string&,const std::string&)
    : MHAPlugin::plugin_t<example5_t>("example plugin configuration structure",iac),
      /* initialzing variable 'scale_ch' with MHAParser::int_t(char* name, .... ) */
      scale_ch("channel number to be scaled","0","[0,["),
      /* initialzing variable 'factor' with MHAParser::float_t(char* name, .... ) */
      factor("scale factor","1.0","[0,2]")
{
    /* Register variables to the configuration parser: */
    insert_item("channel",&scale_ch);
    insert_item("factor",&factor);
    /*
     * On write access to the parser variables a notify callback of
     * this class will be called. That funtion will update the runtime
     * configuration.
     */
    patchbay.connect(&scale_ch.writeaccess,this,&plugin_interface_t::update_cfg);
    patchbay.connect(&factor.writeaccess,this,&plugin_interface_t::update_cfg);
}
```

The processing function can gather the latest valid runtime configuration by a call of `poll_config`. On success, the class member `cfg` points to this configuration. On error, if there is no usable runtime configuration instance, an exception is thrown. In this example, the prepare method ensures that there is a valid runtime configuration, so that in this example, no error can be raised at this point. The prepare method is always executed before the process method is called. The runtime configuration class in this example provides a signal processing method. The process method of the plugin interface calls the process method of this instance to perform the actual signal processing.

```
mha_spec_t* plugin_interface_t::process(mha_spec_t* spec)
{
    poll_config();
    return cfg->process(spec);
}
```

The prepare method ensures that a valid runtime configuration exists by creating a new runtime configuration from the current configuration language variables. If the configuraion is invalid, then an exception of type **MHA_Error** (p. 445) is raised and the preparation of the openMHA fails with an error message.

```
void plugin_interface_t::prepare(mhaconfig_t& tfcfg)
{
    if( tfcfg.domain != MHA_SPECTRUM )
        throw MHA_Error(__FILE__,__LINE__,
                        "Example5: Only spectral processing is supported.");
    /* remember the transform configuration (i.e. channel numbers): */
    tftype = tfcfg;
    /* make sure that a valid runtime configuration exists: */
    update_cfg();
}
```

The update_cfg member function is called when the value of a configuration language variable changes, or from the prepare method. It allocates a new runtime configuration and registers it for later access from the real time processing thread. The function **push_config** (p. 726) stores the configuration in a FiFo queue of runtime configurations. Once they are inserted in the FiFo, the **MHAPlugin::plugin_t** (p. 728) template is responsible for deleting runtime configuration instances stored in the FiFo. You don't need to keep track of the created instances, and you must not delete them yourself.

```
void plugin_interface_t::update_cfg()
{
    if( tftype.channels )
        push_config(new example5_t(scale_ch.data,tftype.channels,factor.data));
}
```

In the end of the example code file, the macro **MHAPLUGIN_CALLBACKS** (p. 10) defines all ANSI-C interface functions and passes them to the corresponding C++ class member functions (partly defined by the **MHAPlugin::plugin_t** (p. 728) template class). All exceptions of type **MHA_Error** (p. 445) are caught and transformed into an appropriate error code and error message.

```
MHAPLUGIN_CALLBACKS(example5,plugin_interface_t,spec,spec)
```

3.3.6 example6.cpp

This last example is the same as the previous one, but it additionally creates an 'Algorithm Communication Variable' (AC variable). It calculates the RMS level of a given channel and stores it into this variable. The variable can be accessed by any other algorithm in the same chain. To store the data onto disk, the 'acsave' plugin can be used. 'acmon' is a plugin which converts AC variables into parsable monitor variables.

In the constructor of the plugin class the variable `rmsdb` is registered under the name `example6_rmslev` as a one-dimensional AC variable of type float. For registration of other types, read access and other detailed informations please see **Communication between algorithms** (p. 28).

```
example6_t::example6_t(const algo_comm_t& iac,
                       const std::string&,const std::string&)
    : MHAPlugin::plugin_t<cfg_t>("example plugin configuration structure",iac),
      /* initialzing variable 'channel_no' with MHAParser::int_t(char* name, .... ) */
      channel_no("channel in which the RMS level is measured","0","[0,[")
{
    /* Register variables to the configuration parser: */
    insert_item("channel",&channel_no);
    /*
     * On write access to the parser variables a notify callback of
     * this class will be called. That funtion will update the runtime
     * configuration.
     */
    patchbay.connect(&channel_no.writeaccess,this,&example6_t::update_cfg);
    /*
     * Propagate the level variable to all algorithms in the
     * processing chain. If multiple instances of this algorithm are
     * required, than it is necessary to use different names for this
     * variable (i.e. prefixing the name with the algorithm name
     * passed to MHAInit).
     */
    ac.insert_var_float( ac.handle, "example6_rmslev", &rmsdb );
}
```

3.3.7 Debugging openMHA plugins

Suppose you would want to step through the code of your openMHA plugin with a debugger. This example details how to use the linux gdb debugger to inspect the **example6_t::prepare()** (p. 317) and **example6_t::process()** (p. 317) routines of **example6.cpp** (p. 24) example 6.

First, make sure that your plugin is compiled with the compiler option to include debugging symbols: Apply the -ggdb switch to all gcc, g++ invocations.

Once the plugin is compiled, with debugging symbols, create a test configuration. For example 6, assuming there is an audio file named input.wav in your working directory, you could create a configuration file named 'debugexample6.cfg', with the following content:

```
# debugexample6.cfg
fragsize = 64
srate = 44100
nchannels_in = 2
iolib = MHAIOFile

io.in = input.wav
io.out = output.wav
mhalib = example6
mha.channel = 1
cmd=start
```

Assuming all your binaries and shared-object libraries are in your 'bin' directory (see README.md), you could start gdb using

```
$ export MHA_LIBRARY_PATH=$PWD/bin
$ gdb $MHA_LIBRARY_PATH/mha
```

Set breakpoints in prepare and process methods, and start execution. Note that specifying the breakpoint by symbol (**example6_t::prepare** (p. 317)) does not yet work, as the symbol lives in the openMHA plugin that has not yet been loaded. Specifying by line number works, however. Specifying the breakpoint by symbol also works once the plugin is loaded (i.e. when the debugger stops in the first break point). You can set the breakpoints like this (example shown here is run in gdb version 7.11.1):

```
(gdb) run ?read:debugexample6.cfg
Starting program: {openMHA_directory}/bin/mha ?read:debugexample6.cfg
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
The Open Master Hearing Aid (openMHA) server
Copyright (c) 2005-2019 HoerTech gGmbH, D-26129 Oldenburg, Germany

This program comes with ABSOLUTELY NO WARRANTY; for details see file COPYING.
This is free software, and you are welcome to redistribute it
under the terms of the GNU AFFERO GENERAL PUBLIC LICENSE, Version 3;
for details see file COPYING.


Breakpoint 1, example6_t::prepare (this=0x6478b0, tfcfg=...)
    at example6.cpp:192
192         if( tfcfg.domain != MHA_WAVEFORM )
(gdb) b example6.cpp:162
Breakpoint 2 at 0x7ffff589744a: file example6.cpp, line 162.
(gdb) c
Continuing.
```

Where ‘{openMHA_directory}' is the directory where openMHA is located (which should also be your working directory in this case). Next stop is the `process()` method. You can now examine and change the variables, step through the program as needed (using, for example ‘n' to step in the next line):

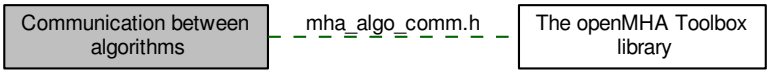
```
Breakpoint 2, example6_t::process (this=0x7ffff6a06c0d, wave=0x10a8b550)
    at example6.cpp:162
162     {
(gdb) n
163         poll_config();
(gdb)
```

3.4 The MHA Framework interface

3.5 Communication between algorithms

Algorithms within one chain can share variables for communication with other algorithms.

Collaboration diagram for Communication between algorithms:



Files

- file **mha_algo_comm.h**
  Header file for Algorithm Communication.

Namespaces

- **MHA_AC**
  Functions and classes for Algorithm Communication (AC) support.

Classes

- class **MHA_AC::spectrum_t**
  *Insert a **MHASignal::spectrum_t** (p. 800) class into the AC space.*
- class **MHA_AC::waveform_t**
  *Insert a **MHASignal::waveform_t** (p. 812) class into the AC space.*
- class **MHA_AC::int_t**
  Insert a integer variable into the AC space.
- class **MHA_AC::float_t**
  Insert a float point variable into the AC space.
- class **MHA_AC::double_t**
  Insert a double precision floating point variable into the AC space.
- class **MHA_AC::ac2matrix_t**
  Copy AC variable to a matrix.
- class **MHA_AC::acspace2matrix_t**
  Copy all or a subset of all numeric AC variables into an array of matrixes.
- struct **algo_comm_t**
  A reference handle for algorithm communication variables.
- struct **comm_var_t**
  Algorithm communication variable structure.

Functions

- **mha_spec_t MHA_AC::get_var_spectrum** (**algo_comm_t** ac, const std::string &name)

  Convert an AC variable into a spectrum.
- **mha_wave_t MHA_AC::get_var_waveform** (**algo_comm_t** ac, const std::string &name)

  Convert an AC variable into a waveform.
- int **MHA_AC::get_var_int** (**algo_comm_t** ac, const std::string &name)

  Return value of an integer scalar AC variable.
- float **MHA_AC::get_var_float** (**algo_comm_t** ac, const std::string &name)

  Return value of an floating point scalar AC variable.
- std::vector< float > **MHA_AC::get_var_vfloat** (**algo_comm_t** ac, const std::string &name)

  Return value of an floating point vector AC variable as standard vector of floats.

3.5.1 Detailed Description

Algorithms within one chain can share variables for communication with other algorithms.

This mechanism allows interaction between algorithms (i.e. separation of noise estimation and noise reduction algorithms, combination of dynamic compression and noise estimation). Through a set of simple C functions, algorithms can propagate variables of any type, even C++ classes, to other algorithms.

An algorithm communication handle (**algo_comm_t** (p. 201)) is passed at initialisation time to the constructor of each plugin class **constructor** (p. 728). This handle contains a reference handle, **algo_comm_t::handle** (p. 201), and a number of function pointers, **algo_comm_t**↩ **::insert_var** (p. 201) etc.. An algorithm communication variable is an object of type **comm_**↩ **var_t** (p. 246).

For AC variables of numeric types, openMHA Plugins for conversion into parsable monitor variables, acmon, and storage into Matlab or text files, acsave, are available.

3.5.2 Function Documentation

3.5.2.1 mha_spec_t MHA_AC::get_var_spectrum (algo_comm_t *ac,* const std::string & *name*)

Convert an AC variable into a spectrum.

This function reads an AC variable and tries to convert it into a valid spectrum. The Spectrum variable is granted to be valid only for one call of the processing function.

Parameters

ac	AC handle
name	Name of the variable

Returns

Spectrum structure

3.5.2.2 mha_wave_t MHA_AC::get_var_waveform (algo_comm_t *ac*, const std::string & *name*)

Convert an AC variable into a waveform.

This function reads an AC variable and tries to convert it into a valid waveform. The waveform variable is granted to be valid only for one call of the processing function.

Parameters

ac	AC handle
name	Name of the variable

Returns

waveform structure

3.5.2.3 int MHA_AC::get_var_int (algo_comm_t *ac*, const std::string & *name*)

Return value of an integer scalar AC variable.

Parameters

ac	AC handle
name	Name of the variable

Returns

Variable value

3.5.2.4 float MHA_AC::get_var_float (algo_comm_t *ac*, const std::string & *name*)

Return value of an floating point scalar AC variable.

Parameters

ac	AC handle
name	Name of the variable

Returns

Variable value

3.5.2.5 std::vector< float > MHA_AC::get_var_vfloat (algo_comm_t *ac*, const std::string & *name*)

Return value of an floating point vector AC variable as standard vector of floats.

Parameters

ac	AC handle
name	Name of the variable

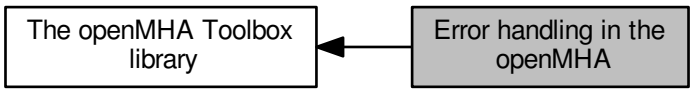
Returns

Variable value

3.6 Error handling in the openMHA

Errors are reported to the user via the **MHA_Error** (p. 445) exception.

Collaboration diagram for Error handling in the openMHA:



Classes

- class **MHA_Error**
  Error reporting exception class.

Macros

- #define **MHA_ErrorMsg**(x) **MHA_Error**(__FILE__,__LINE__,"%s",x)
  Throw an openMHA error with a text message.
- #define **MHA_assert**(x) if(!(x)) throw **MHA_Error**(__FILE__,__LINE__,"\"%s\" is false.",#x)
  *Assertion macro, which throws an **MHA_Error** (p. 445).*
- #define **MHA_assert_equal**(a, b) if(a != b) throw **MHA_Error**(__FILE__,__LINE__↩ ,"\"%s == %s\" is false (%s = %g, %s = %g).",#a,#b,#a,(double)(a),#b,(double)(b))
  *Equality assertion macro, which throws an **MHA_Error** (p. 445) with the values.*

Functions

- void **mha_debug** (const char *fmt,...)
  Print an info message (stderr on Linux, OutputDebugString in Windows).

3.6.1 Detailed Description

Errors are reported to the user via the **MHA_Error** (p. 445) exception.

3.6.2 Macro Definition Documentation

3.6.2.1 #define MHA_ErrorMsg(*x*) MHA_Error(__FILE__,__LINE__,"%s",x)

Throw an openMHA error with a text message.

Parameters

x	Text message.

3.6.2.2 #define MHA_assert(*x*) if(!(x)) throw MHA_Error(__FILE__,__LINE__,"\"%s\" is false.",#x)

Assertion macro, which throws an **MHA_Error** (p. 445).

Parameters

x	Boolean expression which should be true.

3.6.2.3 #define MHA_assert_equal(*a,* *b*) if(a != b) throw MHA_Error(__FILE__,__LINE__,"\"%s == %s\" is false (%s = %g, %s = %g).",#a,#b,#a,(double)(a),#b,(double)(b))

Equality assertion macro, which throws an **MHA_Error** (p. 445) with the values.

Parameters

a	Numeric expression which can be converted to double (for printing).
b	Numeric expression which should be equal to a

3.6.3 Function Documentation

3.6.3.1 void mha_debug (const char * *fmt,* ...)

Print an info message (stderr on Linux, OutputDebugString in Windows).

3.7 The openMHA configuration language

openMHA Plugins that should use the openMHA configuration language for their configuration have to be implemented in C++ and need to include **mha_parser.hh** (p. 1017).

openMHA Plugins that should use the openMHA configuration language for their configuration have to be implemented in C++ and need to include **mha_parser.hh** (p. 1017).

All required classes and functions for parser access are declared in the namespace **MHAParser** (p. 107). The plugin class should be derived from the class **MHAParser::parser_t** (p. 688) (or **MHAPlugin::plugin_t** (p. 728)), which symbolises a sub-parser node in the openMHA script hierarchy. Variables of many types can be registered to the sub-parser node by calling the member function **insert_item** (p. 690).

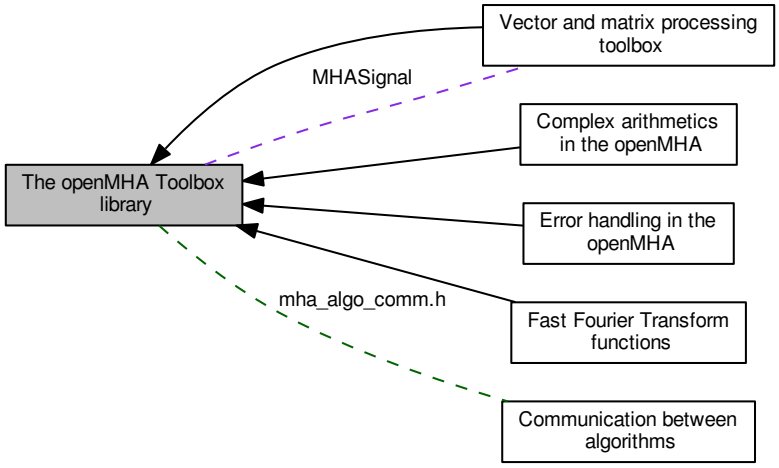
The openMHA Plugin template class **MHAPlugin::plugin_t** (p. 728) together with the Plugin macro **MHAPLUGIN_CALLBACKS** (p. 10) provide the callback mappings and correct inheritance. If your plugin is based on that template class, you simply have to use the insert_item command to give access to your variables, everything else is managed internally.

A complete list of all openMHA script items is given in the description of the **MHAParser** (p. 107) namespace.

3.8 The openMHA Toolbox library

The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins.

Collaboration diagram for The openMHA Toolbox library:



Modules

- **Error handling in the openMHA**

  *Errors are reported to the user via the **MHA_Error** (p. 445) exception.*
- **Vector and matrix processing toolbox**

  *The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 118), and many functions and operators for use with the structures **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465).*
- **Complex arithmetics in the openMHA**
- **Fast Fourier Transform functions**

Files

- file **mha_algo_comm.h**

  Header file for Algorithm Communication.
- file **mha_filter.hh**

  Header file for IIR filter classes.
- file **mha_signal.hh**

  Header file for audio signal handling and processing classes.
- file **mha_tablelookup.hh**

  Header file for table lookup classes.

Namespaces

- **MHAOvlFilter**

  Namespace for overlapping FFT based filter bank classes and functions.
- **MHAFilter**

  Namespace for IIR and FIR filter classes.
- **MHAParser**

  Name space for the openMHA-Parser configuration language.
- **MHASignal**

  Namespace for audio signal handling and processing classes.
- **MHATableLookup**

  Namespace for table lookup classes.

3.8.1 Detailed Description

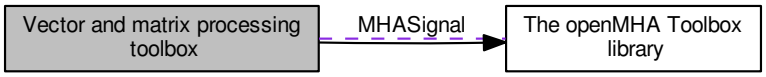
The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins.

It contains the openMHA script language classes.

3.9 Vector and matrix processing toolbox

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 118), and many functions and operators for use with the structures **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465).

Collaboration diagram for Vector and matrix processing toolbox:



Namespaces

- **MHASignal**
  Namespace for audio signal handling and processing classes.
- **MHAWindow**
  Collection of Window types.

Classes

- struct **mha_wave_t**
  Waveform signal structure.
- struct **mha_spec_t**
  Spectrum signal structure.
- struct **mha_audio_descriptor_t**
  *Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 504)).*
- struct **mha_audio_t**
  *An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465)).*
- class **MHASignal::spectrum_t**
  *a signal processing class for spectral data (based on **mha_spec_t** (p. 465))*
- class **MHASignal::waveform_t**
  *signal processing class for waveform data (based on **mha_wave_t** (p. 496))*
- class **MHASignal::doublebuffer_t**
  Double-buffering class.
- class **MHASignal::hilbert_t**
  Hilbert transformation of a waveform segment.
- class **MHASignal::minphase_t**
  Minimal phase function.
- class **MHASignal::uint_vector_t**

Vector of unsigned values, used for size and index description of n-dimensional matrixes.

- class **MHASignal::matrix_t**
  n-dimensional matrix with real or complex floating point values.
- class **MHAParser::window_t**
  MHA configuration interface for a window function generator.
- class **MHASignal::delay_wave_t**
  Delayline containing wave fragments.
- class **MHASignal::async_rmslevel_t**
  Class for asynchronous level metering.

Typedefs

- typedef float **mha_real_t**
  openMHA type for real numbers

Functions

- **mha_wave_t range** (**mha_wave_t** s, unsigned int k0, unsigned int len)
  Return a time interval from a waveform chunk.
- **mha_spec_t channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)
  Return a channel interval from a spectrum.
- void **MHASignal::for_each** (**mha_wave_t** ∗s, **mha_real_t**(∗fun)(**mha_real_t**))
  *Apply a function to each element of a **mha_wave_t** (p. 496).*
- **mha_real_t MHASignal::lin2db** (**mha_real_t** x)
  Conversion from linear scale to dB (no SPL reference)
- **mha_real_t MHASignal::db2lin** (**mha_real_t** x)
  Conversion from dB scale to linear (no SPL reference)
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x)
  Conversion from linear Pascal scale to dB SPL.
- **mha_real_t MHASignal::pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=1e-20f)
  Conversion from squared Pascal scale to dB SPL.
- **mha_real_t MHASignal::dbspl2pa** (**mha_real_t** x)
  Conversion from dB SPL to linear Pascal scale.
- **mha_real_t MHASignal::smp2sec** (**mha_real_t** n, **mha_real_t** srate)
  conversion from samples to seconds
- **mha_real_t MHASignal::sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
  conversion from seconds to samples
- **mha_real_t MHASignal::bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)
  conversion from fft bin index to frequency
- **mha_real_t MHASignal::freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)
  conversion from frequency to fft bin index
- **mha_real_t MHASignal::smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
  conversion from delay in samples to phase shift

- **mha_real_t MHASignal::rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)

  conversion from phase shift to delay in samples
- template<class elem_type >
  std::vector< elem_type > **MHASignal::dupvec** (std::vector< elem_type > vec, unsigned n)

  Duplicate last vector element to match desired size.
- template<class elem_type >
  std::vector< elem_type > **MHASignal::dupvec_chk** (std::vector< elem_type > vec, unsigned n)

  Duplicate last vector element to match desired size, check for dimension.
- bool **equal_dim** (const **mha_wave_t** &a, const **mha_wave_t** &b)

  Test for equal dimension of waveform structures.
- bool **equal_dim** (const **mha_wave_t** &a, const **mhaconfig_t** &b)

  Test for match of waveform dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_spec_t** &b)

  Test for equal dimension of spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mhaconfig_t** &b)

  Test for match of spectrum dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_wave_t** &a, const **mha_spec_t** &b)

  Test for equal dimension of waveform/spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_wave_t** &b)

  Test for equal dimension of waveform/spectrum structures.
- void **integrate** (**mha_wave_t** &s)

  Numeric integration of a signal vector (real values)
- void **integrate** (**mha_spec_t** &s)

  Numeric integration of a signal vector (complex values)
- unsigned int **size** (const **mha_wave_t** &s)

  Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** &s)

  Return size of a spectrum structure.
- unsigned int **size** (const **mha_wave_t** *s)

  Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** *s)

  Return size of a spectrum structure.
- void **clear** (**mha_wave_t** &s)

  Set all values of waveform to zero.
- void **clear** (**mha_wave_t** *s)

  Set all values of waveform to zero.
- void **clear** (**mha_spec_t** &s)

  Set all values of spectrum to zero.
- void **clear** (**mha_spec_t** *s)

  Set all values of spectrum to zero.
- void **assign** (**mha_wave_t** self, **mha_real_t** val)

  Set all values of waveform 'self' to 'val'.

- void **assign** (**mha_wave_t** self, const **mha_wave_t** &val)

  Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_spec_t** self, const **mha_spec_t** &val)

  Set all values of spectrum 'self' to 'val'.
- void **timeshift** (**mha_wave_t** &self, int shift)

  Time shift of waveform chunk.
- **mha_real_t** & **value** (**mha_wave_t** *s, unsigned int fr, unsigned int ch)

  Access an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** *s, unsigned int fr, unsigned int ch)

  Constant access to an element of a waveform structure.
- **mha_complex_t** & **value** (**mha_spec_t** *s, unsigned int fr, unsigned int ch)

  Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** *s, unsigned int fr, unsigned int ch)

  Constant access to an element of a spectrum.
- **mha_real_t** & **value** (**mha_wave_t** &s, unsigned int fr, unsigned int ch)

  Access to an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** &s, unsigned int fr, unsigned int ch)

  Constant access to an element of a waveform structure.
- **mha_complex_t** & **value** (**mha_spec_t** &s, unsigned int fr, unsigned int ch)

  Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** &s, unsigned int fr, unsigned int ch)

  Constant access to an element of a spectrum.
- std::vector< float > **std_vector_float** (const **mha_wave_t** &)

  *Converts a **mha_wave_t** (p. 496) structure into a std::vector<float> (interleaved order).*
- std::vector< std::vector< float > > **std_vector_vector_float** (const **mha_wave_t** &)

  *Converts a **mha_wave_t** (p. 496) structure into a std::vector< std::vector<float> > (outer vector represents channels).*
- std::vector< std::vector< **mha_complex_t** > > **std_vector_vector_complex** (const **mha_spec_t** &)

  *Converts a **mha_spec_t** (p. 465) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).*
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_real_t** &)

  Addition operator.
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_wave_t** &)

  Addition operator.
- **mha_wave_t** & **operator-=** (**mha_wave_t** &, const **mha_wave_t** &)

  Subtraction operator.
- **mha_spec_t** & **operator-=** (**mha_spec_t** &, const **mha_spec_t** &)

  Subtraction operator.
- **mha_wave_t** & **operator*=** (**mha_wave_t** &, const **mha_real_t** &)

  Element-wise multiplication operator.
- **mha_wave_t** & **operator*=** (**mha_wave_t** &, const **mha_wave_t** &)

  Element-wise multiplication operator.
- **mha_spec_t** & **operator*=** (**mha_spec_t** &, const **mha_real_t** &)

  Element-wise multiplication operator.

- **mha_spec_t** & **operator∗=** (**mha_spec_t** &, const **mha_wave_t** &)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &, const **mha_spec_t** &)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator/=** (**mha_spec_t** &, const **mha_spec_t** &)
  Element-wise division operator.
- **mha_wave_t** & **operator/=** (**mha_wave_t** &, const **mha_wave_t** &)
  Element-wise division operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &, const **mha_spec_t** &)
  Addition operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &, const **mha_real_t** &)
  Addition operator.
- **mha_wave_t** & **operator^=** (**mha_wave_t** &self, const **mha_real_t** &arg)
  Exponent operator.
- void **MHASignal::copy_channel** (**mha_spec_t** &self, const **mha_spec_t** &src, unsigned sch, unsigned dch)
  Copy one channel of a source signal.
- void **MHASignal::copy_channel** (**mha_wave_t** &self, const **mha_wave_t** &src, unsigned src_channel, unsigned dest_channel)
  Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen)
  Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** sqfreq_response[])
  Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs** (const **mha_spec_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel** (const **mha_wave_t** &s, unsigned int channel)
  Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s)
  Find maximal absolute value.
- **mha_real_t MHASignal::max** (const **mha_wave_t** &s)
  Find maximal value.
- **mha_real_t MHASignal::min** (const **mha_wave_t** &s)
  Find minimal value.
- **mha_real_t MHASignal::sumsqr_channel** (const **mha_wave_t** &s, unsigned int channel)
  Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsqr_frame** (const **mha_wave_t** &s, unsigned int frame)
  Calculate sum over all channels of squared values.
- void **conjugate** (**mha_spec_t** &self)
  *Replace (!) the value of this **mha_spec_t** (p. 465) with its conjugate.*

3.9.1 Detailed Description

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 118), and many functions and operators for use with the structures **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465).

3.9.2 Typedef Documentation

3.9.2.1 typedef float mha_real_t

openMHA type for real numbers

This type is expected to be allways the C-type 'float' (IEEE 754 single).

3.9.3 Function Documentation

3.9.3.1 mha_wave_t range (mha_wave_t *s*, unsigned int *k0*, unsigned int *len*)

Return a time interval from a waveform chunk.

A waveform chunk containing a time intervall of a larger waveform chunk is returned. The number of channels remains constant. The data of the output waveform structure points to the data of the input structure, i.e., write access to the output waveform chunk modifies the corresponding entries in the input chunk.

Parameters

s	Waveform structure
k0	Index of first value in output
len	Number of frames in output

Returns

Waveform structure representing the sub-interval.

3.9.3.2 mha_spec_t channels (mha_spec_t *s*, unsigned int *ch_start*, unsigned int *nch*)

Return a channel interval from a spectrum.

Parameters

s	Input spectrum
ch_start	Index of first channel in output
nch	Number of channels in output

Returns

Spectrum structure representing the sub-interval.

3.9.3.3 void MHASignal::for_each (mha_wave_t * s, mha_real_t(*)(mha_real_t) fun) [inline]

Apply a function to each element of a **mha_wave_t** (p. 496).

Parameters

s	Pointer to a **mha_wave_t** (p. 496) structure
fun	Function to be applied (one argument)

3.9.3.4 mha_real_t MHASignal::lin2db (mha_real_t x) [inline]

Conversion from linear scale to dB (no SPL reference)

Parameters

x	Linear input.

3.9.3.5 mha_real_t MHASignal::db2lin (mha_real_t x) [inline]

Conversion from dB scale to linear (no SPL reference)

Parameters

x	dB input.

3.9.3.6 mha_real_t MHASignal::pa2dbspl (mha_real_t x) [inline]

Conversion from linear Pascal scale to dB SPL.

Parameters

x	Linear input.

3.9.3.7 mha_real_t MHASignal::pa22dbspl (mha_real_t *x,* mha_real_t *eps* = `1e-20f`) [inline]

Conversion from squared Pascal scale to dB SPL.

Parameters

x	squared pascal input
eps	minimum squared-pascal value

3.9.3.8 mha_real_t MHASignal::dbspl2pa (mha_real_t *x*) [inline]

Conversion from dB SPL to linear Pascal scale.

Parameters

x	Linear input.

3.9.3.9 mha_real_t MHASignal::smp2sec (mha_real_t *n,* mha_real_t *srate*) [inline]

conversion from samples to seconds

Parameters

n	number of samples
srate	sampling rate / Hz

3.9.3.10 mha_real_t MHASignal::sec2smp (mha_real_t *sec,* mha_real_t *srate*) [inline]

conversion from seconds to samples

Parameters

sec	time in seconds
srate	sampling rate / Hz

Returns

number of samples, generally has non-zero fractional part

3.9.3.11 mha_real_t MHASignal::bin2freq (mha_real_t *bin*, unsigned *fftlen*, mha_real_t *srate*) [inline]

conversion from fft bin index to frequency

Parameters

bin	index of fft bin, index 0 has dc
fftlen	FFT length
srate	sampling frequency / Hz

Returns

frequency of fft bin / Hz

3.9.3.12 mha_real_t MHASignal::freq2bin (mha_real_t *freq*, unsigned *fftlen*, mha_real_t *srate*) [inline]

conversion from frequency to fft bin index

Parameters

freq	frequency / Hz
fftlen	FFT length
srate	sampling frequency / Hz

Returns

0-based index of fft bin, generally has non-zero fractional part

3.9.3.13 mha_real_t MHASignal::smp2rad (mha_real_t *samples*, unsigned *bin*, unsigned *fftlen*) [inline]

conversion from delay in samples to phase shift

Compute phase shift that needs to be applied to fft spectrum to achieve the desired delay.

Parameters

samples	delay in samples. Positive delay: shift current signal to future.
bin	index of fft bin, index 0 has dc (index 0 and nyqvist bin cannot be delayed)
fftlen	FFT length

Returns

The phase shift in radiant that needs to be applied to fft bin to achieve the desired delay. A positive delay requires a negative phase shift. If required phase shift is >pi or <-pi, then the desired delay cannot be applied in the fft domain with given parameters. Required phase shifts close to pi should not be used. If bin is 0 or nyqvist, returns 0 phase shift.

3.9.3.14 mha_real_t MHASignal::rad2smp (mha_real_t *phase_shift,* unsigned *bin,* unsigned *fftlen*) [inline]

conversion from phase shift to delay in samples

Compute delay in samples that is achieved by a phase shift.

Parameters

phase_shift	phase shift in radiant
bin	index of fft bin, index 0 has dc (index 0 and nyqvist bin cannot be delayed)
fftlen	FFT length

Returns

The delay in samples achieved by applying the phase shift. A negative phase shift causes a positive delay: shifts current signal to future.

3.9.3.15 template<class elem_type > std::vector<elem_type> MHASignal::dupvec (std::vector< elem_type > *vec,* unsigned *n*)

Duplicate last vector element to match desired size.

Parameters

vec	Input vector.
n	Target number of elements.

Return values

Resized	vector.

3.9.3.16 template<class elem_type > std::vector<elem_type> MHASignal::dupvec_chk (std::vector< elem_type > *vec,* unsigned *n*)

Duplicate last vector element to match desired size, check for dimension.

The input dimension can be either 1 or the target length.

Parameters

vec	Input vector.
n	Target number of elements.

Return values

Resized	vector.

3.9.3.17 bool equal_dim (const mha_wave_t & *a,* const mha_wave_t & *b*) [inline]

Test for equal dimension of waveform structures.

3.9.3.18 bool equal_dim (const mha_wave_t & *a,* const mhaconfig_t & *b*) [inline]

Test for match of waveform dimension with mhaconfig structure.

3.9.3.19 bool equal_dim (const mha_spec_t & *a,* const mha_spec_t & *b*) [inline]

Test for equal dimension of spectrum structures.

3.9.3.20 bool equal_dim (const mha_spec_t & *a,* const mhaconfig_t & *b*) [inline]

Test for match of spectrum dimension with mhaconfig structure.

3.9.3.21 bool equal_dim (const mha_wave_t & *a*, const mha_spec_t & *b*) [inline]

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 496) use interleaved data order, while spectrum structures **mha_spec_t** (p. 465) use non-interleaved.

3.9.3.22 bool equal_dim (const mha_spec_t & *a*, const mha_wave_t & *b*) [inline]

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 496) use interleaved data order, while spectrum structures **mha_spec_t** (p. 465) use non-interleaved.

3.9.3.23 void integrate (mha_wave_t & *s*)

Numeric integration of a signal vector (real values)

Parameters

s	Input signal vector

3.9.3.24 void integrate (mha_spec_t & *s*)

Numeric integration of a signal vector (complex values)

Parameters

s	Input signal vector

3.9.3.25 unsigned int size (const mha_wave_t & *s*) [inline]

Return size of a waveform structure.

3.9.3.26 unsigned int size (const mha_spec_t & *s*) [inline]

Return size of a spectrum structure.

3.9.3.27 unsigned int size (const mha_wave_t * *s*) [inline]

Return size of a waveform structure.

3.9.3.28 unsigned int size (const mha_spec_t * *s*) [inline]

Return size of a spectrum structure.

3.9.3.29 void clear (mha_wave_t & *s*) [inline]

Set all values of waveform to zero.

3.9.3.30 void clear (mha_wave_t * *s*) [inline]

Set all values of waveform to zero.

3.9.3.31 void clear (mha_spec_t & *s*) [inline]

Set all values of spectrum to zero.

3.9.3.32 void clear (mha_spec_t * *s*) [inline]

Set all values of spectrum to zero.

3.9.3.33 void assign (mha_wave_t *self*, mha_real_t *val*) [inline]

Set all values of waveform 'self' to 'val'.

Parameters

self	Waveform to be modified.
val	Value to be assigned to all entries of waveform.

3.9.3.34 void assign (

mha_wave_t ***self,***
const mha_wave_t & ***val*** **)**

Set all values of waveform 'self' to 'val'.

Parameters

self	Waveform to be modified.
val	Source waveform structure.

3.9.3.35 void assign (

mha_spec_t ***self,***
const mha_spec_t & ***val*** **)**

Set all values of spectrum 'self' to 'val'.

Parameters

self	Spectrum to be modified.
val	Source spectrum.

3.9.3.36 void timeshift (

mha_wave_t & ***self,***
int ***shift*** **)**

Time shift of waveform chunk.

Shifted areas are filled with zeros.

Parameters

self	Waveform chunk to be shifted
shift	Shift amount, positive values shift to later times

3.9.3.37 mha_real_t& value (

mha_wave_t $*$ ***s,***
unsigned int ***fr,***
unsigned int ***ch*** **)** `[inline]`

Access an element of a waveform structure.

Parameters

s	Waveform structure
fr	Frame number
ch	Channel number

Returns

Reference to element

3.9.3.38 const mha_real_t& value (const mha_wave_t * s, unsigned int fr, unsigned int ch) [inline]

Constant access to an element of a waveform structure.

Parameters

s	Waveform structure
fr	Frame number
ch	Channel number

Returns

Reference to element

3.9.3.39 mha_complex_t& value (mha_spec_t * s, unsigned int fr, unsigned int ch) [inline]

Access to an element of a spectrum.

Parameters

s	Spectrum structure
fr	Bin number
ch	Channel number

Returns

Reference to element

3.9.3.40 const mha_complex_t& value (const mha_spec_t * s, unsigned int fr, unsigned int ch) [inline]

Constant access to an element of a spectrum.

Parameters

s	Spectrum structure
fr	Bin number
ch	Channel number

Returns

Reference to element

3.9.3.41 mha_real_t& value (mha_wave_t & *s*, unsigned int *fr*, unsigned int *ch*) [inline]

Access to an element of a waveform structure.

Parameters

s	Waveform structure
fr	Frame number
ch	Channel number

Returns

Reference to element

3.9.3.42 const mha_real_t& value (const mha_wave_t & *s*, unsigned int *fr*, unsigned int *ch*) [inline]

Constant access to an element of a waveform structure.

Parameters

s	Waveform structure
fr	Frame number
ch	Channel number

Returns

Reference to element

3.9.3.43 mha_complex_t& value (mha_spec_t & *s*, unsigned int *fr*, unsigned int *ch*) [inline]

Access to an element of a spectrum.

Parameters

s	Spectrum structure
fr	Bin number
ch	Channel number

Returns

Reference to element

3.9.3.44 const mha_complex_t& value (const mha_spec_t & *s*, unsigned int *fr*, unsigned int *ch*) [inline]

Constant access to an element of a spectrum.

Parameters

s	Spectrum structure
fr	Bin number
ch	Channel number

Returns

Reference to element

3.9.3.45 std::vector<float> std_vector_float (const mha_wave_t &)

Converts a **mha_wave_t** (p. 496) structure into a std::vector<float> (interleaved order).

Warning

This function is not real-time safe. Do not use in signal processing thread.

3.9.3.46 std::vector<std::vector<float> > std_vector_vector_float (const mha_wave_t &)

Converts a **mha_wave_t** (p. 496) structure into a std::vector< std::vector<float> > (outer vector represents channels).

Warning

This function is not real-time safe. Do not use in signal processing thread.

3.9.3.47 std::vector<std::vector<mha_complex_t> > std_vector_vector_complex (const mha_spec_t &)

Converts a **mha_spec_t** (p. 465) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).

Warning

This function is not real-time safe. Do not use in signal processing thread.

3.9.3.48 mha_wave_t& operator+= (mha_wave_t & , const mha_real_t &)

Addition operator.

3.9.3.49 mha_wave_t& operator+= (mha_wave_t & , const mha_wave_t &)

Addition operator.

3.9.3.50 mha_wave_t& operator-= (mha_wave_t & , const mha_wave_t &)

Subtraction operator.

3.9.3.51 mha_spec_t& operator-= (mha_spec_t & , const mha_spec_t &)

Subtraction operator.

3.9.3.52 mha_wave_t& operator∗= (mha_wave_t & , const mha_real_t &)

Element-wise multiplication operator.

3.9.3.53 mha_wave_t& operator∗= (mha_wave_t & , const mha_wave_t &)

Element-wise multiplication operator.

3.9.3.54 mha_spec_t& operator∗= (mha_spec_t & , const mha_real_t &)

Element-wise multiplication operator.

3.9.3.55 mha_spec_t& operator∗= (mha_spec_t & , const mha_wave_t &)

Element-wise multiplication operator.

3.9.3.56 mha_spec_t& operator∗= (mha_spec_t & , const mha_spec_t &)

Element-wise multiplication operator.

3.9.3.57 mha_spec_t& operator/= (mha_spec_t & , const mha_spec_t &)

Element-wise division operator.

3.9.3.58 mha_wave_t& operator/= (mha_wave_t & , const mha_wave_t &)

Element-wise division operator.

3.9.3.59 mha_spec_t& operator+= (mha_spec_t & , const mha_spec_t &)

Addition operator.

3.9.3.60 mha_spec_t& operator+= (mha_spec_t & , const mha_real_t &)

Addition operator.

3.9.3.61 mha_wave_t& operator^= (mha_wave_t & *self,* const mha_real_t & *arg*)

Exponent operator.

Warning

This overwrites the xor operator!

3.9.3.62 void MHASignal::copy_channel (mha_spec_t & *self,* const mha_spec_t & *src,* unsigned *sch,* unsigned *dch*)

Copy one channel of a source signal.

Parameters

self	Destination.
src	Source
sch	Source channel number
dch	Destination channel number

3.9.3.63 void MHASignal::copy_channel (mha_wave_t & *self,* const mha_wave_t & *src,* unsigned *src_channel,* unsigned *dest_channel*)

Copy one channel of a source signal.

Parameters

self	Destination.
src	Source
src_channel	Source channel number
dest_channel	Destination channel number

3.9.3.64 mha_real_t MHASignal::rmslevel (const mha_spec_t & *s,* unsigned int *channel,* unsigned int *fftlen*)

Return RMS level of a spectrum channel.

Parameters

s	Input spectrum
channel	Channel number to be tested
fftlen	FFT length (to correctly count the level of the Nyquist bin)

Returns

RMS level in Pa

3.9.3.65 mha_real_t MHASignal::colored_intensity (const mha_spec_t & *s,* unsigned int *channel,* unsigned int *fftlen,* mha_real_t *sqfreq_response[ ]*)

Colored spectrum intensity.

computes the squared sum of the spectrum after filtering with the frequency response

Parameters

s	Input spectrum
channel	Channel number to be tested
fftlen	FFT length (to correctly count the level of the Nyquist bin)
sqfreq_response	A squared weighting factor for every fft bin.

Returns

sum of squares. Root of this is the colored level in Pa

3.9.3.66 mha_real_t MHASignal::maxabs (const mha_spec_t & *s,* unsigned int *channel*)

Find maximal absolute value.

Parameters

s	Input signal
channel	Channel to be tested

Returns

maximum absolute value

3.9.3.67 mha_real_t MHASignal::rmslevel (const mha_wave_t & *s*, unsigned int *channel*)

Return RMS level of a waveform channel.

Parameters

s	Input waveform signal
channel	Channel number to be tested

Returns

RMS level in Pa

3.9.3.68 mha_real_t MHASignal::maxabs (const mha_wave_t & *s*, unsigned int *channel*)

Find maximal absolute value.

Parameters

s	Input signal
channel	Channel to be tested

Returns

maximum absolute value

3.9.3.69 mha_real_t MHASignal::maxabs (const mha_wave_t & *s*)

Find maximal absolute value.

Parameters

s	Input signal

Returns

maximum absolute value

3.9.3.70 mha_real_t MHASignal::max (const mha_wave_t & *s*)

Find maximal value.

Parameters

s	Input signal

Returns

maximum absolute value

3.9.3.71 mha_real_t MHASignal::min (const mha_wave_t & *s*)

Find minimal value.

Parameters

s	Input signal

Returns

maximum absolute value

3.9.3.72 mha_real_t MHASignal::sumsqr_channel (const mha_wave_t & *s*, unsigned int *channel*)

Calculate sum of squared values in one channel.

Parameters

s	Input signal
channel	Channel

Returns

$\sum x^2$

3.9.3.73 mha_real_t MHASignal::sumsqr_frame (const mha_wave_t & *s*, unsigned int *frame*)

Calculate sum over all channels of squared values.

Parameters

s	Input signal
frame	Frame number

Returns

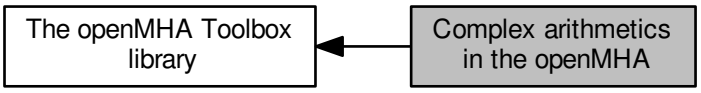
$\sum x^2$

3.9.3.74 void conjugate (mha_spec_t & *self*) [inline]

Replace (!) the value of this **mha_spec_t** (p. 465) with its conjugate.

3.10 Complex arithmetics in the openMHA

Collaboration diagram for Complex arithmetics in the openMHA:



Classes

- struct **mha_complex_t**
  Type for complex floating point values.

Functions

- **mha_complex_t** & **set** (**mha_complex_t** &self, **mha_real_t** real, **mha_real_t** imag=0)
  *Assign real and imaginary parts to a **mha_complex_t** (p. 431) variable.*
- **mha_complex_t mha_complex** (**mha_real_t** real, **mha_real_t** imag=0)
  *Create a new **mha_complex_t** (p. 431) with specified real and imaginary parts.*
- **mha_complex_t** & **set** (**mha_complex_t** &self, const std::complex< **mha_real_t** > &**stdcomplex**)
  *Assign a **mha_complex_t** (p. 431) variable from a std::complex.*
- std::complex< **mha_real_t** > **stdcomplex** (const **mha_complex_t** &self)
  *Create a std::complex from **mha_complex_t** (p. 431).*
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t angle**)
  *replaces the value of the given **mha_complex_t** (p. 431) with exp(i∗b).*
- double **angle** (const **mha_complex_t** &self)
  Computes the angle of a complex number in the complex plane.
- **mha_complex_t** & **operator+=** (**mha_complex_t** &self, const **mha_complex_t** &other)
  Addition of two complex numbers, overwriting the first.
- **mha_complex_t operator+** (const **mha_complex_t** &self, const **mha_complex_t** &other)
  Addition of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator+=** (**mha_complex_t** &self, **mha_real_t** other_real)
  Addition of a complex and a real number, overwriting the complex.
- **mha_complex_t operator+** (const **mha_complex_t** &self, **mha_real_t** other_real)
  Addition of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **operator-=** (**mha_complex_t** &self, const **mha_complex_t** &other)
  Subtraction of two complex numbers, overwriting the first.

- **mha_complex_t operator-** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Subtraction of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator-=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Subtraction of a complex and a real number, overwriting the complex.
- **mha_complex_t operator-** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Subtraction of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **operator*=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Multiplication of two complex numbers, overwriting the first.
- **mha_complex_t operator*** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Multiplication of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator*=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Multiplication of a complex and a real number, overwriting the complex.
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t angle**, **mha_real_t** factor)

  *replaces (!) the value of the given **mha_complex_t*** (p. 431) *with a * exp(i*b)*
- **mha_complex_t operator*** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Multiplication of a complex and a real number, result is a temporary object.
- **mha_real_t abs2** (const **mha_complex_t** &self)

  Compute the square of the absolute value of a complex value.
- **mha_real_t abs** (const **mha_complex_t** &self)

  Compute the absolute value of a complex value.
- **mha_complex_t** & **operator/=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Division of a complex and a real number, overwriting the complex.
- **mha_complex_t operator/** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Division of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **safe_div** (**mha_complex_t** &self, const **mha_complex_t** &other, **mha_real_t** eps, **mha_real_t** eps2)

  Safe division of two complex numbers, overwriting the first.
- **mha_complex_t** & **operator/=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Division of two complex numbers, overwriting the first.
- **mha_complex_t operator/** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Division of two complex numbers, result is a temporary object.
- **mha_complex_t operator-** (const **mha_complex_t** &self)

  Unary minus on a complex results in a negative temporary object.
- bool **operator==** (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compare two complex numbers for equality.
- bool **operator!=** (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compare two complex numbers for inequality.
- void **conjugate** (**mha_complex_t** &self)

  *Replace (!) the value of this **mha_complex_t*** (p. 431) *with its conjugate.*
- **mha_complex_t _conjugate** (const **mha_complex_t** &self)

  Compute the cojugate of this complex value.
- void **reciprocal** (**mha_complex_t** &self)

Replace the value of this complex with its reciprocal.

- **mha_complex_t _reciprocal** (const **mha_complex_t** &self)

  compute the reciprocal of this complex value.
- void **normalize** (**mha_complex_t** &self)

  Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle).
- void **normalize** (**mha_complex_t** &self, **mha_real_t** margin)

  Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle), with a safety margin.
- bool **almost** (const **mha_complex_t** &self, const **mha_complex_t** &other, **mha_real_t** times_epsilon=1e2)

  Compare two complex numbers for equality except for a small relative error.
- bool **operator**< (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compares the absolute values of two complex numbers.

3.10.1 Detailed Description

3.10.2 Function Documentation

3.10.2.1 mha_complex_t& set (mha_complex_t & *self*, mha_real_t *real*, mha_real_t *imag* = 0) [inline]

Assign real and imaginary parts to a **mha_complex_t** (p. 431) variable.

Parameters

self	The **mha_complex_t** (p. 431) variable whose value is about to change.
real	The new real part.
imag	The new imaginary part.

Returns

A reference to the changed variable.

3.10.2.2 mha_complex_t mha_complex (mha_real_t *real*, mha_real_t *imag* = 0) [inline]

Create a new **mha_complex_t** (p. 431) with specified real and imaginary parts.

Parameters

real	The real part.
imag	The imaginary part.

Returns

The new value.

3.10.2.3 mha_complex_t& set (mha_complex_t & *self*, const std::complex< mha_real_t > & *stdcomplex*) [inline]

Assign a **mha_complex_t** (p. 431) variable from a std::complex.

Parameters

self	The **mha_complex_t** (p. 431) variable whose value is about to change.
stdcomplex	The new complex value.

Returns

A reference to the changed variable.

3.10.2.4 std::complex<mha_real_t> stdcomplex (const mha_complex_t & *self*) [inline]

Create a std::complex from **mha_complex_t** (p. 431).

3.10.2.5 mha_complex_t& expi (mha_complex_t & *self*, mha_real_t *angle*) [inline]

replaces the value of the given **mha_complex_t** (p. 431) with exp(i∗b).

Parameters

self	The **mha_complex_t** (p. 431) variable whose value is about to change.
angle	The angle in the complex plane [rad].

Returns

A reference to the changed variable.

3.10.2.6 double angle (const mha_complex_t & *self*) [inline]

Computes the angle of a complex number in the complex plane.

Parameters

self	The complex number whose angle is needed.

Returns

The angle of a complex number in the complex plane.

3.10.2.7 mha_complex_t& operator+= (mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Addition of two complex numbers, overwriting the first.

3.10.2.8 mha_complex_t operator+ (const mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Addition of two complex numbers, result is a temporary object.

3.10.2.9 mha_complex_t& operator+= (mha_complex_t & *self,* mha_real_t *other_real*) [inline]

Addition of a complex and a real number, overwriting the complex.

3.10.2.10 mha_complex_t operator+ (const mha_complex_t & *self,* mha_real_t *other_real*) [inline]

Addition of a complex and a real number, result is a temporary object.

3.10.2.11 mha_complex_t& operator-= (mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Subtraction of two complex numbers, overwriting the first.

3.10.2.12 mha_complex_t operator- (const mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Subtraction of two complex numbers, result is a temporary object.

3.10.2.13 mha_complex_t& operator-= (mha_complex_t & *self,* mha_real_t *other_real*) [inline]

Subtraction of a complex and a real number, overwriting the complex.

3.10.2.14 mha_complex_t operator- (const mha_complex_t & *self,* mha_real_t *other_real*) [inline]

Subtraction of a complex and a real number, result is a temporary object.

3.10.2.15 mha_complex_t& operator∗= (mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Multiplication of two complex numbers, overwriting the first.

3.10.2.16 mha_complex_t operator∗ (const mha_complex_t & *self,* const mha_complex_t & *other*) [inline]

Multiplication of two complex numbers, result is a temporary object.

3.10.2.17 mha_complex_t& operator∗= (mha_complex_t & *self,* mha_real_t *other_real*) [inline]

Multiplication of a complex and a real number, overwriting the complex.

3.10.2.18 mha_complex_t& expi (mha_complex_t & *self,* mha_real_t *angle,* mha_real_t *factor*) [inline]

replaces (!) the value of the given **mha_complex_t** (p. 431) with a ∗ exp(i∗b)

Parameters

self	The **mha_complex_t** (p. 431) variable whose value is about to change.
angle	The imaginary exponent.
factor	The absolute value of the result.

Returns

A reference to the changed variable.

3.10.2.19 mha_complex_t operator∗ (
const mha_complex_t & *self,*
mha_real_t *other_real*) [inline]

Multiplication of a complex and a real number, result is a temporary object.

3.10.2.20 mha_real_t abs2 (
const mha_complex_t & *self*) [inline]

Compute the square of the absolute value of a complex value.

Returns

The square of the absolute value of self.

3.10.2.21 mha_real_t abs (
const mha_complex_t & *self*) [inline]

Compute the absolute value of a complex value.

Returns

The absolute value of self.

3.10.2.22 mha_complex_t& operator/= (
mha_complex_t & *self,*
mha_real_t *other_real*) [inline]

Division of a complex and a real number, overwriting the complex.

3.10.2.23 mha_complex_t operator/ (
const mha_complex_t & *self,*
mha_real_t *other_real*) [inline]

Division of a complex and a real number, result is a temporary object.

3.10.2.24 mha_complex_t& safe_div (
mha_complex_t & *self,*
const mha_complex_t & *other,*
mha_real_t *eps,*
mha_real_t *eps2*) [inline]

Safe division of two complex numbers, overwriting the first.

If abs(divisor) $<$ eps, then divisor is replaced by eps. eps2 = eps$*$eps.

3.10.2.25 mha_complex_t& operator/= (
mha_complex_t & *self,*
const mha_complex_t & *other*) [inline]

Division of two complex numbers, overwriting the first.

3.10.2.26 mha_complex_t operator/ (
const mha_complex_t & *self,*
const mha_complex_t & *other*) [inline]

Division of two complex numbers, result is a temporary object.

3.10.2.27 mha_complex_t operator- (
const mha_complex_t & *self*) [inline]

Unary minus on a complex results in a negative temporary object.

3.10.2.28 bool operator== (
const mha_complex_t & *x,*
const mha_complex_t & *y*) [inline]

Compare two complex numbers for equality.

3.10.2.29 bool operator!= (
const mha_complex_t & *x,*
const mha_complex_t & *y*) [inline]

Compare two complex numbers for inequality.

3.10.2.30 void conjugate (
mha_complex_t & *self*) [inline]

Replace (!) the value of this **mha_complex_t** (p. 431) with its conjugate.

3.10.2.31 mha_complex_t _conjugate (
const mha_complex_t & *self*) [inline]

Compute the cojugate of this complex value.

Returns

A temporary object holding the conjugate value.

3.10.2.32 void reciprocal (mha_complex_t & *self*) [inline]

Replace the value of this complex with its reciprocal.

3.10.2.33 mha_complex_t _reciprocal (const mha_complex_t & *self*) [inline]

compute the reciprocal of this complex value.

Returns

A temporary object holding the reciprocal value.

3.10.2.34 void normalize (mha_complex_t & *self*) [inline]

Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle).

3.10.2.35 void normalize (mha_complex_t & *self,* mha_real_t *margin*) [inline]

Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle), with a safety margin.

3.10.2.36 bool almost (const mha_complex_t & *self,* const mha_complex_t & *other,* mha_real_t *times_epsilon* = 1e2) [inline]

Compare two complex numbers for equality except for a small relative error.

Parameters

self	The first complex number.
other	The second complex number.
times_epsilon	Permitted relative error is this number multiplied with the machine accuracy for this Floating point format (std::numeric_limits<mha_real_t>::epsilon)

Returns

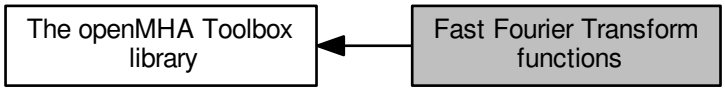
true if the relative difference is below times_epsilon $*$ std::numeric_limits<mha_real_t>↩ ::epsilon

3.10.2.37 bool operator< (
const **mha_complex_t** & *x*,
const **mha_complex_t** & *y*) [inline]

Compares the absolute values of two complex numbers.

3.11 Fast Fourier Transform functions

Collaboration diagram for Fast Fourier Transform functions:



Typedefs

- typedef void * **mha_fft_t**
  Handle for an FFT object.

Functions

- **mha_fft_t mha_fft_new** (unsigned int n)
  Create a new FFT handle.
- void **mha_fft_free** (**mha_fft_t** h)
  Destroy an FFT handle.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out)
  Tranform waveform segment into spectrum.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out, bool swaps)
  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out)
  Tranform spectrum into waveform segment.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out, unsigned int offset)
  Tranform spectrum into waveform segment.
- void **mha_fft_forward** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (forward).
- void **mha_fft_backward** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (backward).
- void **mha_fft_forward_scale** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (forward).
- void **mha_fft_backward_scale** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (backward).
- void **mha_fft_wave2spec_scale** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out)
  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave_scale** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out)
  Tranform spectrum into waveform segment.

3.11.1 Detailed Description

3.11.2 Typedef Documentation

3.11.2.1 typedef void* mha_fft_t

Handle for an FFT object.

This FFT object is used by the functions mha_fft_wave2spec and mha_fft_spec2wave. The FFT back-end is the FFTW library. The back-end is completely hidden, including external header files or linking external libraries is not required.

3.11.3 Function Documentation

3.11.3.1 mha_fft_t mha_fft_new (unsigned int *n*)

Create a new FFT handle.

Parameters

n	FFT length.

Create a new FFT handle.

Parameters

n	FFT length

Return values

FFT	object

3.11.3.2 void mha_fft_free (mha_fft_t *h*)

Destroy an FFT handle.

Parameters

h	Handle to be destroyed.

Destroy an FFT handle.

Parameters

h	FFT object to be removed

3.11.3.3 void mha_fft_wave2spec (mha_fft_t *h,* const mha_wave_t * *in,* mha_spec_t * *out*)

Tranform waveform segment into spectrum.

Parameters

h	FFT handle.
in	Input waveform segment.
out	Output spectrum.

Tranform waveform segment into spectrum.

Parameters

h	FFT object handle
in	pointer to input waveform signal
out	pointer to output spectrum signal (has to be allocated)

3.11.3.4 void mha_fft_wave2spec (mha_fft_t *h,* const mha_wave_t * *in,* mha_spec_t * *out,* bool *swaps*)

Tranform waveform segment into spectrum.

Like normal wave2spec, but swaps wave buffer halves before transforming if the swaps parameter is true.

Warning: These openMHA FFTs adopt a nonstandard scaling scheme in which the forward transform scales by 1/N and the backward does not scale. We would recommend using the '_scale' methods instead.

Parameters

h	FFT handle.
in	Input waveform segment.
out	Output spectrum.
swaps	Function swaps the first and second half of the waveform buffer before the FFT transform when this parameter is set to true.

3.11.3.5 void mha_fft_spec2wave (mha_fft_t *h,* const mha_spec_t * *in,* mha_wave_t * *out*)

Tranform spectrum into waveform segment.

Warning: These openMHA FFTs adopt a nonstandard scaling scheme in which the forward transform scales by 1/N and the backward does not scale. We would recommend using the '_scale' methods instead.

Parameters

h	FFT handle.
in	Input spectrum.
out	Output waveform segment.

Tranform spectrum into waveform segment.

Parameters

h	FFT object handle
in	pointer to input spectrum
out	pointer to output waveform signal (has to be allocated)

3.11.3.6 void mha_fft_spec2wave (mha_fft_t *h,* const mha_spec_t * *in,* mha_wave_t * *out,* unsigned int *offset*)

Tranform spectrum into waveform segment.

out may have fewer number of frames than needed for a complete iFFT. Only as many frames are written into out as fit, starting with offset offset of the complete iFFT.

Warning: These openMHA FFTs adopt a nonstandard scaling scheme in which the forward transform scales by 1/N and the backward does not scale. We would recommend using the '_scale' methods instead.

Parameters

h	FFT handle.
in	Input spectrum.
out	Output waveform segment.
offset	Offset into iFFT wave buffer

Tranform spectrum into waveform segment.

Only part of the iFFT is tranferred into the out buffer.

Out may have fewer number of freames than needed for a complete iFFT. Only as many frames are written into out as fit, starting with offset offset of the complete iFFT.

Parameters

h	FFT object handle
in	pointer to input spectrum
out	pointer to output waveform signal (has to be allocated)
offset	Offset into complete iFFT buffer.

3.11.3.7 void mha_fft_forward (

mha_fft_t ***h,***
mha_spec_t ∗ ***sIn,***
mha_spec_t ∗ ***sOut***)

Complex to complex FFT (forward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 465) typically has nfft/2+1 bins for half-complex representation).

Warning: These openMHA FFTs adopt a nonstandard scaling scheme in which the forward transform scales by 1/N and the backward does not scale. We would recommend using the '_scale' methods instead.

Parameters

h	FFT handle.
sIn	Input spectrum.
sOut	Output spectrum.

3.11.3.8 void mha_fft_backward (

mha_fft_t ***h,***
mha_spec_t ∗ ***sIn,***
mha_spec_t ∗ ***sOut***)

Complex to complex FFT (backward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 465) typically has nfft/2+1 bins for half-complex representation).

Warning: These openMHA FFTs adopt a nonstandard scaling scheme in which the forward transform scales by 1/N and the backward does not scale. We would recommend using the '_scale' methods instead.

Parameters

h	FFT handle.
sIn	Input spectrum.
sOut	Output spectrum.

3.11.3.9 void mha_fft_forward_scale (mha_fft_t *h*, mha_spec_t * *sIn*, mha_spec_t * *sOut*)

Complex to complex FFT (forward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 465) typically has nfft/2+1 bins for half-complex representation).

The _scale methods use standard DFT scaling: There is no scaling in the forward transformation, and 1/N scaling for the backward.

Parameters

h	FFT handle.
sIn	Input spectrum.
sOut	Output spectrum.

3.11.3.10 void mha_fft_backward_scale (mha_fft_t *h*, mha_spec_t * *sIn*, mha_spec_t * *sOut*)

Complex to complex FFT (backward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 465) typically has nfft/2+1 bins for half-complex representation).

The _scale methods use standard DFT scaling: There is no scaling in the forward transformation, and 1/N scaling for the backward.

Parameters

h	FFT handle.
sIn	Input spectrum.
sOut	Output spectrum.

3.11.3.11 void mha_fft_wave2spec_scale (mha_fft_t *h,* const mha_wave_t * *in,* mha_spec_t * *out*)

Tranform waveform segment into spectrum.

The _scale methods use standard DFT scaling: There is no scaling in the forward transformation, and 1/N scaling for the backward.

Parameters

h	FFT handle.
in	Input waveform segment.
out	Output spectrum.

3.11.3.12 void mha_fft_spec2wave_scale (mha_fft_t *h,* const mha_spec_t * *in,* mha_wave_t * *out*)

Tranform spectrum into waveform segment.

The _scale methods use standard DFT scaling: There is no scaling in the forward transformation, and 1/N scaling for the backward.

Parameters

h	FFT handle.
in	Input spectrum.
out	Output waveform segment.

4 Namespace Documentation

4.1 ac2lsl Namespace Reference

All types for the **ac2lsl** (p. 78) plugins live in this namespace.

Classes

- class **ac2lsl_t**
  *Plugin class of **ac2lsl** (p. 78).*
- class **cfg_t**
  *Runtime configuration class of the **ac2lsl** (p. 78) plugin.*
- class **save_var_base_t**
  Interface for ac to lsl bridge variable.
- class **save_var_t**
  Implementation for all ac to lsl bridges except complex types.
- class **save_var_t**< **mha_complex_t** >
  *Template specialization of the **ac2lsl** (p. 78) bridge to take care of complex numbers.*
- struct **type_info**

Variables

- const std::map< int, **type_info** > **types**

4.1.1 Detailed Description

All types for the **ac2lsl** (p. 78) plugins live in this namespace.

4.1.2 Variable Documentation

4.1.2.1 const std::map<int, type_info> ac2lsl::types

4.2 acmon Namespace Reference

Namespace for displaying ac variables as parser monitors.

Classes

- class **ac_monitor_t**
  A class for converting AC variables to Parser monitors of correct type.
- class **acmon_t**

4.2.1 Detailed Description

Namespace for displaying ac variables as parser monitors.

4.3 acsave Namespace Reference

Classes

- class **acsave_t**
- class **cfg_t**
- struct **mat4head_t**
- class **save_var_t**

4.4 addsndfile Namespace Reference

Classes

- class **addsndfile_if_t**
- class **level_adapt_t**
- class **resampled_soundfile_t**

  Reads sound from file and resamples it if necessary and wanted.
- class **sndfile_t**
- class **waveform_proxy_t**

  Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in ***resampled_soundfile_t*** *(p. 184).*

Typedefs

- typedef **MHAPlugin::config_t**< **level_adapt_t** > **level_adaptor**
- typedef **MHAPlugin::plugin_t**< **sndfile_t** > **wave_reader**

Enumerations

Functions

- static unsigned **resampled_num_frames** (unsigned num_source_frames, float source↩ _rate, float target_rate, **addsndfile_resampling_mode_t** resampling_mode)

4.4.1 Typedef Documentation

4.4.1.1 typedef MHAPlugin::config_t<level_adapt_t> addsndfile::level_adaptor

4.4.1.2 typedef MHAPlugin::plugin_t<sndfile_t> addsndfile::wave_reader

4.4.2 Enumeration Type Documentation

4.4.2.1 enum addsndfile::addsndfile_resampling_mode_t

Specifies the resampling mode in **resampled_soundfile_t** (p. 184).

Enumerator

DONT_RESAMPLE_PERMISSIVE

DONT_RESAMPLE_STRICT Do not resample, if the sample rate of the MHA differs from the sample rate of the sound file, raise an error.

DO_RESAMPLE Resample.

4.4.3 Function Documentation

4.4.3.1 static unsigned addsndfile::resampled_num_frames (unsigned *num_source_frames,* float *source_rate,* float *target_rate,* addsndfile_resampling_mode_t *resampling_mode*) [static]

4.5 ADM Namespace Reference

Classes

- class **ADM**

  Adaptive differential microphone, working for speech frequency range.
- class **Delay**

  A delay-line class which can also do subsample-delays for a limited frequency range below fs/4.
- class **Linearphase_FIR**

  An efficient linear-phase fir filter implementation.

Functions

- static double **subsampledelay_coeff** (double samples, double f_design, double fs=1.0)

  compute IIR coefficient for subsample delay

Variables

- const double **PI** = 3.14159265358979312
- const double **C** = 340
- const double **DELAY_FREQ** = 2000
- const double **START_BETA** = 0.5

4.5.1 Function Documentation

4.5.1.1 static double ADM::subsampledelay_coeff (double *samples,* double *f_design,* double *fs =* `1.0`) [static]

compute IIR coefficient for subsample delay

Parameters

samples	Constraint: 0.0 <= samples < 1.0; Amount of sub-sample delay
f_design	design frequency (subsample delay is accurate for this frequency)
fs	sampling rate

Returns

IIR coefficient for subsample delay

4.5.2 Variable Documentation

4.5.2.1 const double ADM::PI = 3.14159265358979312

4.5.2.2 const double ADM::C = 340

4.5.2.3 const double ADM::DELAY_FREQ = 2000

4.5.2.4 const double ADM::START_BETA = 0.5

4.6 AuditoryProfile Namespace Reference

Namespace for classes and functions around the auditory profile (e.g., audiogram handling)

Classes

- class **fmap_t**
  A class to store frequency dependent data (e.g., HTL and UCL).
- class **parser_t**
  Class to make the auditory profile accessible through the parser interface.
- class **profile_t**
  The Auditory Profile class.

4.6.1 Detailed Description

Namespace for classes and functions around the auditory profile (e.g., audiogram handling)

The auditory profile as defined by HearCom or BMBF Modellbasierte Hoergeraete is stored in the class **AuditoryProfile::profile_t** (p. 226). Until a complete definition is available, only the currently needed elements are implemented.

4.7 coherence Namespace Reference

Classes

- class **cohflt_if_t**
- class **cohflt_t**
- class **vars_t**

Functions

- void **getcipd** (**mha_complex_t** &c, **mha_real_t** &a, const **mha_complex_t** &xl, const **mha_complex_t** &xr)

4.7.1 Function Documentation

4.7.1.1 void coherence::getcipd (mha_complex_t & *c,* mha_real_t & *a,* const mha_complex_t & *xl,* const mha_complex_t & *xr*) [inline]

4.8 dc Namespace Reference

Classes

- class **dc_if_t**
- class **dc_t**
- class **dc_vars_t**
- class **dc_vars_validator_t**
- class **wb_inhib_cfg_t**
- class **wideband_inhib_vars_t**

Functions

- unsigned int **get_audiochannels** (unsigned int totalchannels, std::string acname, **algo_comm_t** ac)

4.8.1 Function Documentation

4.8.1.1 unsigned int dc::get_audiochannels (unsigned int *totalchannels,* std::string *acname,* algo_comm_t *ac*)

4.9 dc_simple Namespace Reference

Classes

- class **dc_if_t**
- class **dc_t**
- class **dc_vars_t**
- class **dc_vars_validator_t**
- class **level_smoother_t**

Typedefs

- typedef **MHAPlugin::plugin_t**< **dc_t** > **DC**
- typedef **MHAPlugin::config_t**< **level_smoother_t** > **LEVEL**

Functions

- void **test_fail** (const std::vector< float > &v, unsigned int s, const std::string &name)
- std::vector< float > **force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)
- **mha_real_t not_zero** (**mha_real_t** x, const std::string &comment="")

4.9.1 Typedef Documentation

4.9.1.1 typedef MHAPlugin::plugin_t<dc_t> dc_simple::DC

4.9.1.2 typedef MHAPlugin::config_t<level_smoother_t> dc_simple::LEVEL

4.9.2 Function Documentation

4.9.2.1 void dc_simple::test_fail (const std::vector< float > & *v,* unsigned int *s,* const std::string & *name*)

4.9.2.2 std::vector<float> dc_simple::force_resize (const std::vector< float > & *v,* unsigned int *s,* const std::string & *name*)

4.9.2.3 mha_real_t dc_simple::not_zero (mha_real_t *x,* const std::string & *comment* = "")

4.10 delay Namespace Reference

Classes

- class **interface_t**

4.11 delaysum Namespace Reference

This namespace contains the delaysum plugin.

Classes

- class **delaysum_if_t**
  Interface class for the delaysum plugin.
- class **delaysum_t**
  Runtime configuration of the delaysum plugin.

4.11.1 Detailed Description

This namespace contains the delaysum plugin.

4.12 DynComp Namespace Reference

dynamic compression related classes and functions

Classes

- class **dc_afterburn_rt_t**
  Real-time class for after burn effect.
- class **dc_afterburn_t**
  Afterburn class, to be defined as a member of compressors.
- class **dc_afterburn_vars_t**
  *Variables for **dc_afterburn_t*** (p. 294) *class.*
- class **gaintable_t**
  Gain table class.

Functions

- **mha_real_t interp1** (const std::vector< **mha_real_t** > &vX, const std::vector< **mha_real_t** > &vY, **mha_real_t** X)
  One-dimensional linear interpolation.
- **mha_real_t interp2** (const std::vector< **mha_real_t** > &vX, const std::vector< **mha_real_t** > &vY, const std::vector< std::vector< **mha_real_t** > > &mZ, **mha_real_t** X, **mha_real_t** Y)
  Linear interpolation in a two-dimensional field.

4.12.1 Detailed Description

dynamic compression related classes and functions

4.12.2 Function Documentation

4.12.2.1 mha_real_t DynComp::interp1 (const std::vector< mha_real_t > & *vX*, const std::vector< mha_real_t > & *vY*, mha_real_t *X*)

One-dimensional linear interpolation.

Parameters

vX	Vector with input samples.
vY	Vector with values at input samples.
X	Input value to be interpolated.

Return values

Interpolated	value Y(X) at position X.

4.12.2.2 mha_real_t DynComp::interp2 (const std::vector< mha_real_t > & *vX,* const std::vector< mha_real_t > & *vY,* const std::vector< std::vector< mha_real_t > > & *mZ,* mha_real_t *X,* mha_real_t *Y*)

Linear interpolation in a two-dimensional field.

Parameters

vX	Vector with input samples, first dimension.
vY	Vector with input samples, second dimension.
mZ	Field with values at input samples.
X	First dimension of input value to be interpolated.
Y	Second dimension of input value to be interpolated.

Return values

Interpolated	value Z(X,Y) at position X,Y.

4.13 fader_wave Namespace Reference

Classes

- class **fader_wave_if_t**
- class **level_adapt_t**

Typedefs

- typedef **MHAPlugin::plugin_t< level_adapt_t > level_adaptor**

4.13.1 Typedef Documentation

4.13.1.1 typedef MHAPlugin::plugin_t<level_adapt_t> fader_wave::level_adaptor

4.14 fftfbpow Namespace Reference

Namespace for the fftfbpow plugin.

Classes

- class **fftfbpow_interface_t**
  Interface class for fftfbpow plugin.
- class **fftfbpow_t**
  Run time configuration for the fftfbpow plugin.

4.14.1 Detailed Description

Namespace for the fftfbpow plugin.

4.15 fftfilterbank Namespace Reference

Classes

- class **fftfb_interface_t**
- class **fftfb_plug_t**

4.16 fshift Namespace Reference

All types for the fshift plugin live in this namespace.

Classes

- class **fshift_config_t**
  fshift runtime config class
- class **fshift_t**
  fshift plugin interface class

Functions

- int **fft_find_bin** (**mha_real_t** frequency, unsigned fftlen, **mha_real_t** srate)
  Finds bin number of FFT bin nearest to the given frequency.

4.16.1 Detailed Description

All types for the fshift plugin live in this namespace.

4.16.2 Function Documentation

4.16.2.1 int fshift::fft_find_bin (

mha_real_t *frequency,*
unsigned *fftlen,*
mha_real_t *srate*)

Finds bin number of FFT bin nearest to the given frequency.

Parameters

frequency	The frequency for which to look. Has to be in range [-srate/2,+srate/2]
fftlen	Length of the FFT.
srate	Sampling rate of the waveform from which the FFT originates.

Returns

Bin number of the FFT bin corresponding to frequency

4.17 fshift_hilbert Namespace Reference

All types for the hilbert frequency shifter live in this namespace.

Classes

- class **frequency_translator_t**
- class **hilbert_shifter_t**

4.17.1 Detailed Description

All types for the hilbert frequency shifter live in this namespace.

4.18 gain Namespace Reference

Classes

- class **gain_if_t**
- class **scaler_t**

4.19 gtfb_analyzer Namespace Reference

Classes

- struct **gtfb_analyzer_cfg_t**

  Configuration for Gammatone Filterbank Analyzer.
- class **gtfb_analyzer_t**

  Gammatone Filterbank Analyzer Plugin.

4.20 matrixmixer Namespace Reference

Classes

- class **cfg_t**
- class **matmix_t**

4.21 MHA_AC Namespace Reference

Functions and classes for Algorithm Communication (AC) support.

Classes

- class **ac2matrix_helper_t**
- class **ac2matrix_t**
  Copy AC variable to a matrix.
- class **acspace2matrix_t**
  Copy all or a subset of all numeric AC variables into an array of matrixes.
- class **double_t**
  Insert a double precision floating point variable into the AC space.
- class **float_t**
  Insert a float point variable into the AC space.
- class **int_t**
  Insert a integer variable into the AC space.
- class **spectrum_t**
  *Insert a **MHASignal::spectrum_t** (p. 800) class into the AC space.*
- class **stat_t**
- class **waveform_t**
  *Insert a **MHASignal::waveform_t** (p. 812) class into the AC space.*

Functions

- **mha_spec_t get_var_spectrum** (**algo_comm_t** ac, const std::string &name)
  Convert an AC variable into a spectrum.
- **mha_wave_t get_var_waveform** (**algo_comm_t** ac, const std::string &name)
  Convert an AC variable into a waveform.
- int **get_var_int** (**algo_comm_t** ac, const std::string &name)
  Return value of an integer scalar AC variable.
- float **get_var_float** (**algo_comm_t** ac, const std::string &name)
  Return value of an floating point scalar AC variable.
- std::vector< float > **get_var_vfloat** (**algo_comm_t** ac, const std::string &name)
  Return value of an floating point vector AC variable as standard vector of floats.

4.21.1 Detailed Description

Functions and classes for Algorithm Communication (AC) support.

4.22 mha_error_helpers Namespace Reference

Functions

- unsigned **digits** (unsigned n)

  Compute number of decimal digits required to represent an unsigned integer.
- unsigned **snprintf_required_length** (const char *formatstring,...)

  snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.

4.22.1 Function Documentation

4.22.1.1 unsigned mha_error_helpers::digits (unsigned *n*)

Compute number of decimal digits required to represent an unsigned integer.

Parameters

n	The unsigned integer that we want to know the number of required decimal digits for. return The number of decimal digits in `n`.

4.22.1.2 unsigned mha_error_helpers::snprintf_required_length (const char * *formatstring,* ...)

snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.

Parameters

formatstring	The format string with standard printf formatstring

Returns

the number of bytes required by printf without the terminating nul

4.23 MHA_TCP Namespace Reference

A Namespace for TCP helper classes.

Classes

- class **Async_Notify**
  Portable Multiplexable cross-thread notification.
- class **Client**
  A portable class for a tcp client connections.
- class **Connection**
  Connection *(p. 471) handles Communication between client and server, is used on both sides.*
- class **Event_Watcher**
  OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.
- struct **OS_EVENT_TYPE**
- class **Server**
- class **sock_init_t**
- class **Sockaccept_Event**
- class **Sockread_Event**
  Watch socket for incoming data.
- class **Sockwrite_Event**
- class **Thread**
  A very simple class for portable threads.
- class **Timeout_Event**
- class **Timeout_Watcher**
  OS-independent event watcher with internal fixed-end-time timeout.
- class **Wakeup_Event**
  A base class for asynchronous wakeup events.

Typedefs

- typedef int **SOCKET**

Functions

- std::string **STRERROR** (int err)
  Portable conversion from error number to error string.
- std::string **HSTRERROR** (int err)
  Portable conversion from hostname error number to error string.
- int **N_ERRNO** ()
  Portable access to last network error number.
- int **H_ERRNO** ()
  Portable access to last hostname error number.

- int **G_ERRNO** ()

  Portable access to last non-network error number.
- double **dtime** ()

  Time access function for system's high resolution time, retrieve current time as double.
- double **dtime** (const struct timeval &tv)

  Time access function for unix' high resolution time, converts struct timeval to double.
- struct timeval **stime** (double d)

  Time access function for unix' high resolution time, converts time from double to struct timeval.

Variables

- class **MHA_TCP::sock_init_t sock_initializer**

4.23.1 Detailed Description

A Namespace for TCP helper classes.

4.23.2 Typedef Documentation

4.23.2.1 typedef int MHA_TCP::SOCKET

4.23.3 Function Documentation

4.23.3.1 std::string MHA_TCP::STRERROR (int *err*)

Portable conversion from error number to error string.

4.23.3.2 std::string MHA_TCP::HSTRERROR (int *err*)

Portable conversion from hostname error number to error string.

4.23.3.3 int MHA_TCP::N_ERRNO ( )

Portable access to last network error number.

4.23.3.4 int MHA_TCP::H_ERRNO ( )

Portable access to last hostname error number.

4.23.3.5 int MHA_TCP::G_ERRNO ( )

Portable access to last non-network error number.

4.23.3.6 double MHA_TCP::dtime ( )

Time access function for system's high resolution time, retrieve current time as double.

4.23.3.7 double MHA_TCP::dtime (const struct timeval & *tv*)

Time access function for unix' high resolution time, converts struct timeval to double.

4.23.3.8 struct timeval MHA_TCP::stime (double *d*)

Time access function for unix' high resolution time, converts time from double to struct timeval.

4.23.4 Variable Documentation

4.23.4.1 class MHA_TCP::sock_init_t MHA_TCP::sock_initializer

4.24 mhachain Namespace Reference

Classes

- class **chain_base_t**
- class **mhachain_t**
- class **plugs_t**

4.25 MHAEvents Namespace Reference

Collection of event handling classes.

Classes

- class **connector_base_t**
- class **connector_t**
- class **emitter_t**
  Class for emitting openMHA events.
- class **patchbay_t**
  Patchbay which connects any event emitter with any member function of the parameter class.

4.25.1 Detailed Description

Collection of event handling classes.

4.26 MHAFilter Namespace Reference

Namespace for IIR and FIR filter classes.

Classes

- class **adapt_filter_param_t**
- class **adapt_filter_state_t**
- class **adapt_filter_t**
  Adaptive filter.
- class **blockprocessing_polyphase_resampling_t**
  A class that does polyphase resampling and takes into account block processing.
- class **complex_bandpass_t**
  Complex bandpass filter.
- class **diff_t**
  Differentiator class (non-normalized)
- class **fftfilter_t**
  FFT based FIR filter implementation.
- class **fftfilterbank_t**
  FFT based FIR filterbank implementation.
- class **filter_t**
  Generic IIR filter class.
- class **gamma_flt_t**
  Class for gammatone filter.
- class **iir_filter_state_t**
- class **iir_filter_t**
  IIR filter class wrapper for integration into parser structure.
- class **iir_ord1_real_t**
  First order recursive filter.
- class **o1_ar_filter_t**
  First order attack-release lowpass filter.
- class **o1flt_lowpass_t**
  First order low pass filter.
- class **o1flt_maxtrack_t**
  First order maximum tracker.
- class **o1flt_mintrack_t**
  First order minimum tracker.
- class **partitioned_convolution_t**
  A filter class for partitioned convolution.

- class **polyphase_resampling_t**

  A class that performs polyphase resampling.
- class **resampling_filter_t**

  Hann shaped low pass filter for resampling.
- class **smoothspec_t**

  Smooth spectral gains, create a windowed impulse response.
- class **thirdoctave_analyzer_t**
- struct **transfer_function_t**

  a structure containing a source channel number, a target channel number, and an impulse response.
- struct **transfer_matrix_t**

  A sparse matrix of transfer function partitionss.

Functions

- void **make_friendly_number** (**mha_real_t** &x)
- void **make_friendly_number** (**mha_complex_t** &x)
- void **make_friendly_number** (double &x)
- void **o1_lp_coeffs** (const **mha_real_t** tau, const **mha_real_t** fs, **mha_real_t** &c1, **mha_real_t** &c2)

  Set first order filter coefficients from time constant and sampling rate.
- void **butter_stop_ord1** (double *A, double *B, double f1, double f2, double fs)

  Setup a first order butterworth band stop filter.
- **MHASignal::waveform_t** * **spec2fir** (const **mha_spec_t** *spec, const unsigned int fftlen, const **MHAWindow::base_t** &window, const bool minphase)

  Create a windowed impulse response/FIR filter coefficients from a spectrum.
- unsigned **gcd** (unsigned a, unsigned b)

  greatest common divisor
- double **sinc** (double x)

  sin(x)/x function, coping with x=0.
- std::pair< unsigned, unsigned > **resampling_factors** (float source_sampling_rate, float target_sampling_rate, float factor=1.0f)

  Computes rational resampling factor from two sampling rates.

4.26.1 Detailed Description

Namespace for IIR and FIR filter classes.

4.26.2 Function Documentation

4.26.2.1 void MHAFilter::make_friendly_number (mha_real_t & *x*) [inline]

4.26.2.2 void MHAFilter::make_friendly_number (mha_complex_t & *x*) [inline]

4.26.2.3 void MHAFilter::make_friendly_number (double & *x*) [inline]

4.26.2.4 void MHAFilter::o1_lp_coeffs (const mha_real_t *tau,* const mha_real_t *fs,* mha_real_t & *c1,* mha_real_t & *c2*)

Set first order filter coefficients from time constant and sampling rate.

Parameters

tau	Time constant
fs	Sampling rate

Return values

c1	Recursive filter coefficient
c2	Non-recursive filter coefficient

4.26.2.5 void MHAFilter::butter_stop_ord1 (double * *A,* double * *B,* double *f1,* double *f2,* double *fs*)

Setup a first order butterworth band stop filter.

This function calculates the filter coefficients of a first order butterworth band stop filter.

Return values

A	recursive filter coefficients
B	non recursive filter coefficients

Parameters

f1	lower frequency
f2	upper frequency
fs	sample frequency

4.26.2.6 MHASignal::waveform_t * MHAFilter::spec2fir (const mha_spec_t * *spec,* const unsigned int *fftlen,* const MHAWindow::base_t & *window,* const bool *minphase*)

Create a windowed impulse response/FIR filter coefficients from a spectrum.

Parameters

spec	Input spectrum
fftlen	FFT length of spectrum
window	Window shape (with length, e.g. initialized with MHAWindow::hanning(54)).
minphase	Flag, true if original phase should be discarded and replaced by a minimal phase function.

4.26.2.7 unsigned MHAFilter::gcd (unsigned *a,* unsigned *b*) [inline]

greatest common divisor

4.26.2.8 double MHAFilter::sinc (double *x*)

sin(x)/x function, coping with x=0.

This is the historical sinc function, not the normalized sinc function.

4.26.2.9 std::pair< unsigned, unsigned > MHAFilter::resampling_factors (float *source_sampling_rate,* float *target_sampling_rate,* float *factor* = 1.0f)

Computes rational resampling factor from two sampling rates.

The function will fail if either sampling_rate * factor is not an integer

Parameters

source_sampling_rate	The original sampling rate
target_sampling_rate	The desired sampling rate
factor	A helper factor to use for non-integer sampling rates

Returns

a pair that contains first the upsampling factor and second the downsampling factor required for the specified resampling.

Exceptions

MHA_Error *(p. 445)*	if no rational resampling factor can be found.

4.27 MHAIOJack Namespace Reference

JACK IO.

Classes

- class **io_jack_t**
  Main class for JACK IO.

4.27.1 Detailed Description

JACK IO.

4.28 MHAIOJackdb Namespace Reference

Classes

- class **io_jack_t**
  Main class for JACK IO.

4.29 MHAIOPortAudio Namespace Reference

Classes

- class **device_info_t**
- class **io_portaudio_t**
  Main class for Portaudio sound IO.

Functions

- static std::string **parserFriendlyName** (const std::string &in)

4.29.1 Function Documentation

4.29.1.1 static std::string MHAIOPortAudio::parserFriendlyName (const std::string & *in*) [static]

4.30 MHAJack Namespace Reference

Classes and functions for openMHA and JACK interaction.

Classes

- class **client_avg_t**

  Generic JACK client for averaging a system response across time.
- class **client_noncont_t**

  Generic client for synchronous playback and recording of waveform fragments.
- class **client_t**

  Generic asynchronous JACK client.
- class **port_t**

  Class for one channel/port.

Functions

- void **io** (**mha_wave_t** ∗s_out, **mha_wave_t** ∗s_in, const std::string &name, const std↩ ::vector< std::string > &p_out, const std::vector< std::string > &p_in, float ∗srate=NULL, unsigned int ∗fragsize=NULL, bool use_jack_transport=false)

  Functional form of generic client for synchronous playback and recording of waveform fragments.
- std::vector< unsigned int > **get_port_capture_latency** (const std::vector< std::string > &ports)

  Return the JACK port latency of ports.
- std::vector< int > **get_port_capture_latency_int** (const std::vector< std::string > &ports)

  Return the JACK port latency of ports.
- std::vector< unsigned int > **get_port_playback_latency** (const std::vector< std::string > &ports)

  Return the JACK port latency of ports.
- std::vector< int > **get_port_playback_latency_int** (const std::vector< std::string > &ports)

4.30.1 Detailed Description

Classes and functions for openMHA and JACK interaction.

4.30.2 Function Documentation

4.30.2.1 void MHAJack::io (mha_wave_t * *s_out*, mha_wave_t * *s_in*, const std::string & *name*, const std::vector< std::string > & *p_out*, const std::vector< std::string > & *p_in*, float * *srate* = NULL, unsigned int * *fragsize* = NULL, bool *use_jack_transport* = false)

Functional form of generic client for synchronous playback and recording of waveform fragments.

4.30.2.2 std::vector< unsigned int > MHAJack::get_port_capture_latency (const std::vector< std::string > & *ports*)

Return the JACK port latency of ports.

Parameters

ports	Ports to be tested

Returns

Latency vector (one entry for each port)

4.30.2.3 std::vector< int > MHAJack::get_port_capture_latency_int (const std::vector< std::string > & *ports*)

Return the JACK port latency of ports.

Parameters

ports	Ports to be tested

Returns

Latency vector (one entry for each port)

4.30.2.4 std::vector< unsigned int > MHAJack::get_port_playback_latency (const std::vector< std::string > & *ports*)

Return the JACK port latency of ports.

Parameters

ports	Ports to be tested

Returns

Latency vector (one entry for each port)

4.30.2.5 std::vector< int > MHAJack::get_port_playback_latency_int (const std::vector< std::string > & *ports*)

4.31 MHAKernel Namespace Reference

Classes

- class **algo_comm_class_t**
- class **comm_var_map_t**

Functions

- **algo_comm_class_t** * **algo_comm_safe_cast** (void *)

4.31.1 Function Documentation

4.31.1.1 MHAKernel::algo_comm_class_t * MHAKernel::algo_comm_safe_cast (void * *h*)

4.32 MHAMultiSrc Namespace Reference

Collection of classes for selecting audio chunks from multiple sources.

Classes

- class **base_t**

  Base class for source selection.
- class **channel_t**
- class **channels_t**
- class **spectrum_t**
- class **waveform_t**

4.32.1 Detailed Description

Collection of classes for selecting audio chunks from multiple sources.

4.33 MHAOvlFilter Namespace Reference

Namespace for overlapping FFT based filter bank classes and functions.

Namespaces

- **barkscale**
- **FreqScaleFun**

  Transform functions from linear scale in Hz to new frequency scales.
- **ShapeFun**

  Shape functions for overlapping filters.

Classes

- class **band_descriptor_t**
- class **fftfb_ac_info_t**
- class **fftfb_t**

  FFT based overlapping filter bank.
- class **fftfb_vars_t**

  Set of configuration variables for FFT-based overlapping filters.
- class **fscale_bw_t**
- class **fscale_t**
- class **fspacing_t**

  Class for frequency spacing, used by filterbank shape generator class.
- class **overlap_save_filterbank_analytic_t**
- class **overlap_save_filterbank_t**

  *A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 616).*
- class **scale_var_t**

Typedefs

- typedef **mha_real_t**(**scale_fun_t**) (**mha_real_t**)

4.33.1 Detailed Description

Namespace for overlapping FFT based filter bank classes and functions.

4.33.2 Typedef Documentation

4.33.2.1 typedef mha_real_t(MHAOvlFilter::scale_fun_t) (mha_real_t)

4.34 MHAOvlFilter::barkscale Namespace Reference

Classes

- class **bark2hz_t**
- class **hz2bark_t**

Variables

- **mha_real_t vfreq** [**BARKSCALE_ENTRIES**]
- **mha_real_t vbark** [**BARKSCALE_ENTRIES**]

4.34.1 Variable Documentation

4.34.1.1 mha_real_t MHAOvlFilter::barkscale::vfreq

4.34.1.2 mha_real_t MHAOvlFilter::barkscale::vbark

4.35 MHAOvlFilter::FreqScaleFun Namespace Reference

Transform functions from linear scale in Hz to new frequency scales.

Functions

- **mha_real_t hz2hz** (**mha_real_t** x)
  Dummy scale transformation Hz to Hz.
- **mha_real_t hz2khz** (**mha_real_t** x)
- **mha_real_t hz2octave** (**mha_real_t** x)
- **mha_real_t hz2third_octave** (**mha_real_t** x)
- **mha_real_t hz2bark** (**mha_real_t** x)
  Transformation to bark scale.
- **mha_real_t hz2bark_analytic** (**mha_real_t**)
- **mha_real_t hz2erb** (**mha_real_t**)
- **mha_real_t hz2erb_glasberg1990** (**mha_real_t**)
- **mha_real_t hz2log** (**mha_real_t** x)
  Third octave frequency scale.
- **mha_real_t inv_scale** (**mha_real_t**, **mha_real_t**(∗)(**mha_real_t**))

4.35.1 Detailed Description

Transform functions from linear scale in Hz to new frequency scales.

4.35.2 Function Documentation

4.35.2.1 mha_real_t MHAOvlFilter::FreqScaleFun::hz2hz (mha_real_t *x*)

Dummy scale transformation Hz to Hz.

This function implements a dummy scale transformation (linear frequency scale).

Parameters

x	Input frequency in Hz

Returns

Frequency in Hz

4.35.2.2 mha_real_t MHAOvlFilter::FreqScaleFun::hz2khz (mha_real_t *x*)

4.35.2.3 mha_real_t MHAOvlFilter::FreqScaleFun::hz2octave (mha_real_t *x*)

4.35.2.4 mha_real_t MHAOvlFilter::FreqScaleFun::hz2third_octave (mha_real_t *x*)

4.35.2.5 mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark (mha_real_t *x*)

Transformation to bark scale.

This function implements a critical band rate (bark) scale.

Parameters

x	Input frequency in Hz

Returns

Critical band rate in Bark

4.35.2.6 mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark_analytic (mha_real_t *x*)

4.35.2.7 mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb (mha_real_t *x*)

4.35.2.8 mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb_glasberg1990 (mha_real_t *x*)

4.35.2.9 mha_real_t MHAOvlFilter::FreqScaleFun::hz2log (mha_real_t *x*)

Third octave frequency scale.

This function implements a third octave scale. Frequencies below 16 Hz are mapped to 16 Hz.

Parameters

x	Frequency in Hz

Returns

Third octaves relative to 1000 Hz

4.35.2.10 mha_real_t MHAOvlFilter::FreqScaleFun::inv_scale (mha_real_t *y*, mha_real_t(∗)(mha_real_t) *fun*)

4.36 MHAOvlFilter::ShapeFun Namespace Reference

Shape functions for overlapping filters.

Functions

- **mha_real_t rect** (**mha_real_t** x)
  Filter shape function for rectangular filters.
- **mha_real_t linear** (**mha_real_t** x)
  Filter shape function for sawtooth filters.
- **mha_real_t hann** (**mha_real_t** x)
  Filter shape function for hanning shaped filters.
- **mha_real_t expflt** (**mha_real_t**)
- **mha_real_t gauss** (**mha_real_t**)

4.36.1 Detailed Description

Shape functions for overlapping filters.

4.36.2 Function Documentation

4.36.2.1 mha_real_t MHAOvlFilter::ShapeFun::rect (mha_real_t *x*)

Filter shape function for rectangular filters.

This function creates rectangular filter shapes. The edge is exactly half way between two center frequencies (on a given scale).

Parameters

x	Input value in the range [-1,1].

Returns

Weigth function in the range [0,1]

4.36.2.2 mha_real_t MHAOvlFilter::ShapeFun::linear (mha_real_t *x*)

Filter shape function for sawtooth filters.

This function creates sawtooth filter shapes. They rise linearly form 0 to 1 in the interval from the lower neighbor center frequency to the band center frequency and from 1 to 0 in the interval from the band center frequency to the upper neighbour band center frequency. Linear means linear on a given frequency scale.

Parameters

x	Input value in the range [-1,1].

Returns

Weigth function in the range [0,1]

4.36.2.3 mha_real_t MHAOvlFilter::ShapeFun::hann (mha_real_t *x*)

Filter shape function for hanning shaped filters.

This function creates hanning window shaped filters.

Parameters

x	Input value in the range [-1,1].

Returns

Weigth function in the range [0,1]

4.36.2.4 mha_real_t MHAOvlFilter::ShapeFun::expflt (mha_real_t *x*)

4.36.2.5 mha_real_t MHAOvlFilter::ShapeFun::gauss (mha_real_t *x*)

4.37 MHAParser Namespace Reference

Name space for the openMHA-Parser configuration language.

Namespaces

- **StrCnv**

  String converter namespace.

Classes

- class **base_t**
  Base class for all parser items.
- class **bool_mon_t**
  Monitor with string value.
- class **bool_t**
  Variable with a boolean value ("yes"/"no")
- class **c_ifc_parser_t**
- class **commit_t**
  Parser variable with event-emission functionality.
- class **complex_mon_t**
  Monitor with complex value.
- class **complex_t**
  Variable with complex value.
- class **entry_t**
- class **expression_t**
- class **float_mon_t**
  Monitor with float value.
- class **float_t**
  Variable with float value.
- class **int_mon_t**
  Monitor variable with int value.
- class **int_t**
  Variable with integer value.
- class **keyword_list_t**
  Keyword list class.
- class **kw_t**
  Variable with keyword list value.
- class **mcomplex_mon_t**
  Matrix of complex numbers monitor.
- class **mcomplex_t**
  Matrix variable with complex value.
- class **mfloat_mon_t**
  Matrix of floats monitor.
- class **mfloat_t**
  Matrix variable with float value.
- class **mhaconfig_mon_t**
- class **mhapluginloader_t**
  Class to create a plugin loader in a parser, including the load logic.
- class **mint_mon_t**
  Matrix of ints monitor.
- class **monitor_t**
  Base class for monitors and variable nodes.
- class **parser_t**

Parser node class.

- class **range_var_t**

  Base class for all variables with a numeric value range.

- class **string_mon_t**

  Monitor with string value.

- class **string_t**

  Variable with a string value.

- class **variable_t**

  Base class for variable nodes.

- class **vcomplex_mon_t**

  Monitor with vector of complex values.

- class **vcomplex_t**

  Vector variable with complex value.

- class **vfloat_mon_t**

  Vector of floats monitor.

- class **vfloat_t**

  Vector variable with float value.

- class **vint_mon_t**

  Vector of ints monitor.

- class **vint_t**

  Variable with vector<int> value.

- class **vstring_mon_t**

  Vector of monitors with string value.

- class **vstring_t**

  Vector variable with string values.

- class **window_t**

  MHA configuration interface for a window function generator.

Typedefs

- typedef std::string(base_t::* **opact_t**) (**expression_t** &)
- typedef std::string(base_t::* **query_t**) (const std::string &)
- typedef std::map< std::string, **opact_t** > **opact_map_t**
- typedef std::map< std::string, **query_t** > **query_map_t**
- typedef std::list< **entry_t** > **entry_map_t**
- typedef int(* **c_parse_cmd_t**) (void *, const char *, char *, unsigned int)
- typedef const char *(* **c_parse_err_t**) (void *, int)

Functions

- int **get_precision** ()
- std::string **commentate** (const std::string &s)
- void **trim** (std::string &s)
- std::string **cfg_dump** (**base_t** ∗, const std::string &)
- std::string **cfg_dump_short** (**base_t** ∗, const std::string &)
- std::string **all_dump** (**base_t** ∗, const std::string &)
- std::string **mon_dump** (**base_t** ∗, const std::string &)
- std::string **all_ids** (**base_t** ∗, const std::string &, const std::string &="")
- void **strreplace** (std::string &, const std::string &, const std::string &)

  string replace function
- void **envreplace** (std::string &s)

4.37.1 Detailed Description

Name space for the openMHA-Parser configuration language.

This namespace contains all classes which are needed for the implementation of the open↩ MHA configuration language. For details on the script language itself please see section **The openMHA configuration language** (p. 34).

4.37.2 List of valid MHAParser items

- **Sub-parser**: **parser_t** (p. 688)
- **Variables**:
  Numeric variables: **int_t** (p. 664), **vint_t** (p. 713), **float_t** (p. 659), **vfloat_t** (p. 709), **mfloat_t** (p. 677)
  Other variables: **string_t** (p. 697), **vstring_t** (p. 717), **kw_t** (p. 669), **bool_t** (p. 645)
- **Monitors**:
  Numeric monitors: **int_mon_t** (p. 662), **vint_mon_t** (p. 711), **float_mon_t** (p. 658), **vfloat_mon_t** (p. 707)
  mfloat_mon_t (p. 676)
  mcomplex_mon_t (p. 672)
  Other monitors: **bool_mon_t** (p. 644), **string_mon_t** (p. 696), **vstring_mon_t** (p. 715)

Members can be inserted into the configuration namespace by using MHAParser::insert_item() or the **insert_member()** (p. 1021) macro.

4.37.3 Typedef Documentation

4.37.3.1 typedef std::string(base_t::* MHAParser::opact_t) (expression_t &)

4.37.3.2 typedef std::string(base_t::* MHAParser::query_t) (const std::string &)

4.37.3.3 typedef std::map<std::string,opact_t> MHAParser::opact_map_t

4.37.3.4 typedef std::map<std::string,query_t> MHAParser::query_map_t

4.37.3.5 typedef std::list<entry_t> MHAParser::entry_map_t

4.37.3.6 typedef int(* MHAParser::c_parse_cmd_t) (void *, const char *, char *, unsigned int)

4.37.3.7 typedef const char*(* MHAParser::c_parse_err_t) (void *, int)

4.37.4 Function Documentation

4.37.4.1 int MHAParser::get_precision ( )

4.37.4.2 std::string MHAParser::commentate (const std::string & *s*)

4.37.4.3 void MHAParser::trim (std::string & *s*)

4.37.4.4 std::string MHAParser::cfg_dump (base_t * *p*, const std::string & *pref*)

4.37.4.5 std::string MHAParser::cfg_dump_short (base_t * *p*, const std::string & *pref*)

4.37.4.6 std::string MHAParser::all_dump (base_t * *p*, const std::string & *pref*)

4.37.4.7 std::string MHAParser::mon_dump (base_t * *p*, const std::string & *pref*)

4.37.4.8 std::string MHAParser::all_ids (base_t * *p*, const std::string & *pref*, const std::string & *id* = " ")

4.37.4.9 void MHAParser::strreplace (std::string & *s*, const std::string & *arg*, const std::string & *rep*)

string replace function

Parameters

s	target string
arg	search pattern
rep	replace pattern

4.37.4.10 void MHAParser::envreplace (std::string & *s*)

4.38 MHAParser::StrCnv Namespace Reference

String converter namespace.

Functions

- int **num_brackets** (const std::string &s)
  Return number of brackets at beginning and end of string.
- int **bracket_balance** (const std::string &s)
- void **str2val** (const std::string &, bool &)
  Convert from string.
- void **str2val** (const std::string &, float &)
  Convert from string.
- void **str2val** (const std::string &, **mha_complex_t** &)
  Convert from string.
- void **str2val** (const std::string &, int &)
  Convert from string.
- void **str2val** (const std::string &, **keyword_list_t** &)
  Convert from string.
- void **str2val** (const std::string &, std::string &)
  Convert from string.
- template<class arg_t >
  void **str2val** (const std::string &s, std::vector< arg_t > &val)
  Converter for vector types.
- template<>
  void **str2val**< **mha_real_t** > (const std::string &s, std::vector< **mha_real_t** > &v)
  Converter for vector<mha_real_t> with Matlab-style expansion.
- template<class arg_t >
  void **str2val** (const std::string &s, std::vector< std::vector< arg_t > > &val)
  Converter for matrix types.
- std::string **val2str** (const bool &)
  Convert to string.
- std::string **val2str** (const float &)
  Convert to string.
- std::string **val2str** (const **mha_complex_t** &)

Convert to string.

- std::string **val2str** (const int &)
  Convert to string.
- std::string **val2str** (const **keyword_list_t** &)
  Convert to string.
- std::string **val2str** (const std::string &)
  Convert to string.
- std::string **val2str** (const std::vector< float > &)
  Convert to string.
- std::string **val2str** (const std::vector< **mha_complex_t** > &)
  Convert to string.
- std::string **val2str** (const std::vector< int > &)
  Convert to string.
- std::string **val2str** (const std::vector< std::vector< int > > &)
  Convert to string.
- std::string **val2str** (const std::vector< std::string > &)
  Convert to string.
- std::string **val2str** (const std::vector< std::vector< float > > &)
  Convert to string.
- std::string **val2str** (const std::vector< std::vector< **mha_complex_t** > > &)
  Convert to string.

4.38.1 Detailed Description

String converter namespace.

The functions defined in this namespace manage the conversions from C++ variables to strings and back. It was tried to keep a matlab compatible string format for vectors and vectors of vectors.

4.38.2 Function Documentation

4.38.2.1 int MHAParser::StrCnv::num_brackets (const std::string & *s*)

Return number of brackets at beginning and end of string.

Parameters

s	String

Returns

Number of brackets, or -1 for empty string

4.38.2.2 int MHAParser::StrCnv::bracket_balance (const std::string & *s*)

4.38.2.3 void MHAParser::StrCnv::str2val (const std::string & *s*, bool & *v*)

Convert from string.

4.38.2.4 void MHAParser::StrCnv::str2val (const std::string & *s*, float & *v*)

Convert from string.

4.38.2.5 void MHAParser::StrCnv::str2val (const std::string & *s*, mha_complex_t & *v*)

Convert from string.

4.38.2.6 void MHAParser::StrCnv::str2val (const std::string & *s*, int & *v*)

Convert from string.

4.38.2.7 void MHAParser::StrCnv::str2val (const std::string & *s*, MHAParser::keyword_list_t & *v*)

Convert from string.

4.38.2.8 void MHAParser::StrCnv::str2val (const std::string & *s*, std::string & *v*)

Convert from string.

4.38.2.9 template<class arg_t > void MHAParser::StrCnv::str2val (const std::string & *s*, std::vector< arg_t > & *val*)

Converter for vector types.

4.38.2.10 template<> void MHAParser::StrCnv::str2val< mha_real_t > (const std::string & *s,* std::vector< mha_real_t > & *v*)

Converter for vector<mha_real_t> with Matlab-style expansion.

4.38.2.11 template<class arg_t > void MHAParser::StrCnv::str2val (const std::string & *s,* std::vector< std::vector< arg_t > > & *val*)

Converter for matrix types.

4.38.2.12 std::string MHAParser::StrCnv::val2str (const bool & *v*)

Convert to string.

4.38.2.13 std::string MHAParser::StrCnv::val2str (const float & *v*)

Convert to string.

4.38.2.14 std::string MHAParser::StrCnv::val2str (const mha_complex_t & *v*)

Convert to string.

4.38.2.15 std::string MHAParser::StrCnv::val2str (const int & *v*)

Convert to string.

4.38.2.16 std::string MHAParser::StrCnv::val2str (const keyword_list_t & *v*)

Convert to string.

4.38.2.17 std::string MHAParser::StrCnv::val2str (const std::string & *v*)

Convert to string.

4.38.2.18 std::string MHAParser::StrCnv::val2str (const std::vector< float > & *v*)

Convert to string.

4.38.2.19 std::string MHAParser::StrCnv::val2str (const std::vector< mha_complex_t > & *v*)

Convert to string.

4.38.2.20 std::string MHAParser::StrCnv::val2str (const std::vector< int > & *v*)

Convert to string.

4.38.2.21 std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< int > > & *v*)

Convert to string.

4.38.2.22 std::string MHAParser::StrCnv::val2str (const std::vector< std::string > & *v*)

Convert to string.

4.38.2.23 std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< float > > & *v*)

Convert to string.

4.38.2.24 std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< mha_complex_t > > & *v*)

Convert to string.

4.39 MHAPlugin Namespace Reference

Namespace for openMHA plugin class templates and thread-safe runtime configurations.

Classes

- class **cfg_chain_t**
- class **config_t**

  Template class for thread safe configuration.
- class **plugin_t**

  The template class for C++ openMHA plugins.

4.39.1 Detailed Description

Namespace for openMHA plugin class templates and thread-safe runtime configurations.

4.40 MHAPlugin_Resampling Namespace Reference

Classes

- class **resampling_if_t**
- class **resampling_t**

4.41 MHAPlugin_Split Namespace Reference

Classes

- class **domain_handler_t**

  Handles domain-specific partial input and output signal.
- class **dummy_threads_t**

  Dummy specification of a thread platform: This class implements everything in a single thread.
- class **posix_threads_t**

  Posix threads specification of thread platform.
- class **split_t**

  Implements split plugin.
- class **splitted_part_t**

  *The **splitted_part_t** (p. 751) instance manages the plugin that performs processing on the reduced set of channels.*
- class **thread_platform_t**

  Basic interface for encapsulating thread creation, thread priority setting, and synchronization on any threading platform (i.e., pthreads or win32threads).
- class **uni_processor_t**

  An interface to a class that sports a process method with no parameters and no return value.

Enumerations

4.41.1 Detailed Description

A namespace for the split plugin. Helps testability and documentation.

4.41.2 Enumeration Type Documentation

4.41.2.1 anonymous enum

Invalid thread priority.

Enumerator

INVALID_THREAD_PRIORITY

4.42 MHASignal Namespace Reference

Namespace for audio signal handling and processing classes.

Classes

- class **async_rmslevel_t**
  Class for asynchronous level metering.
- class **delay_spec_t**
- class **delay_t**
  Class to realize a simple delay of waveform streams.
- class **delay_wave_t**
  Delayline containing wave fragments.
- class **doublebuffer_t**
  Double-buffering class.
- class **fft_t**
- class **hilbert_fftw_t**
- class **hilbert_t**
  Hilbert transformation of a waveform segment.
- class **loop_wavefragment_t**
  Copy a fixed waveform fragment to a series of waveform fragments of other size.
- class **matrix_t**
  n-dimensional matrix with real or complex floating point values.
- class **minphase_t**
  Minimal phase function.
- class **quantizer_t**
  Simple simulation of fixpoint quantization.
- class **ringbuffer_t**
  A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.
- class **schroeder_t**
  Schroeder tone complex class.
- class **spectrum_t**
  *a signal processing class for spectral data (based on **mha_spec_t** (p. 465))*

- class **stat_t**
- class **subsample_delay_t**

  implements subsample delay in spectral domain.
- class **uint_vector_t**

  Vector of unsigned values, used for size and index description of n-dimensional matrixes.
- class **waveform_t**

  *signal processing class for waveform data (based on **mha_wave_t** (p. 496))*

Functions

- void **for_each** (**mha_wave_t** *s, **mha_real_t**(*fun)(**mha_real_t**))

  *Apply a function to each element of a **mha_wave_t** (p. 496).*
- **mha_real_t lin2db** (**mha_real_t** x)

  Conversion from linear scale to dB (no SPL reference)
- **mha_real_t db2lin** (**mha_real_t** x)

  Conversion from dB scale to linear (no SPL reference)
- **mha_real_t pa2dbspl** (**mha_real_t** x)

  Conversion from linear Pascal scale to dB SPL.
- **mha_real_t pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=1e-20f)

  Conversion from squared Pascal scale to dB SPL.
- **mha_real_t dbspl2pa** (**mha_real_t** x)

  Conversion from dB SPL to linear Pascal scale.
- **mha_real_t smp2sec** (**mha_real_t** n, **mha_real_t** srate)

  conversion from samples to seconds
- **mha_real_t sec2smp** (**mha_real_t** sec, **mha_real_t** srate)

  conversion from seconds to samples
- **mha_real_t bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)

  conversion from fft bin index to frequency
- **mha_real_t freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)

  conversion from frequency to fft bin index
- **mha_real_t smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)

  conversion from delay in samples to phase shift
- **mha_real_t rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)

  conversion from phase shift to delay in samples
- template<class elem_type >
  std::vector< elem_type > **dupvec** (std::vector< elem_type > vec, unsigned n)

  Duplicate last vector element to match desired size.
- template<class elem_type >
  std::vector< elem_type > **dupvec_chk** (std::vector< elem_type > vec, unsigned n)

  Duplicate last vector element to match desired size, check for dimension.
- void **copy_channel** (**mha_spec_t** &self, const **mha_spec_t** &src, unsigned sch, unsigned dch)

  Copy one channel of a source signal.
- void **copy_channel** (**mha_wave_t** &self, const **mha_wave_t** &src, unsigned src_↩ channel, unsigned dest_channel)

Copy one channel of a source signal.

- **mha_real_t rmslevel** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen)
  Return RMS level of a spectrum channel.
- **mha_real_t colored_intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** sqfreq_response[ ])
  Colored spectrum intensity.
- **mha_real_t maxabs** (const **mha_spec_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t rmslevel** (const **mha_wave_t** &s, unsigned int channel)
  Return RMS level of a waveform channel.
- **mha_real_t maxabs** (const **mha_wave_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t maxabs** (const **mha_wave_t** &s)
  Find maximal absolute value.
- **mha_real_t max** (const **mha_wave_t** &s)
  Find maximal value.
- **mha_real_t min** (const **mha_wave_t** &s)
  Find minimal value.
- **mha_real_t sumsqr_channel** (const **mha_wave_t** &s, unsigned int channel)
  Calculate sum of squared values in one channel.
- **mha_real_t sumsqr_frame** (const **mha_wave_t** &s, unsigned int frame)
  Calculate sum over all channels of squared values.
- void **scale** (**mha_spec_t** ∗dest, const **mha_wave_t** ∗src)
- void **limit** (**mha_wave_t** &s, const **mha_real_t** &**min**, const **mha_real_t** &**max**)
  Limit the singal in the waveform buffer to the range [min, max].
- template<class elem_type >
  elem_type **kth_smallest** (elem_type array[ ], unsigned n, unsigned k)
  Fast search for the kth smallest element of an array.
- template<class elem_type >
  elem_type **median** (elem_type array[ ], unsigned n)
  Fast median search.
- template<class elem_type >
  elem_type **mean** (const std::vector< elem_type > &data, elem_type start_val)
  Calculate average of elements in a vector.
- template<class elem_type >
  std::vector< elem_type > **quantile** (std::vector< elem_type > data, const std::vector< elem_type > &p)
  Calculate quantile of elements in a vector.
- void **saveas_mat4** (const **mha_spec_t** &data, const std::string &varname, FILE ∗fh)
  Save a openMHA spectrum as a variable in a Matlab4 file.
- void **saveas_mat4** (const **mha_wave_t** &data, const std::string &varname, FILE ∗fh)
  Save a openMHA waveform as a variable in a Matlab4 file.
- void **saveas_mat4** (const std::vector< **mha_real_t** > &data, const std::string &varname, FILE ∗fh)
  Save a float vector as a variable in a Matlab4 file.
- void **copy_permuted** (**mha_wave_t** ∗dest, const **mha_wave_t** ∗src)
  Copy contents of a waveform to a permuted waveform.

Variables

- unsigned long int **signal_counter** = 0
  Signal counter to produce signal ID strings.

4.42.1 Detailed Description

Namespace for audio signal handling and processing classes.

4.42.2 Function Documentation

4.42.2.1 void MHASignal::scale (mha_spec_t * *dest,* const mha_wave_t * *src*)

4.42.2.2 void MHASignal::limit (mha_wave_t & *s,* const mha_real_t & *min,* const mha_real_t & *max*)

Limit the singal in the waveform buffer to the range [min, max].

Parameters

s	The signal to limit. The signal in this wave buffer is modified.
min	lower limit
max	upper limit

4.42.2.3 template<class elem_type > elem_type MHASignal::kth_smallest (elem_type *array[ ],* unsigned *n,* unsigned *k*)

Fast search for the kth smallest element of an array.

The order of elements is altered, but not completely sorted. Using the algorithm from N. Wirth, published in "Algorithms + data structures = programs", Prentice-Hall, 1976

Parameters

array	Element array

Postcondition

The order of elements in the array is altered. array[k] then holds the result.

Parameters

n	number of elements in array

Precondition

n >= 1

Parameters

k	The k'th smalles element is returned: k = 0 returns the minimum, k = (n-1)/2 returns the median, k=(n-1) returns the maximum

Precondition

k < n

Returns

The kth smallest array element

4.42.2.4 template<class elem_type > elem_type MHASignal::median (elem_type *array[ ],* unsigned *n*) [inline]

Fast median search.

The order of elements is altered, but not completely sorted.

Parameters

array	Element array

Postcondition

The order of elements in the array is altered. array[(n-1)/2] then holds the median.

Parameters

n	number of elements in array

Precondition

n >= 1

Returns

The median of the array elements

4.42.2.5 template<class elem_type > elem_type MHASignal::mean (const std::vector< elem_type > & *data,* elem_type *start_val*) [inline]

Calculate average of elements in a vector.

Parameters

data	Input vector
start_val	Value for initialization of the return value before sum.

Returns

The average of the vector elements

4.42.2.6 template<class elem_type > std::vector<elem_type> MHASignal::quantile (std::vector< elem_type > *data,* const std::vector< elem_type > & *p*) [inline]

Calculate quantile of elements in a vector.

Parameters

data	Input vector
p	Vector of probability values.

Returns

Vector of quantiles of input data, one entry for each probability value.

4.42.2.7 void MHASignal::saveas_mat4 (const mha_spec_t & *data,* const std::string & *varname,* FILE * *fh*)

Save a openMHA spectrum as a variable in a Matlab4 file.

Parameters

data	openMHA spectrum to be saved.
varname	Matlab variable name (Matlab4 limitations on maximal length are not checked).
fh	File handle to Matlab4 file.

4.42.2.8 void MHASignal::saveas_mat4 (const mha_wave_t & *data*, const std::string & *varname*, FILE * *fh*)

Save a openMHA waveform as a variable in a Matlab4 file.

Parameters

data	openMHA waveform to be saved.
varname	Matlab variable name (Matlab4 limitations on maximal length are not checked).
fh	File handle to Matlab4 file.

4.42.2.9 void MHASignal::saveas_mat4 (const std::vector< mha_real_t > & *data*, const std::string & *varname*, FILE * *fh*)

Save a float vector as a variable in a Matlab4 file.

Parameters

data	Float vector to be saved.
varname	Matlab variable name (Matlab4 limitations on maximal length are not checked).
fh	File handle to Matlab4 file.

4.42.2.10 void MHASignal::copy_permuted (mha_wave_t * *dest*, const mha_wave_t * *src*)

Copy contents of a waveform to a permuted waveform.

Parameters

dest	Destination waveform
src	Source waveform

The total size of src and dest must be the same, num_frames and num_channels must be

exchanged in dest.

4.42.3 Variable Documentation

4.42.3.1 unsigned long int MHASignal::signal_counter = 0

Signal counter to produce signal ID strings.

4.43 MHASndFile Namespace Reference

Classes

- class **sf_t**
- class **sf_wave_t**

4.44 MHATableLookup Namespace Reference

Namespace for table lookup classes.

Classes

- class **linear_table_t**
  Class for interpolation with equidistant x values.
- class **table_t**
- class **xy_table_t**
  Class for interpolation with non-equidistant x values.

4.44.1 Detailed Description

Namespace for table lookup classes.

4.45 MHAWindow Namespace Reference

Collection of Window types.

Classes

- class **bartlett_t**
  Bartlett window.
- class **base_t**
  Common base for window types.
- class **blackman_t**
  Blackman window.
- class **fun_t**
  Generic window based on a generator function.
- class **hamming_t**
  Hamming window.
- class **hanning_t**
  von-Hann window
- class **rect_t**
  Rectangular window.
- class **user_t**
  User defined window.

Functions

- float **rect** (float)
  Rectangular window function.
- float **bartlett** (float)
  Bartlett window function.
- float **hanning** (float)
  Hanning window function.
- float **hamming** (float)
  Hamming window function.
- float **blackman** (float)
  Blackman window function.

4.45.1 Detailed Description

Collection of Window types.

4.45.2 Function Documentation

4.45.2.1 float MHAWindow::rect (float *x*)

Rectangular window function.

4.45.2.2 float MHAWindow::bartlett (float *x*)

Bartlett window function.

4.45.2.3 float MHAWindow::hanning (float *x*)

Hanning window function.

4.45.2.4 float MHAWindow::hamming (float *x*)

Hamming window function.

4.45.2.5 float MHAWindow::blackman (float *x*)

Blackman window function.

4.46 multibandcompressor Namespace Reference

Classes

- class **fftfb_plug_t**
- class **interface_t**
- class **plugin_signals_t**

4.47 noisePowProposedScale Namespace Reference

Classes

- class **interface_t**
- class **noisePowProposed**

4.48 overlapadd Namespace Reference

Classes

- class **overlapadd_if_t**
- class **overlapadd_t**

4.49 plingploing Namespace Reference

All classes for the plingploing music generator live in this namespace.

Classes

- class **if_t**
  Plugin class of the plingploing music generator.
- class **plingploing_t**
  Run-time configuration of the plingploing music generator.

Functions

- double **drand** (double a, double b)

4.49.1 Detailed Description

All classes for the plingploing music generator live in this namespace.

4.49.2 Function Documentation

4.49.2.1 double plingploing::drand (double *a,* double *b*)

4.50 PluginLoader Namespace Reference

Classes

- class **config_file_splitter_t**
- class **fourway_processor_t**
  This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.
- class **mhapluginloader_t**

Functions

- const char * **mhastrdomain** (**mha_domain_t**)
- void **mhaconfig_compare** (const **mhaconfig_t** &req, const **mhaconfig_t** &avail, const std::string &pref="")
  *Compare two **mhaconfig_t** (p. 504) structures, and report differences as an error.*

4.50.1 Function Documentation

4.50.1.1 const char * PluginLoader::mhastrdomain (mha_domain_t *d*)

4.50.1.2 void PluginLoader::mhaconfig_compare (const mhaconfig_t & *req*, const mhaconfig_t & *avail*, const std::string & *pref* = " ")

Compare two **mhaconfig_t** (p. 504) structures, and report differences as an error.

Parameters

req	Expected **mhaconfig_t** (p. 504) structure
avail	Available **mhaconfig_t** (p. 504) structure
pref	Prefix for error messages

4.51 route Namespace Reference

Classes

- class **interface_t**
- class **process_t**

4.52 shadowfilter_begin Namespace Reference

Classes

- class **cfg_t**
- class **shadowfilter_begin_t**

4.53 shadowfilter_end Namespace Reference

Classes

- class **cfg_t**
- class **shadowfilter_end_t**

4.54 smoothgains_bridge Namespace Reference

Classes

- class **overlapadd_if_t**
- class **smoothspec_wrap_t**

4.55 testplugin Namespace Reference

Classes

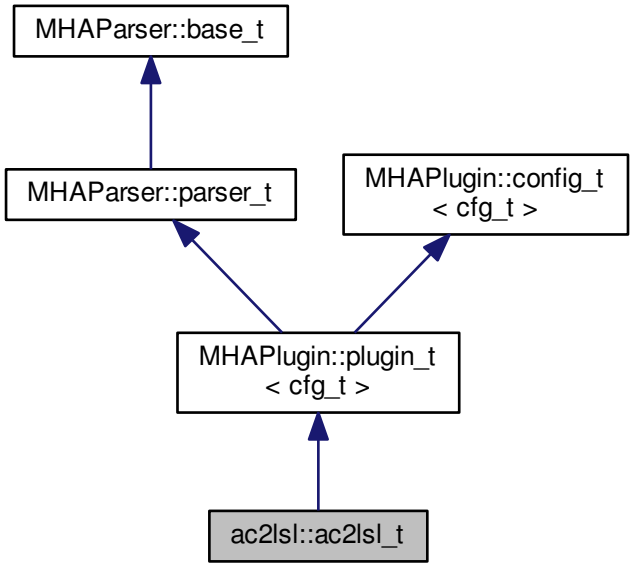
- class **ac_parser_t**
- class **config_parser_t**
- class **if_t**
- class **signal_parser_t**

5 Class Documentation

5.1 ac2lsl::ac2lsl_t Class Reference

Plugin class of **ac2lsl** (p. 78).

Inheritance diagram for ac2lsl::ac2lsl_t:



Public Member Functions

- **ac2lsl_t** (**algo_comm_t** iac, const char *chain, const char *algo)
- void **prepare** (**mhaconfig_t** &)

  *Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 133).*
- **mha_wave_t** * **process** (**mha_wave_t** *s)

  Processing fct for waveforms.
- **mha_spec_t** * **process** (**mha_spec_t** *s)

  Processing fct for spectra.
- void **process** ()

  Process function.
- void **release** ()

  Release fct.

Private Member Functions

- std::vector< std::string > **get_all_names_from_ac_space** (const **algo_comm_t** &**ac**) const

  Retrieves all variable names from the AC space.
- void **update** ()

  Construct new runtime configuration.

Private Attributes

- **MHAParser::vstring_t vars**
- **MHAParser::string_t source_id**
- **MHAParser::bool_t rt_strict**
- **MHAParser::bool_t activate**
- **MHAParser::int_t skip**
- **MHAEvents::patchbay_t< ac2lsl_t > patchbay**
- bool **is_first_run**

Additional Inherited Members

5.1.1 Detailed Description

Plugin class of **ac2lsl** (p. 78).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ac2lsl::ac2lsl_t::ac2lsl_t (
algo_comm_t *iac,*
const char ∗ *chain,*
const char ∗ *algo*)

5.1.3 Member Function Documentation

5.1.3.1 void ac2lsl::ac2lsl_t::prepare (
mhaconfig_t & *cf*) [virtual]

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 133).

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 730).

5.1.3.2 mha_wave_t∗ ac2lsl::ac2lsl_t::process (
mha_wave_t ∗ *s*) [inline]

Processing fct for waveforms.

Calls **process(void)** (p. 132).

5.1.3.3 mha_spec_t∗ ac2lsl::ac2lsl_t::process (
mha_spec_t ∗ *s*) [inline]

Processing fct for spectra.

Calls **process(void)** (p. 132).

5.1.3.4 void ac2lsl::ac2lsl_t::process (
void)

Process function.

Checks once if the plugin is run in a real-time thread and throws if rt_strict is true, then forwards to **cfg_t::process()** (p. 135).

5.1.3.5 void ac2lsl::ac2lsl_t::release ( ) [virtual]

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugin::plugin_t**< **cfg_t** > (p. 731).

5.1.3.6 std::vector< std::string > ac2lsl::ac2lsl_t::get_all_names_from_ac_space (
const algo_comm_t & *ac*) **const** [private]

Retrieves all variable names from the AC space.

Parameters

ac	AC space

Returns

Vector of variable names

5.1.3.7 void ac2lsl::ac2lsl_t::update ( ) [private]

Construct new runtime configuration.

5.1.4 Member Data Documentation

5.1.4.1 MHAParser::vstring_t ac2lsl::ac2lsl_t::vars [private]

5.1.4.2 MHAParser::string_t ac2lsl::ac2lsl_t::source_id [private]

5.1.4.3 MHAParser::bool_t ac2lsl::ac2lsl_t::rt_strict [private]

5.1.4.4 MHAParser::bool_t ac2lsl::ac2lsl_t::activate [private]

5.1.4.5 MHAParser::int_t ac2lsl::ac2lsl_t::skip [private]

5.1.4.6 MHAEvents::patchbay_t<ac2lsl_t> ac2lsl::ac2lsl_t::patchbay [private]

5.1.4.7 bool ac2lsl::ac2lsl_t::is_first_run [private]

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

- **ac2lsl.cpp**

5.2 ac2lsl::cfg_t Class Reference

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

Public Member Functions

- **cfg_t** (const **algo_comm_t** &ac_, unsigned skip_, const std::string &**source_id**, const std::vector< std::string > &varnames_, double rate)
  *C'tor of **ac2lsl** (p. 78) run time configuration.*
- void **process** ()

Private Member Functions

- void **create_or_replace_var** (const std::string &name, const **comm_var_t** &v)
- void **check_vars** ()
- void **update_varlist** ()

Private Attributes

- std::map< std::string, std::unique_ptr< **save_var_base_t** > > **varlist**
  Maps variable name to unique ptr's of ac to lsl bridges.
- unsigned **skipcnt**
  Counter of frames to skip.
- const unsigned **skip**
  Number of frames to skip after each send.
- const double **srate**
  Sampling rate of the stream.
- const std::string **source_id**
  User configurable source id.
- const **algo_comm_t** & **ac**
  Handle to the ac space.

5.2.1 Detailed Description

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 cfg_t::cfg_t (

const algo_comm_t & *ac_*,
unsigned *skip_*,
const std::string & *source_id*,
const std::vector< std::string > & *varnames_*,
double *rate*)

C'tor of **ac2lsl** (p. 78) run time configuration.

Parameters

*ac_*	AC space, source of data to send over LSL
*skip_*	Number of frames to skip after each send
*source_id_*	LSL identifier for this data stream
*varnames_*	Names of AC variables to send over LSL
rate	Rate with wich chunks of data are sent to the LSL stream. Usually the rate with which process calls happen, but may be lower due to the subsampling caused by skip_

5.2.3 Member Function Documentation

5.2.3.1 void cfg_t::create_or_replace_var (const std::string & *name,* const **comm_var_t** & *v*) [private]

5.2.3.2 void ac2lsl::cfg_t::check_vars ( ) [private]

5.2.3.3 void cfg_t::update_varlist ( ) [private]

5.2.3.4 void cfg_t::process (void)

5.2.4 Member Data Documentation

5.2.4.1 std::map<std::string, std::unique_ptr<**save_var_base_t**> > ac2lsl::cfg_t::varlist [private]

Maps variable name to unique ptr's of ac to lsl bridges.

5.2.4.2 unsigned ac2lsl::cfg_t::skipcnt [private]

Counter of frames to skip.

5.2.4.3 const unsigned ac2lsl::cfg_t::skip [private]

Number of frames to skip after each send.

5.2.4.4 const double ac2lsl::cfg_t::srate [private]

Sampling rate of the stream.

5.2.4.5 const std::string ac2lsl::cfg_t::source_id [private]

User configurable source id.

5.2.4.6 const **algo_comm_t**& ac2lsl::cfg_t::ac [private]

Handle to the ac space.

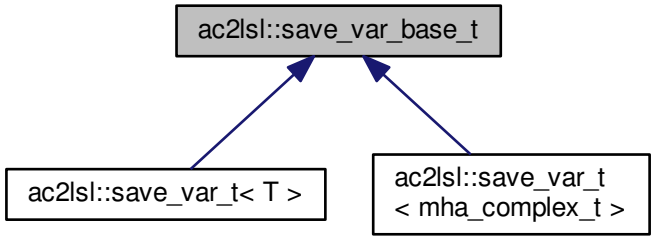
The documentation for this class was generated from the following file:

- **ac2lsl.cpp**

5.3 ac2lsl::save_var_base_t Class Reference

Interface for ac to lsl bridge variable.

Inheritance diagram for ac2lsl::save_var_base_t:



Public Member Functions

- virtual void **send_frame** ()=0
- virtual void * **get_buf_address** () const noexcept=0
- virtual void **set_buf_address** (void *data)=0
- virtual lsl::stream_info **info** () const noexcept=0
- virtual unsigned **data_type** () const noexcept=0
- virtual unsigned **num_entries** () const noexcept=0
- virtual ~**save_var_base_t** ()=default

5.3.1 Detailed Description

Interface for ac to lsl bridge variable.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 virtual ac2lsl::save_var_base_t::~save_var_base_t ( ) [virtual], [default]

5.3.3 Member Function Documentation

5.3.3.1 virtual void ac2lsl::save_var_base_t::send_frame ( ) [pure virtual]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 143), and **ac2lsl::save_var_t< T >** (p. 140).

5.3.3.2 virtual void* ac2lsl::save_var_base_t::get_buf_address ( ) const [pure virtual], [noexcept]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 142), and **ac2lsl::save_var_t< T >** (p. 139).

5.3.3.3 virtual void ac2lsl::save_var_base_t::set_buf_address (void * *data*) [pure virtual]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 142), and **ac2lsl::save_var_t< T >** (p. 139).

5.3.3.4 virtual lsl::stream_info ac2lsl::save_var_base_t::info ( ) const [pure virtual], [noexcept]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 142), and **ac2lsl::save_var_t< T >** (p. 139).

5.3.3.5 virtual unsigned ac2lsl::save_var_base_t::data_type ( ) const [pure virtual], [noexcept]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 142), and **ac2lsl::save_var_t< T >** (p. 140).

5.3.3.6 virtual unsigned ac2lsl::save_var_base_t::num_entries ( ) const [pure virtual], [noexcept]

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 142), and **ac2lsl::save_var_t< T >** (p. 139).

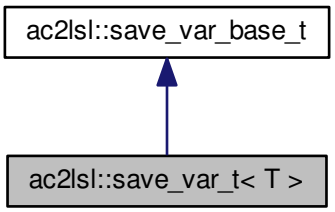
The documentation for this class was generated from the following file:

- **ac2lsl.cpp**

5.4 ac2lsl::save_var_t< T > Class Template Reference

Implementation for all ac to lsl bridges except complex types.

Inheritance diagram for ac2lsl::save_var_t< T >:



Public Member Functions

- **save_var_t** (const std::string &name_, const std::string &type_, unsigned num_entries_, const **mha_real_t** rate_, const lsl::channel_format_t format_, const std::string &source↩ _id_, void ∗data_, const unsigned **data_type_**)
  C'tor of generic ac to lsl bridge.
- virtual void ∗ **get_buf_address** () const noexceptoverride
  Get buffer address as void pointer.
- virtual void **set_buf_address** (void ∗data) override
  Cast the input pointer to the appropriate type and set the buffer address.
- virtual lsl::stream_info **info** () const noexceptoverride
  Get stream info object from stream outlet.
- virtual unsigned **num_entries** () const noexceptoverride
  Get number of entries in the stream object.
- virtual unsigned **data_type** () const noexceptoverride
  Get data type id according MHA convention.
- virtual ∼**save_var_t** ()=default
- virtual void **send_frame** () override
  Send a frame to lsl.

Private Attributes

- lsl::stream_outlet **stream**
  LSL stream outlet.
- T ∗ **buf**
  Pointer to data buffer of the ac variable.
- const unsigned **data_type_**
  Data type id according to MHA convention.

5.4.1 Detailed Description

template<typename T>
class ac2lsl::save_var_t< T >

Implementation for all ac to lsl bridges except complex types.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 template<typename T > ac2lsl::save_var_t< T >::save_var_t (

const std::string & *name_*,
const std::string & *type_*,
unsigned *num_entries_*,
const mha_real_t *rate_*,
const lsl::channel_format_t *format_*,
const std::string & *source_id_*,
void ∗ *data_*,
const unsigned *data_type_*) `[inline]`

C'tor of generic ac to lsl bridge.

Parameters

info	LSL stream info object containing metadata
data	Pointer to data buffer of the ac variable
data_type	Type id of the stream, in mha convention. Should be set to one if not a vector.

5.4.2.2 template<typename T > virtual ac2lsl::save_var_t< T >::~save_var_t ( ) [virtual], [default]

5.4.3 Member Function Documentation

5.4.3.1 template<typename T > virtual void* ac2lsl::save_var_t< T >::get_buf_address ( ) const [inline], [override], [virtual], [noexcept]

Get buffer address as void pointer.

Returns

Adress of the data buffer

Implements **ac2lsl::save_var_base_t** (p. 137).

5.4.3.2 template<typename T > virtual void ac2lsl::save_var_t< T >::set_buf_address (void * *data*) [inline], [override], [virtual]

Cast the input pointer to the appropriate type and set the buffer address.

Parameters

data	New buffer address

Implements **ac2lsl::save_var_base_t** (p. 137).

5.4.3.3 template<typename T > virtual lsl::stream_info ac2lsl::save_var_t< T >::info ( ) const [inline], [override], [virtual], [noexcept]

Get stream info object from stream outlet.

Implements **ac2lsl::save_var_base_t** (p. 137).

5.4.3.4 template<typename T > virtual unsigned ac2lsl::save_var_t< T >::num_entries ( ) const [inline], [override], [virtual], [noexcept]

Get number of entries in the stream object.

Implements **ac2lsl::save_var_base_t** (p. 137).

5.4.3.5 template<typename T > virtual unsigned ac2lsl::save_var_t< T >::data_type ( ) const [inline], [override], [virtual], [noexcept]

Get data type id according MHA convention.

Implements **ac2lsl::save_var_base_t** (p. 137).

5.4.3.6 template<typename T > virtual void ac2lsl::save_var_t< T >::send_frame ( ) [inline], [override], [virtual]

Send a frame to lsl.

Implements **ac2lsl::save_var_base_t** (p. 136).

5.4.4 Member Data Documentation

5.4.4.1 template<typename T > lsl::stream_outlet ac2lsl::save_var_t< T >::stream [private]

LSL stream outlet.

Interface to lsl

5.4.4.2 template<typename T > T* ac2lsl::save_var_t< T >::buf [private]

Pointer to data buffer of the ac variable.

5.4.4.3 template<typename T > const unsigned ac2lsl::save_var_t< T >::data_type_ [private]

Data type id according to MHA convention.

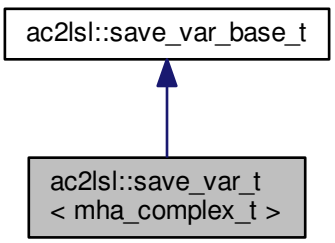
The documentation for this class was generated from the following file:

- **ac2lsl.cpp**

5.5 ac2lsl::save_var_t< mha_complex_t > Class Template Reference

Template specialization of the **ac2lsl** (p. 78) bridge to take care of complex numbers.

Inheritance diagram for ac2lsl::save_var_t< mha_complex_t >:



Public Member Functions

- **save_var_t** (const std::string &name_, const std::string &type_, const unsigned num_↩ entries_, const **mha_real_t** rate_, const lsl::channel_format_t format_, const std::string &source_id_, void *data_)
  C'tor of specialization for complex types.
- virtual void * **get_buf_address** () const noexceptoverride
- virtual void **set_buf_address** (void *data) override
- virtual lsl::stream_info **info** () const noexceptoverride
  Get buffer address as void pointer.
- virtual unsigned **num_entries** () const noexceptoverride
  Get number of entries in the stream object.
- virtual unsigned **data_type** () const noexceptoverride
  Cast the input pointer to the appropriate type and set the buffer address.
- virtual ~**save_var_t** ()=default
- virtual void **send_frame** () override
  Send a frame of complex types.

Private Attributes

- lsl::stream_outlet **stream**
  LSL stream outlet.
- **mha_complex_t** * **buf**
  Pointer to data buffer of the ac variable.

5.5.1 Detailed Description

template<>
class ac2lsl::save_var_t< mha_complex_t >

Template specialization of the **ac2lsl** (p. 78) bridge to take care of complex numbers.

This specialization is needed because lsl does not support complex numbers. Order is [re(0), im(0), re(1), im(1), ...]

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ac2lsl::save_var_t< mha_complex_t >::save_var_t (

const std::string & *name_*,
const std::string & *type_*,
const unsigned *num_entries_*,
const mha_real_t *rate_*,
const lsl::channel_format_t *format_*,
const std::string & *source_id_*,
void * *data_*) `[inline]`

C'tor of specialization for complex types.

See generic c'tor for details.

5.5.2.2 virtual ac2lsl::save_var_t< mha_complex_t >::~save_var_t ( ) [virtual], [default]

5.5.3 Member Function Documentation

5.5.3.1 virtual void* ac2lsl::save_var_t< mha_complex_t >::get_buf_address ( ) const [inline], [override], [virtual], [noexcept]

Implements **ac2lsl::save_var_base_t** (p. 137).

5.5.3.2 virtual void ac2lsl::save_var_t< mha_complex_t >::set_buf_address (void * *data*) [inline], [override], [virtual]

Implements **ac2lsl::save_var_base_t** (p. 137).

5.5.3.3 virtual lsl::stream_info ac2lsl::save_var_t< mha_complex_t >::info ( ) const [inline], [override], [virtual], [noexcept]

Get buffer address as void pointer.

Returns

Adress of the data buffer

Implements **ac2lsl::save_var_base_t** (p. 137).

5.5.3.4 virtual unsigned ac2lsl::save_var_t< mha_complex_t >::num_entries ( ) const [inline], [override], [virtual], [noexcept]

Get number of entries in the stream object.

Implements **ac2lsl::save_var_base_t** (p. 137).

5.5.3.5 virtual unsigned ac2lsl::save_var_t< mha_complex_t >::data_type ( ) const [inline], [override], [virtual], [noexcept]

Cast the input pointer to the appropriate type and set the buffer address.

Parameters

data	New buffer address

Implements **ac2lsl::save_var_base_t** (p. 137).

5.5.3.6 virtual void ac2lsl::save_var_t< mha_complex_t >::send_frame ( ) [inline], [override], [virtual]

Send a frame of complex types.

Complex numbers are stored as alternating real and imaginary parts. An array of complex numbers in memory can be reinterpreted as a vector of real numbers that correspond to real and imaginary parts. LSL does not support complex types directly. Send one vector containing {buf[0].re,buf[0].im,buf[1].re,buf[1].im,...} instead.

Implements **ac2lsl::save_var_base_t** (p. 136).

5.5.4 Member Data Documentation

5.5.4.1 lsl::stream_outlet ac2lsl::save_var_t< mha_complex_t >::stream [private]

LSL stream outlet.

Interface to lsl

5.5.4.2 mha_complex_t* ac2lsl::save_var_t< mha_complex_t >::buf [private]

Pointer to data buffer of the ac variable.

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

- **ac2lsl.cpp**

5.6 ac2lsl::type_info Struct Reference

Public Attributes

- const std::string **name**
- const lsl::channel_format_t **format**

5.6.1 Member Data Documentation

5.6.1.1 const std::string ac2lsl::type_info::name

5.6.1.2 const lsl::channel_format_t ac2lsl::type_info::format

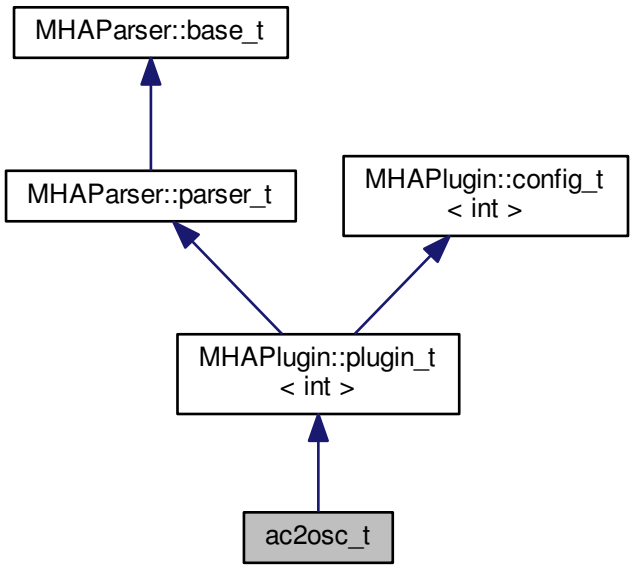
The documentation for this struct was generated from the following file:

- **ac2lsl.cpp**

5.7 ac2osc_t Class Reference

Plugin class of the ac2osc plugin.

Inheritance diagram for ac2osc_t:



Public Member Functions

- **ac2osc_t** (**algo_comm_t** iac, const char *chain, const char *algo)
  C'tor of plugin class.
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_wave_t** *s)
  Processing fct for waveforms.
- **mha_spec_t** * **process** (**mha_spec_t** *s)
  Processing fct for spectra.
- void **process** ()
  Process function.
- void **release** ()
  Release frees osc related memory, does cleanup.

Private Member Functions

- void **send_osc_float** ()
- void **update_mode** ()
  Start/Stop sending of messages.

Private Attributes

- **MHAParser::string_t host**
  OSC server host name.
- **MHAParser::string_t port**
  OSC server port.
- **MHAParser::int_t ttl**
  Time-to-live of UDP packages.
- **MHAParser::vstring_t vars**
  List of AC variables to be saved, empty for all.
- **MHAParser::kw_t mode**
  Record mode.
- **MHAParser::int_t skip**
  number of frames to skip after sending
- **MHAParser::bool_t rt_strict**
  abort if used in real-time thread?
- **MHA_AC::acspace2matrix_t** $*$ **acspace**
- **MHAEvents::patchbay_t**$<$ **ac2osc_t** $>$ **patchbay**
- bool **b_record**
- uint8_t $*$ **rtmem**
- float **framerate**
- int **skipcnt**
- lo_address **lo_addr**
- bool **is_first_run**

Additional Inherited Members

5.7.1 Detailed Description

Plugin class of the ac2osc plugin.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ac2osc_t::ac2osc_t (algo_comm_t *iac,* const char $*$ *chain,* const char $*$ *algo*)

C'tor of plugin class.

5.7.3 Member Function Documentation

5.7.3.1 void ac2osc_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.7.3.2 mha_wave_t* ac2osc_t::process (mha_wave_t * *s*) [inline]

Processing fct for waveforms.

Calls **process(void)** (p. 146).

5.7.3.3 mha_spec_t* ac2osc_t::process (mha_spec_t * *s*) [inline]

Processing fct for spectra.

Calls **process(void)** (p. 146).

5.7.3.4 void ac2osc_t::process (void)

Process function.

Checks once if the plugin is run in a real-time thread and throws if rt_strict is true, sends osc messages according to config.

5.7.3.5 void ac2osc_t::release (void) [virtual]

Release frees osc related memory, does cleanup.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.7.3.6 void ac2osc_t::send_osc_float ( ) [private]

5.7.3.7 void ac2osc_t::update_mode ( ) [private]

Start/Stop sending of messages.

5.7.4 Member Data Documentation

5.7.4.1 MHAParser::string_t ac2osc_t::host [private]

OSC server host name.

5.7.4.2 MHAParser::string_t ac2osc_t::port [private]

OSC server port.

5.7.4.3 MHAParser::int_t ac2osc_t::ttl [private]

Time-to-live of UDP packages.

5.7.4.4 MHAParser::vstring_t ac2osc_t::vars [private]

List of AC variables to be saved, empty for all.

5.7.4.5 MHAParser::kw_t ac2osc_t::mode [private]

Record mode.

5.7.4.6 MHAParser::int_t ac2osc_t::skip [private]

number of frames to skip after sending

5.7.4.7 MHAParser::bool_t ac2osc_t::rt_strict [private]

abort if used in real-time thread?

5.7.4.8 MHA_AC::acspace2matrix_t* ac2osc_t::acspace [private]

5.7.4.9 MHAEvents::patchbay_t<ac2osc_t> ac2osc_t::patchbay [private]

5.7.4.10 bool ac2osc_t::b_record [private]

5.7.4.11 uint8_t* ac2osc_t::rtmem [private]

5.7.4.12 float ac2osc_t::framerate [private]

5.7.4.13 int ac2osc_t::skipcnt [private]

5.7.4.14 lo_address ac2osc_t::lo_addr [private]

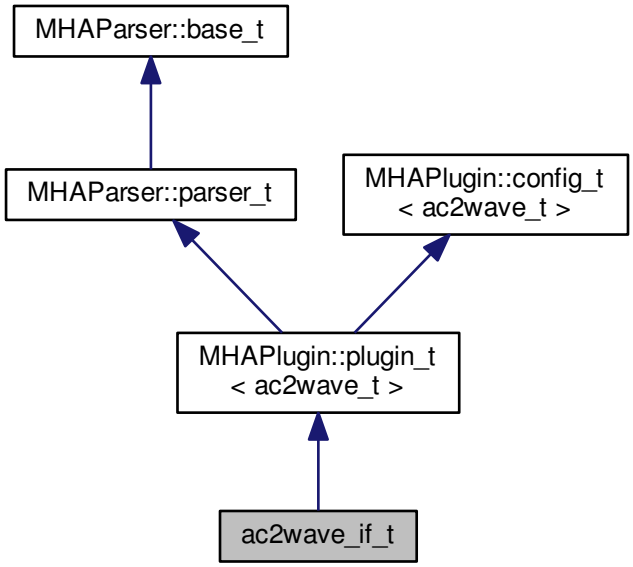
5.7.4.15 bool ac2osc_t::is_first_run [private]

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

- **ac2osc.cpp**

5.8 ac2wave_if_t Class Reference

Inheritance diagram for ac2wave_if_t:



Public Member Functions

- **ac2wave_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::string_t name**
- **MHAParser::float_t gain_in**
- **MHAParser::float_t gain_ac**
- **MHAParser::int_t delay_in**
- **MHAParser::int_t delay_ac**
- **MHASignal::waveform_t** ∗ **zeros**
- bool **prepared**
- **MHAEvents::patchbay_t**< **ac2wave_if_t** > **patchbay**

Additional Inherited Members

5.8.1 Constructor & Destructor Documentation

5.8.1.1 ac2wave_if_t::ac2wave_if_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.8.2 Member Function Documentation

5.8.2.1 mha_wave_t * ac2wave_if_t::process (mha_spec_t *)

5.8.2.2 mha_wave_t * ac2wave_if_t::process (mha_wave_t * *s*)

5.8.2.3 void ac2wave_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< ac2wave_t >** (p. 730).

5.8.2.4 void ac2wave_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< ac2wave_t >** (p. 731).

5.8.2.5 void ac2wave_if_t::update ( ) [private]

5.8.3 Member Data Documentation

5.8.3.1 MHAParser::string_t ac2wave_if_t::name [private]

5.8.3.2 MHAParser::float_t ac2wave_if_t::gain_in [private]

5.8.3.3 MHAParser::float_t ac2wave_if_t::gain_ac [private]

5.8.3.4 MHAParser::int_t ac2wave_if_t::delay_in [private]

5.8.3.5 MHAParser::int_t ac2wave_if_t::delay_ac [private]

5.8.3.6 MHASignal::waveform_t* ac2wave_if_t::zeros [private]

5.8.3.7 bool ac2wave_if_t::prepared [private]

5.8.3.8 MHAEvents::patchbay_t<ac2wave_if_t> ac2wave_if_t::patchbay [private]

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

- **ac2wave.cpp**

5.9 ac2wave_t Class Reference

Public Member Functions

- **ac2wave_t** (unsigned int frames_, unsigned int channels_, **algo_comm_t** ac_, std::string name_, float gain_in_, float gain_ac_, unsigned int delay_in_, unsigned int delay_ac_)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- unsigned int **frames**
- unsigned int **channels**
- **mha_wave_t w**
- **algo_comm_t ac**
- std::string **name**
- **MHASignal::delay_wave_t delay_in**
- **MHASignal::delay_wave_t delay_ac**
- **mha_real_t gain_in**
- **mha_real_t gain_ac**

5.9.1 Constructor & Destructor Documentation

5.9.1.1 ac2wave_t::ac2wave_t (
unsigned int *frames_*,
unsigned int *channels_*,
algo_comm_t *ac_*,
std::string *name_*,
float *gain_in_*,
float *gain_ac_*,
unsigned int *delay_in_*,
unsigned int *delay_ac_*)

5.9.2 Member Function Documentation

5.9.2.1 mha_wave_t ∗ ac2wave_t::process (
mha_wave_t ∗ *s*)

5.9.3 Member Data Documentation

5.9.3.1 unsigned int ac2wave_t::frames [private]

5.9.3.2 unsigned int ac2wave_t::channels [private]

5.9.3.3 mha_wave_t ac2wave_t::w [private]

5.9.3.4 algo_comm_t ac2wave_t::ac [private]

5.9.3.5 std::string ac2wave_t::name [private]

5.9.3.6 MHASignal::delay_wave_t ac2wave_t::delay_in [private]

5.9.3.7 MHASignal::delay_wave_t ac2wave_t::delay_ac [private]

5.9.3.8 mha_real_t ac2wave_t::gain_in [private]

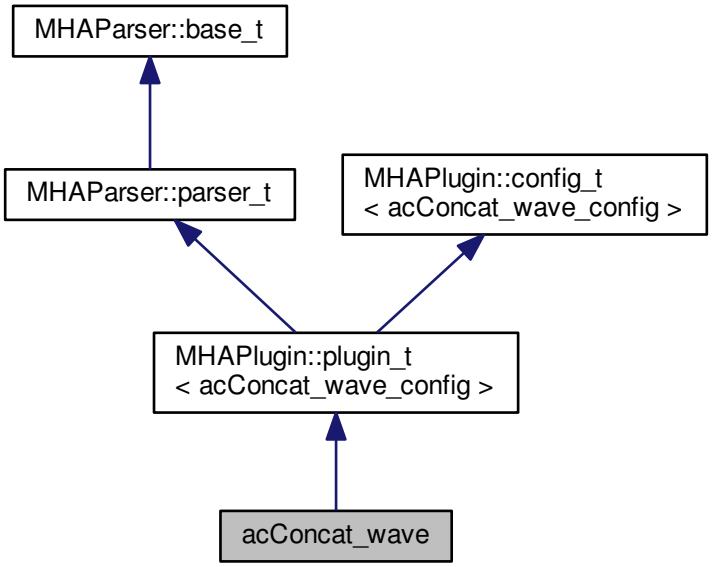
5.9.3.9 mha_real_t ac2wave_t::gain_ac [private]

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

- **ac2wave.cpp**

5.10 acConcat_wave Class Reference

Inheritance diagram for acConcat_wave:



Public Member Functions

- **acConcat_wave** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  Constructs our plugin.
- ∼**acConcat_wave** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t num_AC**
- **MHAParser::string_t prefix_names_AC**
- **MHAParser::vint_t samples_AC**
- **MHAParser::string_t name_con_AC**
- **MHAParser::int_t numchannels**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **acConcat_wave** > **patchbay**

Additional Inherited Members

5.10.1 Constructor & Destructor Documentation

5.10.1.1 acConcat_wave::acConcat_wave (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.10.1.2 acConcat_wave::~acConcat_wave ( )

5.10.2 Member Function Documentation

5.10.2.1 mha_wave_t * acConcat_wave::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.10.2.2 void acConcat_wave::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< acConcat_wave_config >** (p. 730).

5.10.2.3 void acConcat_wave::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< acConcat_wave_config >** (p. 731).

5.10.2.4 void acConcat_wave::update_cfg ( ) [private]

5.10.3 Member Data Documentation

5.10.3.1 MHAParser::int_t acConcat_wave::num_AC

5.10.3.2 MHAParser::string_t acConcat_wave::prefix_names_AC

5.10.3.3 MHAParser::vint_t acConcat_wave::samples_AC

5.10.3.4 MHAParser::string_t acConcat_wave::name_con_AC

5.10.3.5 MHAParser::int_t acConcat_wave::numchannels

5.10.3.6 MHAEvents::patchbay_t<acConcat_wave> acConcat_wave::patchbay [private]

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

- **acConcat_wave.h**
- **acConcat_wave.cpp**

5.11 acConcat_wave_config Class Reference

Public Member Functions

- **acConcat_wave_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **acConcat_↩ wave** ∗_concat)
- ∼**acConcat_wave_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Public Attributes

- **algo_comm_t** & **ac**
- std::vector< std::string > **strNames_AC**
- std::vector< int > **numSamples_AC**
- **mha_wave_t vGCC**
- **MHA_AC::waveform_t** ∗ **vGCC_con**

5.11.1 Constructor & Destructor Documentation

5.11.1.1 acConcat_wave_config::acConcat_wave_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* acConcat_wave ∗ *_concat*)

5.11.1.2 acConcat_wave_config::∼acConcat_wave_config ( )

5.11.2 Member Function Documentation

5.11.2.1 mha_wave_t ∗ acConcat_wave_config::process (mha_wave_t ∗ *wave*)

5.11.3 Member Data Documentation

5.11.3.1 algo_comm_t& acConcat_wave_config::ac

5.11.3.2 std::vector<std::string> acConcat_wave_config::strNames_AC

5.11.3.3 std::vector<int> acConcat_wave_config::numSamples_AC

5.11.3.4 mha_wave_t acConcat_wave_config::vGCC

5.11.3.5 MHA_AC::waveform_t∗ acConcat_wave_config::vGCC_con

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

- **acConcat_wave.h**
- **acConcat_wave.cpp**

5.12 acmon::ac_monitor_t Class Reference

A class for converting AC variables to Parser monitors of correct type.

Public Member Functions

- **ac_monitor_t** (**MHAParser::parser_t** &parent, const std::string &name_, **algo_comm_t** ac, bool use_matrix)
  Converts AC variable to parser monitor.
- void **getvar** (**algo_comm_t** ac)
  Update values of monitor.

Public Attributes

- std::string **name**
  name of AC variable and parser monitor
- std::string **dimstr**
  columns x rows
- **MHAParser::vfloat_mon_t mon**
  Monitor used for real vectors.
- **MHAParser::mfloat_mon_t mon_mat**
  Monitor used for real matrices.
- **MHAParser::vcomplex_mon_t mon_complex**
  monitor used for complex vectors
- **MHAParser::mcomplex_mon_t mon_mat_complex**
  monitor used for complex matrices
- **MHAParser::parser_t** & **p_parser**
  parent parser to insert monitor into

Private Attributes

- bool **use_mat**
  if true, use matrix monitor, else use vector monitor

5.12.1 Detailed Description

A class for converting AC variables to Parser monitors of correct type.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 acmon::ac_monitor_t::ac_monitor_t (MHAParser::parser_t & *parent*, const std::string & *name_*, algo_comm_t *ac*, bool *use_matrix*)

Converts AC variable to parser monitor.

Parameters

parent	The parser to insert a monitor into
*name_*	The name of the AC variable and the monitor variable
ac	Handle to algorithm communication space
use_matrix	Indicates if a matrix monitor type should be used.

5.12.3 Member Function Documentation

5.12.3.1 void acmon::ac_monitor_t::getvar (algo_comm_t *ac*)

Update values of monitor.

Parameters

ac	Handle to algorithm communication space

5.12.4 Member Data Documentation

5.12.4.1 std::string acmon::ac_monitor_t::name

name of AC variable and parser monitor

5.12.4.2 std::string acmon::ac_monitor_t::dimstr

columns x rows

5.12.4.3 MHAParser::vfloat_mon_t acmon::ac_monitor_t::mon

Monitor used for real vectors.

5.12.4.4 MHAParser::mfloat_mon_t acmon::ac_monitor_t::mon_mat

Monitor used for real matrices.

5.12.4.5 MHAParser::vcomplex_mon_t acmon::ac_monitor_t::mon_complex

monitor used for complex vectors

5.12.4.6 MHAParser::mcomplex_mon_t acmon::ac_monitor_t::mon_mat_complex

monitor used for complex matrices

5.12.4.7 MHAParser::parser_t& acmon::ac_monitor_t::p_parser

parent parser to insert monitor into

5.12.4.8 bool acmon::ac_monitor_t::use_mat [private]

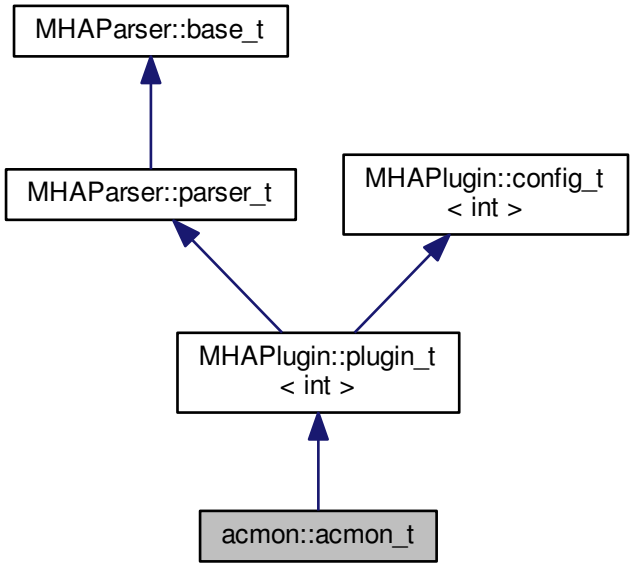
if true, use matrix monitor, else use vector monitor

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

- **ac_monitor_type.hh**
- **ac_monitor_type.cpp**

5.13 acmon::acmon_t Class Reference

Inheritance diagram for acmon::acmon_t:



Public Member Functions

- **acmon_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- ∼**acmon_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **save_vars** ()
- void **update_recmode** ()

Private Attributes

- **algo_comm_t ac**
- **MHAParser::vstring_mon_t varlist**
- **MHAParser::vstring_mon_t dimensions**
- **MHAParser::kw_t dispmode**
- **MHAParser::kw_t recmode**
- std::vector< **ac_monitor_t** ∗ > **vars**
- **MHAEvents::patchbay_t**< **acmon_t** > **patchbay**
- std::string **chain**
- std::string **algo**
- bool **b_cont**
- bool **b_snapshot**

Additional Inherited Members

5.13.1 Constructor & Destructor Documentation

5.13.1.1 acmon::acmon_t::acmon_t (
const **algo_comm_t** & *iac,*
const **std::string** & *ith,*
const **std::string** & *ial*)

5.13.1.2 acmon::acmon_t::∼acmon_t (
void)

5.13.2 Member Function Documentation

5.13.2.1 void acmon::acmon_t::prepare (
mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **int** > (p. 730).

5.13.2.2 void acmon::acmon_t::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.13.2.3 mha_spec_t * acmon::acmon_t::process (mha_spec_t * *s*)

5.13.2.4 mha_wave_t * acmon::acmon_t::process (mha_wave_t * *s*)

5.13.2.5 void acmon::acmon_t::save_vars ( ) [private]

5.13.2.6 void acmon::acmon_t::update_recmode ( ) [private]

5.13.3 Member Data Documentation

5.13.3.1 algo_comm_t acmon::acmon_t::ac [private]

5.13.3.2 MHAParser::vstring_mon_t acmon::acmon_t::varlist [private]

5.13.3.3 MHAParser::vstring_mon_t acmon::acmon_t::dimensions [private]

5.13.3.4 MHAParser::kw_t acmon::acmon_t::dispmode [private]

5.13.3.5 MHAParser::kw_t acmon::acmon_t::recmode [private]

5.13.3.6 std::vector<ac_monitor_t*> acmon::acmon_t::vars [private]

5.13.3.7 MHAEvents::patchbay_t<acmon_t> acmon::acmon_t::patchbay [private]

5.13.3.8 std::string acmon::acmon_t::chain [private]

5.13.3.9 std::string acmon::acmon_t::algo [private]

5.13.3.10 bool acmon::acmon_t::b_cont [private]

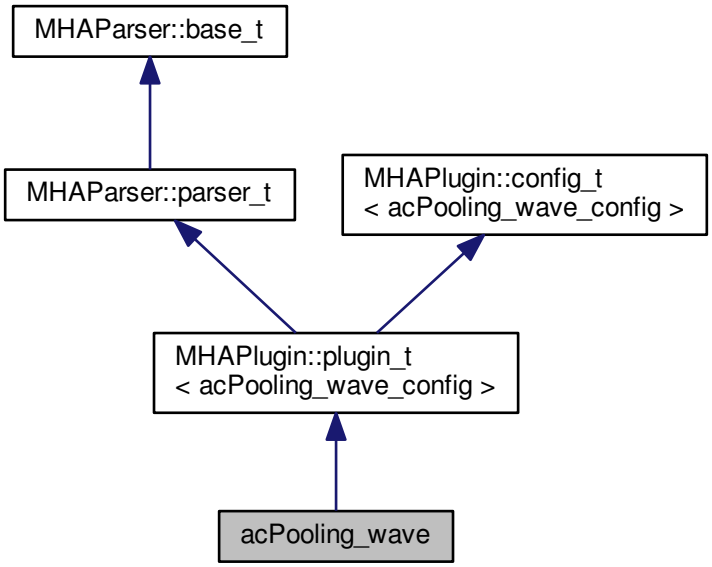
5.13.3.11 bool acmon::acmon_t::b_snapshot [private]

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

- **acmon.cpp**

5.14 acPooling_wave Class Reference

Inheritance diagram for acPooling_wave:



Public Member Functions

- **acPooling_wave** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
  Constructs our plugin.
- ∼**acPooling_wave** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t numsamples**
- **MHAParser::int_t pooling_wndlen**
- **MHAParser::kw_t pooling_type**
- **MHAParser::float_t upper_threshold**
- **MHAParser::float_t lower_threshold**
- **MHAParser::int_t neighbourhood**

- **MHAParser::float_t alpha**
- **MHAParser::string_t p_name**
- **MHAParser::string_t p_biased_name**
- **MHAParser::string_t pool_name**
- **MHAParser::string_t max_pool_ind_name**
- **MHAParser::string_t like_ratio_name**
- **MHAParser::vfloat_t prob_bias**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **acPooling_wave** > **patchbay**

Additional Inherited Members

5.14.1 Constructor & Destructor Documentation

5.14.1.1 acPooling_wave::acPooling_wave (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.14.1.2 acPooling_wave::~acPooling_wave ( )

5.14.2 Member Function Documentation

5.14.2.1 mha_wave_t * acPooling_wave::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.14.2.2 void acPooling_wave::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< acPooling_wave_config >** (p. 730).

5.14.2.3 void acPooling_wave::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< acPooling_wave_config >** (p. 731).

5.14.2.4 void acPooling_wave::update_cfg ( ) [private]

5.14.3 Member Data Documentation

5.14.3.1 MHAParser::int_t acPooling_wave::numsamples

5.14.3.2 MHAParser::int_t acPooling_wave::pooling_wndlen

5.14.3.3 MHAParser::kw_t acPooling_wave::pooling_type

5.14.3.4 MHAParser::float_t acPooling_wave::upper_threshold

5.14.3.5 MHAParser::float_t acPooling_wave::lower_threshold

5.14.3.6 MHAParser::int_t acPooling_wave::neighbourhood

5.14.3.7 MHAParser::float_t acPooling_wave::alpha

5.14.3.8 MHAParser::string_t acPooling_wave::p_name

5.14.3.9 MHAParser::string_t acPooling_wave::p_biased_name

5.14.3.10 MHAParser::string_t acPooling_wave::pool_name

5.14.3.11 MHAParser::string_t acPooling_wave::max_pool_ind_name

5.14.3.12 MHAParser::string_t acPooling_wave::like_ratio_name

5.14.3.13 MHAParser::vfloat_t acPooling_wave::prob_bias

5.14.3.14 MHAEvents::patchbay_t<acPooling_wave> acPooling_wave::patchbay [private]

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

- **acPooling_wave.h**
- **acPooling_wave.cpp**

5.15 acPooling_wave_config Class Reference

Public Member Functions

- **acPooling_wave_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **acPooling_wave** ∗_pooling)
- ∼**acPooling_wave_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **insert** ()

Public Attributes

- **algo_comm_t** & **ac**
- std::string **raw_p_name**
- **MHA_AC::waveform_t p**
- **MHA_AC::waveform_t p_biased**
- **MHA_AC::waveform_t p_max**
- **MHA_AC::waveform_t like_ratio**
- **mha_wave_t c**
- unsigned int **pooling_ind**
- unsigned int **pooling_option**
- unsigned int **pooling_size**
- float **up_thresh**
- float **low_thresh**
- int **neigh**
- float **alpha**
- **MHASignal::waveform_t pool**
- **MHASignal::waveform_t prob_bias_func**

5.15.1 Constructor & Destructor Documentation

5.15.1.1 acPooling_wave_config::acPooling_wave_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* acPooling_wave ∗ *_pooling*)

5.15.1.2 acPooling_wave_config::∼acPooling_wave_config ( )

5.15.2 Member Function Documentation

5.15.2.1 mha_wave_t ∗ acPooling_wave_config::process (mha_wave_t ∗ *wave*)

5.15.2.2 void acPooling_wave_config::insert ( )

5.15.3 Member Data Documentation

5.15.3.1 algo_comm_t& acPooling_wave_config::ac

5.15.3.2 std::string acPooling_wave_config::raw_p_name

5.15.3.3 MHA_AC::waveform_t acPooling_wave_config::p

5.15.3.4 MHA_AC::waveform_t acPooling_wave_config::p_biased

5.15.3.5 MHA_AC::waveform_t acPooling_wave_config::p_max

5.15.3.6 MHA_AC::waveform_t acPooling_wave_config::like_ratio

5.15.3.7 mha_wave_t acPooling_wave_config::c

5.15.3.8 unsigned int acPooling_wave_config::pooling_ind

5.15.3.9 unsigned int acPooling_wave_config::pooling_option

5.15.3.10 unsigned int acPooling_wave_config::pooling_size

5.15.3.11 float acPooling_wave_config::up_thresh

5.15.3.12 float acPooling_wave_config::low_thresh

5.15.3.13 int acPooling_wave_config::neigh

5.15.3.14 float acPooling_wave_config::alpha

5.15.3.15 MHASignal::waveform_t acPooling_wave_config::pool

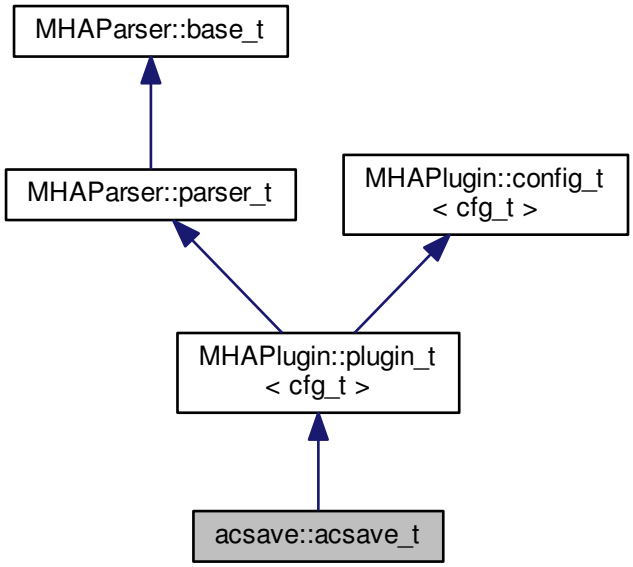
5.15.3.16 MHASignal::waveform_t acPooling_wave_config::prob_bias_func

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

- **acPooling_wave.h**
- **acPooling_wave.cpp**

5.16 acsave::acsave_t Class Reference

Inheritance diagram for acsave::acsave_t:



Public Member Functions

- **acsave_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **event_start_recording** ()
- void **event_stop_and_flush** ()

Private Types

- typedef std::vector< **save_var_t** ∗ > **varlist_t**

Private Member Functions

- void **process** ()

Private Attributes

- **MHAParser::bool_t bflush**
- **MHAParser::kw_t fileformat**
- **MHAParser::string_t fname**
- **MHAParser::float_t reclen**
- **MHAParser::vstring_t variables**
- **varlist_t varlist**
- std::string **chain**
- std::string **algo**
- bool **b_prepared**
- bool **b_flushed**
- **MHAEvents::patchbay_t**< **acsave_t** > **patchbay**

Additional Inherited Members

5.16.1 Member Typedef Documentation

5.16.1.1 typedef std::vector<save_var_t*> acsave::acsave_t::varlist_t [private]

5.16.2 Constructor & Destructor Documentation

5.16.2.1 acsave::acsave_t::acsave_t (
const algo_comm_t & *iac,*
const std::string & *ith,*
const std::string & *ial*)

5.16.3 Member Function Documentation

5.16.3.1 void acsave::acsave_t::prepare (
mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 730).

5.16.3.2 void acsave::acsave_t::release (
void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **cfg_t** > (p. 731).

5.16.3.3 mha_spec_t * acsave::acsave_t::process (mha_spec_t * *s*)

5.16.3.4 mha_wave_t * acsave::acsave_t::process (mha_wave_t * *s*)

5.16.3.5 void acsave::acsave_t::event_start_recording ( )

5.16.3.6 void acsave::acsave_t::event_stop_and_flush ( )

5.16.3.7 void acsave::acsave_t::process (void) [private]

5.16.4 Member Data Documentation

5.16.4.1 MHAParser::bool_t acsave::acsave_t::bflush [private]

5.16.4.2 MHAParser::kw_t acsave::acsave_t::fileformat [private]

5.16.4.3 MHAParser::string_t acsave::acsave_t::fname [private]

5.16.4.4 MHAParser::float_t acsave::acsave_t::reclen [private]

5.16.4.5 MHAParser::vstring_t acsave::acsave_t::variables [private]

5.16.4.6 varlist_t acsave::acsave_t::varlist [private]

5.16.4.7 std::string acsave::acsave_t::chain [private]

5.16.4.8 std::string acsave::acsave_t::algo [private]

5.16.4.9 bool acsave::acsave_t::b_prepared [private]

5.16.4.10 bool acsave::acsave_t::b_flushed [private]

5.16.4.11 MHAEvents::patchbay_t<acsave_t> acsave::acsave_t::patchbay [private]

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

- **acsave.cpp**

5.17 acsave::cfg_t Class Reference

Public Member Functions

- **cfg_t** (const **algo_comm_t** &iac, unsigned int imax_frames, std::vector< std::string > &var_names)
- ~**cfg_t** ()
- void **store_frame** ()
- void **flush_data** (const std::string &, unsigned int)

Private Attributes

- **algo_comm_t ac**
- unsigned int **nvars**
- **save_var_t** ** **varlist**
- unsigned int **rec_frames**
- unsigned int **max_frames**

5.17.1 Constructor & Destructor Documentation

5.17.1.1 cfg_t::cfg_t (

const **algo_comm_t** & *iac,*
unsigned int *imax_frames,*
std::vector< std::string > & *var_names*)

5.17.1.2 cfg_t::~cfg_t ( )

5.17.2 Member Function Documentation

5.17.2.1 void cfg_t::store_frame ( )

This function is called in the processing thread.

5.17.2.2 void cfg_t::flush_data (

const std::string & *filename,*
unsigned int *fmt*)

This function is called in the configuration thread.

Parameters

filename	Output file name
fmt	Output file format

5.17.3 Member Data Documentation

5.17.3.1 algo_comm_t acsave::cfg_t::ac [private]

5.17.3.2 unsigned int acsave::cfg_t::nvars [private]

5.17.3.3 save_var_t** acsave::cfg_t::varlist [private]

5.17.3.4 unsigned int acsave::cfg_t::rec_frames [private]

5.17.3.5 unsigned int acsave::cfg_t::max_frames [private]

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

- **acsave.cpp**

5.18 acsave::mat4head_t Struct Reference

Public Attributes

- int32_t **t**
- int32_t **rows**
- int32_t **cols**
- int32_t **imag**
- int32_t **namelen**

5.18.1 Member Data Documentation

5.18.1.1 int32_t acsave::mat4head_t::t

5.18.1.2 int32_t acsave::mat4head_t::rows

5.18.1.3 int32_t acsave::mat4head_t::cols

5.18.1.4 int32_t acsave::mat4head_t::imag

5.18.1.5 int32_t acsave::mat4head_t::namelen

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

- **acsave.cpp**

5.19 acsave::save_var_t Class Reference

Public Member Functions

- **save_var_t** (const std::string &, int, const **algo_comm_t** &)
- ∼**save_var_t** ()
- void **store_frame** ()
- void **save_txt** (FILE ∗, unsigned int)
- void **save_mat4** (FILE ∗, unsigned int)
- void **save_m** (FILE ∗, unsigned int)

Public Attributes

- double ∗ **data**

Private Attributes

- std::string **name**
- unsigned int **nframes**
- unsigned int **ndim**
- unsigned int **maxframe**
- **algo_comm_t ac**
- unsigned int **framecnt**
- bool **b_complex**

5.19.1 Constructor & Destructor Documentation

5.19.1.1 acsave::save_var_t::save_var_t (const std::string & *nm,* int *n,* const algo_comm_t & *iac*)

5.19.1.2 acsave::save_var_t::∼save_var_t ( )

5.19.2 Member Function Documentation

5.19.2.1 void acsave::save_var_t::store_frame ( )

5.19.2.2 void acsave::save_var_t::save_txt (FILE ∗ *fh,* unsigned int *writeframes*)

5.19.2.3 void acsave::save_var_t::save_mat4 (FILE * *fh,* unsigned int *writeframes*)

5.19.2.4 void acsave::save_var_t::save_m (FILE * *fh,* unsigned int *writeframes*)

5.19.3 Member Data Documentation

5.19.3.1 double* acsave::save_var_t::data

5.19.3.2 std::string acsave::save_var_t::name [private]

5.19.3.3 unsigned int acsave::save_var_t::nframes [private]

5.19.3.4 unsigned int acsave::save_var_t::ndim [private]

5.19.3.5 unsigned int acsave::save_var_t::maxframe [private]

5.19.3.6 algo_comm_t acsave::save_var_t::ac [private]

5.19.3.7 unsigned int acsave::save_var_t::framecnt [private]

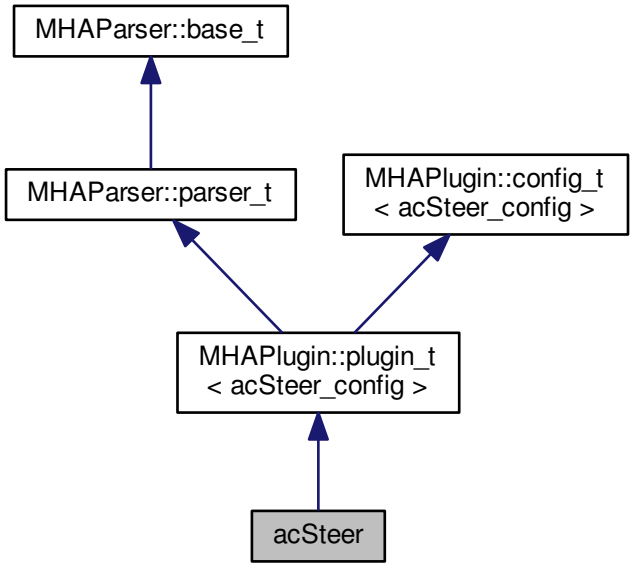
5.19.3.8 bool acsave::save_var_t::b_complex [private]

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

- **acsave.cpp**

5.20 acSteer Class Reference

Inheritance diagram for acSteer:



Public Member Functions

- **acSteer** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_↩ name)

  Constructs our plugin.
- ∼**acSteer** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

  Thos method is a NOOP.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::string_t steerFile**
- **MHAParser::string_t acSteerName1**
- **MHAParser::string_t acSteerName2**
- **MHAParser::int_t nsteerchan**
- **MHAParser::int_t nrefmic**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< acSteer > patchbay**

Additional Inherited Members

5.20.1 Constructor & Destructor Documentation

5.20.1.1 acSteer::acSteer (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.20.1.2 acSteer::~acSteer ( )

5.20.2 Member Function Documentation

5.20.2.1 mha_spec_t * acSteer::process (mha_spec_t * *signal*)

Thos method is a NOOP.

5.20.2.2 void acSteer::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< acSteer_config >** (p. 730).

5.20.2.3 void acSteer::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< acSteer_config >** (p. 731).

5.20.2.4 void acSteer::update_cfg ( ) [private]

5.20.3 Member Data Documentation

5.20.3.1 MHAParser::string_t acSteer::steerFile

5.20.3.2 MHAParser::string_t acSteer::acSteerName1

5.20.3.3 MHAParser::string_t acSteer::acSteerName2

5.20.3.4 MHAParser::int_t acSteer::nsteerchan

5.20.3.5 MHAParser::int_t acSteer::nrefmic

5.20.3.6 MHAEvents::patchbay_t<acSteer> acSteer::patchbay [private]

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

- **acSteer.h**
- **acSteer.cpp**

5.21 acSteer_config Class Reference

Public Member Functions

- **acSteer_config** (**algo_comm_t** &ac, const **mhaconfig_t** in_cfg, **acSteer** *__acSteer__)
- ~**acSteer_config** ()
- void **insert** ()

Public Attributes

- unsigned int **nchan**
- unsigned int **nfreq**
- unsigned int **nsteerchan**
- unsigned int **nrefmic**
- unsigned int **nangle**
- **MHA_AC::spectrum_t specSteer1**
- **MHA_AC::spectrum_t specSteer2**

5.21.1 Constructor & Destructor Documentation

5.21.1.1 acSteer_config::acSteer_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* acSteer * *acSteer*)

5.21.1.2 acSteer_config::~acSteer_config ( )

5.21.2 Member Function Documentation

5.21.2.1 void acSteer_config::insert ( )

5.21.3 Member Data Documentation

5.21.3.1 unsigned int acSteer_config::nchan

5.21.3.2 unsigned int acSteer_config::nfreq

5.21.3.3 unsigned int acSteer_config::nsteerchan

5.21.3.4 unsigned int acSteer_config::nrefmic

5.21.3.5 unsigned int acSteer_config::nangle

5.21.3.6 MHA_AC::spectrum_t acSteer_config::specSteer1

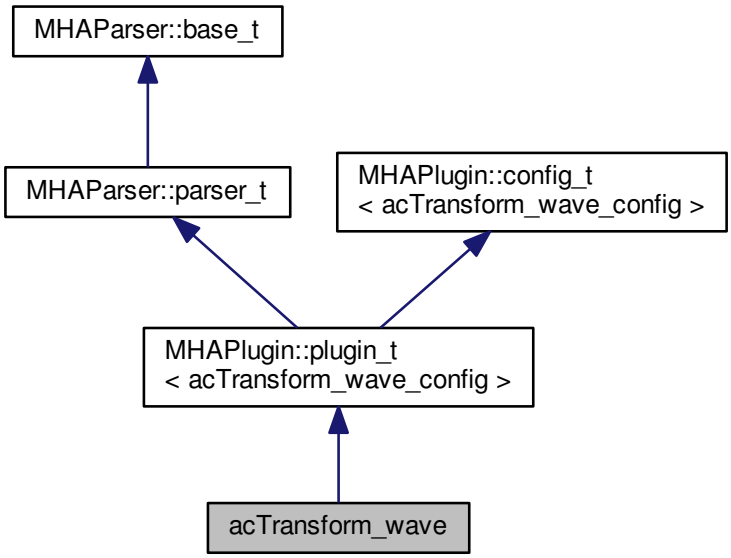
5.21.3.7 MHA_AC::spectrum_t acSteer_config::specSteer2

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

- **acSteer.h**
- **acSteer.cpp**

5.22 acTransform_wave Class Reference

Inheritance diagram for acTransform_wave:



Public Member Functions

- **acTransform_wave** (**algo_comm_t** &**ac**, const std::string &chain_name, const std↩ ::string &algo_name)

  Constructs our plugin.
- ∼**acTransform_wave** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::string_t ang_name**
- **MHAParser::string_t raw_p_name**
- **MHAParser::string_t raw_p_max_name**
- **MHAParser::string_t rotated_p_name**
- **MHAParser::string_t rotated_p_max_name**
- **MHAParser::int_t numsamples**
- **MHAParser::bool_t to_from**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< acTransform_wave > patchbay**

Additional Inherited Members

5.22.1 Constructor & Destructor Documentation

5.22.1.1 acTransform_wave::acTransform_wave (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.22.1.2 acTransform_wave::~acTransform_wave ( )

5.22.2 Member Function Documentation

5.22.2.1 mha_wave_t * acTransform_wave::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.22.2.2 void acTransform_wave::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< acTransform_wave_config >** (p. 730).

5.22.2.3 void acTransform_wave::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **acTransform_wave_config** > (p. 731).

5.22.2.4 void acTransform_wave::update_cfg ( ) [private]

5.22.3 Member Data Documentation

5.22.3.1 MHAParser::string_t acTransform_wave::ang_name

5.22.3.2 MHAParser::string_t acTransform_wave::raw_p_name

5.22.3.3 MHAParser::string_t acTransform_wave::raw_p_max_name

5.22.3.4 MHAParser::string_t acTransform_wave::rotated_p_name

5.22.3.5 MHAParser::string_t acTransform_wave::rotated_p_max_name

5.22.3.6 MHAParser::int_t acTransform_wave::numsamples

5.22.3.7 MHAParser::bool_t acTransform_wave::to_from

5.22.3.8 MHAEvents::patchbay_t<acTransform_wave> acTransform_wave::patchbay [private]

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

5.23 acTransform_wave_config Class Reference

Public Member Functions

- **acTransform_wave_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **ac↩ Transform_wave** *_transform)
- ~**acTransform_wave_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Public Attributes

- **algo_comm_t** & **ac**
- std::string **ang_name**
- std::string **raw_p_name**
- std::string **raw_p_max_name**
- **MHA_AC::waveform_t rotated_p**
- **MHA_AC::int_t rotated_i**
- unsigned int **offset**
- unsigned int **resolution**
- unsigned int **to_from**

5.23.1 Constructor & Destructor Documentation

5.23.1.1 acTransform_wave_config::acTransform_wave_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* acTransform_wave * *_transform*)

5.23.1.2 acTransform_wave_config::~acTransform_wave_config ( )

5.23.2 Member Function Documentation

5.23.2.1 mha_wave_t * acTransform_wave_config::process (mha_wave_t * *wave*)

5.23.3 Member Data Documentation

5.23.3.1 algo_comm_t& acTransform_wave_config::ac

5.23.3.2 std::string acTransform_wave_config::ang_name

5.23.3.3 std::string acTransform_wave_config::raw_p_name

5.23.3.4 std::string acTransform_wave_config::raw_p_max_name

5.23.3.5 MHA_AC::waveform_t acTransform_wave_config::rotated_p

5.23.3.6 MHA_AC::int_t acTransform_wave_config::rotated_i

5.23.3.7 unsigned int acTransform_wave_config::offset

5.23.3.8 unsigned int acTransform_wave_config::resolution

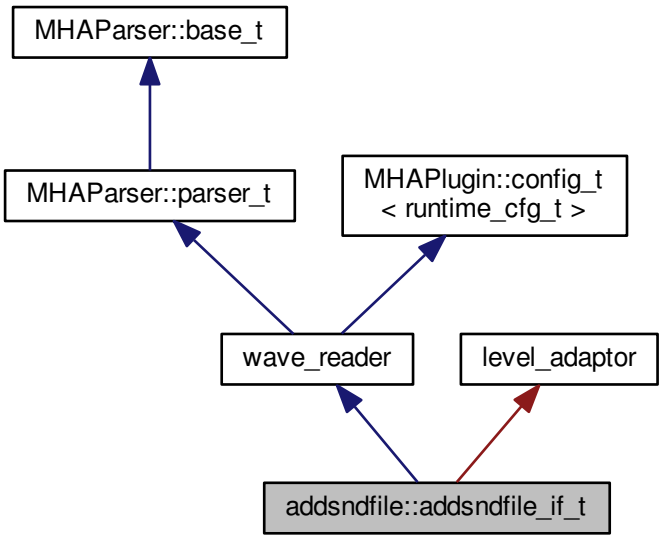
5.23.3.9 unsigned int acTransform_wave_config::to_from

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

5.24 addsndfile::addsndfile_if_t Class Reference

Inheritance diagram for addsndfile::addsndfile_if_t:



Public Member Functions

- **addsndfile_if_t** (**algo_comm_t**, const char *, const char *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update** ()
- void **change_mode** ()
- void **set_level** ()
- void **scan_dir** ()

Private Attributes

- **MHAParser::string_t filename**
- **MHAParser::string_t path**
- **MHAParser::bool_t loop**
- **MHAParser::float_t level**

- **MHAParser::kw_t levelmode**
- **MHAParser::kw_t resamplingmode**
- **MHAParser::vint_t channels**
- **MHAParser::kw_t mode**
- **MHAParser::float_t ramplen**
- **MHAParser::int_t startpos**
- **MHAParser::vint_mon_t mapping**
- **MHAParser::int_mon_t numchannels**
- **MHAParser::int_mon_t mhachannels**
- **MHAParser::int_mon_t active**
- **MHAParser::string_t search_pattern**
- **MHAParser::vstring_mon_t search_result**
- unsigned int **uint_mode**
- **MHAEvents::patchbay_t< addsndfile_if_t > patchbay**

Additional Inherited Members

5.24.1 Constructor & Destructor Documentation

5.24.1.1 addsndfile::addsndfile_if_t::addsndfile_if_t (
algo_comm_t *iac,*
const char * ,
const char *)

5.24.2 Member Function Documentation

5.24.2.1 mha_wave_t * addsndfile::addsndfile_if_t::process (
mha_wave_t * *s*)

5.24.2.2 void addsndfile::addsndfile_if_t::prepare (
mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 730).

5.24.2.3 void addsndfile::addsndfile_if_t::release (
void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 731).

5.24.2.4 void addsndfile::addsndfile_if_t::update ( ) [private]

5.24.2.5 void addsndfile::addsndfile_if_t::change_mode ( ) [private]

5.24.2.6 void addsndfile::addsndfile_if_t::set_level ( ) [private]

5.24.2.7 void addsndfile::addsndfile_if_t::scan_dir ( ) [private]

5.24.3 Member Data Documentation

5.24.3.1 MHAParser::string_t addsndfile::addsndfile_if_t::filename [private]

5.24.3.2 MHAParser::string_t addsndfile::addsndfile_if_t::path [private]

5.24.3.3 MHAParser::bool_t addsndfile::addsndfile_if_t::loop [private]

5.24.3.4 MHAParser::float_t addsndfile::addsndfile_if_t::level [private]

5.24.3.5 MHAParser::kw_t addsndfile::addsndfile_if_t::levelmode [private]

5.24.3.6 MHAParser::kw_t addsndfile::addsndfile_if_t::resamplingmode [private]

5.24.3.7 MHAParser::vint_t addsndfile::addsndfile_if_t::channels [private]

5.24.3.8 MHAParser::kw_t addsndfile::addsndfile_if_t::mode [private]

5.24.3.9 MHAParser::float_t addsndfile::addsndfile_if_t::ramplen [private]

5.24.3.10 MHAParser::int_t addsndfile::addsndfile_if_t::startpos [private]

5.24.3.11 MHAParser::vint_mon_t addsndfile::addsndfile_if_t::mapping [private]

5.24.3.12 MHAParser::int_mon_t addsndfile::addsndfile_if_t::numchannels [private]

5.24.3.13 MHAParser::int_mon_t addsndfile::addsndfile_if_t::mhachannels [private]

5.24.3.14 MHAParser::int_mon_t addsndfile::addsndfile_if_t::active [private]

5.24.3.15 MHAParser::string_t addsndfile::addsndfile_if_t::search_pattern [private]

5.24.3.16 MHAParser::vstring_mon_t addsndfile::addsndfile_if_t::search_result [private]

5.24.3.17 unsigned int addsndfile::addsndfile_if_t::uint_mode [private]

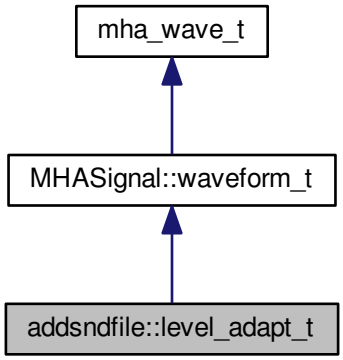
5.24.3.18 MHAEvents::patchbay_t<addsndfile_if_t> addsndfile::addsndfile_if_t::patchbay [private]

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

- **addsndfile.cpp**

5.25 addsndfile::level_adapt_t Class Reference

Inheritance diagram for addsndfile::level_adapt_t:



Public Member Functions

- **level_adapt_t** (**mhaconfig_t** cf, **mha_real_t** adapt_len, **mha_real_t** l_new_, **mha_real_t** l_old_)
- void **update_frame** ()
- **mha_real_t get_level** () const
- bool **can_update** () const

Private Attributes

- unsigned int **ilen**
- unsigned int **pos**
- **MHAWindow::fun_t wnd**
- **mha_real_t l_new**
- **mha_real_t l_old**

Additional Inherited Members

5.25.1 Constructor & Destructor Documentation

5.25.1.1 addsndfile::level_adapt_t::level_adapt_t (mhaconfig_t *cf,* mha_real_t *adapt_len,* mha_real_t *l_new_,* mha_real_t *l_old_*)

5.25.2 Member Function Documentation

5.25.2.1 void addsndfile::level_adapt_t::update_frame ( )

5.25.2.2 mha_real_t addsndfile::level_adapt_t::get_level ( ) const [inline]

5.25.2.3 bool addsndfile::level_adapt_t::can_update ( ) const [inline]

5.25.3 Member Data Documentation

5.25.3.1 unsigned int addsndfile::level_adapt_t::ilen [private]

5.25.3.2 unsigned int addsndfile::level_adapt_t::pos [private]

5.25.3.3 MHAWindow::fun_t addsndfile::level_adapt_t::wnd [private]

5.25.3.4 mha_real_t addsndfile::level_adapt_t::l_new [private]

5.25.3.5 mha_real_t addsndfile::level_adapt_t::l_old [private]

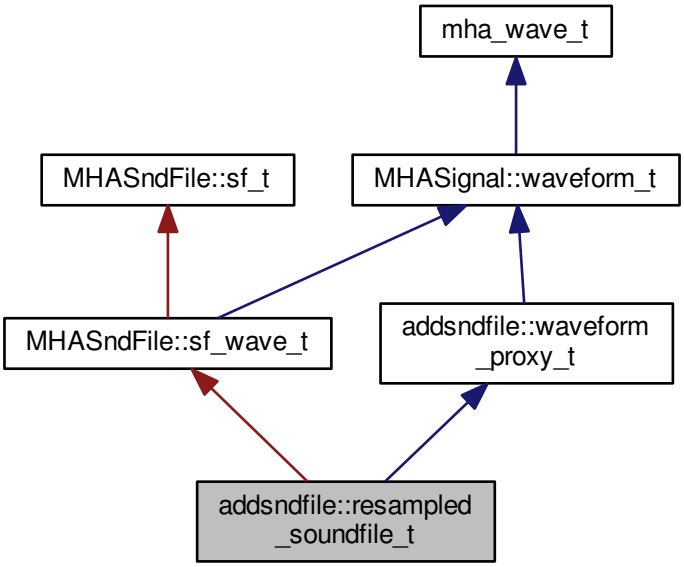
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

5.26 addsndfile::resampled_soundfile_t Class Reference

Reads sound from file and resamples it if necessary and wanted.

Inheritance diagram for addsndfile::resampled_soundfile_t:



Public Member Functions

- **resampled_soundfile_t** (const std::string &name, float mha_sampling_rate, **addsndfile_resampling_mode_t** resampling_mode)

  Reads sound from file and resamples if necessary and wanted.

Additional Inherited Members

5.26.1 Detailed Description

Reads sound from file and resamples it if necessary and wanted.

Sound data can then be used by addsndfile.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 addsndfile::resampled_soundfile_t::resampled_soundfile_t (const std::string & *name,* float *mha_sampling_rate,* addsndfile_resampling_mode_t *resampling_mode*)

Reads sound from file and resamples if necessary and wanted.

If the sound file does not specify a sampling rate, then the sound data is always used without resampling.

Parameters

name	Sound file name
mha_sampling_rate	The sampling rate of the MHA signal processing at the point of the addsndfile plugin
resampling_mode	DONT_RESAMPLE_STRICT: Do not resample, just use the samples from the sound file at the current sample rate, even if the sample rate of the sound file differs. DONT_RESAMPLE_PERMISSIVE: Do not resample, if the sample rate of the MHA differs from the sample rate of the sound file, raise an error. DO_RESAMPLE: Resample.

Exceptions

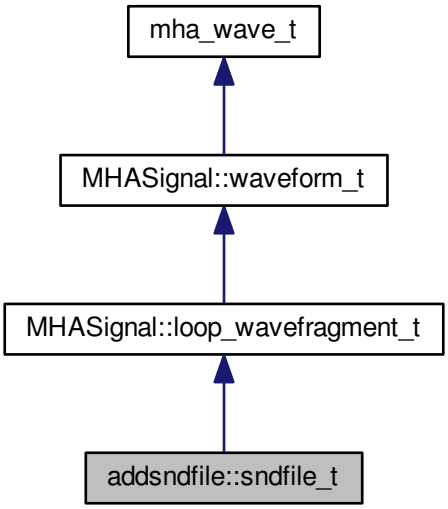
MHA_Error *(p. 445)*	If the sampling rate of the file does not match the sampling rate of the MHA and DONT_RESAMPLE_STRICT was requested. If resampling failed (e.g. due to non-rational quotient of MHA sampling rate and sound file sampling rate).

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

- **addsndfile.cpp**

5.27 addsndfile::sndfile_t Class Reference

Inheritance diagram for addsndfile::sndfile_t:



Public Member Functions

- **sndfile_t** (const std::string &name, bool loop, unsigned int level_mode, std::vector< int > channels_, unsigned int nchannels, std::vector< int > &mapping, int &numchannels, unsigned int startpos, float mha_sampling_rate, **addsndfile_resampling_mode_t** resampling_mode)

Additional Inherited Members

5.27.1 Constructor & Destructor Documentation

5.27.1.1 addsndfile::sndfile_t::sndfile_t (
const std::string & *name,*
bool *loop,*
unsigned int *level_mode,*
std::vector< int > *channels_,*
unsigned int *nchannels,*
std::vector< int > & *mapping,*
int & *numchannels,*
unsigned int *startpos,*
float *mha_sampling_rate,*
addsndfile_resampling_mode_t *resampling_mode*)

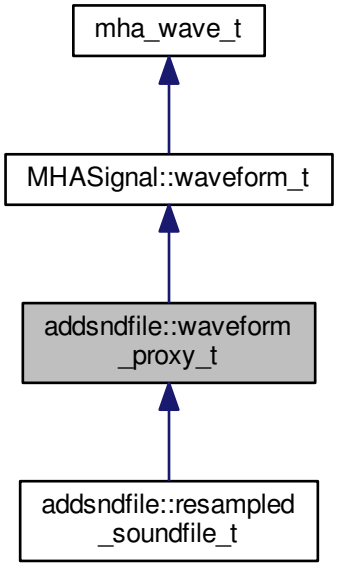
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

5.28 addsndfile::waveform_proxy_t Class Reference

Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. 184).

Inheritance diagram for addsndfile::waveform_proxy_t:



Public Member Functions

- **waveform_proxy_t** (unsigned frames, unsigned **channels**)

Additional Inherited Members

5.28.1 Detailed Description

Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. 184).

5.28.2 Constructor & Destructor Documentation

5.28.2.1 addsndfile::waveform_proxy_t::waveform_proxy_t (unsigned *frames,* unsigned *channels*) [inline]

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

- **addsndfile.cpp**

5.29 ADM::ADM< F > Class Template Reference

Adaptive differential microphone, working for speech frequency range.

Public Member Functions

- **ADM** (F fs, F dist, unsigned lp_order, const F *lp_alphas, unsigned decomb_order, const F *decomb_alphas, F tau_beta=F(50e-3), F mu_beta=F(1e-4))
  Create Adaptive Differential Microphone.
- F **process** (const F &front, const F &back, const F &external_beta=F(-1))
  ADM *(p. 188) processes one frame.*
- F **beta** () const

Private Attributes

- **Delay**< F > **m_delay_front**
- **Delay**< F > **m_delay_back**
- **Linearphase_FIR**< F > **m_lp_bf**
- **Linearphase_FIR**< F > **m_lp_result**
- **Linearphase_FIR**< F > **m_decomb**
- F **m_beta**
- F **m_mu_beta**
- F **m_powerfilter_coeff**
- F **m_powerfilter_norm**
- F **m_powerfilter_state**

5.29.1 Detailed Description

template<class F>
class ADM::ADM< F >

Adaptive differential microphone, working for speech frequency range.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 template<class F > ADM::ADM< F >::ADM (

F *fs*,
F *dist*,
unsigned *lp_order*,
const F * *lp_alphas*,
unsigned *decomb_order*,
const F * *decomb_alphas*,
F *tau_beta* = `F(50e-3)`**,**
F *mu_beta* = `F(1e-4)` **)**

Create Adaptive Differential Microphone.

Parameters

fs	Sampling rate / Hz
dist	Distance between physical microphones / m
lp_order	Filter order of FIR lowpass filter used for adaptation
lp_alphas	Pointer to array of alpha coefficients for the lowpass filter used for adaptation. Since this class uses linear phase FIR filters only, only the first half (order/2 + 1) of the coefficients will be read (coefficients for linear-phase FIR filters are symmetric).
decomb_order	Filter order of FIR compensation filter (compensates for comb filter characteristic)
decomb_alphas	Pointer to array of alpha coefficients for the compensation filter used to compensate for the comb filter characteristic. Since this class uses linear phase FIR filters only, only the first half (order/2 + 1)of the coefficients will be read (coefficients for linear-phase FIR filters are symmetric).
tau_beta	Time constant of the lowpass filter used for averaging the power of the output signal
mu_beta	adaption speed

5.29.3 Member Function Documentation

5.29.3.1 template<class F > F ADM::ADM< F >::process (const F & *front,* const F & *back,* const F & *external_beta =* `F(-1)`) [inline]

ADM (p. 188) processes one frame.

Parameters

front	The current front input signal sample
back	The current rear input signal sample
external_beta	If >= 0, this is used as the "beta" parameter for direction to filter out. Else, the beta parameter is adapted to filtered out a direction so that best reduction of signal intensity from the back hemisphere is achieved.

Returns

The computed output sample

5.29.3.2 template<class F > F ADM::ADM< F >::beta ( ) const [inline]

5.29.4 Member Data Documentation

5.29.4.1 template<class F > Delay<F> ADM::ADM< F >::m_delay_front [private]

5.29.4.2 template<class F > Delay<F> ADM::ADM< F >::m_delay_back [private]

5.29.4.3 template<class F > Linearphase_FIR<F> ADM::ADM< F >::m_lp_bf [private]

5.29.4.4 template<class F > Linearphase_FIR<F> ADM::ADM< F >::m_lp_result [private]

5.29.4.5 template<class F > Linearphase_FIR<F> ADM::ADM< F >::m_decomb [private]

5.29.4.6 template<class F > F ADM::ADM< F >::m_beta [private]

5.29.4.7 template<class F > F ADM::ADM< F >::m_mu_beta [private]

5.29.4.8 template<class F > F ADM::ADM< F >::m_powerfilter_coeff [private]

5.29.4.9 template<class F > F ADM::ADM< F >::m_powerfilter_norm [private]

5.29.4.10 template<class F > F ADM::ADM< F >::m_powerfilter_state [private]

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

- **adm.hh**

5.30 ADM::Delay< F > Class Template Reference

A delay-line class which can also do subsample-delays for a limited frequency range below fs/4.

Public Member Functions

- **Delay** (F samples, F f_design, F fs)
  Create a signal delay object.
- ~**Delay** ()
- F **process** (const F &in_sample)
  Apply delay to signal.

Private Attributes

- unsigned **m_fullsamples**
  Integer part of delay.
- F **m_coeff**
  coefficient for 1st order IIR lowpass filter which does the subsample delay
- F **m_norm**
  normalization for the IIR subsample delay filter
- F * **m_state**
  Ringbuffer: Delayline.
- unsigned **m_now_in**
  current position for inserting new samples into m_state ringbuffer

5.30.1 Detailed Description

template<class F>
class ADM::Delay< F >

A delay-line class which can also do subsample-delays for a limited frequency range below fs/4.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 template<class F > ADM::Delay< F >::Delay (F *samples,* F *f_design,* F *fs*)

Create a signal delay object.

Parameters

samples	number of samples to delay (may be non-integer)
f_design	subsampledelay is exact for this frequency
fs	sampling frequency

5.30.2.2 template<class F > ADM::Delay< F >::~Delay ( )

5.30.3 Member Function Documentation

5.30.3.1 template<class F > F ADM::Delay< F >::process (const F & *in_sample*) [inline]

Apply delay to signal.

Parameters

in_sample	The current input signal sample

Returns

The computed output sample

5.30.4 Member Data Documentation

5.30.4.1 template<class F > unsigned ADM::Delay< F >::m_fullsamples [private]

Integer part of delay.

5.30.4.2 template<class F > F ADM::Delay< F >::m_coeff [private]

coefficient for 1st order IIR lowpass filter which does the subsample delay

5.30.4.3 template<class F > F ADM::Delay< F >::m_norm [private]

normalization for the IIR subsample delay filter

5.30.4.4 template<class F > F* ADM::Delay< F >::m_state [private]

Ringbuffer: Delayline.

5.30.4.5 template<class F > unsigned ADM::Delay< F >::m_now_in [private]

current position for inserting new samples into m_state ringbuffer

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

- **adm.hh**

5.31 ADM::Linearphase_FIR< F > Class Template Reference

An efficient linear-phase fir filter implementation.

Public Member Functions

- **Linearphase_FIR** (unsigned order, const F *alphas)
  Create linear-phase FIR filter.
- ~**Linearphase_FIR** ()
- F **process** (const F &in_sample)
  Filter one sample with this linear-phase FIR filter.

Private Attributes

- unsigned **m_order**
  The filter order of this linear-phase FIR filter.
- F * **m_alphas**
  FIR filter coefficients.
- F * **m_output**
  Ringbuffer for building future output.
- unsigned **m_now**
  current start of ringbuffer

5.31.1 Detailed Description

template<class F>
class ADM::Linearphase_FIR< F >

An efficient linear-phase fir filter implementation.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 template<class F > ADM::Linearphase_FIR< F >::Linearphase_FIR (unsigned *order,* const F * *alphas*)

Create linear-phase FIR filter.

Parameters

order	filter order of this FIR filter. restriction: must be even.
alphas	pointer to Array of alpha coefficients. Since this class is for linear phase FIR filters only, only (order / 2 + 1) coefficients will be read. (Coefficients for linear-phase FIR filters are symmetric.)

5.31.2.2 template<class F > ADM::Linearphase_FIR< F >::~Linearphase_FIR ( )

5.31.3 Member Function Documentation

5.31.3.1 template<class F > F ADM::Linearphase_FIR< F >::process (const F & *in_sample*) [inline]

Filter one sample with this linear-phase FIR filter.

Parameters

in_sample	the current input sample

Returns

the computed output sample

5.31.4 Member Data Documentation

5.31.4.1 template<class F > unsigned ADM::Linearphase_FIR< F >::m_order [private]

The filter order of this linear-phase FIR filter.

5.31.4.2 template<class F > F* ADM::Linearphase_FIR< F >::m_alphas [private]

FIR filter coefficients.

Only m_order / 2 + 1 coefficients need to be stored since coefficients of linear-phase FIR filters are symmetric

5.31.4.3 template<class F > F* ADM::Linearphase_FIR< F >::m_output [private]

Ringbuffer for building future output.

5.31.4.4 template<class F > unsigned ADM::Linearphase_FIR< F >::m_now [private]

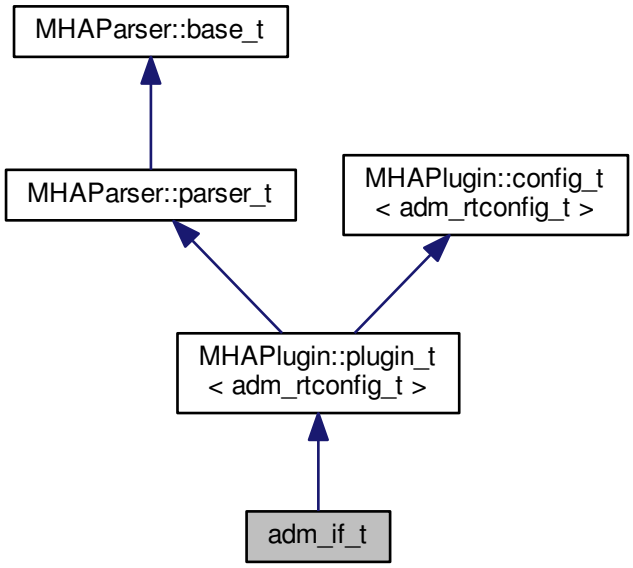
current start of ringbuffer

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

- **adm.hh**

5.32 adm_if_t Class Reference

Inheritance diagram for adm_if_t:



Public Member Functions

- **adm_if_t** (const **algo_comm_t** &**ac**, const std::string &thread_name, const std::string &algo_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗in)
- virtual void **prepare** (**mhaconfig_t** &)
- virtual void **release** ()

Private Member Functions

- void **update** ()
- bool **is_prepared** ()

Private Attributes

- **MHASignal::waveform_t * out**
- **MHAParser::vint_t front_channels**
- **MHAParser::vint_t rear_channels**
- **MHAParser::vfloat_t distances**
- **MHAParser::int_t lp_order**
- **MHAParser::int_t decomb_order**
- **MHAParser::int_t bypass**
- **MHAParser::float_t beta**
- **MHAParser::vfloat_t mu_beta**
- **MHAParser::vfloat_t tau_beta**
- **MHAParser::vfloat_mon_t coeff_lp**
- **MHAParser::vfloat_mon_t coeff_decomb**
- unsigned **input_channels**
- **mha_real_t srate**
- **MHAEvents::patchbay_t< adm_if_t > patchbay**

Additional Inherited Members

5.32.1 Constructor & Destructor Documentation

5.32.1.1 adm_if_t::adm_if_t (

const **algo_comm_t** & *ac,*
const std::string & *thread_name,*
const std::string & *algo_name*)

5.32.2 Member Function Documentation

5.32.2.1 mha_wave_t * adm_if_t::process (

mha_wave_t * *in*)

5.32.2.2 void adm_if_t::prepare (

mhaconfig_t & *cfg*) [virtual]

Implements **MHAPlugin::plugin_t< adm_rtconfig_t >** (p. 730).

5.32.2.3 void adm_if_t::release (

void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< adm_rtconfig_t >** (p. 731).

5.32.2.4 void adm_if_t::update ( ) [private]

5.32.2.5 bool adm_if_t::is_prepared ( ) [inline], [private]

5.32.3 Member Data Documentation

5.32.3.1 MHASignal::waveform_t* adm_if_t::out [private]

5.32.3.2 MHAParser::vint_t adm_if_t::front_channels [private]

5.32.3.3 MHAParser::vint_t adm_if_t::rear_channels [private]

5.32.3.4 MHAParser::vfloat_t adm_if_t::distances [private]

5.32.3.5 MHAParser::int_t adm_if_t::lp_order [private]

5.32.3.6 MHAParser::int_t adm_if_t::decomb_order [private]

5.32.3.7 MHAParser::int_t adm_if_t::bypass [private]

5.32.3.8 MHAParser::float_t adm_if_t::beta [private]

5.32.3.9 MHAParser::vfloat_t adm_if_t::mu_beta [private]

5.32.3.10 MHAParser::vfloat_t adm_if_t::tau_beta [private]

5.32.3.11 MHAParser::vfloat_mon_t adm_if_t::coeff_lp [private]

5.32.3.12 MHAParser::vfloat_mon_t adm_if_t::coeff_decomb [private]

5.32.3.13 unsigned adm_if_t::input_channels [private]

5.32.3.14 mha_real_t adm_if_t::srate [private]

5.32.3.15 MHAEvents::patchbay_t<adm_if_t> adm_if_t::patchbay [private]

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

- **adm.cpp**

5.33 adm_rtconfig_t Class Reference

Public Types

- typedef **ADM::ADM**< **mha_real_t** > **adm_t**

Public Member Functions

- **adm_rtconfig_t** (unsigned nchannels_in, unsigned nchannels_out, const std::vector< int > &**front_channels**, const std::vector< int > &**rear_channels**, const **mha_real_t** fs, const std::vector< **mha_real_t** > &distances, const int lp_order, const int decomb_order, const std::vector< **mha_real_t** > &tau_beta, const std::vector< **mha_real_t** > &mu_↩ beta)

  Construct new ADMs.
- virtual ∼**adm_rtconfig_t** ()
- size_t **num_adms** () const
- **adm_t** & **adm** (unsigned index)

  Returns adm object number index.
- int **front_channel** (unsigned index) const

  Returns index of front channel for adm number index.
- int **rear_channel** (unsigned index) const

  Returns index of rear channel for adm number index.

Private Member Functions

- void **check_index** (unsigned index) const

  Index checking for all internal arrays.

Private Attributes

- std::vector< int > **front_channels**

  Indices of channels containing the signals from the front microphones.
- std::vector< int > **rear_channels**

  Indices of channels containing the signals from the rear microphones.
- **MHASignal::waveform_t** ∗ **lp_coeffs**

  Lowpass filter coefficients.
- std::vector< **MHASignal::waveform_t** ∗ > **decomb_coeffs**

  Decomb-Filter coefficients.
- std::vector< **adm_t** ∗ > **adms**

  ADMs.

5.33.1 Member Typedef Documentation

5.33.1.1 typedef ADM::ADM<mha_real_t> adm_rtconfig_t::adm_t

5.33.2 Constructor & Destructor Documentation

5.33.2.1 adm_rtconfig_t::adm_rtconfig_t (

unsigned *nchannels_in,*
unsigned *nchannels_out,*
const std::vector< int > & *front_channels,*
const std::vector< int > & *rear_channels,*
const **mha_real_t** *fs,*
const std::vector< **mha_real_t** > & *distances,*
const int *lp_order,*
const int *decomb_order,*
const std::vector< **mha_real_t** > & *tau_beta,*
const std::vector< **mha_real_t** > & *mu_beta*)

Construct new ADMs.

Used when configuration changes.

Parameters

nchannels_in	Number of input channels
nchannels_out	Number of output channels
front_channels	Parser's front_channels setting
rear_channels	Parser's front_channels setting
fs	Sampling rate / Hz
distances	Distances between microphones / m
lp_order	Filter order of FIR lowpass filter for adaptation
decomb_order	Filter order of FIR compensation filter (compensates for comb filter characteristic)
tau_beta	Time constants of the lowpass filter used for averaging the power of the output signal used for adaptation
mu_beta	Adaptation step sizes

5.33.2.2 adm_rtconfig_t::~adm_rtconfig_t ( ) [virtual]

5.33.3 Member Function Documentation

5.33.3.1 void adm_rtconfig_t::check_index (

unsigned *index*) const [inline], [private]

Index checking for all internal arrays.

Exceptions

MHA_Error *(p. 445)*	if index out of range.

5.33.3.2 size_t adm_rtconfig_t::num_adms ( ) const [inline]

5.33.3.3 adm_t& adm_rtconfig_t::adm (unsigned *index*) [inline]

Returns adm object number index.

5.33.3.4 int adm_rtconfig_t::front_channel (unsigned *index*) const [inline]

Returns index of front channel for adm number index.

5.33.3.5 int adm_rtconfig_t::rear_channel (unsigned *index*) const [inline]

Returns index of rear channel for adm number index.

5.33.4 Member Data Documentation

5.33.4.1 std::vector<int> adm_rtconfig_t::front_channels [private]

Indices of channels containing the signals from the front microphones.

5.33.4.2 std::vector<int> adm_rtconfig_t::rear_channels [private]

Indices of channels containing the signals from the rear microphones.

5.33.4.3 MHASignal::waveform_t* adm_rtconfig_t::lp_coeffs [private]

Lowpass filter coefficients.

5.33.4.4 std::vector<MHASignal::waveform_t*> adm_rtconfig_t::decomb_coeffs [private]

Decomb-Filter coefficients.

5.33.4.5 std::vector<adm_t *> adm_rtconfig_t::adms [private]

ADMs.

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

- **adm.cpp**

5.34 algo_comm_t Struct Reference

A reference handle for algorithm communication variables.

Public Attributes

- void * **handle**
  AC variable control handle.
- int(* **insert_var**)(void *, const char *, **comm_var_t**)
  Register an AC variable.
- int(* **insert_var_int**)(void *, const char *, int *)
  Register an int as an AC variable.
- int(* **insert_var_float**)(void *, const char *, float *)
  Register a float as an AC variable.
- int(* **remove_var**)(void *, const char *)
  Remove an AC variable.
- int(* **remove_ref**)(void *, void *)
  Remove all AC variable which refer to address.
- int(* **is_var**)(void *, const char *)
  Test if an AC variable exists.
- int(* **get_var**)(void *, const char *, **comm_var_t** *)
  Get the variable handle of an AC variable.
- int(* **get_var_int**)(void *, const char *, int *)
  Get the value of an int AC variable.
- int(* **get_var_float**)(void *, const char *, float *)
  Get the value of a float AC variable.
- int(* **get_entries**)(void *, char *, unsigned int)
  Return a space separated list of all variable names.
- const char *(* **get_error**)(int)
  Convert AC error codes into human readable error messages.

5.34.1 Detailed Description

A reference handle for algorithm communication variables.

This structure contains a coontrol handle and a set of function pointers for sharing variables within one processing chain. See also section **Communication between algorithms** (p. 28).

5.34.2 Member Data Documentation

5.34.2.1 algo_comm_t::handle

AC variable control handle.

5.34.2.2 algo_comm_t::insert_var

Register an AC variable.

This function can register a variable to be shared within one chain. If a variable of this name exists it will be overwritten.

Parameters

h	AC handle
n	name of variable. May not be empty. Must not contain space character. The name is copied, therefore it is allowed that the char array pointed to gets invalid after return.
v	variable handle of type **comm_var_t** (p. 246)

Returns

Error code or zero on success

5.34.2.3 algo_comm_t::insert_var_int

Register an int as an AC variable.

This function can register an int variable to be shared with other algorithms. It behaves similar to ac.insert_var.

Parameters

h	AC handle
n	name of variable
v	pointer on the variable

Returns

Error code or zero on success

5.34.2.4 algo_comm_t::insert_var_float

Register a float as an AC variable.

This function can register a float variable to be shared with other algorithms. It behaves similar to ac.insert_var.

Parameters

h	AC handle
n	name of variable
v	pointer on the variable

Returns

Error code or zero on success

5.34.2.5 algo_comm_t::remove_var

Remove an AC variable.

Remove (unregister) an AC variable. After calling this function, the variable is not available to ac.is_var or ac.get_var. The data pointer is not affected.

Parameters

h	AC handle
n	name of variable to be removed

Returns

Error code or zero on success

5.34.2.6 algo_comm_t::remove_ref

Remove all AC variable which refer to address.

This function removes all AC variables whos data field points to the given address.

Parameters

h	AC handle
p	address which should not be referred to any more

Returns

Error code or zero on success

5.34.2.7 algo_comm_t::is_var

Test if an AC variable exists.

This function tests if an AC variable of a given name exists. Use ac.get_var to get information about the variables type and dimension.

Parameters

h	AC handle
n	name of variable

Returns

1 if the variable exists, 0 otherwise

5.34.2.8 algo_comm_t::get_var

Get the variable handle of an AC variable.

This function returns the variable handle **comm_var_t** (p. 246) of a variable of the given name. If no variable of that name exists, an error code is returned.

Parameters

h	AC handle
n	name of variable
v	pointer to a AC variable object

Returns

Error code or zero on success

5.34.2.9 algo_comm_t::get_var_int

Get the value of an int AC variable.

This function returns the value of an int AC variable of the given name. If no variable exists, the variable type is mismatching or more than one entry is registered, a corresponding error code is returned. This is a special version of ac.get_var.

Parameters

h	AC handle
n	name of variable
v	pointer on an int variable to store the result

Returns

Error code or zero on success

5.34.2.10 algo_comm_t::get_var_float

Get the value of a float AC variable.

This function returns the value of a float AC variable of the given name. If no variable exists, the variable type is mismatching or more than one entry is registered, a corresponding error code is returned. This is a special version of ac.get_var.

Parameters

h	AC handle
n	name of variable
v	pointer on a float variable to store the result

Returns

Error code or zero on success

5.34.2.11 algo_comm_t::get_entries

Return a space separated list of all variable names.

This function returns the names of all registered variables, separated by a single space.

Parameters

h	AC handle

Return values

ret	Character buffer for return value

Parameters

len	length of character buffer

Returns

Error code or zero on success. -1: invalid ac handle. -3: not enough room in character buffer to store all variable names.

5.34.2.12 algo_comm_t::get_error

Convert AC error codes into human readable error messages.

Parameters

e	Error code

Returns

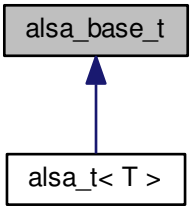
Error message

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

- **mha.hh**
- **mha_algo_comm.cpp**

5.35 alsa_base_t Class Reference

Inheritance diagram for alsa_base_t:



Public Member Functions

- **alsa_base_t** ()
- virtual ∼**alsa_base_t** ()=default
- virtual void **start** ()=0

  start puts alsa device in usable state
- virtual void **stop** ()=0

  stop informs alsa device that we do not need any more samples / will not provide any more samples
- virtual int **read** (**mha_wave_t** ∗∗)=0

  *read audio samples from the device into an internal **mha_wave_t*** (p. 496) *buffer, then update the pointer given as parameter to point to the internal structure.*
- virtual int **write** (**mha_wave_t** ∗)=0

  write audio samples from the given waveform buffer to the sound device.

Public Attributes

- snd_pcm_t ∗ **pcm**

  The underlying alsa handle to this sound card.

5.35.1 Constructor & Destructor Documentation

5.35.1.1 alsa_base_t::alsa_base_t ( ) [inline]

5.35.1.2 virtual alsa_base_t::~alsa_base_t ( ) [virtual], [default]

5.35.2 Member Function Documentation

5.35.2.1 virtual void alsa_base_t::start ( ) [pure virtual]

start puts alsa device in usable state

Implemented in **alsa_t**< **T** > (p. 211).

5.35.2.2 virtual void alsa_base_t::stop ( ) [pure virtual]

stop informs alsa device that we do not need any more samples / will not provide any more samples

Implemented in **alsa_t**< **T** > (p. 212).

5.35.2.3 virtual int alsa_base_t::read (**mha_wave_t** **) [pure virtual]

read audio samples from the device into an internal **mha_wave_t** (p. 496) buffer, then update the pointer given as parameter to point to the internal structure.

Converts sound samples from the integer data type provided by the sound card to floating-point values needed by the MHA in the range [-1.0,1.0]

Implemented in **alsa_t**< **T** > (p. 212).

5.35.2.4 virtual int alsa_base_t::write (**mha_wave_t** *) [pure virtual]

write audio samples from the given waveform buffer to the sound device.

converts the floating point values coming from the MHA to the integer samples required by the sound card.

Implemented in **alsa_t**< **T** > (p. 212).

5.35.3 Member Data Documentation

5.35.3.1 snd_pcm_t* alsa_base_t::pcm

The underlying alsa handle to this sound card.

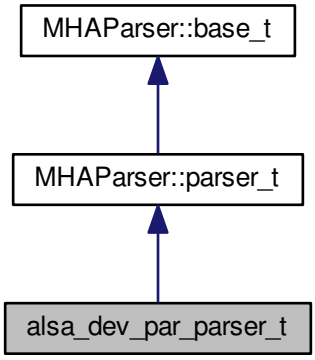
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.36 alsa_dev_par_parser_t Class Reference

Parser variables corresponding to one alsa device.

Inheritance diagram for alsa_dev_par_parser_t:



Public Member Functions

- **alsa_dev_par_parser_t** (snd_pcm_stream_t **stream_dir**)
  Constructor inserts the parser variables into this sub-parser.

Public Attributes

- **MHAParser::string_t device**
  Name of the device in the alsa world, like "hw:0.0", "default", etc.
- **MHAParser::int_t nperiods**
  Number of buffers of fragsize to hold in the alsa buffer.
- snd_pcm_stream_t **stream_dir**
  Remember the direction (capture/playback) of this device.

Additional Inherited Members

5.36.1 Detailed Description

Parser variables corresponding to one alsa device.

ALSA separates audio capture and audio playback into two different devices that have to be opened separately. This class encapsulates the parser variables that pertain to one such direction.

5.36.2 Constructor & Destructor Documentation

5.36.2.1 alsa_dev_par_parser_t::alsa_dev_par_parser_t (snd_pcm_stream_t *stream_dir*)

Constructor inserts the parser variables into this sub-parser.

Parameters

stream_dir	capture or playback

5.36.3 Member Data Documentation

5.36.3.1 MHAParser::string_t alsa_dev_par_parser_t::device

Name of the device in the alsa world, like "hw:0.0", "default", etc.

5.36.3.2 MHAParser::int_t alsa_dev_par_parser_t::nperiods

Number of buffers of fragsize to hold in the alsa buffer.

Usually 2, the minimum possible.

5.36.3.3 snd_pcm_stream_t alsa_dev_par_parser_t::stream_dir

Remember the direction (capture/playback) of this device.

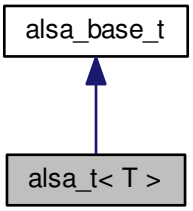
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.37 alsa_t< T > Class Template Reference

Our representation of one alsa device.

Inheritance diagram for alsa_t< T >:



Public Member Functions

- **alsa_t** (const **alsa_dev_par_parser_t** &par, unsigned int rate, unsigned int **fragsize**, unsigned int **channels**)

  Constructor receives the parameters for this device.
- ~**alsa_t** ()

  Destructor closes the sound device.
- void **start** () override

  start puts alsa device in usable state
- void **stop** () override

  stop informs alsa device that we do not need any more samples / will not provide any more samples
- int **read** (**mha_wave_t** **) override

  *read audio samples from the device into an internal **mha_wave_t** (p. 496) buffer, then update the pointer given as parameter to point to the internal structure.*
- int **write** (**mha_wave_t** *) override

  write audio samples from the given waveform buffer to the sound device.

Private Attributes

- unsigned int **channels**
- unsigned int **fragsize**
- T * **buffer**
- std::vector< **mha_real_t** > **frame_data**
- **MHASignal::waveform_t wave**

  internal buffer to store sound samples coming from the sound card.
- const **mha_real_t gain**
- const **mha_real_t invgain**
- const **mha_real_t val_min**
- const **mha_real_t val_max**
- snd_pcm_format_t **pcm_format**

Additional Inherited Members

5.37.1 Detailed Description

template<typename T>
class alsa_t< T >

Our representation of one alsa device.

We can start and stop the device, and depending on the direction, read or write samples.

5.37.2 Constructor & Destructor Documentation

5.37.2.1 template<typename T > alsa_t< T >::alsa_t (const alsa_dev_par_parser_t & *par,* unsigned int *rate,* unsigned int *fragsize,* unsigned int *channels*)

Constructor receives the parameters for this device.

It opens the sound device using the alsa library and selects the given parameters, but does not yet start the sound device to perform real I/O.

Parameters

par	our parser variable aggregator (containing direction, device name, and number of periods to place in alsa buffer)
rate	sampling rate in Hz
fragsize	samples per block per channel
channels	number of audio channels to open

5.37.2.2 template<typename T > alsa_t< T >::~alsa_t ( )

Destructor closes the sound device.

5.37.3 Member Function Documentation

5.37.3.1 template<typename T > void alsa_t< T >::start ( ) [override], [virtual]

start puts alsa device in usable state

Implements **alsa_base_t** (p. 207).

5.37.3.2 template<typename T > void alsa_t< T >::stop ( ) [override], [virtual]

stop informs alsa device that we do not need any more samples / will not provide any more samples

Implements **alsa_base_t** (p. 207).

5.37.3.3 template<typename T > int alsa_t< T >::read (mha_wave_t ** *s*) [override], [virtual]

read audio samples from the device into an internal **mha_wave_t** (p. 496) buffer, then update the pointer given as parameter to point to the internal structure.

Converts sound samples from the integer data type provided by the sound card to floating-point values needed by the MHA in the range [-1.0,1.0]

Implements **alsa_base_t** (p. 207).

5.37.3.4 template<typename T > int alsa_t< T >::write (mha_wave_t * *s*) [override], [virtual]

write audio samples from the given waveform buffer to the sound device.

converts the floating point values coming from the MHA to the integer samples required by the sound card.

Implements **alsa_base_t** (p. 207).

5.37.4 Member Data Documentation

5.37.4.1 template<typename T > unsigned int alsa_t< T >::channels [private]

5.37.4.2 template<typename T > unsigned int alsa_t< T >::fragsize [private]

5.37.4.3 template<typename T > T* alsa_t< T >::buffer [private]

5.37.4.4 template<typename T > std::vector<mha_real_t> alsa_t< T >::frame_data [private]

5.37.4.5 template<typename T > MHASignal::waveform_t alsa_t< T >::wave [private]

internal buffer to store sound samples coming from the sound card.

5.37.4.6 template<typename T > const mha_real_t alsa_t< T >::gain [private]

5.37.4.7 template<typename T > const mha_real_t alsa_t< T >::invgain [private]

5.37.4.8 template<typename T > const mha_real_t alsa_t< T >::val_min [private]

5.37.4.9 template<typename T > const mha_real_t alsa_t< T >::val_max [private]

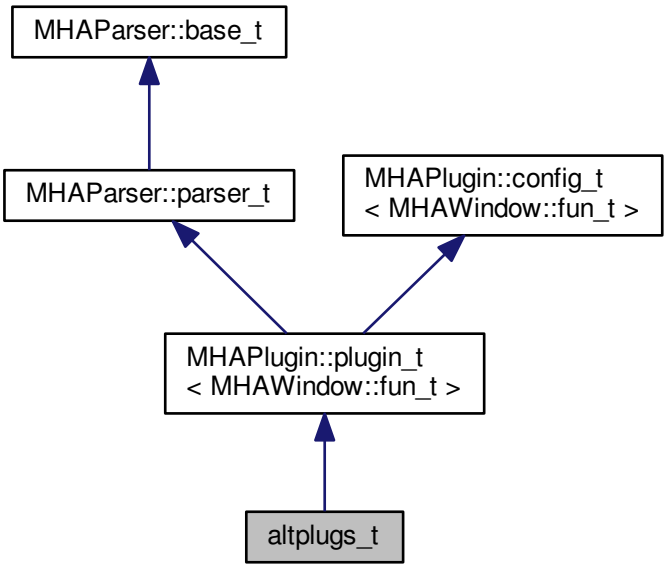
5.37.4.10 template<typename T > snd_pcm_format_t alsa_t< T >::pcm_format [private]

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

- **MHAIOalsa.cpp**

5.38 altplugs_t Class Reference

Inheritance diagram for altplugs_t:



Public Member Functions

- **altplugs_t** (**algo_comm_t** iac, const char *chain, const char *algo)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** (**mha_wave_t** *, **mha_wave_t** **)
- void **process** (**mha_spec_t** *, **mha_wave_t** **)
- void **process** (**mha_wave_t** *, **mha_spec_t** **)
- void **process** (**mha_spec_t** *, **mha_spec_t** **)
- virtual std::string **parse** (const std::string &arg)
- virtual void **parse** (const char *a1, char *a2, unsigned int a3)

Private Member Functions

- void **event_set_plugs** ()
- void **event_add_plug** ()
- void **event_delete_plug** ()
- void **event_select_plug** ()
- void **update_selector_list** ()
- void **update_ramplen** ()
- void **proc_ramp** (**mha_wave_t** ∗s)

Private Attributes

- **MHAParser::bool_t use_own_ac**
- **MHAParser::vstring_t parser_plugs**
- **MHAParser::string_t add_plug**
- **MHAParser::string_t delete_plug**
- **MHAParser::float_t ramplen**
- **MHAParser::kw_t select_plug**
- **MHAParser::parser_t current**
- **MHAParser::vstring_mon_t nondefault_labels**
- std::vector< **mhaplug_cfg_t** ∗ > **plugs**
- **mhaplug_cfg_t** ∗ **selected_plug**
- **MHAEvents::patchbay_t**< **altplugs_t** > **patchbay**
- **MHASignal::waveform_t** ∗ **fallback_wave**
- **MHASignal::spectrum_t** ∗ **fallback_spec**
- **mhaconfig_t cfin**
- **mhaconfig_t cfout**
- bool **prepared**
- bool **added_via_plugs**
- unsigned int **ramp_counter**
- unsigned int **ramp_len**

Additional Inherited Members

5.38.1 Constructor & Destructor Documentation

5.38.1.1 altplugs_t::altplugs_t (algo_comm_t *iac,* const char ∗ *chain,* const char ∗ *algo*)

5.38.2 Member Function Documentation

5.38.2.1 void altplugs_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t**< **MHAWindow::fun_t** > (p. 730).

5.38.2.2 void altplugs_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< MHAWindow::fun_t >** (p. 731).

5.38.2.3 void altplugs_t::process (mha_wave_t * *sIn*, mha_wave_t ** *sOut*)

5.38.2.4 void altplugs_t::process (mha_spec_t * *sIn*, mha_wave_t ** *sOut*)

5.38.2.5 void altplugs_t::process (mha_wave_t * *sIn*, mha_spec_t ** *sOut*)

5.38.2.6 void altplugs_t::process (mha_spec_t * *sIn*, mha_spec_t ** *sOut*)

5.38.2.7 std::string altplugs_t::parse (const std::string & *arg*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 637).

5.38.2.8 virtual void altplugs_t::parse (const char * *a1*, char * *a2*, unsigned int *a3*) [inline], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.38.2.9 void altplugs_t::event_set_plugs ( ) [private]

5.38.2.10 void altplugs_t::event_add_plug ( ) [private]

5.38.2.11 void altplugs_t::event_delete_plug ( ) [private]

5.38.2.12 void altplugs_t::event_select_plug ( ) [private]

5.38.2.13 void altplugs_t::update_selector_list ( ) [private]

5.38.2.14 void altplugs_t::update_ramplen ( ) [private]

5.38.2.15 void altplugs_t::proc_ramp (mha_wave_t * s) [private]

5.38.3 Member Data Documentation

5.38.3.1 MHAParser::bool_t altplugs_t::use_own_ac [private]

5.38.3.2 MHAParser::vstring_t altplugs_t::parser_plugs [private]

5.38.3.3 MHAParser::string_t altplugs_t::add_plug [private]

5.38.3.4 MHAParser::string_t altplugs_t::delete_plug [private]

5.38.3.5 MHAParser::float_t altplugs_t::ramplen [private]

5.38.3.6 MHAParser::kw_t altplugs_t::select_plug [private]

5.38.3.7 MHAParser::parser_t altplugs_t::current [private]

5.38.3.8 MHAParser::vstring_mon_t altplugs_t::nondefault_labels [private]

5.38.3.9 std::vector<mhaplug_cfg_t*> altplugs_t::plugs [private]

5.38.3.10 mhaplug_cfg_t* altplugs_t::selected_plug [private]

5.38.3.11 MHAEvents::patchbay_t<altplugs_t> altplugs_t::patchbay [private]

5.38.3.12 MHASignal::waveform_t* altplugs_t::fallback_wave [private]

5.38.3.13 MHASignal::spectrum_t* altplugs_t::fallback_spec [private]

5.38.3.14 mhaconfig_t altplugs_t::cfin [private]

5.38.3.15 mhaconfig_t altplugs_t::cfout [private]

5.38.3.16 bool altplugs_t::prepared [private]

5.38.3.17 bool altplugs_t::added_via_plugs [private]

5.38.3.18 unsigned int altplugs_t::ramp_counter [private]

5.38.3.19 unsigned int altplugs_t::ramp_len [private]

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

- **altplugs.cpp**

5.39 analysepath_t Class Reference

Public Member Functions

- **analysepath_t** (unsigned int nchannels_in, unsigned int outer_fragsize, unsigned int inner_fragsize, int **priority**, **MHAProc_wave2wave_t** inner_proc_wave2wave, **MHAProc_wave2spec_t** inner_proc_wave2spec, void *ilibdata, **algo_comm_t outer_ac**, const **MHA_AC::acspace2matrix_t** &acspace_template, **mha_domain_t inner_out_domain**, unsigned int fifo_len_blocks)
- virtual ~**analysepath_t** ()
- void **rt_process** (**mha_wave_t** *)
- virtual int **svc** ()

Private Attributes

- **MHAProc_wave2wave_t inner_process_wave2wave**
- **MHAProc_wave2spec_t inner_process_wave2spec**
- **MHASignal::waveform_t inner_input**
- void * **libdata**
- **mha_fifo_t**< **mha_real_t** > **wave_fifo**
- **mha_fifo_t**< **MHA_AC::acspace2matrix_t** > **ac_fifo**
- **MHA_AC::acspace2matrix_t inner_ac_copy**
- **MHA_AC::acspace2matrix_t outer_ac_copy**
- **algo_comm_t outer_ac**
- **mha_domain_t inner_out_domain**
- **MHA_Error inner_error**
- bool **has_inner_error**
- bool **flag_terminate_inner_thread**
- int **input_to_process**
- pthread_mutex_t **ProcessMutex**
- pthread_attr_t **attr**
- struct sched_param **priority**
- int **scheduler**
- pthread_t **thread**
- pthread_cond_t **cond_to_process**

5.39.1 Constructor & Destructor Documentation

5.39.1.1 analysepath_t::analysepath_t (

unsigned int *nchannels_in,*
unsigned int *outer_fragsize,*
unsigned int *inner_fragsize,*
int *priority,*
MHAProc_wave2wave_t *inner_proc_wave2wave,*
MHAProc_wave2spec_t *inner_proc_wave2spec,*
void * *ilibdata,*
algo_comm_t *outer_ac,*
const MHA_AC::acspace2matrix_t & *acspace_template,*
mha_domain_t *inner_out_domain,*
unsigned int *fifo_len_blocks*)

5.39.1.2 analysepath_t::~analysepath_t ( ) [virtual]

5.39.2 Member Function Documentation

5.39.2.1 void analysepath_t::rt_process (mha_wave_t * *outer_input*)

5.39.2.2 int analysepath_t::svc ( ) [virtual]

5.39.3 Member Data Documentation

5.39.3.1 MHAProc_wave2wave_t analysepath_t::inner_process_wave2wave [private]

5.39.3.2 MHAProc_wave2spec_t analysepath_t::inner_process_wave2spec [private]

5.39.3.3 MHASignal::waveform_t analysepath_t::inner_input [private]

5.39.3.4 void* analysepath_t::libdata [private]

5.39.3.5 mha_fifo_t<mha_real_t> analysepath_t::wave_fifo [private]

5.39.3.6 mha_fifo_t<MHA_AC::acspace2matrix_t> analysepath_t::ac_fifo [private]

5.39.3.7 MHA_AC::acspace2matrix_t analysepath_t::inner_ac_copy [private]

5.39.3.8 MHA_AC::acspace2matrix_t analysepath_t::outer_ac_copy [private]

5.39.3.9 algo_comm_t analysepath_t::outer_ac [private]

5.39.3.10 mha_domain_t analysepath_t::inner_out_domain [private]

5.39.3.11 MHA_Error analysepath_t::inner_error [private]

5.39.3.12 bool analysepath_t::has_inner_error [private]

5.39.3.13 bool analysepath_t::flag_terminate_inner_thread [private]

5.39.3.14 int analysepath_t::input_to_process [private]

5.39.3.15 pthread_mutex_t analysepath_t::ProcessMutex [private]

5.39.3.16 pthread_attr_t analysepath_t::attr [private]

5.39.3.17 struct sched_param analysepath_t::priority [private]

5.39.3.18 int analysepath_t::scheduler [private]

5.39.3.19 pthread_t analysepath_t::thread [private]

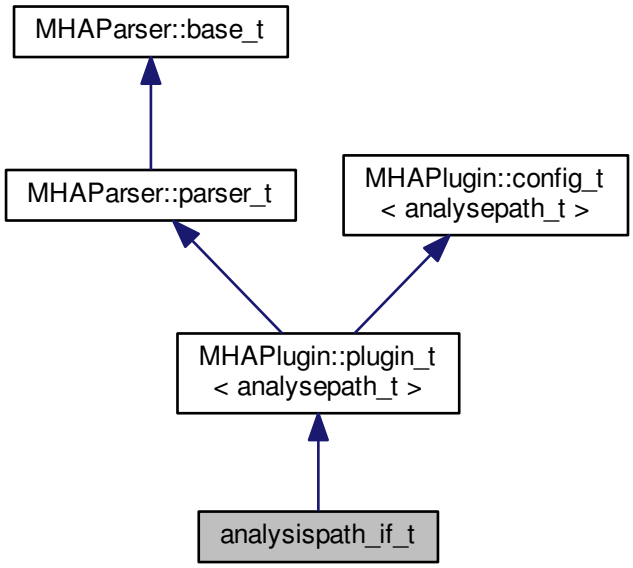
5.39.3.20 pthread_cond_t analysepath_t::cond_to_process [private]

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

- **analysispath.cpp**

5.40 analysispath_if_t Class Reference

Inheritance diagram for analysispath_if_t:



Public Member Functions

- **analysispath_if_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- ∼**analysispath_if_t** ()

Private Member Functions

- void **loadlib** ()

Private Attributes

- **MHAEvents::patchbay_t**< **analysispath_if_t** > **patchbay**
- **MHAParser::string_t libname**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t fifolen**
- **MHAParser::int_t priority**
- **MHAParser::vstring_t vars**
- **plug_t** ∗ **plug**
- std::string **chain**
- std::string **algo**
- **MHA_AC::acspace2matrix_t** ∗ **acspace_template**

Additional Inherited Members

5.40.1 Constructor & Destructor Documentation

5.40.1.1 analysispath_if_t::analysispath_if_t (algo_comm_t *iac,* std::string *th,* std::string *al*)

5.40.1.2 analysispath_if_t::∼analysispath_if_t ( )

5.40.2 Member Function Documentation

5.40.2.1 mha_wave_t * analysispath_if_t::process (mha_wave_t * *s*)

5.40.2.2 void analysispath_if_t::prepare (mhaconfig_t & *conf*) [virtual]

Implements **MHAPlugin::plugin_t< analysepath_t >** (p. 730).

5.40.2.3 void analysispath_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< analysepath_t >** (p. 731).

5.40.2.4 void analysispath_if_t::loadlib ( ) [private]

5.40.3 Member Data Documentation

5.40.3.1 MHAEvents::patchbay_t< analysispath_if_t > analysispath_if_t::patchbay [private]

5.40.3.2 MHAParser::string_t analysispath_if_t::libname [private]

5.40.3.3 MHAParser::int_t analysispath_if_t::fragsize [private]

5.40.3.4 MHAParser::int_t analysispath_if_t::fifolen [private]

5.40.3.5 MHAParser::int_t analysispath_if_t::priority [private]

5.40.3.6 MHAParser::vstring_t analysispath_if_t::vars [private]

5.40.3.7 plug_t* analysispath_if_t::plug [private]

5.40.3.8 std::string analysispath_if_t::chain [private]

5.40.3.9 std::string analysispath_if_t::algo [private]

5.40.3.10 MHA_AC::acspace2matrix_t* analysispath_if_t::acspace_template [private]

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

- **analysispath.cpp**

5.41 AuditoryProfile::fmap_t Class Reference

A class to store frequency dependent data (e.g., HTL and UCL).

Inherits map< mha_real_t, mha_real_t >.

Public Member Functions

- std::vector< **mha_real_t** > **get_frequencies** () const
  Return configured frequencies.
- std::vector< **mha_real_t** > **get_values** () const
  Return stored values corresponding to the frequencies.
- bool **isempty** () const

5.41.1 Detailed Description

A class to store frequency dependent data (e.g., HTL and UCL).

5.41.2 Member Function Documentation

5.41.2.1 std::vector< mha_real_t > AuditoryProfile::fmap_t::get_frequencies ( ) const

Return configured frequencies.

5.41.2.2 std::vector< mha_real_t > AuditoryProfile::fmap_t::get_values ( ) const

Return stored values corresponding to the frequencies.

5.41.2.3 bool AuditoryProfile::fmap_t::isempty ( ) const [inline]

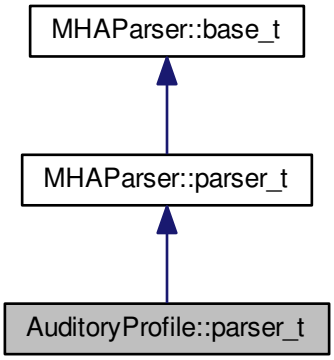
The documentation for this class was generated from the following files:

- **auditory_profile.h**
- **auditory_profile.cpp**

5.42 AuditoryProfile::parser_t Class Reference

Class to make the auditory profile accessible through the parser interface.

Inheritance diagram for AuditoryProfile::parser_t:



Classes

- class **ear_t**
- class **fmap_t**

Public Member Functions

- **parser_t** ()
- **AuditoryProfile::profile_t get_current_profile** ()

Private Attributes

- **AuditoryProfile::parser_t::ear_t L**
- **AuditoryProfile::parser_t::ear_t R**

Additional Inherited Members

5.42.1 Detailed Description

Class to make the auditory profile accessible through the parser interface.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 AuditoryProfile::parser_t::parser_t ( )

5.42.3 Member Function Documentation

5.42.3.1 AuditoryProfile::profile_t AuditoryProfile::parser_t::get_current_profile ( )

5.42.4 Member Data Documentation

5.42.4.1 AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::L [private]

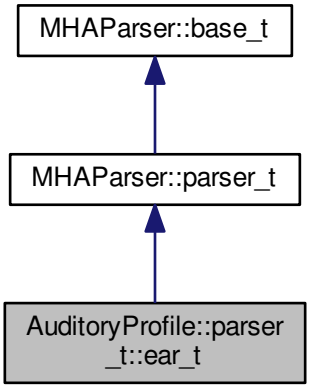
5.42.4.2 AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::R [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.43 AuditoryProfile::parser_t::ear_t Class Reference

Inheritance diagram for AuditoryProfile::parser_t::ear_t:



Public Member Functions

- **ear_t** ()
- **AuditoryProfile::profile_t::ear_t get_ear** () const

Private Attributes

- **AuditoryProfile::parser_t::fmap_t HTL**
- **AuditoryProfile::parser_t::fmap_t UCL**

Additional Inherited Members

5.43.1 Constructor & Destructor Documentation

5.43.1.1 AuditoryProfile::parser_t::ear_t::ear_t ( )

5.43.2 Member Function Documentation

5.43.2.1 AuditoryProfile::profile_t::ear_t AuditoryProfile::parser_t::ear_t::get_ear ( ) const

5.43.3 Member Data Documentation

5.43.3.1 AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::HTL [private]

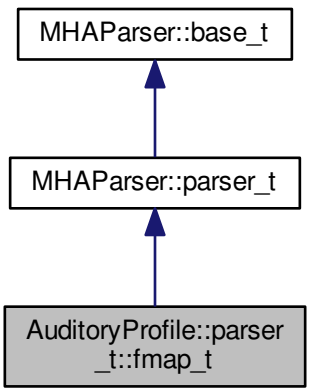
5.43.3.2 AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::UCL [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.44 AuditoryProfile::parser_t::fmap_t Class Reference

Inheritance diagram for AuditoryProfile::parser_t::fmap_t:



Public Member Functions

- **fmap_t** (const std::string &name, const std::string &**help**)
- **AuditoryProfile::fmap_t get_fmap** () const

Private Member Functions

- void **validate** ()

Private Attributes

- **MHAEvents::patchbay_t**< **AuditoryProfile::parser_t::fmap_t** > **patchbay**
- **MHAParser::vfloat_t f**
- **MHAParser::vfloat_t value**
- std::string **name_**

Additional Inherited Members

5.44.1 Constructor & Destructor Documentation

5.44.1.1 AuditoryProfile::parser_t::fmap_t::fmap_t (const std::string & *name,* const std::string & *help*)

5.44.2 Member Function Documentation

5.44.2.1 AuditoryProfile::fmap_t AuditoryProfile::parser_t::fmap_t::get_fmap ( ) const

5.44.2.2 void AuditoryProfile::parser_t::fmap_t::validate ( ) [private]

5.44.3 Member Data Documentation

5.44.3.1 MHAEvents::patchbay_t<AuditoryProfile::parser_t::fmap_t> AuditoryProfile::parser_t::fmap_t::patchbay [private]

5.44.3.2 MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::f [private]

5.44.3.3 MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::value [private]

5.44.3.4 std::string AuditoryProfile::parser_t::fmap_t::name_ [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.45 AuditoryProfile::profile_t Class Reference

The Auditory Profile class.

Classes

- class **ear_t**
  Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

Public Member Functions

- **AuditoryProfile::profile_t::ear_t get_ear** (unsigned int channel) const
  Return ear information of channel number.

Public Attributes

- **AuditoryProfile::profile_t::ear_t L**
  Left ear data.
- **AuditoryProfile::profile_t::ear_t R**
  Right ear data.

5.45.1 Detailed Description

The Auditory Profile class.

See definition of auditory profile

Todo Give more documentation; implement all parts of the auditory profile.

Currently only the audiogram data is stored.

5.45.2 Member Function Documentation

5.45.2.1 AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::get_ear (unsigned int *channel*) const [inline]

Return ear information of channel number.

5.45.3 Member Data Documentation

5.45.3.1 AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::L

Left ear data.

5.45.3.2 AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::R

Right ear data.

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

- **auditory_profile.h**

5.46 AuditoryProfile::profile_t::ear_t Class Reference

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

Public Member Functions

- void **convert_empty2normal** ()

Public Attributes

- **AuditoryProfile::fmap_t HTL**
- **AuditoryProfile::fmap_t UCL**

5.46.1 Detailed Description

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

5.46.2 Member Function Documentation

5.46.2.1 void AuditoryProfile::profile_t::ear_t::convert_empty2normal ( )

5.46.3 Member Data Documentation

5.46.3.1 AuditoryProfile::fmap_t AuditoryProfile::profile_t::ear_t::HTL

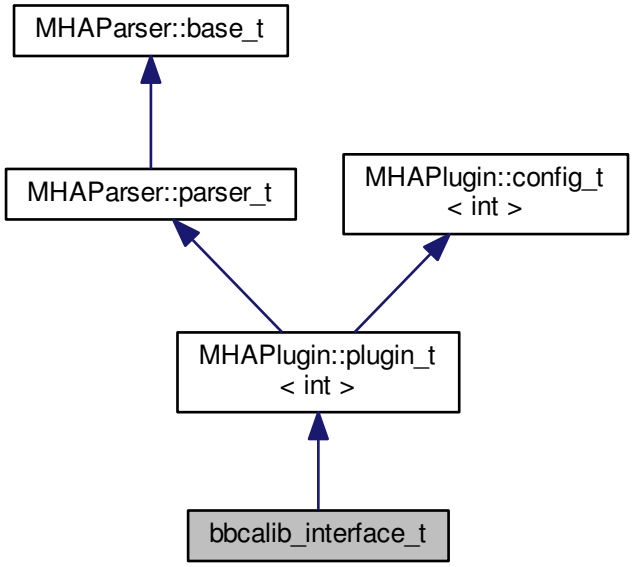
5.46.3.2 AuditoryProfile::fmap_t AuditoryProfile::profile_t::ear_t::UCL

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.47 bbcalib_interface_t Class Reference

Inheritance diagram for bbcalib_interface_t:



Public Member Functions

- **bbcalib_interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- ~**bbcalib_interface_t** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Attributes

- **calibrator_t calib_in**
- **calibrator_t calib_out**
- **MHAParser::mhapluginloader_t plugloader**

Additional Inherited Members

5.47.1 Constructor & Destructor Documentation

5.47.1.1 bbcalib_interface_t::bbcalib_interface_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.47.1.2 bbcalib_interface_t::~bbcalib_interface_t ( )

5.47.2 Member Function Documentation

5.47.2.1 mha_wave_t * bbcalib_interface_t::process (mha_wave_t * *s*)

5.47.2.2 void bbcalib_interface_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.47.2.3 void bbcalib_interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.47.3 Member Data Documentation

5.47.3.1 calibrator_t bbcalib_interface_t::calib_in [private]

5.47.3.2 calibrator_t bbcalib_interface_t::calib_out [private]

5.47.3.3 MHAParser::mhapluginloader_t bbcalib_interface_t::plugloader [private]

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

- **transducers.cpp**

5.48 calibrator_runtime_layer_t Class Reference

Public Member Functions

- **calibrator_runtime_layer_t** (bool is_input, const **mhaconfig_t** &tf, **calibrator_variables_t** &vars)
- **mha_real_t process** (**mha_wave_t** **)

Static Private Member Functions

- static unsigned int **firfirlen** (const std::vector< std::vector< float > > &)
- static unsigned int **firfir2fftlen** (unsigned int, const std::vector< std::vector< float > > &)

Private Attributes

- **MHAFilter::fftfilter_t fir**
- **MHASignal::quantizer_t quant**
- **MHASignal::waveform_t gain**
- **softclipper_t softclip**
- bool **b_is_input**
- bool **b_use_fir**
- bool **b_use_clipping**
- **MHASignal::loop_wavefragment_t speechnoise**
- **MHASignal::loop_wavefragment_t::playback_mode_t pmode**

5.48.1 Constructor & Destructor Documentation

5.48.1.1 calibrator_runtime_layer_t::calibrator_runtime_layer_t (bool *is_input,* const mhaconfig_t & *tf,* calibrator_variables_t & *vars*)

5.48.2 Member Function Documentation

5.48.2.1 mha_real_t calibrator_runtime_layer_t::process (mha_wave_t ** *s*)

5.48.2.2 unsigned int calibrator_runtime_layer_t::firfirlen (const std::vector< std::vector< float > > & *fir*) [static], [private]

5.48.2.3 unsigned int calibrator_runtime_layer_t::firfir2fftlen (unsigned int *fragsize,* const std::vector< std::vector< float > > & *fir*) [static], [private]

5.48.3 Member Data Documentation

5.48.3.1 MHAFilter::fftfilter_t calibrator_runtime_layer_t::fir [private]

5.48.3.2 MHASignal::quantizer_t calibrator_runtime_layer_t::quant [private]

5.48.3.3 MHASignal::waveform_t calibrator_runtime_layer_t::gain [private]

5.48.3.4 softclipper_t calibrator_runtime_layer_t::softclip [private]

5.48.3.5 bool calibrator_runtime_layer_t::b_is_input [private]

5.48.3.6 bool calibrator_runtime_layer_t::b_use_fir [private]

5.48.3.7 bool calibrator_runtime_layer_t::b_use_clipping [private]

5.48.3.8 MHASignal::loop_wavefragment_t calibrator_runtime_layer_t::speechnoise [private]

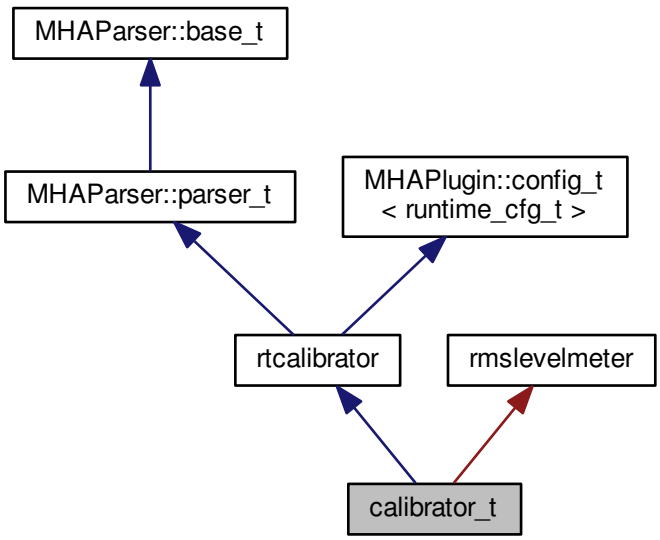
5.48.3.9 MHASignal::loop_wavefragment_t::playback_mode_t calibrator_runtime_layer_t::pmode [private]

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

- **transducers.cpp**

5.49 calibrator_t Class Reference

Inheritance diagram for calibrator_t:



Public Member Functions

- **calibrator_t** (**algo_comm_t**, bool is_input)
- void **prepare** (**mhaconfig_t** &tf)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)

Private Member Functions

- void **update** ()
- void **update_tau_level** ()
- void **read_levels** ()

Private Attributes

- bool **b_is_input**
- **MHAEvents::patchbay_t**< **calibrator_t** > **patchbay**
- **calibrator_variables_t vars**
- bool **prepared**

Additional Inherited Members

5.49.1 Constructor & Destructor Documentation

5.49.1.1 calibrator_t::calibrator_t (algo_comm_t *iac,* bool *is_input*)

5.49.2 Member Function Documentation

5.49.2.1 void calibrator_t::prepare (mhaconfig_t & *tf*) [inline], [virtual]

Implements **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 730).

5.49.2.2 void calibrator_t::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 731).

5.49.2.3 mha_wave_t * calibrator_t::process (mha_wave_t * *s*)

5.49.2.4 void calibrator_t::update ( ) [private]

5.49.2.5 void calibrator_t::update_tau_level ( ) [private]

5.49.2.6 void calibrator_t::read_levels ( ) [private]

5.49.3 Member Data Documentation

5.49.3.1 bool calibrator_t::b_is_input [private]

5.49.3.2 MHAEvents::patchbay_t<calibrator_t> calibrator_t::patchbay [private]

5.49.3.3 calibrator_variables_t calibrator_t::vars [private]

5.49.3.4 bool calibrator_t::prepared [private]

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

- **transducers.cpp**

5.50 calibrator_variables_t Class Reference

Public Member Functions

- **calibrator_variables_t** (bool is_input, **MHAParser::parser_t** &parent)

Public Attributes

- **MHAParser::vfloat_t peaklevel**
- **MHAParser::mfloat_t fir**
- **MHAParser::int_t nbits**
- **MHAParser::float_t tau_level**
- **MHAParser::kw_t spnoise_mode**
- **MHAParser::vint_t spnoise_channels**
- **MHAParser::float_t spnoise_level**
- **MHAParser::vfloat_mon_t rmslevel**
- **MHAParser::parser_t spnoise_parser**
- **MHAParser::float_mon_t srate**
- **MHAParser::int_mon_t fragsize**
- **MHAParser::int_mon_t num_channels**
- **MHAParser::parser_t config_parser**
- **softclipper_variables_t softclip**
- **MHAParser::bool_t do_clipping**

5.50.1 Constructor & Destructor Documentation

5.50.1.1 calibrator_variables_t::calibrator_variables_t (bool *is_input,* MHAParser::parser_t & *parent*)

5.50.2 Member Data Documentation

5.50.2.1 MHAParser::vfloat_t calibrator_variables_t::peaklevel

5.50.2.2 MHAParser::mfloat_t calibrator_variables_t::fir

5.50.2.3 MHAParser::int_t calibrator_variables_t::nbits

5.50.2.4 MHAParser::float_t calibrator_variables_t::tau_level

5.50.2.5 MHAParser::kw_t calibrator_variables_t::spnoise_mode

5.50.2.6 MHAParser::vint_t calibrator_variables_t::spnoise_channels

5.50.2.7 MHAParser::float_t calibrator_variables_t::spnoise_level

5.50.2.8 MHAParser::vfloat_mon_t calibrator_variables_t::rmslevel

5.50.2.9 MHAParser::parser_t calibrator_variables_t::spnoise_parser

5.50.2.10 MHAParser::float_mon_t calibrator_variables_t::srate

5.50.2.11 MHAParser::int_mon_t calibrator_variables_t::fragsize

5.50.2.12 MHAParser::int_mon_t calibrator_variables_t::num_channels

5.50.2.13 MHAParser::parser_t calibrator_variables_t::config_parser

5.50.2.14 softclipper_variables_t calibrator_variables_t::softclip

5.50.2.15 MHAParser::bool_t calibrator_variables_t::do_clipping

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

- **transducers.cpp**

5.51 cfg_t Class Reference

Public Member Functions

- **cfg_t** (unsigned int, unsigned int)
- **cfg_t** (**mhaconfig_t** chcfg, **mha_real_t** newlev, bool replace, **mha_real_t** len)
- void **process** (**mha_wave_t** ∗)
- void **process** (**mha_spec_t** ∗)
- **cfg_t** (**mha_real_t** tau_attack, **mha_real_t** tau_decay, unsigned int nch, **mha_real_↩ t** start_limit, **mha_real_t** slope_db, **mha_real_t** fs)

Public Attributes

- unsigned int **channel**
- **mha_real_t start_lin**
- **mha_real_t alpha**
- **MHAFilter::o1flt_lowpass_t attack**
- **MHAFilter::o1flt_maxtrack_t decay**

Private Attributes

- **mha_real_t gain_wave_**
- **mha_real_t gain_spec_**
- bool **replace_**
- bool **use_frozen_**
- **MHASignal::waveform_t frozen_noise_**
- unsigned int **pos**

5.51.1 Constructor & Destructor Documentation

5.51.1.1 cfg_t::cfg_t (unsigned int *ichannel,* unsigned int *numchannels*)

5.51.1.2 cfg_t::cfg_t (mhaconfig_t *chcfg,* mha_real_t *newlev,* bool *replace,* mha_real_t *len*)

5.51.1.3 cfg_t::cfg_t (mha_real_t *tau_attack,* mha_real_t *tau_decay,* unsigned int *nch,* mha_real_t *start_limit,* mha_real_t *slope_db,* mha_real_t *fs*)

5.51.2 Member Function Documentation

5.51.2.1 void cfg_t::process (mha_wave_t * s) [inline]

5.51.2.2 void cfg_t::process (mha_spec_t * s) [inline]

5.51.3 Member Data Documentation

5.51.3.1 unsigned int cfg_t::channel

5.51.3.2 mha_real_t cfg_t::gain_wave_ [private]

5.51.3.3 mha_real_t cfg_t::gain_spec_ [private]

5.51.3.4 bool cfg_t::replace_ [private]

5.51.3.5 bool cfg_t::use_frozen_ [private]

5.51.3.6 MHASignal::waveform_t cfg_t::frozen_noise_ [private]

5.51.3.7 unsigned int cfg_t::pos [private]

5.51.3.8 mha_real_t cfg_t::start_lin

5.51.3.9 mha_real_t cfg_t::alpha

5.51.3.10 MHAFilter::o1flt_lowpass_t cfg_t::attack

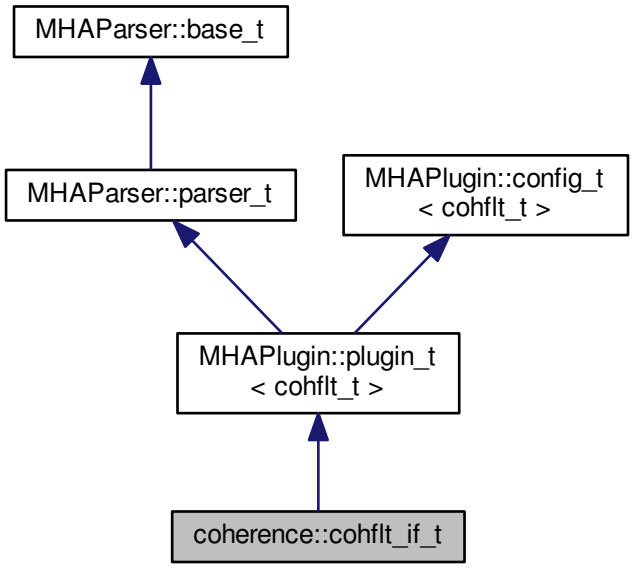
5.51.3.11 MHAFilter::o1flt_maxtrack_t cfg_t::decay

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

- **example6.cpp**
- **noise.cpp**
- **softclip.cpp**

5.52 coherence::cohflt_if_t Class Reference

Inheritance diagram for coherence::cohflt_if_t:



Public Member Functions

- **cohflt_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **cohflt_if_t** > **patchbay**
- **vars_t vars**
- const std::string **algo**

Additional Inherited Members

5.52.1 Constructor & Destructor Documentation

5.52.1.1 coherence::cohflt_if_t::cohflt_if_t (const algo_comm_t & *ac,* const std::string & *th,* const std::string & *al*)

5.52.2 Member Function Documentation

5.52.2.1 void coherence::cohflt_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< cohflt_t >** (p. 730).

5.52.2.2 void coherence::cohflt_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< cohflt_t >** (p. 731).

5.52.2.3 mha_spec_t * coherence::cohflt_if_t::process (mha_spec_t * *s*)

5.52.2.4 void coherence::cohflt_if_t::update ( ) [private]

5.52.3 Member Data Documentation

5.52.3.1 MHAEvents::patchbay_t<cohflt_if_t> coherence::cohflt_if_t::patchbay [private]

5.52.3.2 vars_t coherence::cohflt_if_t::vars [private]

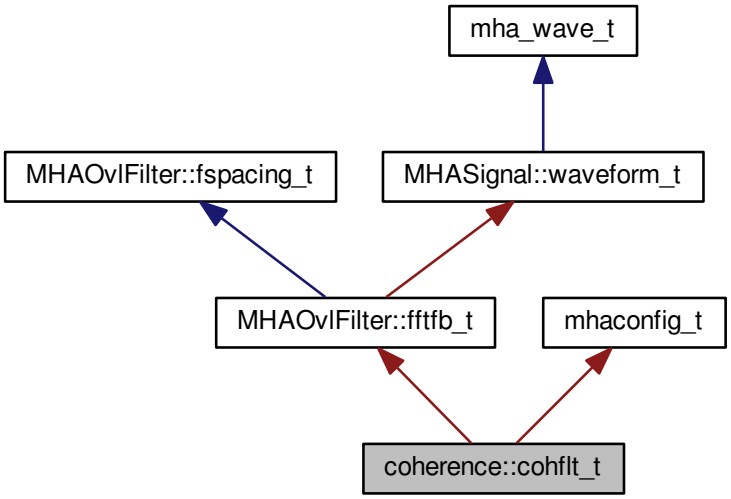
5.52.3.3 const std::string coherence::cohflt_if_t::algo [private]

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

- **coherence.cpp**

5.53 coherence::cohflt_t Class Reference

Inheritance diagram for coherence::cohflt_t:



Public Member Functions

- **cohflt_t** (**vars_t** &v, const **mhaconfig_t** &icf, **algo_comm_t** iac, const std::string &name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **insert** ()

Private Attributes

- unsigned int **nbands**
- bool **avg_ipd**
- **mha_complex_t cg**
- float **g**
- float **c_scale**
- float **c_min**
- **MHASignal::waveform_t alpha**
- float **limit**
- **MHAFilter::o1flt_lowpass_t lp1r**
- **MHAFilter::o1flt_lowpass_t lp1i**
- **MHA_AC::spectrum_t coh_c**
- **MHA_AC::waveform_t coh_rlp**
- **MHASignal::waveform_t gain**
- **MHASignal::delay_wave_t gain_delay**

- **MHASignal::spectrum_t s_out**
- bool **bInvert**
- **MHAFilter::o1flt_lowpass_t lp1ltg**
- bool **b_ltg**
- std::vector< float > **staticgain**

Additional Inherited Members

5.53.1 Constructor & Destructor Documentation

5.53.1.1 coherence::cohflt_t::cohflt_t (
vars_t & *v,*
const mhaconfig_t & *icf,*
algo_comm_t *iac,*
const std::string & *name*)

5.53.2 Member Function Documentation

5.53.2.1 mha_spec_t * coherence::cohflt_t::process (
mha_spec_t * *s*)

5.53.2.2 void coherence::cohflt_t::insert ( )

5.53.3 Member Data Documentation

5.53.3.1 unsigned int coherence::cohflt_t::nbands [private]

5.53.3.2 bool coherence::cohflt_t::avg_ipd [private]

5.53.3.3 mha_complex_t coherence::cohflt_t::cg [private]

5.53.3.4 float coherence::cohflt_t::g [private]

5.53.3.5 float coherence::cohflt_t::c_scale [private]

5.53.3.6 float coherence::cohflt_t::c_min [private]

5.53.3.7 MHASignal::waveform_t coherence::cohflt_t::alpha [private]

5.53.3.8 float coherence::cohflt_t::limit [private]

5.53.3.9 MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1r [private]

5.53.3.10 MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1i [private]

5.53.3.11 MHA_AC::spectrum_t coherence::cohflt_t::coh_c [private]

5.53.3.12 MHA_AC::waveform_t coherence::cohflt_t::coh_rlp [private]

5.53.3.13 MHASignal::waveform_t coherence::cohflt_t::gain [private]

5.53.3.14 MHASignal::delay_wave_t coherence::cohflt_t::gain_delay [private]

5.53.3.15 MHASignal::spectrum_t coherence::cohflt_t::s_out [private]

5.53.3.16 bool coherence::cohflt_t::bInvert [private]

5.53.3.17 MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1ltg [private]

5.53.3.18 bool coherence::cohflt_t::b_ltg [private]

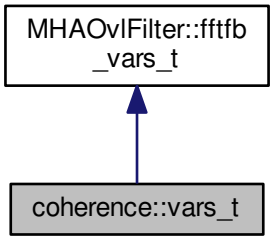
5.53.3.19 std::vector<float> coherence::cohflt_t::staticgain [private]

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

- **coherence.cpp**

5.54 coherence::vars_t Class Reference

Inheritance diagram for coherence::vars_t:



Public Member Functions

- **vars_t** (**MHAParser::parser_t** ∗)

Public Attributes

- **MHAParser::kw_t tau_unit**
- **MHAParser::vfloat_t tau**
- **MHAParser::vfloat_t alpha**
- **MHAParser::float_t limit**
- **MHAParser::vfloat_t mapping**
- **MHAParser::kw_t average**
- **MHAParser::bool_t invert**
- **MHAParser::bool_t ltgcomp**
- **MHAParser::vfloat_t ltgtau**
- **MHAParser::vfloat_t staticgain**
- **MHAParser::int_t delay**

5.54.1 Constructor & Destructor Documentation

5.54.1.1 coherence::vars_t::vars_t (MHAParser::parser_t * *p*)

5.54.2 Member Data Documentation

5.54.2.1 MHAParser::kw_t coherence::vars_t::tau_unit

5.54.2.2 MHAParser::vfloat_t coherence::vars_t::tau

5.54.2.3 MHAParser::vfloat_t coherence::vars_t::alpha

5.54.2.4 MHAParser::float_t coherence::vars_t::limit

5.54.2.5 MHAParser::vfloat_t coherence::vars_t::mapping

5.54.2.6 MHAParser::kw_t coherence::vars_t::average

5.54.2.7 MHAParser::bool_t coherence::vars_t::invert

5.54.2.8 MHAParser::bool_t coherence::vars_t::ltgcomp

5.54.2.9 MHAParser::vfloat_t coherence::vars_t::ltgtau

5.54.2.10 MHAParser::vfloat_t coherence::vars_t::staticgain

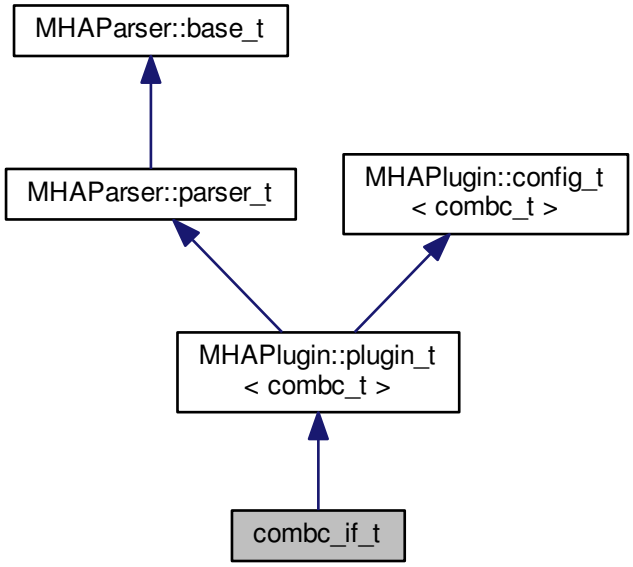
5.54.2.11 MHAParser::int_t coherence::vars_t::delay

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

- **coherence.cpp**

5.55 combc_if_t Class Reference

Inheritance diagram for combc_if_t:



Public Member Functions

- **combc_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- **MHAParser::int_t outchannels**
- **MHAParser::bool_t interleaved**
- **MHAParser::string_t channel_gain_name**
- **MHAParser::string_t element_gain_name**

Additional Inherited Members

5.55.1 Constructor & Destructor Documentation

5.55.1.1 combc_if_t::combc_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.55.2 Member Function Documentation

5.55.2.1 void combc_if_t::prepare (mhaconfig_t & *chcfg*) [virtual]

Implements **MHAPlugin::plugin_t**< **combc_t** > (p. 730).

5.55.2.2 mha_wave_t * combc_if_t::process (mha_wave_t * *s*)

5.55.2.3 mha_spec_t * combc_if_t::process (mha_spec_t * *s*)

5.55.3 Member Data Documentation

5.55.3.1 MHAParser::int_t combc_if_t::outchannels [private]

5.55.3.2 MHAParser::bool_t combc_if_t::interleaved [private]

5.55.3.3 MHAParser::string_t combc_if_t::channel_gain_name [private]

5.55.3.4 MHAParser::string_t combc_if_t::element_gain_name [private]

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

- **combinechannels.cpp**

5.56 combc_t Class Reference

Public Member Functions

- **combc_t** (**algo_comm_t** ac, **mhaconfig_t** cfg_input, **mhaconfig_t** cfg_output, std↩ ::vector< float > channel_gains, const std::string &element_gain_name, bool interleaved)
- **mha_wave_t** * **process** (**mha_wave_t** *s)
- **mha_spec_t** * **process** (**mha_spec_t** *s)

Private Attributes

- **algo_comm_t ac_**
- bool **interleaved_**
- unsigned int **nbands**
- **MHASignal::waveform_t w_out**
- **MHASignal::spectrum_t s_out**
- std::vector< **mha_real_t** > **channel_gains_**
- std::string **element_gain_name_**

5.56.1 Constructor & Destructor Documentation

5.56.1.1 combc_t::combc_t (

algo_comm_t *ac,*
mhaconfig_t *cfg_input,*
mhaconfig_t *cfg_output,*
std::vector< float > *channel_gains,*
const std::string & *element_gain_name,*
bool *interleaved*)

5.56.2 Member Function Documentation

5.56.2.1 mha_wave_t * combc_t::process (

mha_wave_t * *s*)

5.56.2.2 mha_spec_t * combc_t::process (

mha_spec_t * *s*)

5.56.3 Member Data Documentation

5.56.3.1 algo_comm_t combc_t::ac_ [private]

5.56.3.2 bool combc_t::interleaved_ [private]

5.56.3.3 unsigned int combc_t::nbands [private]

5.56.3.4 MHASignal::waveform_t combc_t::w_out [private]

5.56.3.5 MHASignal::spectrum_t combc_t::s_out [private]

5.56.3.6 std::vector<mha_real_t> combc_t::channel_gains_ [private]

5.56.3.7 std::string combc_t::element_gain_name_ [private]

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

- **combinechannels.cpp**

5.57 comm_var_t Struct Reference

Algorithm communication variable structure.

Public Attributes

- unsigned int **data_type**
  Type of data.
- unsigned int **num_entries**
  Number of entries.
- unsigned int **stride**
  length of one row (C interpretation) or of one column (Fortran interpretation)
- void * **data**
  Pointer to variable data.

5.57.1 Detailed Description

Algorithm communication variable structure.

Algorithm communication variables (AC variables) are objects of this type. The member data is a pointer to the variable ‘data'. This pointer has to be valid for the lifetime of this AC variable. The member ‘data_type' can be one of the predefined types or any user defined type. The member ‘num_entries' describes the number of elements of this base type stored at the pointer address.

```
An AC variable can be registered with the \ref
algo_comm_t::insert_var "insert_var" function.
```

5.57.2 Member Data Documentation

5.57.2.1 comm_var_t::data_type

Type of data.

This can be one of the predefined types

- MHA_AC_CHAR
- MHA_AC_INT
- MHA_AC_MHAREAL
- MHA_AC_FLOAT
- MHA_AC_DOUBLE
- MHA_AC_MHACOMPLEX
- MHA_AC_VEC_FLOAT or any user defined type with a value greater than
- MHA_AC_USER

5.57.2.2 comm_var_t::num_entries

Number of entries.

5.57.2.3 comm_var_t::stride

length of one row (C interpretation) or of one column (Fortran interpretation)

5.57.2.4 comm_var_t::data

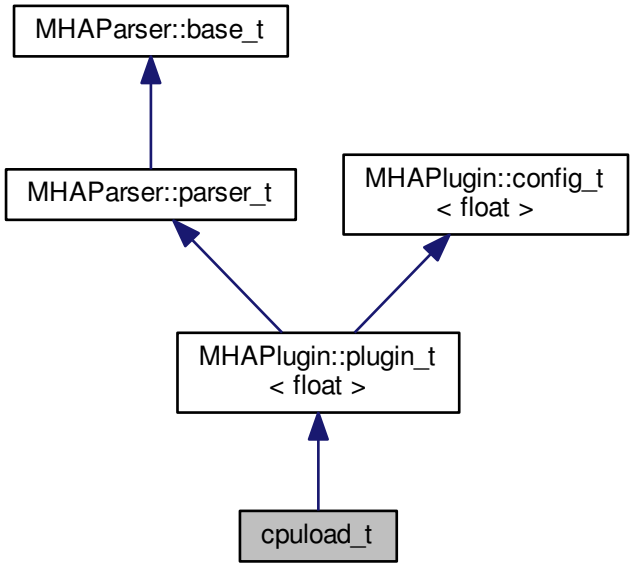
Pointer to variable data.

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

- **mha.hh**
- **mha_algo_comm.cpp**

5.58 cpuload_t Class Reference

Inheritance diagram for cpuload_t:



Public Member Functions

- **cpuload_t** (**algo_comm_t**, const char *, const char *)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **compute_something** ()
- void **compute_something_else** ()

Private Attributes

- **MHAParser::float_t factor**
- **MHAParser::bool_t use_sine**
- float **phase**
- volatile float **result**
- std::vector< float > **table**

Additional Inherited Members

5.58.1 Constructor & Destructor Documentation

5.58.1.1 cpuload_t::cpuload_t (algo_comm_t *iac,* const char * , const char *)

5.58.2 Member Function Documentation

5.58.2.1 mha_spec_t * cpuload_t::process (mha_spec_t * *s*)

5.58.2.2 mha_wave_t * cpuload_t::process (mha_wave_t * *s*)

5.58.2.3 void cpuload_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< float >** (p. 730).

5.58.2.4 void cpuload_t::compute_something ( ) [inline], [private]

5.58.2.5 void cpuload_t::compute_something_else ( ) [inline], [private]

5.58.3 Member Data Documentation

5.58.3.1 MHAParser::float_t cpuload_t::factor [private]

5.58.3.2 MHAParser::bool_t cpuload_t::use_sine [private]

5.58.3.3 float cpuload_t::phase [private]

5.58.3.4 volatile float cpuload_t::result [private]

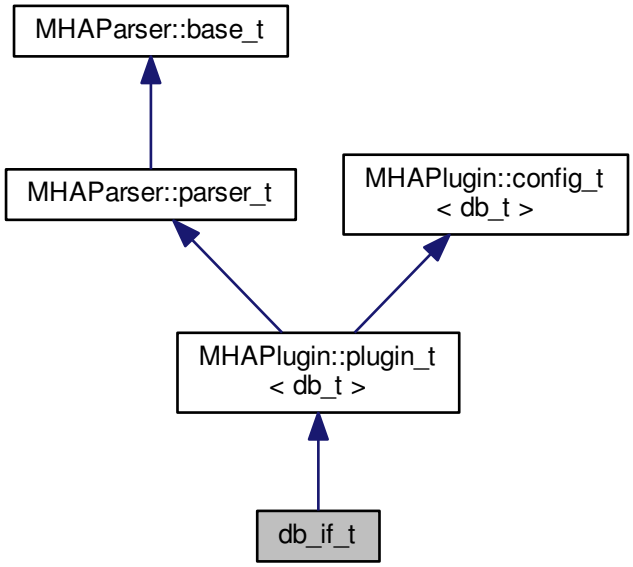
5.58.3.5 std::vector<float> cpuload_t::table [private]

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

- **cpuload.cpp**

5.59 db_if_t Class Reference

Inheritance diagram for db_if_t:



Public Member Functions

- **db_if_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- ∼**db_if_t** ()

Private Attributes

- **MHAEvents::patchbay_t**< **db_if_t** > **patchbay**
- **MHAParser::int_t fragsize**
- **MHAParser::mhapluginloader_t plugloader**
- std::string **chain**
- std::string **algo**
- bool **bypass**

Additional Inherited Members

5.59.1 Constructor & Destructor Documentation

5.59.1.1 db_if_t::db_if_t (algo_comm_t *iac,* std::string *th,* std::string *al*)

5.59.1.2 db_if_t::∼db_if_t ( )

5.59.2 Member Function Documentation

5.59.2.1 mha_wave_t ∗ db_if_t::process (mha_wave_t ∗ *s*)

5.59.2.2 void db_if_t::prepare (mhaconfig_t & *conf*) [virtual]

Implements **MHAPlugin::plugin_t**< **db_t** > (p. 730).

5.59.2.3 void db_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **db_t** > (p. 731).

5.59.3 Member Data Documentation

5.59.3.1 MHAEvents::patchbay_t< db_if_t > db_if_t::patchbay [private]

5.59.3.2 MHAParser::int_t db_if_t::fragsize [private]

5.59.3.3 MHAParser::mhapluginloader_t db_if_t::plugloader [private]

5.59.3.4 std::string db_if_t::chain [private]

5.59.3.5 std::string db_if_t::algo [private]

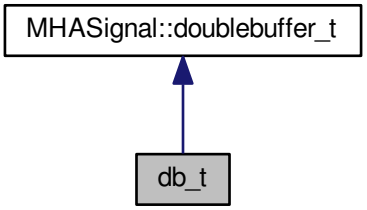
5.59.3.6 bool db_if_t::bypass [private]

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

- **db.cpp**

5.60 db_t Class Reference

Inheritance diagram for db_t:



Public Member Functions

- **db_t** (unsigned int outer_fragsize, unsigned int inner_fragsize, unsigned int nch_in, unsigned int nch_out, **MHAParser::mhapluginloader_t** &plug)
- **mha_wave_t** ∗ **inner_process** (**mha_wave_t** ∗)

Private Attributes

- **MHAParser::mhapluginloader_t** & **plugloader**

Additional Inherited Members

5.60.1 Constructor & Destructor Documentation

5.60.1.1 db_t::db_t (

unsigned int *outer_fragsize,*
unsigned int *inner_fragsize,*
unsigned int *nch_in,*
unsigned int *nch_out,*
MHAParser::mhapluginloader_t & *plug*)

5.60.2 Member Function Documentation

5.60.2.1 mha_wave_t * db_t::inner_process (

mha_wave_t * *s*) [virtual]

Implements **MHASignal::doublebuffer_t** (p. 771).

5.60.3 Member Data Documentation

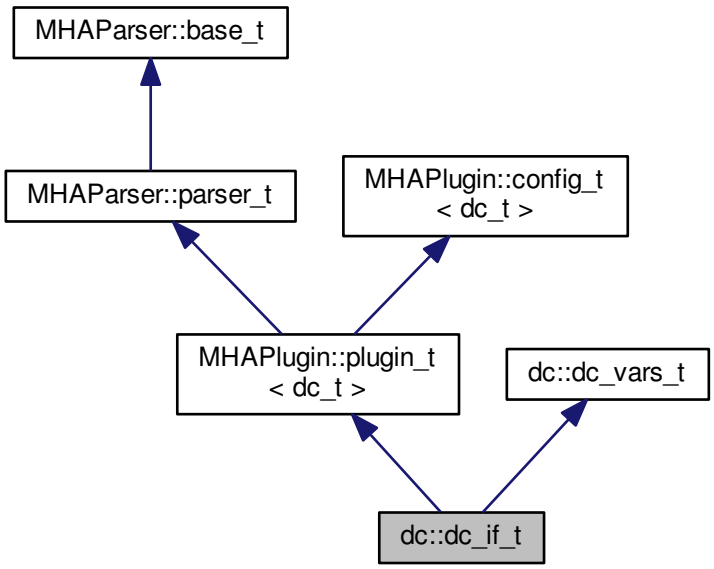
5.60.3.1 MHAParser::mhapluginloader_t& db_t::plugloader [private]

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

- **db.cpp**

5.61 dc::dc_if_t Class Reference

Inheritance diagram for dc::dc_if_t:



Public Member Functions

- **dc_if_t** (const **algo_comm_t** &ac_, const std::string &th_, const std::string &al_)
- void **prepare** (**mhaconfig_t** &tf)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update_monitors** ()

  Called from within the processing routines: updates the monitor variables.
- void **update** ()

  Called by MHA configuration change event mechanism: creates new runtime configuration.

Private Attributes

- std::string **algo**
- **wideband_inhib_vars_t wbinhib**
- **MHAEvents::patchbay_t**< **dc_if_t** > **patchbay**

Additional Inherited Members

5.61.1 Constructor & Destructor Documentation

5.61.1.1 dc::dc_if_t::dc_if_t (const algo_comm_t & *ac_*, const std::string & *th_*, const std::string & *al_*)

5.61.2 Member Function Documentation

5.61.2.1 void dc::dc_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< dc_t >** (p. 730).

5.61.2.2 mha_wave_t * dc::dc_if_t::process (mha_wave_t * *s_in*)

5.61.2.3 mha_spec_t * dc::dc_if_t::process (mha_spec_t * *s_in*)

5.61.2.4 void dc::dc_if_t::update_monitors ( ) [private]

Called from within the processing routines: updates the monitor variables.

5.61.2.5 void dc::dc_if_t::update ( ) [private]

Called by MHA configuration change event mechanism: creates new runtime configuration.

5.61.3 Member Data Documentation

5.61.3.1 std::string dc::dc_if_t::algo [private]

5.61.3.2 wideband_inhib_vars_t dc::dc_if_t::wbinhib [private]

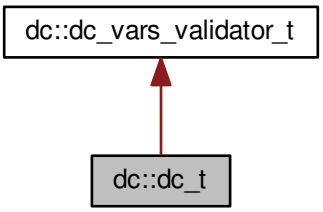
5.61.3.3 MHAEvents::patchbay_t<dc_if_t> dc::dc_if_t::patchbay [private]

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

- **dc.cpp**

5.62 dc::dc_t Class Reference

Inheritance diagram for dc::dc_t:



Public Member Functions

- **dc_t** (**dc_vars_t** vars, **mha_real_t** filter_rate, unsigned int nch, **algo_comm_t** ac, **mha_domain_t** domain, unsigned int fftlen, std::string algo)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *, **wb_inhib_cfg_t** *wbinhib)
- void **explicit_insert** ()
- unsigned **get_nbands** () const

  Number of frequency bands accessor.
- const **MHASignal::waveform_t** & **get_level_in_db** () const
- const **MHASignal::waveform_t** & **get_level_in_db_adjusted** () const

Private Attributes

- std::vector< **MHATableLookup::linear_table_t** > **gt**
- **MHAFilter::o1flt_lowpass_t rmslevel**
- **MHAFilter::o1flt_lowpass_t attack**
- **MHAFilter::o1flt_maxtrack_t decay**
- bool **powersum**
- bool **bypass**
- unsigned int **naudiochannels**
- unsigned int **nbands**
- **MHA_AC::waveform_t level_in_db**
- **MHA_AC::waveform_t level_in_db_adjusted**
- **MHA_AC::waveform_t inhib_gain**
- **MHASignal::waveform_t max_level_difference**
- unsigned int **k_nyquist**

Additional Inherited Members

5.62.1 Constructor & Destructor Documentation

5.62.1.1 dc::dc_t::dc_t (

dc_vars_t *vars,*
mha_real_t *filter_rate,*
unsigned int *nch,*
algo_comm_t *ac,*
mha_domain_t *domain,*
unsigned int *fftlen,*
std::string *algo*)

5.62.2 Member Function Documentation

5.62.2.1 mha_wave_t * dc::dc_t::process (

mha_wave_t * *s*)

5.62.2.2 mha_spec_t * dc::dc_t::process (

mha_spec_t * *s,*
wb_inhib_cfg_t * *wbinhib*)

5.62.2.3 void dc::dc_t::explicit_insert ( )

5.62.2.4 unsigned dc::dc_t::get_nbands ( ) const [inline]

Number of frequency bands accessor.

5.62.2.5 const MHASignal::waveform_t& dc::dc_t::get_level_in_db ( ) const [inline]

5.62.2.6 const MHASignal::waveform_t& dc::dc_t::get_level_in_db_adjusted ( ) const [inline]

5.62.3 Member Data Documentation

5.62.3.1 std::vector<MHATableLookup::linear_table_t> dc::dc_t::gt [private]

5.62.3.2 MHAFilter::o1flt_lowpass_t dc::dc_t::rmslevel [private]

5.62.3.3 MHAFilter::o1flt_lowpass_t dc::dc_t::attack [private]

5.62.3.4 MHAFilter::o1flt_maxtrack_t dc::dc_t::decay [private]

5.62.3.5 bool dc::dc_t::powersum [private]

5.62.3.6 bool dc::dc_t::bypass [private]

5.62.3.7 unsigned int dc::dc_t::naudiochannels [private]

5.62.3.8 unsigned int dc::dc_t::nbands [private]

5.62.3.9 MHA_AC::waveform_t dc::dc_t::level_in_db [private]

5.62.3.10 MHA_AC::waveform_t dc::dc_t::level_in_db_adjusted [private]

5.62.3.11 MHA_AC::waveform_t dc::dc_t::inhib_gain [private]

5.62.3.12 MHASignal::waveform_t dc::dc_t::max_level_difference [private]

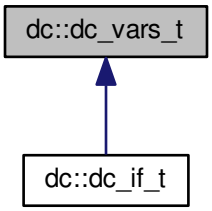
5.62.3.13 unsigned int dc::dc_t::k_nyquist [private]

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

- **dc.cpp**

5.63 dc::dc_vars_t Class Reference

Inheritance diagram for dc::dc_vars_t:



Public Member Functions

- **dc_vars_t** (**MHAParser::parser_t** &)

Public Attributes

- **MHAParser::bool_t powersum**
- **MHAParser::mfloat_t gtdata**
- **MHAParser::vfloat_t gtmin**
- **MHAParser::vfloat_t gtstep**
- **MHAParser::vfloat_t taurmslevel**
- **MHAParser::vfloat_t tauattack**
- **MHAParser::vfloat_t taudecay**
- **MHAParser::string_t filterbank**
- std::string **cf_name**
- std::string **ef_name**
- std::string **bw_name**
- **MHAParser::string_t chname**
- **MHAParser::bool_t bypass**
- **MHAParser::string_t clientid**
- **MHAParser::string_t gainrule**
- **MHAParser::string_t preset**
- **MHAParser::int_mon_t modified**
- **MHAParser::mfloat_t max_level_difference**
- **MHAParser::vfloat_mon_t input_level**
- **MHAParser::vfloat_mon_t filtered_level**
- **MHAParser::vfloat_mon_t center_frequencies**
- **MHAParser::vfloat_mon_t edge_frequencies**
- **MHAParser::vfloat_mon_t band_weights**
- **MHAParser::bool_t use_wbinhib**

5.63.1 Constructor & Destructor Documentation

5.63.1.1 dc::dc_vars_t::dc_vars_t (MHAParser::parser_t & *p*)

5.63.2 Member Data Documentation

5.63.2.1 MHAParser::bool_t dc::dc_vars_t::powersum

5.63.2.2 MHAParser::mfloat_t dc::dc_vars_t::gtdata

5.63.2.3 MHAParser::vfloat_t dc::dc_vars_t::gtmin

5.63.2.4 MHAParser::vfloat_t dc::dc_vars_t::gtstep

5.63.2.5 MHAParser::vfloat_t dc::dc_vars_t::taurmslevel

5.63.2.6 MHAParser::vfloat_t dc::dc_vars_t::tauattack

5.63.2.7 MHAParser::vfloat_t dc::dc_vars_t::taudecay

5.63.2.8 MHAParser::string_t dc::dc_vars_t::filterbank

5.63.2.9 std::string dc::dc_vars_t::cf_name

5.63.2.10 std::string dc::dc_vars_t::ef_name

5.63.2.11 std::string dc::dc_vars_t::bw_name

5.63.2.12 MHAParser::string_t dc::dc_vars_t::chname

5.63.2.13 MHAParser::bool_t dc::dc_vars_t::bypass

5.63.2.14 MHAParser::string_t dc::dc_vars_t::clientid

5.63.2.15 MHAParser::string_t dc::dc_vars_t::gainrule

5.63.2.16 MHAParser::string_t dc::dc_vars_t::preset

5.63.2.17 MHAParser::int_mon_t dc::dc_vars_t::modified

5.63.2.18 MHAParser::mfloat_t dc::dc_vars_t::max_level_difference

5.63.2.19 MHAParser::vfloat_mon_t dc::dc_vars_t::input_level

5.63.2.20 MHAParser::vfloat_mon_t dc::dc_vars_t::filtered_level

5.63.2.21 MHAParser::vfloat_mon_t dc::dc_vars_t::center_frequencies

5.63.2.22 MHAParser::vfloat_mon_t dc::dc_vars_t::edge_frequencies

5.63.2.23 MHAParser::vfloat_mon_t dc::dc_vars_t::band_weights

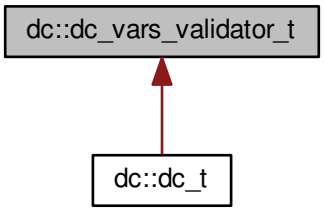
5.63.2.24 MHAParser::bool_t dc::dc_vars_t::use_wbinhib

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

- **dc.cpp**

5.64 dc::dc_vars_validator_t Class Reference

Inheritance diagram for dc::dc_vars_validator_t:



Public Member Functions

- **dc_vars_validator_t** (**dc_vars_t** &v, unsigned int s, **mha_domain_t** domain)

5.64.1 Constructor & Destructor Documentation

5.64.1.1 dc::dc_vars_validator_t::dc_vars_validator_t (dc_vars_t & *v*, unsigned int *s*, mha_domain_t *domain*)

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

- **dc.cpp**

5.65 dc::wb_inhib_cfg_t Class Reference

Public Member Functions

- **wb_inhib_cfg_t** (const **wideband_inhib_vars_t** &vars)

Public Attributes

- std::vector< float > **weights**
- float **dl_map_min**
- float **dl_map_max**
- float **dl_diff**
- float **l_min**
- std::vector< std::vector< float > > **g_scale**

5.65.1 Constructor & Destructor Documentation

5.65.1.1 dc::wb_inhib_cfg_t::wb_inhib_cfg_t (const wideband_inhib_vars_t & *vars*)

5.65.2 Member Data Documentation

5.65.2.1 std::vector<float> dc::wb_inhib_cfg_t::weights

5.65.2.2 float dc::wb_inhib_cfg_t::dl_map_min

5.65.2.3 float dc::wb_inhib_cfg_t::dl_map_max

5.65.2.4 float dc::wb_inhib_cfg_t::dl_diff

5.65.2.5 float dc::wb_inhib_cfg_t::l_min

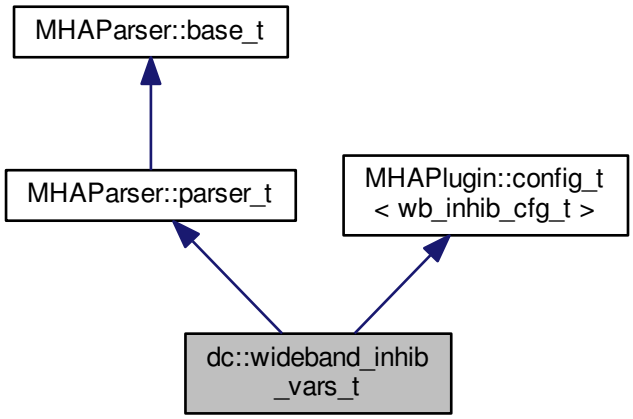
5.65.2.6 std::vector<std::vector<float> > dc::wb_inhib_cfg_t::g_scale

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

- **dc.cpp**

5.66 dc::wideband_inhib_vars_t Class Reference

Inheritance diagram for dc::wideband_inhib_vars_t:



Public Member Functions

- **wideband_inhib_vars_t** ()
- void **setchannels** (unsigned int ch, unsigned int bnds)
- **wb_inhib_cfg_t** ∗ **current** ()
- void **update** ()

Public Attributes

- **MHAParser::vfloat_t weights**
- **MHAParser::float_t dl_map_min**
- **MHAParser::float_t dl_map_max**
- **MHAParser::float_t l_min**
- **MHAParser::mfloat_t g_scale**
- **MHAEvents::patchbay_t**< **wideband_inhib_vars_t** > **patchbay**
- unsigned int **channels**
- unsigned int **bands**

Additional Inherited Members

5.66.1 Constructor & Destructor Documentation

5.66.1.1 dc::wideband_inhib_vars_t::wideband_inhib_vars_t ( )

5.66.2 Member Function Documentation

5.66.2.1 void dc::wideband_inhib_vars_t::setchannels (unsigned int *ch,* unsigned int *bnds*) [inline]

5.66.2.2 wb_inhib_cfg_t∗ dc::wideband_inhib_vars_t::current ( ) [inline]

5.66.2.3 void dc::wideband_inhib_vars_t::update ( )

5.66.3 Member Data Documentation

5.66.3.1 MHAParser::vfloat_t dc::wideband_inhib_vars_t::weights

5.66.3.2 MHAParser::float_t dc::wideband_inhib_vars_t::dl_map_min

5.66.3.3 MHAParser::float_t dc::wideband_inhib_vars_t::dl_map_max

5.66.3.4 MHAParser::float_t dc::wideband_inhib_vars_t::l_min

5.66.3.5 MHAParser::mfloat_t dc::wideband_inhib_vars_t::g_scale

5.66.3.6 MHAEvents::patchbay_t<wideband_inhib_vars_t> dc::wideband_inhib_vars_t::patchbay

5.66.3.7 unsigned int dc::wideband_inhib_vars_t::channels

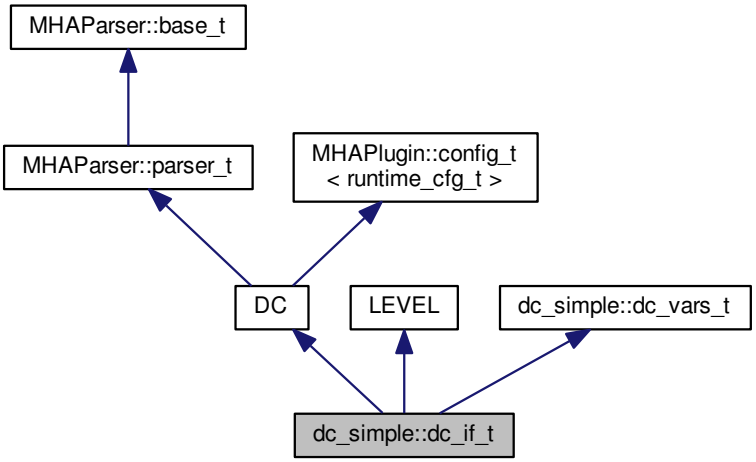
5.66.3.8 unsigned int dc::wideband_inhib_vars_t::bands

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

- **dc.cpp**

5.67 dc_simple::dc_if_t Class Reference

Inheritance diagram for dc_simple::dc_if_t:



Public Member Functions

- **dc_if_t** (const **algo_comm_t** &ac_, const std::string &th_, const std::string &al_)
- void **prepare** (**mhaconfig_t** &tf)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update_dc** ()
- void **update_level** ()
- void **has_been_modified** ()
- void **read_modified** ()
- void **update_level_mon** ()
- void **update_gain_mon** ()

Private Attributes

- **MHAParser::string_t clientid**
- **MHAParser::string_t gainrule**
- **MHAParser::string_t preset**
- **MHAParser::int_mon_t modified**
- **MHAParser::vfloat_mon_t mon_l**
- **MHAParser::vfloat_mon_t mon_g**
- **MHAParser::string_t filterbank**
- **MHAParser::vfloat_mon_t center_frequencies**
- **MHAParser::vfloat_mon_t edge_frequencies**
- **MHAEvents::patchbay_t**< **dc_if_t** > **patchbay**
- bool **prepared**

Additional Inherited Members

5.67.1 Constructor & Destructor Documentation

5.67.1.1 dc_simple::dc_if_t::dc_if_t (

const **algo_comm_t** & *ac_*,
const **std::string** & *th_*,
const **std::string** & *al_*)

5.67.2 Member Function Documentation

5.67.2.1 void dc_simple::dc_if_t::prepare (

mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 730).

5.67.2.2 void dc_simple::dc_if_t::release (

void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 731).

5.67.2.3 mha_spec_t * dc_simple::dc_if_t::process (mha_spec_t * *s*)

5.67.2.4 mha_wave_t * dc_simple::dc_if_t::process (mha_wave_t * *s*)

5.67.2.5 void dc_simple::dc_if_t::update_dc ( ) [private]

5.67.2.6 void dc_simple::dc_if_t::update_level ( ) [private]

5.67.2.7 void dc_simple::dc_if_t::has_been_modified ( ) [inline], [private]

5.67.2.8 void dc_simple::dc_if_t::read_modified ( ) [inline], [private]

5.67.2.9 void dc_simple::dc_if_t::update_level_mon ( ) [private]

5.67.2.10 void dc_simple::dc_if_t::update_gain_mon ( ) [private]

5.67.3 Member Data Documentation

5.67.3.1 MHAParser::string_t dc_simple::dc_if_t::clientid [private]

5.67.3.2 MHAParser::string_t dc_simple::dc_if_t::gainrule [private]

5.67.3.3 MHAParser::string_t dc_simple::dc_if_t::preset [private]

5.67.3.4 MHAParser::int_mon_t dc_simple::dc_if_t::modified [private]

5.67.3.5 MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_l [private]

5.67.3.6 MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_g [private]

5.67.3.7 MHAParser::string_t dc_simple::dc_if_t::filterbank [private]

5.67.3.8 MHAParser::vfloat_mon_t dc_simple::dc_if_t::center_frequencies [private]

5.67.3.9 MHAParser::vfloat_mon_t dc_simple::dc_if_t::edge_frequencies [private]

5.67.3.10 MHAEvents::patchbay_t<dc_if_t> dc_simple::dc_if_t::patchbay [private]

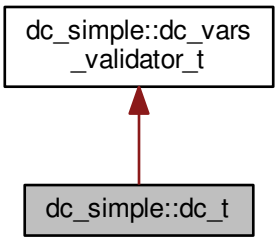
5.67.3.11 bool dc_simple::dc_if_t::prepared [private]

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

- **dc_simple.cpp**

5.68 dc_simple::dc_t Class Reference

Inheritance diagram for dc_simple::dc_t:



Classes

- class **line_t**

Public Member Functions

- **dc_t** (const **dc_vars_t** &vars, **mha_real_t** filter_rate, unsigned int nch, unsigned int fftlen)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗, **mha_wave_t** ∗level_db)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗, **mha_wave_t** ∗level_db)

Public Attributes

- std::vector< float > **mon_l**
- std::vector< float > **mon_g**

Private Attributes

- std::vector< **mha_real_t** > **expansion_threshold**
- std::vector< **mha_real_t** > **limiter_threshold**
- std::vector< **line_t** > **compression**
- std::vector< **line_t** > **expansion**
- std::vector< **line_t** > **limiter**
- std::vector< **mha_real_t** > **maxgain**
- unsigned int **nbands**

Additional Inherited Members

5.68.1 Constructor & Destructor Documentation

5.68.1.1 dc_simple::dc_t::dc_t (
const dc_vars_t & *vars,*
mha_real_t *filter_rate,*
unsigned int *nch,*
unsigned int *fftlen*)

5.68.2 Member Function Documentation

5.68.2.1 mha_spec_t * dc_simple::dc_t::process (
mha_spec_t * *s,*
mha_wave_t * *level_db*)

5.68.2.2 mha_wave_t * dc_simple::dc_t::process (
mha_wave_t * *s,*
mha_wave_t * *level_db*)

5.68.3 Member Data Documentation

5.68.3.1 std::vector<mha_real_t> dc_simple::dc_t::expansion_threshold [private]

5.68.3.2 std::vector<mha_real_t> dc_simple::dc_t::limiter_threshold [private]

5.68.3.3 std::vector<line_t> dc_simple::dc_t::compression [private]

5.68.3.4 std::vector<line_t> dc_simple::dc_t::expansion [private]

5.68.3.5 std::vector<line_t> dc_simple::dc_t::limiter [private]

5.68.3.6 std::vector<mha_real_t> dc_simple::dc_t::maxgain [private]

5.68.3.7 unsigned int dc_simple::dc_t::nbands [private]

5.68.3.8 std::vector<float> dc_simple::dc_t::mon_l

5.68.3.9 std::vector<float> dc_simple::dc_t::mon_g

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

- **dc_simple.cpp**

5.69 dc_simple::dc_t::line_t Class Reference

Public Member Functions

- **line_t** (**mha_real_t** x1, **mha_real_t** y1, **mha_real_t** x2, **mha_real_t** y2)
- **line_t** (**mha_real_t** x1, **mha_real_t** y1, **mha_real_t** slope)
- **mha_real_t operator()** (**mha_real_t** x)

Private Attributes

- **mha_real_t m**
- **mha_real_t y0**

5.69.1 Constructor & Destructor Documentation

5.69.1.1 dc_simple::dc_t::line_t::line_t (
mha_real_t *x1,*
mha_real_t *y1,*
mha_real_t *x2,*
mha_real_t *y2*)

5.69.1.2 dc_simple::dc_t::line_t::line_t (
mha_real_t *x1,*
mha_real_t *y1,*
mha_real_t *slope*)

5.69.2 Member Function Documentation

5.69.2.1 mha_real_t dc_simple::dc_t::line_t::operator() (
mha_real_t *x*) [inline]

5.69.3 Member Data Documentation

5.69.3.1 mha_real_t dc_simple::dc_t::line_t::m [private]

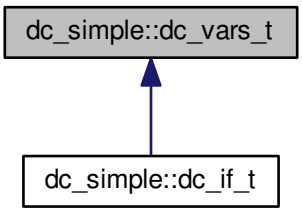
5.69.3.2 mha_real_t dc_simple::dc_t::line_t::y0 [private]

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

- **dc_simple.cpp**

5.70 dc_simple::dc_vars_t Class Reference

Inheritance diagram for dc_simple::dc_vars_t:



Public Member Functions

- **dc_vars_t** (**MHAParser::parser_t** &)

Public Attributes

- **MHAParser::vfloat_t g50**
- **MHAParser::vfloat_t g80**
- **MHAParser::vfloat_t maxgain**
- **MHAParser::vfloat_t expansion_threshold**
- **MHAParser::vfloat_t expansion_slope**
- **MHAParser::vfloat_t limiter_threshold**
- **MHAParser::vfloat_t tauattack**
- **MHAParser::vfloat_t taudecay**
- **MHAParser::bool_t bypass**

5.70.1 Constructor & Destructor Documentation

5.70.1.1 dc_simple::dc_vars_t::dc_vars_t (MHAParser::parser_t & *p*)

5.70.2 Member Data Documentation

5.70.2.1 MHAParser::vfloat_t dc_simple::dc_vars_t::g50

5.70.2.2 MHAParser::vfloat_t dc_simple::dc_vars_t::g80

5.70.2.3 MHAParser::vfloat_t dc_simple::dc_vars_t::maxgain

5.70.2.4 MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_threshold

5.70.2.5 MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_slope

5.70.2.6 MHAParser::vfloat_t dc_simple::dc_vars_t::limiter_threshold

5.70.2.7 MHAParser::vfloat_t dc_simple::dc_vars_t::tauattack

5.70.2.8 MHAParser::vfloat_t dc_simple::dc_vars_t::taudecay

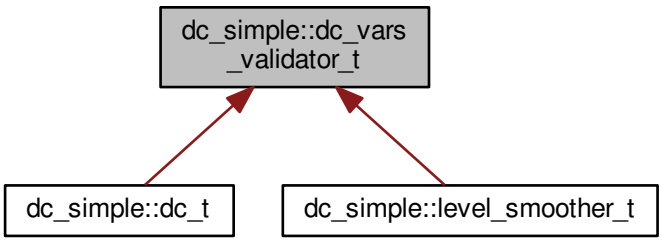
5.70.2.9 MHAParser::bool_t dc_simple::dc_vars_t::bypass

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

- **dc_simple.cpp**

5.71 dc_simple::dc_vars_validator_t Class Reference

Inheritance diagram for dc_simple::dc_vars_validator_t:



Public Member Functions

- **dc_vars_validator_t** (const **dc_vars_t** &v, unsigned int s)

5.71.1 Constructor & Destructor Documentation

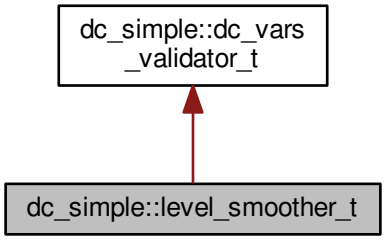
5.71.1.1 dc_simple::dc_vars_validator_t::dc_vars_validator_t (const dc_vars_t & *v*, unsigned int *s*)

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

- **dc_simple.cpp**

5.72 dc_simple::level_smoother_t Class Reference

Inheritance diagram for dc_simple::level_smoother_t:



Public Member Functions

- **level_smoother_t** (const **dc_vars_t** &vars, **mha_real_t** filter_rate, **mhaconfig_t** buscfg)
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- **MHAFilter::o1flt_lowpass_t attack**
- **MHAFilter::o1flt_maxtrack_t decay**
- unsigned int **nbands**
- unsigned int **fftlen**
- **MHASignal::waveform_t level_wave**
- **MHASignal::waveform_t level_spec**

Additional Inherited Members

5.72.1 Constructor & Destructor Documentation

5.72.1.1 dc_simple::level_smoother_t::level_smoother_t (const dc_vars_t & *vars,* mha_real_t *filter_rate,* mhaconfig_t *buscfg*)

5.72.2 Member Function Documentation

5.72.2.1 mha_wave_t * dc_simple::level_smoother_t::process (mha_spec_t * *s*)

5.72.2.2 mha_wave_t * dc_simple::level_smoother_t::process (mha_wave_t * *s*)

5.72.3 Member Data Documentation

5.72.3.1 MHAFilter::o1flt_lowpass_t dc_simple::level_smoother_t::attack [private]

5.72.3.2 MHAFilter::o1flt_maxtrack_t dc_simple::level_smoother_t::decay [private]

5.72.3.3 unsigned int dc_simple::level_smoother_t::nbands [private]

5.72.3.4 unsigned int dc_simple::level_smoother_t::fftlen [private]

5.72.3.5 MHASignal::waveform_t dc_simple::level_smoother_t::level_wave [private]

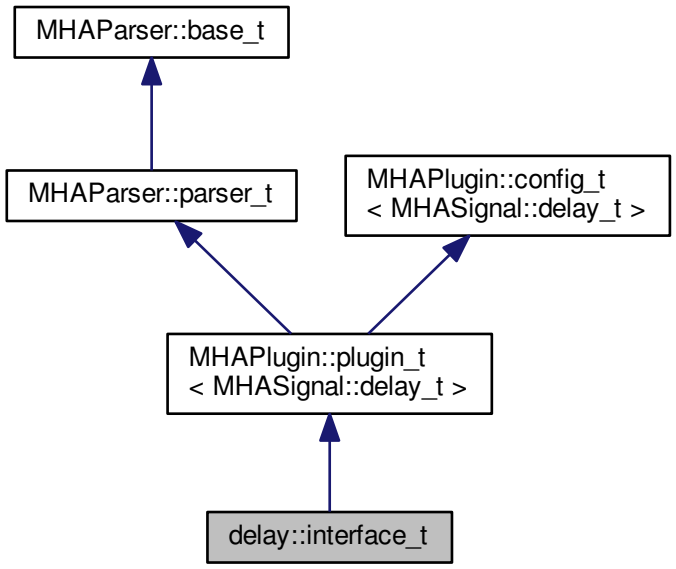
5.72.3.6 MHASignal::waveform_t dc_simple::level_smoother_t::level_spec [private]

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

- **dc_simple.cpp**

5.73 delay::interface_t Class Reference

Inheritance diagram for delay::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::vint_t delays**
- **MHAEvents::patchbay_t**< **interface_t** > **patchbay**

Additional Inherited Members

5.73.1 Constructor & Destructor Documentation

5.73.1.1 delay::interface_t::interface_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.73.2 Member Function Documentation

5.73.2.1 void delay::interface_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< MHASignal::delay_t >** (p. 730).

5.73.2.2 mha_wave_t * delay::interface_t::process (mha_wave_t * *s*)

5.73.2.3 void delay::interface_t::update ( ) [private]

5.73.3 Member Data Documentation

5.73.3.1 MHAParser::vint_t delay::interface_t::delays [private]

5.73.3.2 MHAEvents::patchbay_t<interface_t> delay::interface_t::patchbay [private]

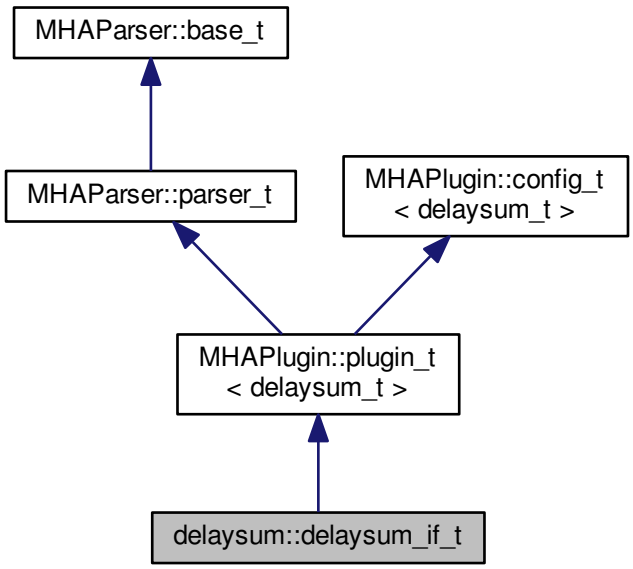
The documentation for this class was generated from the following file:

- **delay.cpp**

5.74 delaysum::delaysum_if_t Class Reference

Interface class for the delaysum plugin.

Inheritance diagram for delaysum::delaysum_if_t:



Public Member Functions

- **delaysum_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::vfloat_t weights**

  Linear weights to be multiplied with the audio signal, one factor for each channel.
- **MHAParser::vint_t delay**

  vector of channel-specific delays, in samples.
- **MHAEvents::patchbay_t**< **delaysum_if_t** > **patchbay**

  The patchbay to react to config changes.

Additional Inherited Members

5.74.1 Detailed Description

Interface class for the delaysum plugin.

This plugin allows to delay and sum multiple input channels using individual delays and weights. After each channel gets delayed it is multiplied with the given weight and then added to the single outout channel.

5.74.2 Constructor & Destructor Documentation

5.74.2.1 delaysum::delaysum_if_t::delaysum_if_t (const algo_comm_t & *iac,* const std::string & , const std::string &)

5.74.3 Member Function Documentation

5.74.3.1 mha_wave_t * delaysum::delaysum_if_t::process (mha_wave_t * *wave*)

5.74.3.2 void delaysum::delaysum_if_t::prepare (mhaconfig_t & *tfcfg*) [virtual]

Implements **MHAPlugin::plugin_t< delaysum_t >** (p. 730).

5.74.3.3 void delaysum::delaysum_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< delaysum_t >** (p. 731).

5.74.3.4 void delaysum::delaysum_if_t::update_cfg ( ) [private]

5.74.4 Member Data Documentation

5.74.4.1 MHAParser::vfloat_t delaysum::delaysum_if_t::weights [private]

Linear weights to be multiplied with the audio signal, one factor for each channel.

Order is [chan0, chan1, ...]

5.74.4.2 MHAParser::vint_t delaysum::delaysum_if_t::delay [private]

vector of channel-specific delays, in samples.

5.74.4.3 MHAEvents::patchbay_t<delaysum_if_t> delaysum::delaysum_if_t::patchbay [private]

The patchbay to react to config changes.

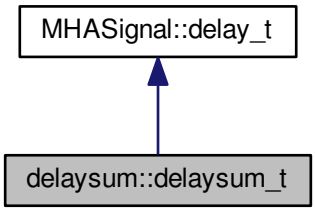
The documentation for this class was generated from the following file:

- **delaysum.cpp**

5.75 delaysum::delaysum_t Class Reference

Runtime configuration of the delaysum plugin.

Inheritance diagram for delaysum::delaysum_t:



Public Member Functions

- **delaysum_t** (unsigned int nch, unsigned int fragsize, const std::vector< **mha_real_t** > &weights_, const std::vector< int > &delays_)

  Constructor of the runtime configuration.
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Attributes

- std::vector< **mha_real_t** > **weights**

  Relative weights for each channel. Order is [chan0, chan1, ...].
- **MHASignal::waveform_t out**

  Output waveform.

5.75.1 Detailed Description

Runtime configuration of the delaysum plugin.

Inherits from the already present delay_t class. The constructor initializes and validates the runtime configuration and forwards the delay vector to the delay_t class. The process function first calls delay_t::process and then multiplies every output channel with its weight and adds them into the output channel.

5.75.2 Constructor & Destructor Documentation

5.75.2.1 delaysum::delaysum_t::delaysum_t (unsigned int *nch,* unsigned int *fragsize,* const std::vector< mha_real_t > & *weights_,* const std::vector< int > & *delays_*)

Constructor of the runtime configuration.

Parameters

nch	Number of input channels.
fragsize	Size of one input fragment in frames.
*weights_*	Vector of weights for each channel.
*delays_*	Vector of delays, one entry per channel.

5.75.3 Member Function Documentation

5.75.3.1 mha_wave_t * delaysum::delaysum_t::process (mha_wave_t * *signal*)

5.75.4 Member Data Documentation

5.75.4.1 std::vector<mha_real_t> delaysum::delaysum_t::weights [private]

Relative weights for each channel. Order is [chan0, chan1, ...].

5.75.4.2 MHASignal::waveform_t delaysum::delaysum_t::out [private]

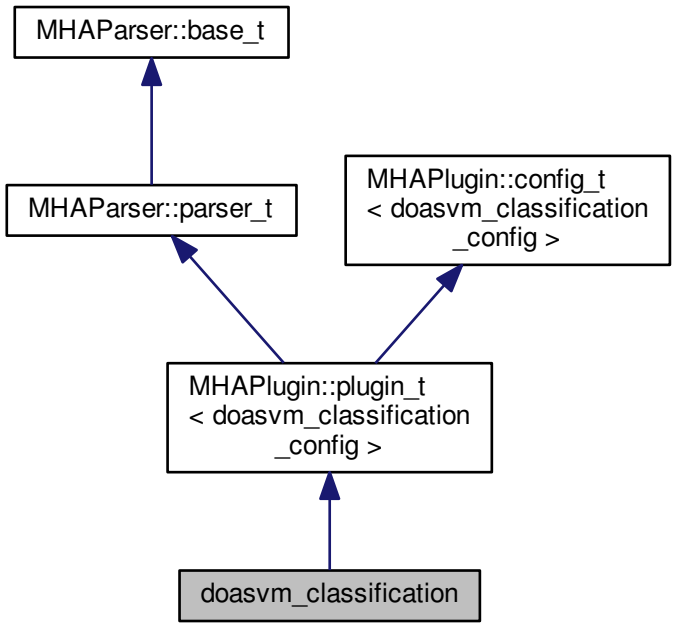
Output waveform.

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

- **delaysum.cpp**

5.76 doasvm_classification Class Reference

Inheritance diagram for doasvm_classification:



Public Member Functions

- **doasvm_classification** (**algo_comm_t** &**ac**, const std::string &chain_name, const std↩ ::string &algo_name)

  Constructs our plugin.
- ∼**doasvm_classification** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::vfloat_t angles**
- **MHAParser::mfloat_t w**
- **MHAParser::vfloat_t b**
- **MHAParser::vfloat_t x**
- **MHAParser::vfloat_t y**
- **MHAParser::string_t p_name**
- **MHAParser::string_t max_p_ind_name**
- **MHAParser::string_t vGCC_name**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< doasvm_classification > patchbay**

Additional Inherited Members

5.76.1 Constructor & Destructor Documentation

5.76.1.1 doasvm_classification::doasvm_classification (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.76.1.2 doasvm_classification::~doasvm_classification ( )

5.76.2 Member Function Documentation

5.76.2.1 mha_wave_t * doasvm_classification::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.76.2.2 void doasvm_classification::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< doasvm_classification_config >** (p. 730).

5.76.2.3 void doasvm_classification::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< doasvm_classification_config >** (p. 731).

5.76.2.4 void doasvm_classification::update_cfg ( ) [private]

5.76.3 Member Data Documentation

5.76.3.1 MHAParser::vfloat_t doasvm_classification::angles

5.76.3.2 MHAParser::mfloat_t doasvm_classification::w

5.76.3.3 MHAParser::vfloat_t doasvm_classification::b

5.76.3.4 MHAParser::vfloat_t doasvm_classification::x

5.76.3.5 MHAParser::vfloat_t doasvm_classification::y

5.76.3.6 MHAParser::string_t doasvm_classification::p_name

5.76.3.7 MHAParser::string_t doasvm_classification::max_p_ind_name

5.76.3.8 MHAParser::string_t doasvm_classification::vGCC_name

5.76.3.9 MHAEvents::patchbay_t<doasvm_classification> doasvm_classification::patchbay [private]

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

5.77 doasvm_classification_config Class Reference

Public Member Functions

- **doasvm_classification_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **doasvm_classification** *_doasvm)
- ~**doasvm_classification_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Public Attributes

- **algo_comm_t** & **ac**
- **doasvm_classification** ∗ **doasvm**
- **MHA_AC::waveform_t p**
- **MHA_AC::int_t p_max**
- **mha_wave_t c**

5.77.1 Constructor & Destructor Documentation

5.77.1.1 doasvm_classification_config::doasvm_classification_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* doasvm_classification ∗ *_doasvm*)

5.77.1.2 doasvm_classification_config::∼doasvm_classification_config ( )

5.77.2 Member Function Documentation

5.77.2.1 mha_wave_t ∗ doasvm_classification_config::process (mha_wave_t ∗ *wave*)

5.77.3 Member Data Documentation

5.77.3.1 algo_comm_t& doasvm_classification_config::ac

5.77.3.2 doasvm_classification∗ doasvm_classification_config::doasvm

5.77.3.3 MHA_AC::waveform_t doasvm_classification_config::p

5.77.3.4 MHA_AC::int_t doasvm_classification_config::p_max

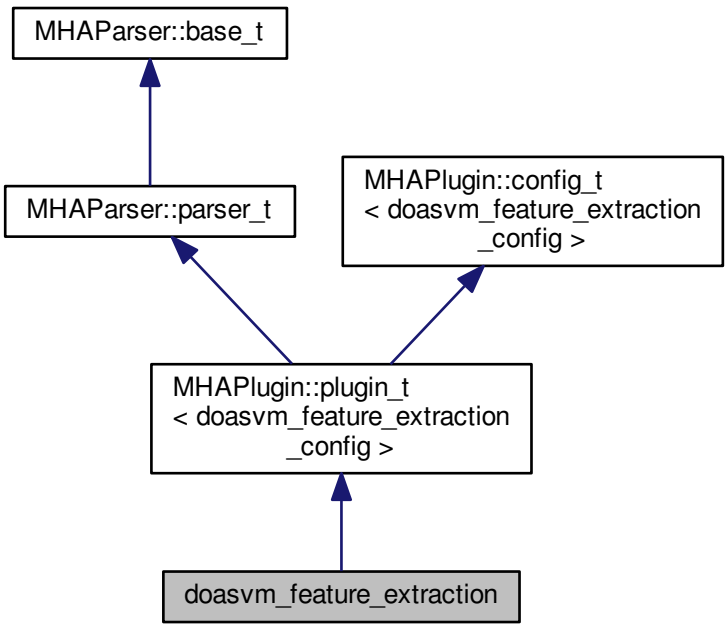
5.77.3.5 mha_wave_t doasvm_classification_config::c

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

5.78 doasvm_feature_extraction Class Reference

Inheritance diagram for doasvm_feature_extraction:



Public Member Functions

- **doasvm_feature_extraction** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  Constructs our plugin.
- ∼**doasvm_feature_extraction** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t fftlen**
- **MHAParser::int_t max_lag**
- **MHAParser::int_t nupsample**
- **MHAParser::string_t vGCC_name**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< doasvm_feature_extraction > patchbay**

Additional Inherited Members

5.78.1 Constructor & Destructor Documentation

5.78.1.1 doasvm_feature_extraction::doasvm_feature_extraction (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.78.1.2 doasvm_feature_extraction::~doasvm_feature_extraction ( )

5.78.2 Member Function Documentation

5.78.2.1 mha_wave_t * doasvm_feature_extraction::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.78.2.2 void doasvm_feature_extraction::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< doasvm_feature_extraction_config >** (p. 730).

5.78.2.3 void doasvm_feature_extraction::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< doasvm_feature_extraction_config >** (p. 731).

5.78.2.4 void doasvm_feature_extraction::update_cfg ( ) [private]

5.78.3 Member Data Documentation

5.78.3.1 MHAParser::int_t doasvm_feature_extraction::fftlen

5.78.3.2 MHAParser::int_t doasvm_feature_extraction::max_lag

5.78.3.3 MHAParser::int_t doasvm_feature_extraction::nupsample

5.78.3.4 MHAParser::string_t doasvm_feature_extraction::vGCC_name

5.78.3.5 MHAEvents::patchbay_t<doasvm_feature_extraction> doasvm_feature_extraction::patchbay [private]

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

- **doasvm_feature_extraction.h**
- **doasvm_feature_extraction.cpp**

5.79 doasvm_feature_extraction_config Class Reference

Public Member Functions

- **doasvm_feature_extraction_config** (**algo_comm_t** &ac, const **mhaconfig_t** in_cfg, **doasvm_feature_extraction** *_doagcc)
- ~**doasvm_feature_extraction_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Public Attributes

- **doasvm_feature_extraction** * **doagcc**
- unsigned int **wndlen**
- unsigned int **fftlen**
- unsigned int **G_length**
- unsigned int **GCC_start**
- unsigned int **GCC_end**
- **MHA_AC::waveform_t vGCC_ac**
- **mha_fft_t fft**
- **mha_fft_t ifft**
- double **hifftwin_sum**
- **MHASignal::waveform_t proc_wave**
- **MHASignal::waveform_t hwin**
- **MHASignal::waveform_t hifftwin**
- **MHASignal::waveform_t vGCC**
- **MHASignal::spectrum_t in_spec**
- **MHASignal::spectrum_t G**

5.79.1 Constructor & Destructor Documentation

5.79.1.1 doasvm_feature_extraction_config::doasvm_feature_extraction_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* doasvm_feature_extraction * *_doagcc*)

5.79.1.2 doasvm_feature_extraction_config::~doasvm_feature_extraction_config ( )

5.79.2 Member Function Documentation

5.79.2.1 mha_wave_t * doasvm_feature_extraction_config::process (mha_wave_t * *wave*)

5.79.3 Member Data Documentation

5.79.3.1 doasvm_feature_extraction* doasvm_feature_extraction_config::doagcc

5.79.3.2 unsigned int doasvm_feature_extraction_config::wndlen

5.79.3.3 unsigned int doasvm_feature_extraction_config::fftlen

5.79.3.4 unsigned int doasvm_feature_extraction_config::G_length

5.79.3.5 unsigned int doasvm_feature_extraction_config::GCC_start

5.79.3.6 unsigned int doasvm_feature_extraction_config::GCC_end

5.79.3.7 MHA_AC::waveform_t doasvm_feature_extraction_config::vGCC_ac

5.79.3.8 mha_fft_t doasvm_feature_extraction_config::fft

5.79.3.9 mha_fft_t doasvm_feature_extraction_config::ifft

5.79.3.10 double doasvm_feature_extraction_config::hifftwin_sum

5.79.3.11 MHASignal::waveform_t doasvm_feature_extraction_config::proc_wave

5.79.3.12 MHASignal::waveform_t doasvm_feature_extraction_config::hwin

5.79.3.13 MHASignal::waveform_t doasvm_feature_extraction_config::hifftwin

5.79.3.14 MHASignal::waveform_t doasvm_feature_extraction_config::vGCC

5.79.3.15 MHASignal::spectrum_t doasvm_feature_extraction_config::in_spec

5.79.3.16 MHASignal::spectrum_t doasvm_feature_extraction_config::G

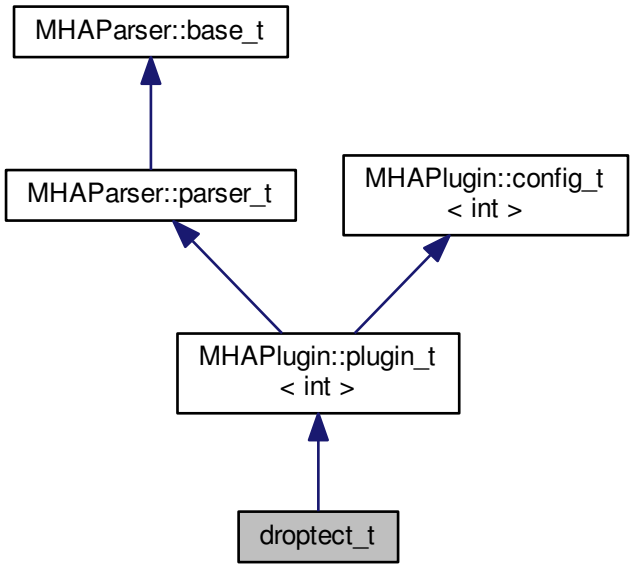
The documentation for this class was generated from the following files:

- **doasvm_feature_extraction.h**
- **doasvm_feature_extraction.cpp**

5.80 droptect_t Class Reference

Detect dropouts in a signal with a constant spectrum.

Inheritance diagram for droptect_t:



Public Member Functions

- **droptect_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo↩ _name)

  This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
- void **release** (void)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗signal)

Private Attributes

- **MHAParser::vint_mon_t dropouts**
- **MHAParser::vint_mon_t consecutive_dropouts**
- **MHAParser::int_mon_t blocks**
- **MHAParser::bool_t reset**
- **MHAParser::float_t threshold**
- **MHASignal::waveform_t** ∗ **current_powspec**

- **MHASignal::waveform_t * filtered_powspec**
- **MHAParser::float_t tau**
- std::vector< bool > **filter_activated**
- float **period**

  The period of the process callback.
- **MHAParser::mfloat_mon_t filtered_powspec_mon**

  User access to filtered spectrum.
- **MHAParser::float_mon_t level_mon**

Additional Inherited Members

5.80.1 Detailed Description

Detect dropouts in a signal with a constant spectrum.

5.80.2 Constructor & Destructor Documentation

5.80.2.1 droptect_t::droptect_t (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.

5.80.3 Member Function Documentation

5.80.3.1 void droptect_t::prepare (mhaconfig_t & *signal_info*) [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.80.3.2 void droptect_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.80.3.3 mha_spec_t * droptect_t::process (mha_spec_t * *signal*)

5.80.4 Member Data Documentation

5.80.4.1 MHAParser::vint_mon_t droptect_t::dropouts [private]

5.80.4.2 MHAParser::vint_mon_t droptect_t::consecutive_dropouts [private]

5.80.4.3 MHAParser::int_mon_t droptect_t::blocks [private]

5.80.4.4 MHAParser::bool_t droptect_t::reset [private]

5.80.4.5 MHAParser::float_t droptect_t::threshold [private]

5.80.4.6 MHASignal::waveform_t* droptect_t::current_powspec [private]

5.80.4.7 MHASignal::waveform_t* droptect_t::filtered_powspec [private]

5.80.4.8 MHAParser::float_t droptect_t::tau [private]

5.80.4.9 std::vector<bool> droptect_t::filter_activated [private]

5.80.4.10 float droptect_t::period [private]

The period of the process callback.

5.80.4.11 MHAParser::mfloat_mon_t droptect_t::filtered_powspec_mon [private]

User access to filtered spectrum.

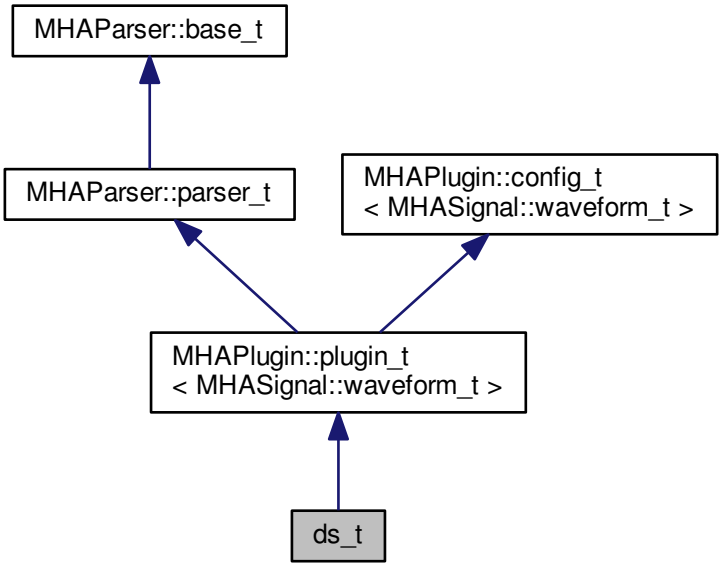
5.80.4.12 MHAParser::float_mon_t droptect_t::level_mon [private]

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

- **droptect.cpp**

5.81 ds_t Class Reference

Inheritance diagram for ds_t:



Public Member Functions

- **ds_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Attributes

- **MHAParser::int_t ratio**
- **MHAFilter::iir_filter_t antialias**

Additional Inherited Members

5.81.1 Constructor & Destructor Documentation

5.81.1.1 ds_t::ds_t (
algo_comm_t *iac,*
std::string ,
std::string)

5.81.2 Member Function Documentation

5.81.2.1 mha_wave_t * ds_t::process (
mha_wave_t * *s*)

5.81.2.2 void ds_t::prepare (
mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 730).

5.81.2.3 void ds_t::release (
void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 731).

5.81.3 Member Data Documentation

5.81.3.1 MHAParser::int_t ds_t::ratio [private]

5.81.3.2 MHAFilter::iir_filter_t ds_t::antialias [private]

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

- **downsample.cpp**

5.82 dynamiclib_t Class Reference

Public Member Functions

- **dynamiclib_t** (const std::string &)
- void * **resolve** (const std::string &)
- void * **resolve_checked** (const std::string &)
- ~**dynamiclib_t** ()
- const std::string & **getmodulename** () const
- const std::string & **getname** () const

Private Attributes

- std::string **fullname**
- std::string **modulename**
- **mha_libhandle_t h**

5.82.1 Constructor & Destructor Documentation

5.82.1.1 dynamiclib_t::dynamiclib_t (const std::string & *n*)

5.82.1.2 dynamiclib_t::∼dynamiclib_t ( )

5.82.2 Member Function Documentation

5.82.2.1 void ∗ dynamiclib_t::resolve (const std::string & *n*)

5.82.2.2 void ∗ dynamiclib_t::resolve_checked (const std::string & *n*)

5.82.2.3 const std::string& dynamiclib_t::getmodulename ( ) const [inline]

5.82.2.4 const std::string& dynamiclib_t::getname ( ) const [inline]

5.82.3 Member Data Documentation

5.82.3.1 std::string dynamiclib_t::fullname [private]

5.82.3.2 std::string dynamiclib_t::modulename [private]

5.82.3.3 mha_libhandle_t dynamiclib_t::h [private]

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

- **mha_os.h**
- **mha_os.cpp**

5.83 DynComp::dc_afterburn_rt_t Class Reference

Real-time class for after burn effect.

Public Member Functions

- **dc_afterburn_rt_t** (const std::vector< float > &cf, unsigned int **channels**, float srate, const **dc_afterburn_vars_t** &vars)
- void **burn** (float &Gin, float Lin, unsigned int band, unsigned int channel)

  gain modifier method (afterburn).

Private Attributes

- std::vector< float > **drain_inv**
- std::vector< float > **conflux**
- std::vector< float > **maxgain**
- std::vector< float > **mpo_inv**
- std::vector< **MHAFilter::o1flt_lowpass_t** > **lp**

5.83.1 Detailed Description

Real-time class for after burn effect.

The constructor processes the parameters and creates pre-processed variables for efficient realtime processing.

5.83.2 Constructor & Destructor Documentation

5.83.2.1 DynComp::dc_afterburn_rt_t::dc_afterburn_rt_t (
const std::vector< float > & *cf,*
unsigned int *channels,*
float *srate,*
const dc_afterburn_vars_t & *vars*)

5.83.3 Member Function Documentation

5.83.3.1 void DynComp::dc_afterburn_rt_t::burn (
float & *Gin,*
float *Lin,*
unsigned int *band,*
unsigned int *channel*) [inline]

gain modifier method (afterburn).

Parameters

Gin	Linear gain.
Lin	Input level (Pascal).
band	Filter band number.
channel	Channel number.

Output level for MPO is estimated by Gin $*$ Lin.

5.83.4 Member Data Documentation

5.83.4.1 std::vector<float> DynComp::dc_afterburn_rt_t::drain_inv [private]

5.83.4.2 std::vector<float> DynComp::dc_afterburn_rt_t::conflux [private]

5.83.4.3 std::vector<float> DynComp::dc_afterburn_rt_t::maxgain [private]

5.83.4.4 std::vector<float> DynComp::dc_afterburn_rt_t::mpo_inv [private]

5.83.4.5 std::vector<MHAFilter::o1flt_lowpass_t> DynComp::dc_afterburn_rt_t::lp [private]

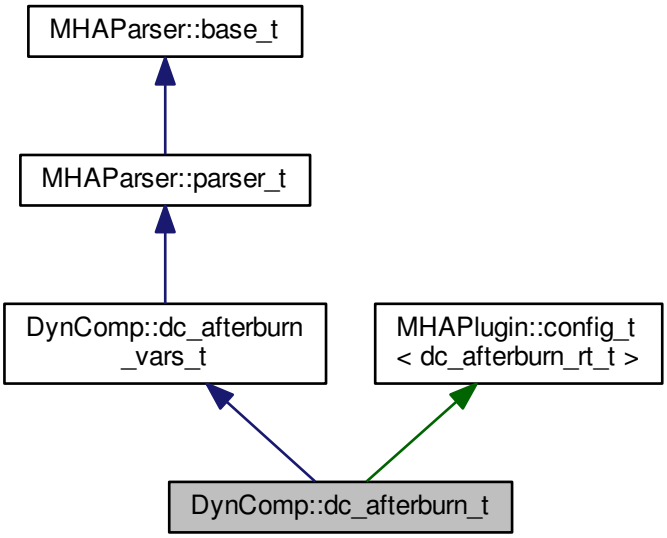
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.84 DynComp::dc_afterburn_t Class Reference

Afterburn class, to be defined as a member of compressors.

Inheritance diagram for DynComp::dc_afterburn_t:



Public Member Functions

- **dc_afterburn_t** ()
- void **set_fb_pars** (const std::vector< float > &cf, unsigned int **channels**, float srate)
- void **unset_fb_pars** ()
- void **update_burner** ()
- void **burn** (float &Gin, float Lin, unsigned int band, unsigned int channel)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t< dc_afterburn_t > patchbay**
- std::vector< float > **_cf**
- unsigned int **_channels**
- float **_srate**
- bool **commit_pending**
- bool **fb_pars_configured**

Additional Inherited Members

5.84.1 Detailed Description

Afterburn class, to be defined as a member of compressors.

5.84.2 Constructor & Destructor Documentation

5.84.2.1 DynComp::dc_afterburn_t::dc_afterburn_t ( )

5.84.3 Member Function Documentation

5.84.3.1 void DynComp::dc_afterburn_t::set_fb_pars (const std::vector< float > & *cf,* unsigned int *channels,* float *srate*)

5.84.3.2 void DynComp::dc_afterburn_t::unset_fb_pars ( )

5.84.3.3 void DynComp::dc_afterburn_t::update_burner ( ) [inline]

5.84.3.4 void DynComp::dc_afterburn_t::burn (float & *Gin,* float *Lin,* unsigned int *band,* unsigned int *channel*) [inline]

5.84.3.5 void DynComp::dc_afterburn_t::update ( ) [private]

5.84.4 Member Data Documentation

5.84.4.1 MHAEvents::patchbay_t<dc_afterburn_t> DynComp::dc_afterburn_t::patchbay [private]

5.84.4.2 std::vector<float> DynComp::dc_afterburn_t::_cf [private]

5.84.4.3 unsigned int DynComp::dc_afterburn_t::_channels [private]

5.84.4.4 float DynComp::dc_afterburn_t::_srate [private]

5.84.4.5 bool DynComp::dc_afterburn_t::commit_pending [private]

5.84.4.6 bool DynComp::dc_afterburn_t::fb_pars_configured [private]

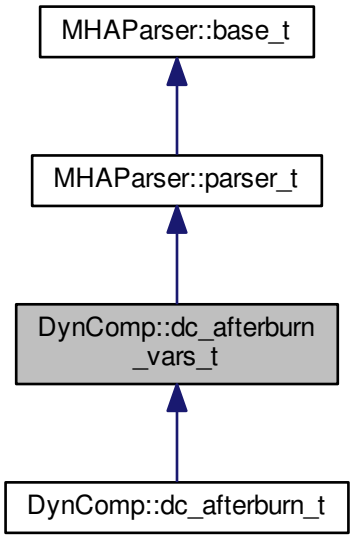
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.85 DynComp::dc_afterburn_vars_t Class Reference

Variables for **dc_afterburn_t** (p. 294) class.

Inheritance diagram for DynComp::dc_afterburn_vars_t:



Public Member Functions

- **dc_afterburn_vars_t** ()

Public Attributes

- **MHAParser::vfloat_t f**
- **MHAParser::vfloat_t drain**
- **MHAParser::vfloat_t conflux**
- **MHAParser::vfloat_t maxgain**
- **MHAParser::vfloat_t mpo**
- **MHAParser::float_t taugain**
- **MHAParser::kw_t commit**
- **MHAParser::bool_t bypass**

Additional Inherited Members

5.85.1 Detailed Description

Variables for **dc_afterburn_t** (p. 294) class.

5.85.2 Constructor & Destructor Documentation

5.85.2.1 DynComp::dc_afterburn_vars_t::dc_afterburn_vars_t ( )

5.85.3 Member Data Documentation

5.85.3.1 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::f

5.85.3.2 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::drain

5.85.3.3 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::conflux

5.85.3.4 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::maxgain

5.85.3.5 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::mpo

5.85.3.6 MHAParser::float_t DynComp::dc_afterburn_vars_t::taugain

5.85.3.7 MHAParser::kw_t DynComp::dc_afterburn_vars_t::commit

5.85.3.8 MHAParser::bool_t DynComp::dc_afterburn_vars_t::bypass

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

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.86 DynComp::gaintable_t Class Reference

Gain table class.

Public Member Functions

- **gaintable_t** (const std::vector< **mha_real_t** > &LInput, const std::vector< **mha_real_t** > &FCenter, unsigned int **channels**)

  Constructor.
- ~**gaintable_t** ()
- void **update** (std::vector< std::vector< std::vector< **mha_real_t** > > > newGain)

  Update gains from an external table.
- **mha_real_t get_gain** (**mha_real_t** Lin, **mha_real_t** Fin, unsigned int channel)

  Read Gain from gain table.
- **mha_real_t get_gain** (**mha_real_t** Lin, unsigned int band, unsigned int channel)

  Read Gain from gain table.
- void **get_gain** (const **mha_wave_t** &Lin, **mha_wave_t** &Gain)

  Read Gains from gain table.
- unsigned int **nbands** () const

  Return number of frequency bands.
- unsigned int **nchannels** () const

  Return number of audio channels.
- std::vector< std::vector< **mha_real_t** > > **get_iofun** () const

  Return current input-output function.
- std::vector< **mha_real_t** > **get_vL** () const
- std::vector< **mha_real_t** > **get_vF** () const

Private Attributes

- unsigned int **num_L**
- unsigned int **num_F**
- unsigned int **num_channels**
- std::vector< **mha_real_t** > **vL**
- std::vector< **mha_real_t** > **vF**
- std::vector< **mha_real_t** > **vFlog**
- std::vector< std::vector< std::vector< **mha_real_t** > > > **data**

5.86.1 Detailed Description

Gain table class.

This gain table is intended to efficient table lookup, i.e, interpolation of levels, and optional interpolation of frequencies. Sample input levels and sample frequencies are given in the constructor. The gain entries can be updated with the **update()** (p. 301) member function via a gain prescription rule from an auditory profile.

5.86.2 Constructor & Destructor Documentation

5.86.2.1 gaintable_t::gaintable_t (const std::vector< mha_real_t > & *LInput,* const std::vector< mha_real_t > & *FCenter,* unsigned int *channels*)

Constructor.

Parameters

LInput	Input level samples, in equivalent LTASS_combined dB SPL.
FCenter	Frequency samples in Hz (e.g., center frequencies of filterbank).
channels	Number of audio channels (typically 2).

5.86.2.2 gaintable_t::~gaintable_t ( )

5.86.3 Member Function Documentation

5.86.3.1 void gaintable_t::update (std::vector< std::vector< std::vector< mha_real_t > > > *newGain*)

Update gains from an external table.

Parameters

newGain	New gain table entries.

Dimension change is not allowed. The number of entries are checked.

5.86.3.2 mha_real_t gaintable_t::get_gain (mha_real_t *Lin*, mha_real_t *Fin*, unsigned int *channel*)

Read Gain from gain table.

Parameters

Lin	Input level
Fin	Input frequency (no match required)
channel	Audio channel

5.86.3.3 mha_real_t gaintable_t::get_gain (mha_real_t *Lin*, unsigned int *band*, unsigned int *channel*)

Read Gain from gain table.

Parameters

Lin	Input level

Parameters

band	Input frequency band
channel	Audio channel

5.86.3.4 void gaintable_t::get_gain (const mha_wave_t & *Lin*, mha_wave_t & *Gain*)

Read Gains from gain table.

Parameters

Lin	Input levels.
Gain	Output gain.

The number of channels in Lin and Gain must match the number of bands times number of channels in the gaintable.

5.86.3.5 unsigned int DynComp::gaintable_t::nbands ( ) const [inline]

Return number of frequency bands.

5.86.3.6 unsigned int DynComp::gaintable_t::nchannels ( ) const [inline]

Return number of audio channels.

5.86.3.7 std::vector< std::vector< mha_real_t > > gaintable_t::get_iofun ( ) const

Return current input-output function.

5.86.3.8 std::vector<mha_real_t> DynComp::gaintable_t::get_vL ( ) const [inline]

5.86.3.9 std::vector<mha_real_t> DynComp::gaintable_t::get_vF ( ) const [inline]

5.86.4 Member Data Documentation

5.86.4.1 unsigned int DynComp::gaintable_t::num_L [private]

5.86.4.2 unsigned int DynComp::gaintable_t::num_F [private]

5.86.4.3 unsigned int DynComp::gaintable_t::num_channels [private]

5.86.4.4 std::vector<mha_real_t> DynComp::gaintable_t::vL [private]

5.86.4.5 std::vector<mha_real_t> DynComp::gaintable_t::vF [private]

5.86.4.6 std::vector<mha_real_t> DynComp::gaintable_t::vFlog [private]

5.86.4.7 std::vector<std::vector<std::vector<mha_real_t> > > DynComp::gaintable_t::data [private]

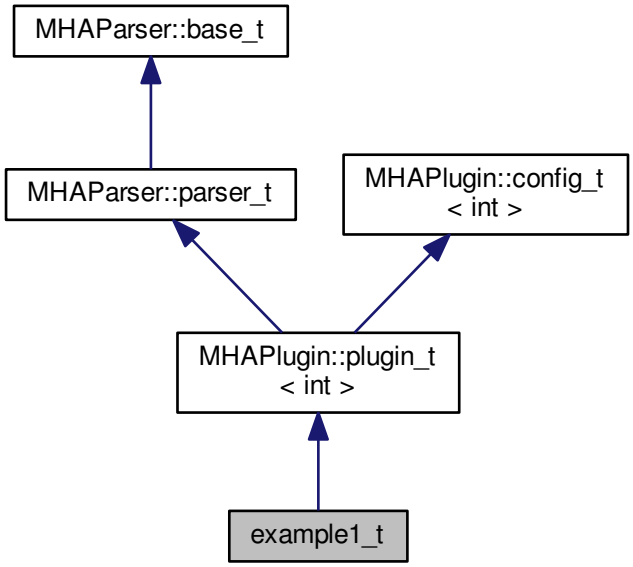
The documentation for this class was generated from the following files:

- **gaintable.h**
- **gaintable.cpp**

5.87 example1_t Class Reference

This C++ class implements the simplest example plugin for the step-by-step tutorial.

Inheritance diagram for example1_t:



Public Member Functions

- **example1_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
  Do-nothing constructor.
- void **release** (void)
  Release may be empty.
- void **prepare** (**mhaconfig_t** &signal_info)
  Plugin preparation.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Signal processing performed by the plugin.

Additional Inherited Members

5.87.1 Detailed Description

This C++ class implements the simplest example plugin for the step-by-step tutorial.

It inherits from **MHAPlugin::plugin_t** (p. 728) for correct integration in the configuration language interface.

5.87.2 Constructor & Destructor Documentation

5.87.2.1 example1_t::example1_t (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*) [inline]

Do-nothing constructor.

The constructor has to take these three arguments, but it does not have to use them. However, the base class has to be initialized.

5.87.3 Member Function Documentation

5.87.3.1 void example1_t::release (void) [inline], [virtual]

Release may be empty.

Reimplemented from **MHAPlugin::plugin_t**< **int** > (p. 731).

5.87.3.2 void example1_t::prepare (mhaconfig_t & *signal_info*) [inline], [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains at least one channel

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.87.3.3 mha_wave_t* example1_t::process (mha_wave_t * *signal*) [inline]

Signal processing performed by the plugin.

This plugin multiplies the signal in the first audio channel by a factor 0.1.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin. (In-place processing)

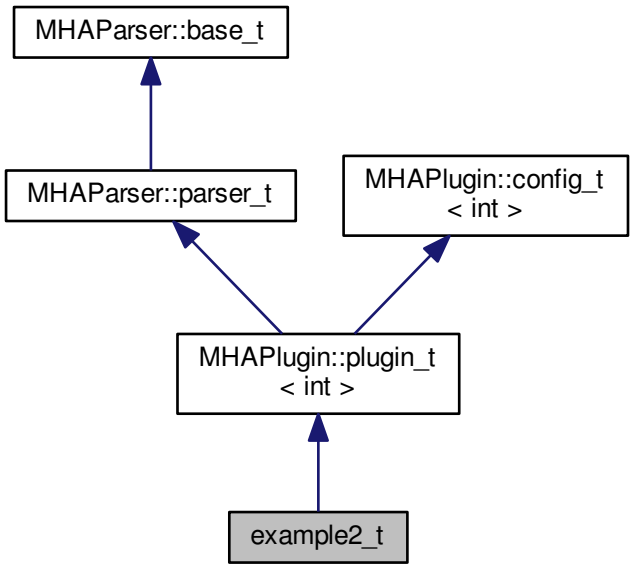
The documentation for this class was generated from the following file:

- **example1.cpp**

5.88 example2_t Class Reference

This C++ class implements the second example plugin for the step-by-step tutorial.

Inheritance diagram for example2_t:



Public Member Functions

- **example2_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)

  Plugin preparation.
- void **release** (void)

  Undo restrictions posed in prepare.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)

  Signal processing performed by the plugin.

Private Attributes

- **MHAParser::int_t scale_ch**

  Index of audio channel to scale.
- **MHAParser::float_t factor**

  The scaling factor applied to the selected channel.

Additional Inherited Members

5.88.1 Detailed Description

This C++ class implements the second example plugin for the step-by-step tutorial.

It extends the first example by using configuration language variables to influence the processing.

5.88.2 Constructor & Destructor Documentation

5.88.2.1 example2_t::example2_t (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

5.88.3 Member Function Documentation

5.88.3.1 void example2_t::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.88.3.2 void example2_t::release (void) [virtual]

Undo restrictions posed in prepare.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.88.3.3 mha_wave_t * example2_t::process (mha_wave_t * *signal*)

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin. (In-place processing)

5.88.4 Member Data Documentation

5.88.4.1 MHAParser::int_t example2_t::scale_ch [private]

Index of audio channel to scale.

5.88.4.2 MHAParser::float_t example2_t::factor [private]

The scaling factor applied to the selected channel.

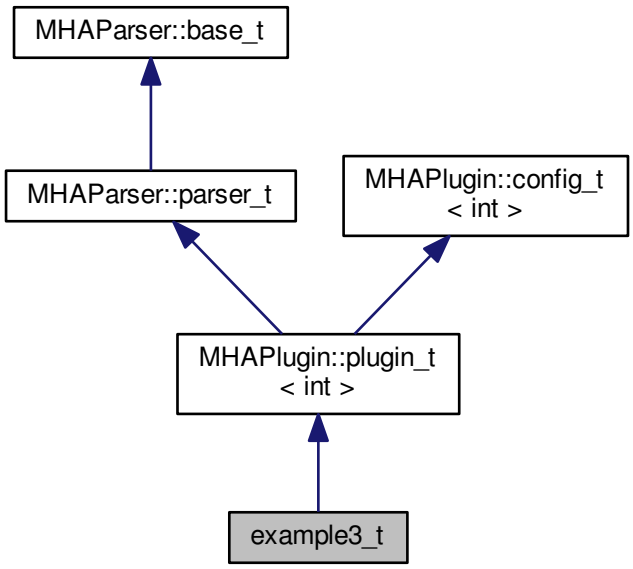
The documentation for this class was generated from the following file:

- **example2.cpp**

5.89 example3_t Class Reference

A Plugin class using the openMHA Event mechanism.

Inheritance diagram for example3_t:



Public Member Functions

- **example3_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)

  Plugin preparation.
- void **release** (void)

  Bookkeeping only.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)

  Signal processing performed by the plugin.

Private Member Functions

- void **on_scale_ch_writeaccess** ()
- void **on_scale_ch_valuechanged** ()
- void **on_scale_ch_readaccess** ()
- void **on_prereadaccess** ()

Private Attributes

- **MHAParser::int_t scale_ch**
  Index of audio channel to scale.
- **MHAParser::float_t factor**
  The scaling factor applied to the selected channel.
- **MHAParser::int_mon_t prepared**
  Keep Track of the prepare/release calls.
- **MHAEvents::patchbay_t< example3_t > patchbay**
  The Event connector.

Additional Inherited Members

5.89.1 Detailed Description

A Plugin class using the openMHA Event mechanism.

This is the third example plugin for the step-by-step tutorial.

5.89.2 Constructor & Destructor Documentation

5.89.2.1 example3_t::example3_t (

algo_comm_t & ***ac,***
const std::string & ***chain_name,***
const std::string & ***algo_name*** **)**

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

5.89.3 Member Function Documentation

5.89.3.1 void example3_t::on_scale_ch_writeaccess ( ) [private]

5.89.3.2 void example3_t::on_scale_ch_valuechanged ( ) [private]

5.89.3.3 void example3_t::on_scale_ch_readaccess ( ) [private]

5.89.3.4 void example3_t::on_prereadaccess ( ) [private]

5.89.3.5 void example3_t::prepare (

mhaconfig_t & ***signal_info*** **)** [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.89.3.6 void example3_t::release (void) [virtual]

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.89.3.7 mha_wave_t * example3_t::process (mha_wave_t * *signal*)

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin. (In-place processing)

5.89.4 Member Data Documentation

5.89.4.1 MHAParser::int_t example3_t::scale_ch [private]

Index of audio channel to scale.

5.89.4.2 MHAParser::float_t example3_t::factor [private]

The scaling factor applied to the selected channel.

5.89.4.3 MHAParser::int_mon_t example3_t::prepared [private]

Keep Track of the prepare/release calls.

5.89.4.4 MHAEvents::patchbay_t<example3_t> example3_t::patchbay [private]

The Event connector.

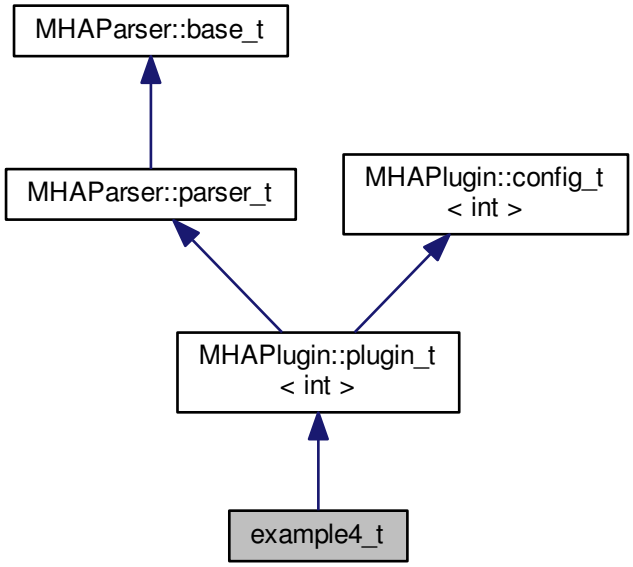
The documentation for this class was generated from the following file:

- **example3.cpp**

5.90 example4_t Class Reference

A Plugin class using the spectral signal.

Inheritance diagram for example4_t:



Public Member Functions

- **example4_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)

  Plugin preparation.
- void **release** (void)

  Bookkeeping only.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗signal)

  Signal processing performed by the plugin.

Private Member Functions

- void **on_scale_ch_writeaccess** ()
- void **on_scale_ch_valuechanged** ()
- void **on_scale_ch_readaccess** ()
- void **on_prereadaccess** ()

Private Attributes

- **MHAParser::int_t scale_ch**

  Index of audio channel to scale.
- **MHAParser::float_t factor**

  The scaling factor applied to the selected channel.
- **MHAParser::int_mon_t prepared**

  Keep Track of the prepare/release calls.
- **MHAEvents::patchbay_t**< **example4_t** > **patchbay**

  The Event connector.

Additional Inherited Members

5.90.1 Detailed Description

A Plugin class using the spectral signal.

This is the fourth example plugin for the step-by-step tutorial.

5.90.2 Constructor & Destructor Documentation

5.90.2.1 example4_t::example4_t (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

5.90.3 Member Function Documentation

5.90.3.1 void example4_t::on_scale_ch_writeaccess ( ) [private]

5.90.3.2 void example4_t::on_scale_ch_valuechanged ( ) [private]

5.90.3.3 void example4_t::on_scale_ch_readaccess ( ) [private]

5.90.3.4 void example4_t::on_prereadaccess ( ) [private]

5.90.3.5 void example4_t::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.90.3.6 void example4_t::release (void) [virtual]

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

5.90.3.7 mha_spec_t * example4_t::process (mha_spec_t * *signal*)

Signal processing performed by the plugin.

This plugin multiplies the spectral signal in the selected audio channel by the configured factor.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin. (In-place processing)

5.90.4 Member Data Documentation

5.90.4.1 MHAParser::int_t example4_t::scale_ch [private]

Index of audio channel to scale.

5.90.4.2 MHAParser::float_t example4_t::factor [private]

The scaling factor applied to the selected channel.

5.90.4.3 MHAParser::int_mon_t example4_t::prepared [private]

Keep Track of the prepare/release calls.

5.90.4.4 MHAEvents::patchbay_t<example4_t> example4_t::patchbay [private]

The Event connector.

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

- **example4.cpp**

5.91 example5_t Class Reference

Public Member Functions

- **example5_t** (unsigned int, unsigned int, **mha_real_t**)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- unsigned int **channel**
- **mha_real_t scale**

5.91.1 Constructor & Destructor Documentation

5.91.1.1 example5_t::example5_t (unsigned int *ichannel,* unsigned int *numchannels,* mha_real_t *iscale*)

5.91.2 Member Function Documentation

5.91.2.1 mha_spec_t * example5_t::process (mha_spec_t * *spec*)

5.91.3 Member Data Documentation

5.91.3.1 unsigned int example5_t::channel [private]

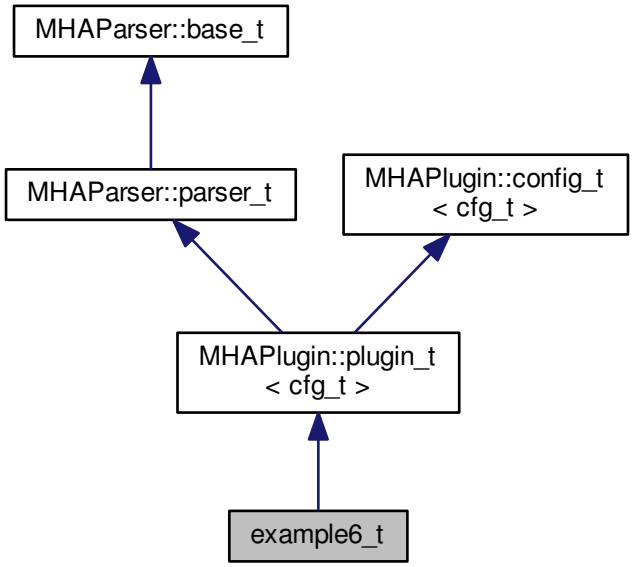
5.91.3.2 mha_real_t example5_t::scale [private]

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

- **example5.cpp**

5.92 example6_t Class Reference

Inheritance diagram for example6_t:



Public Member Functions

- **example6_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::int_t channel_no**
- float **rmsdb**
- **MHAEvents::patchbay_t**< **example6_t** > **patchbay**

Additional Inherited Members

5.92.1 Constructor & Destructor Documentation

5.92.1.1 example6_t::example6_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.92.2 Member Function Documentation

5.92.2.1 mha_wave_t ∗ example6_t::process (mha_wave_t ∗ *wave*)

5.92.2.2 void example6_t::prepare (mhaconfig_t & *tfcfg*) [virtual]

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 730).

5.92.2.3 void example6_t::update_cfg ( ) [private]

5.92.3 Member Data Documentation

5.92.3.1 MHAParser::int_t example6_t::channel_no [private]

5.92.3.2 float example6_t::rmsdb [private]

5.92.3.3 MHAEvents::patchbay_t<example6_t> example6_t::patchbay [private]

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

- **example6.cpp**

5.93 expression_t Class Reference

Class for separating a string into a left hand value and a right hand value.

5.93.1 Detailed Description

Class for separating a string into a left hand value and a right hand value.

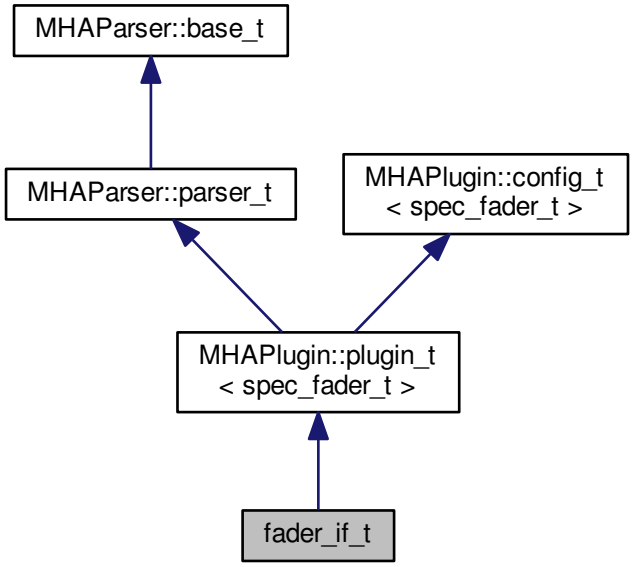
A list of valid operators can be provided. After construction, the class members lval, rval and op contain the apropriate contents.

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

- **mha_parser.cpp**

5.94 fader_if_t Class Reference

Inheritance diagram for fader_if_t:



Public Member Functions

- **fader_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< fader_if_t > patchbay**
- **MHAParser::float_t tau**
- **MHAParser::vfloat_t newgains**
- **mha_real_t * actgains**

Additional Inherited Members

5.94.1 Constructor & Destructor Documentation

5.94.1.1 fader_if_t::fader_if_t (

const **algo_comm_t** & *iac,*
const **std::string** & *,*
const **std::string** &)

5.94.2 Member Function Documentation

5.94.2.1 mha_spec_t * fader_if_t::process (

mha_spec_t * *s*)

5.94.2.2 void fader_if_t::prepare (

mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< spec_fader_t >** (p. 730).

5.94.2.3 void fader_if_t::update_cfg (

void) [private]

5.94.3 Member Data Documentation

5.94.3.1 MHAEvents::patchbay_t<fader_if_t> fader_if_t::patchbay [private]

5.94.3.2 MHAParser::float_t fader_if_t::tau [private]

5.94.3.3 MHAParser::vfloat_t fader_if_t::newgains [private]

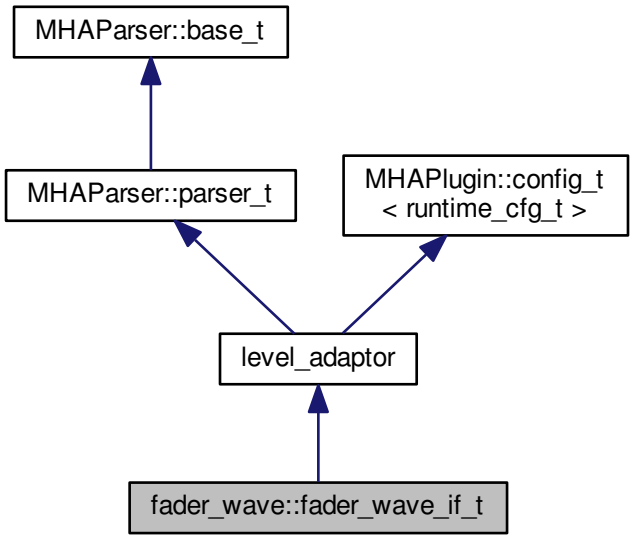
5.94.3.4 mha_real_t* fader_if_t::actgains [private]

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

- **fader_spec.cpp**

5.95 fader_wave::fader_wave_if_t Class Reference

Inheritance diagram for fader_wave::fader_wave_if_t:



Public Member Functions

- **fader_wave_if_t** (**algo_comm_t**, const char *, const char *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **set_level** ()

Private Attributes

- **MHAParser::vfloat_t gain**
- **MHAParser::float_t ramplen**
- **MHAEvents::patchbay_t**< **fader_wave_if_t** > **patchbay**
- bool **prepared**

Additional Inherited Members

5.95.1 Constructor & Destructor Documentation

5.95.1.1 fader_wave::fader_wave_if_t::fader_wave_if_t (algo_comm_t *iac,* const char * , const char *)

5.95.2 Member Function Documentation

5.95.2.1 mha_wave_t * fader_wave::fader_wave_if_t::process (mha_wave_t * *s*)

5.95.2.2 void fader_wave::fader_wave_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 730).

5.95.2.3 void fader_wave::fader_wave_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 731).

5.95.2.4 void fader_wave::fader_wave_if_t::set_level ( ) [private]

5.95.3 Member Data Documentation

5.95.3.1 MHAParser::vfloat_t fader_wave::fader_wave_if_t::gain [private]

5.95.3.2 MHAParser::float_t fader_wave::fader_wave_if_t::ramplen [private]

5.95.3.3 MHAEvents::patchbay_t<fader_wave_if_t> fader_wave::fader_wave_if_t::patchbay [private]

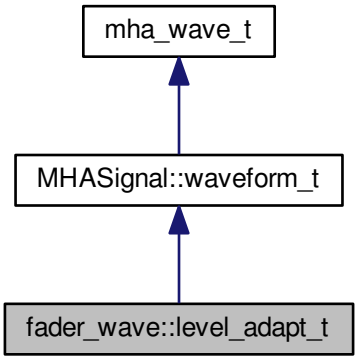
5.95.3.4 bool fader_wave::fader_wave_if_t::prepared [private]

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

- **fader_wave.cpp**

5.96 fader_wave::level_adapt_t Class Reference

Inheritance diagram for fader_wave::level_adapt_t:



Public Member Functions

- **level_adapt_t** (**mhaconfig_t** cf, **mha_real_t** adapt_len, std::vector< float > l_new_↩ , std::vector< float > l_old_)
- void **update_frame** ()
- std::vector< float > **get_level** () const
- bool **can_update** () const

Private Attributes

- unsigned int **ilen**
- unsigned int **pos**
- **MHAWindow::fun_t wnd**
- std::vector< float > **l_new**
- std::vector< float > **l_old**

Additional Inherited Members

5.96.1 Constructor & Destructor Documentation

5.96.1.1 fader_wave::level_adapt_t::level_adapt_t (mhaconfig_t *cf,* mha_real_t *adapt_len,* std::vector< float > *l_new_,* std::vector< float > *l_old_*)

5.96.2 Member Function Documentation

5.96.2.1 void fader_wave::level_adapt_t::update_frame ( )

5.96.2.2 std::vector<float> fader_wave::level_adapt_t::get_level ( ) const [inline]

5.96.2.3 bool fader_wave::level_adapt_t::can_update ( ) const [inline]

5.96.3 Member Data Documentation

5.96.3.1 unsigned int fader_wave::level_adapt_t::ilen [private]

5.96.3.2 unsigned int fader_wave::level_adapt_t::pos [private]

5.96.3.3 MHAWindow::fun_t fader_wave::level_adapt_t::wnd [private]

5.96.3.4 std::vector<float> fader_wave::level_adapt_t::l_new [private]

5.96.3.5 std::vector<float> fader_wave::level_adapt_t::l_old [private]

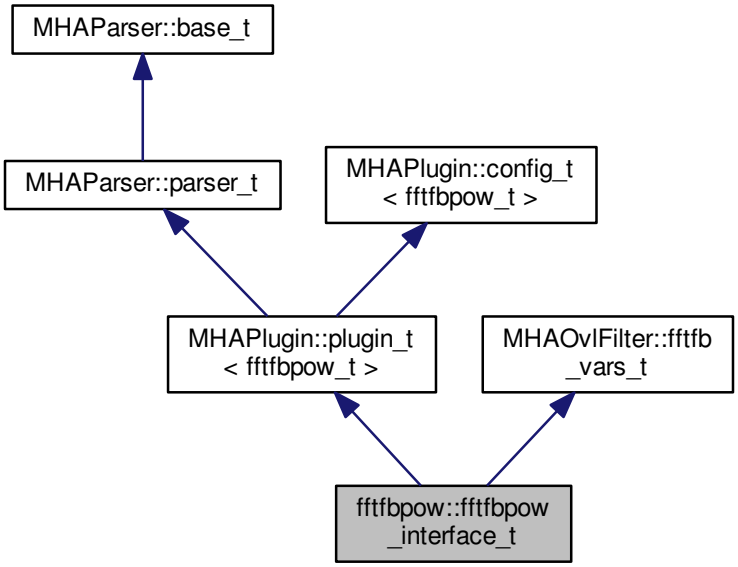
The documentation for this class was generated from the following file:

- **fader_wave.cpp**

5.97 fttfbpow::fttfbpow_interface_t Class Reference

Interface class for fttfbpow plugin.

Inheritance diagram for fttfbpow::fttfbpow_interface_t:



Public Member Functions

- **fftfbpow_interface_t** (const **algo_comm_t** &**ac**, const std::string &, const std::string &algo_name)

  Constructor with standard MHA constructor parameters.
- void **prepare** (**mhaconfig_t** &tf)

  Standard MHA plugin prepare function.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)

  Standard MHA plugin process fct.

Private Member Functions

- void **update_cfg** ()

  Constructs new runtime configuration in thread-safe manner.

Private Attributes

- std::string **name**

  Configured name of this plugin instance.
- **MHAEvents::patchbay_t**< **fftfbpow_interface_t** > **patchbay**

  Patchbay to connect to MHA configuration interface.

Additional Inherited Members

5.97.1 Detailed Description

Interface class for fftfbpow plugin.

5.97.2 Constructor & Destructor Documentation

5.97.2.1 fftfbpow::fftfbpow_interface_t::fftfbpow_interface_t (const algo_comm_t & *ac,* const std::string & *,* const std::string & *algo_name*)

Constructor with standard MHA constructor parameters.

Parameters

ac	Handle to algorithm communication variable space
algo_name	Configured name of this plugin instance

5.97.3 Member Function Documentation

5.97.3.1 void fftfbpow::fftfbpow_interface_t::prepare (mhaconfig_t & *tf*) [virtual]

Standard MHA plugin prepare function.

Ensures that the input is in the frequency domain, calls **update_cfg()** (p. 325) and inserts fbpow into the AC space.

Parameters

tf	Incoming mha configuration structure, contains information about input signal

Implements **MHAPlugin::plugin_t**< **fftfbpow_t** > (p. 730).

5.97.3.2 mha_spec_t * fftfbpow::fftfbpow_interface_t::process (mha_spec_t * *s*)

Standard MHA plugin process fct.

Polls new config and calls **process()** (p. 325) of the runtime configuration.

Parameters

s	Input spectrum

Returns

Unchanged input spectrum

5.97.3.3 void fftfbpow::fftfbpow_interface_t::update_cfg (void) [private]

Constructs new runtime configuration in thread-safe manner.

5.97.4 Member Data Documentation

5.97.4.1 std::string fftfbpow::fftfbpow_interface_t::name [private]

Configured name of this plugin instance.

5.97.4.2 MHAEvents::patchbay_t<fftfbpow_interface_t> fftfbpow::fftfbpow_interface_t::patchbay [private]

Patchbay to connect to MHA configuration interface.

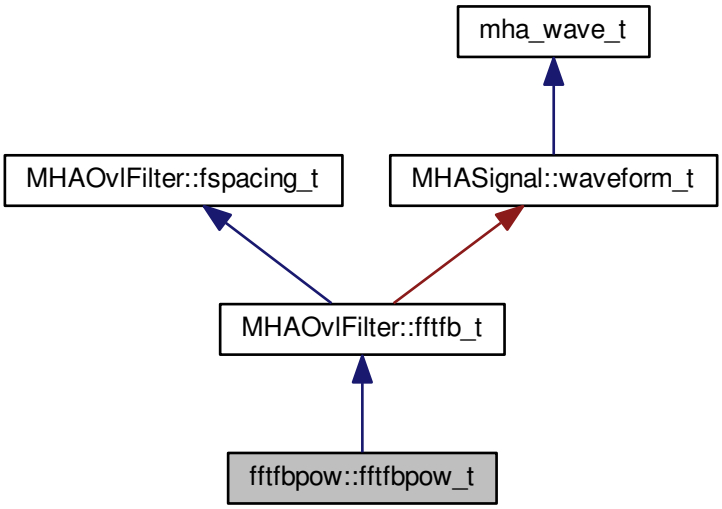
The documentation for this class was generated from the following file:

- **fftfbpow.cpp**

5.98 fftfbpow::fftfbpow_t Class Reference

Run time configuration for the fftfbpow plugin.

Inheritance diagram for fftfbpow::fftfbpow_t:



Public Member Functions

- **fftfbpow_t** (**MHAOvlFilter::fftfb_vars_t** &vars, unsigned int nch, unsigned int nfft, **mha_real_t** fs, **algo_comm_t** ac, std::string name)

  Constructor of the run time configuration.

Public Attributes

- **MHA_AC::waveform_t fbpow**

  AC variable containing the estimated power in each frequency band.

Additional Inherited Members

5.98.1 Detailed Description

Run time configuration for the fftfbpow plugin.

5.98.2 Constructor & Destructor Documentation

5.98.2.1 fftfbpow::fftfbpow_t::fftfbpow_t (MHAOvlFilter::fftfb_vars_t & *vars,* unsigned int *nch,* unsigned int *nfft,* mha_real_t *fs,* algo_comm_t *ac,* std::string *name*)

Constructor of the run time configuration.

Parameters

vars	Set of configuration variables for FFT-based overlapping filters
nch	Number of audio input channels
nfft	Length of FFT
fs	Sampling rate
ac	AC space
name	Configured name of plugin interface, used as prefix for AC variable names

5.98.3 Member Data Documentation

5.98.3.1 MHA_AC::waveform_t fftfbpow::fftfbpow_t::fbpow

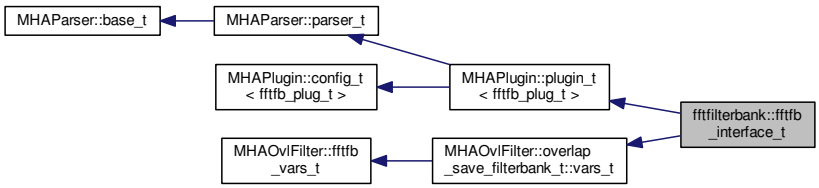
AC variable containing the estimated power in each frequency band.

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

- **fftfbpow.cpp**

5.99 fftfilterbank::fftfb_interface_t Class Reference

Inheritance diagram for fftfilterbank::fftfb_interface_t:



Public Member Functions

- **fftfb_interface_t** (const **algo_comm_t** &**ac**, const std::string &th, const std::string &al)
  Default values are set and MHA configuration variables registered into the parser.
- void **prepare** (**mhaconfig_t** &)
  Prepare all variables for processing.
- void **release** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::bool_t return_imag**
- **MHAEvents::patchbay_t**< **fftfb_interface_t** > **patchbay**
- **MHA_AC::int_t nchannels**
- std::string **algo**
- bool **prepared**
- unsigned int **nbands**

Additional Inherited Members

5.99.1 Constructor & Destructor Documentation

5.99.1.1 fftfilterbank::fftfb_interface_t::fftfb_interface_t (const algo_comm_t & *ac,* const std::string & *th,* const std::string & *al*)

Default values are set and MHA configuration variables registered into the parser.

Parameters

ac	algorithm communication handle
th	chain name
al	algorithm name

5.99.2 Member Function Documentation

5.99.2.1 void fftfilterbank::fftfb_interface_t::prepare (mhaconfig_t & *tf*) [virtual]

Prepare all variables for processing.

In this function, all variables are initialised and the filter shapes for each band are calculated. The filter shapes $W(f)$ are defined as

$$W(f) = W(T(S(f))) = W(x), \quad x = T(S(f)) = T(\hat{f}),$$

$W(x)$ beeing a symmetric window function in the interval $[-1, 1]$ and $S(f)$ the transformation from the linear scale to the given frequency scale (see functions in FreqScaleFun). The function $T(\hat{f})$ transforms the frequency range between the center frequencies $[\hat{f}_{k-1}, \hat{f}_k]$ and $[\hat{f}_k, \hat{f}_{k+1}]$ into the interval $[-1, 0]$ and $[0, 1]$, respectively. This function is realised by the function linscale().

Parameters

tf	Channel configuration

Implements **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 730).

5.99.2.2 void fftfilterbank::fftfb_interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 731).

5.99.2.3 mha_spec_t * fftfilterbank::fftfb_interface_t::process (mha_spec_t * *s*)

5.99.2.4 mha_wave_t * fftfilterbank::fftfb_interface_t::process (mha_wave_t * *s*)

5.99.2.5 void fftfilterbank::fftfb_interface_t::update_cfg (void) [private]

5.99.3 Member Data Documentation

5.99.3.1 MHAParser::bool_t fftfilterbank::fftfb_interface_t::return_imag [private]

5.99.3.2 MHAEvents::patchbay_t<fftfb_interface_t> fftfilterbank::fftfb_interface_t::patchbay [private]

5.99.3.3 MHA_AC::int_t fftfilterbank::fftfb_interface_t::nchannels [private]

5.99.3.4 std::string fftfilterbank::fftfb_interface_t::algo [private]

5.99.3.5 bool fftfilterbank::fftfb_interface_t::prepared [private]

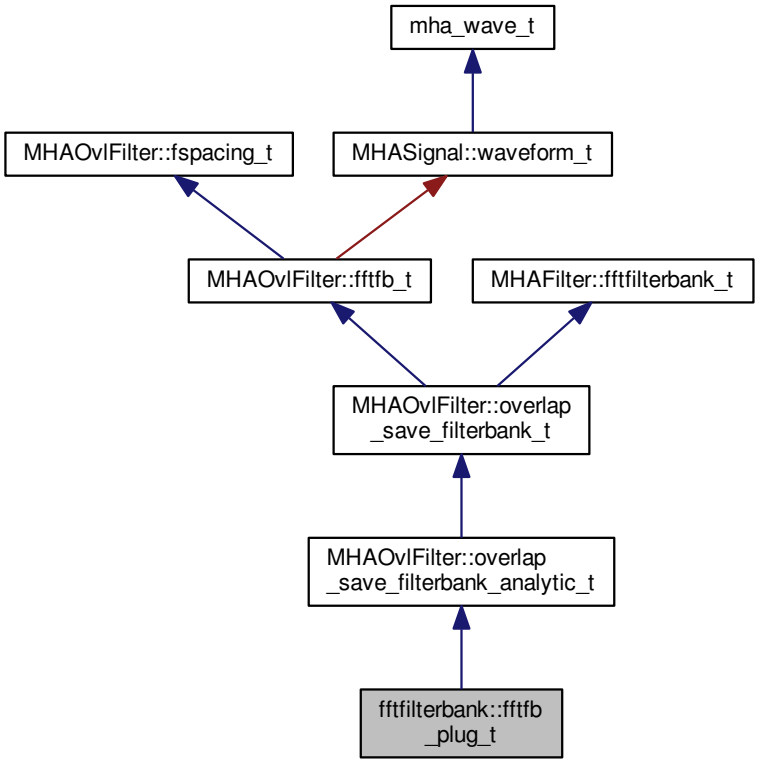
5.99.3.6 unsigned int fftfilterbank::fftfb_interface_t::nbands [private]

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

- **fftfilterbank.cpp**

5.100 fftfilterbank::fftfb_plug_t Class Reference

Inheritance diagram for fftfilterbank::fftfb_plug_t:



Public Member Functions

- **fftfb_plug_t** (**MHAOvlFilter::overlap_save_filterbank_t::vars_t** &, **mhaconfig_t** chcfg, **algo_comm_t** ac, std::string alg, bool return_imag)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **insert** ()

Private Attributes

- **MHAOvlFilter::fftfb_ac_info_t fb_acinfo**
- **MHASignal::spectrum_t s_out**
- **MHA_AC::waveform_t imag**
- bool **return_imag_**

Additional Inherited Members

5.100.1 Constructor & Destructor Documentation

5.100.1.1 fftfilterbank::fftfb_plug_t::fftfb_plug_t (

MHAOvlFilter::overlap_save_filterbank_t::vars_t & *vars,*
mhaconfig_t *chcfg,*
algo_comm_t *ac,*
std::string *alg,*
bool *return_imag*)

5.100.2 Member Function Documentation

5.100.2.1 mha_spec_t ∗ fftfilterbank::fftfb_plug_t::process (

mha_spec_t ∗ *s*)

5.100.2.2 mha_wave_t ∗ fftfilterbank::fftfb_plug_t::process (

mha_wave_t ∗ *s*)

5.100.2.3 void fftfilterbank::fftfb_plug_t::insert ( )

5.100.3 Member Data Documentation

5.100.3.1 MHAOvlFilter::fftfb_ac_info_t fftfilterbank::fftfb_plug_t::fb_acinfo [private]

5.100.3.2 MHASignal::spectrum_t fftfilterbank::fftfb_plug_t::s_out [private]

5.100.3.3 MHA_AC::waveform_t fftfilterbank::fftfb_plug_t::imag [private]

5.100.3.4 bool fftfilterbank::fftfb_plug_t::return_imag_ [private]

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

- **fftfilterbank.cpp**

5.101 fshift::fshift_config_t Class Reference

fshift runtime config class

Public Member Functions

- **fshift_config_t** (**fshift_t** const ∗const plug)
  C'tor of the fshift plugin runtime configuration class.
- ∼**fshift_config_t** ()=default
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- const unsigned int **kmin**
  FFT bin corresponding to fmin.
- const unsigned **kmax**
  FFT bin corresponding to fmax.
- const int **df**
  Frequency shift expressed in FFT bins.
- const **mha_complex_t delta_phi**
  Phase advance per fft frame.
- **mha_complex_t delta_phi_total**
  Sum of all phase advances.

5.101.1 Detailed Description

fshift runtime config class

5.101.2 Constructor & Destructor Documentation

5.101.2.1 fshift::fshift_config_t::fshift_config_t (fshift_t const ∗const *plug*) [explicit]

C'tor of the fshift plugin runtime configuration class.

Parameters

plug	ptr to the plugin interface class. Configuration information is given this way to keep the argument list small.

5.101.2.2 fshift::fshift_config_t::~fshift_config_t ( ) [default]

5.101.3 Member Function Documentation

5.101.3.1 mha_spec_t * fshift::fshift_config_t::process (mha_spec_t * *in*)

5.101.4 Member Data Documentation

5.101.4.1 const unsigned int fshift::fshift_config_t::kmin [private]

FFT bin corresponding to fmin.

5.101.4.2 const unsigned fshift::fshift_config_t::kmax [private]

FFT bin corresponding to fmax.

5.101.4.3 const int fshift::fshift_config_t::df [private]

Frequency shift expressed in FFT bins.

5.101.4.4 const mha_complex_t fshift::fshift_config_t::delta_phi [private]

Phase advance per fft frame.

5.101.4.5 mha_complex_t fshift::fshift_config_t::delta_phi_total [private]

Sum of all phase advances.

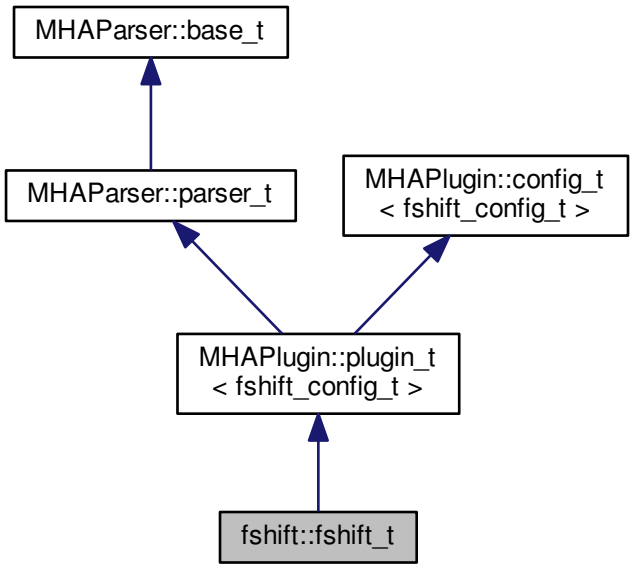
The documentation for this class was generated from the following files:

- **fshift.hh**
- **fshift.cpp**

5.102 fshift::fshift_t Class Reference

fshift plugin interface class

Inheritance diagram for fshift::fshift_t:



Public Member Functions

- **fshift_t** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_↩ name)

  Constructs our plugin.
- ∼**fshift_t** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)
- **mha_real_t fmin** () const
- **mha_real_t fmax** () const
- **mha_real_t df** () const

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< fshift_t > patchbay**
  patch bay for connecting configuration parser events with local member functions:
- **MHAParser::float_t m_fmin**
- **MHAParser::float_t m_fmax**
  upper boundary for frequency shifter
- **MHAParser::float_t m_df**
  Shift frequency in Hz.

Additional Inherited Members

5.102.1 Detailed Description

fshift plugin interface class

5.102.2 Constructor & Destructor Documentation

5.102.2.1 fshift::fshift_t::fshift_t (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.102.2.2 fshift::fshift_t::~fshift_t ( )

5.102.3 Member Function Documentation

5.102.3.1 mha_spec_t * fshift::fshift_t::process (mha_spec_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.102.3.2 void fshift::fshift_t::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< fshift_config_t >** (p. 730).

5.102.3.3 void fshift::fshift_t::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< fshift_config_t >** (p. 731).

5.102.3.4 mha_real_t fshift::fshift_t::fmin ( ) const [inline]

5.102.3.5 mha_real_t fshift::fshift_t::fmax ( ) const [inline]

5.102.3.6 mha_real_t fshift::fshift_t::df ( ) const [inline]

5.102.3.7 void fshift::fshift_t::update_cfg (void) [private]

5.102.4 Member Data Documentation

5.102.4.1 MHAEvents::patchbay_t<fshift_t> fshift::fshift_t::patchbay [private]

patch bay for connecting configuration parser events with local member functions:

5.102.4.2 MHAParser::float_t fshift::fshift_t::m_fmin [private]

5.102.4.3 MHAParser::float_t fshift::fshift_t::m_fmax [private]

upper boundary for frequency shifter

5.102.4.4 MHAParser::float_t fshift::fshift_t::m_df [private]

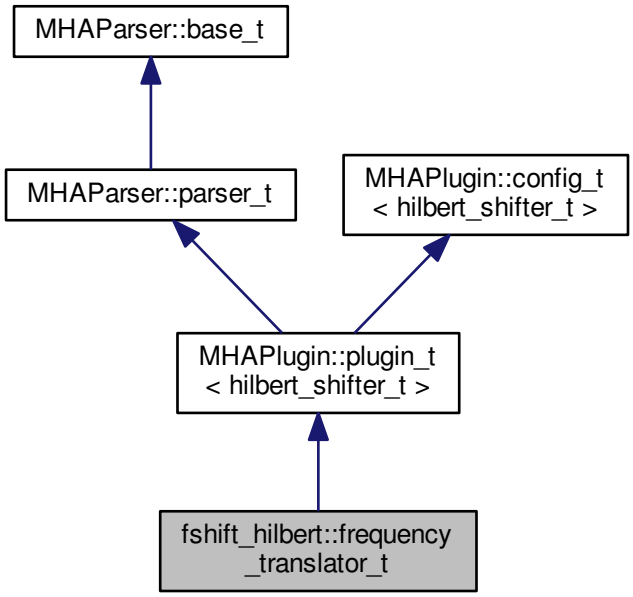
Shift frequency in Hz.

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

- **fshift.hh**
- **fshift.cpp**

5.103 fshift_hilbert::frequency_translator_t Class Reference

Inheritance diagram for fshift_hilbert::frequency_translator_t:



Public Member Functions

- **frequency_translator_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **frequency_translator_t** > **patchbay**
- **MHAParser::vfloat_t df**

  Vector containing the shift frequencies in Hz.
- **MHAParser::float_t fmin**

  Lower boundary for frequency shifter.

- **MHAParser::float_t fmax**

  Upper boundary for frequency shifter.
- **MHAParser::int_t irslen**

  Maximum length of cut off filter response.
- **MHAParser::kw_t phasemode**

  Mode of gain smoothing.

Additional Inherited Members

5.103.1 Constructor & Destructor Documentation

5.103.1.1 fshift_hilbert::frequency_translator_t::frequency_translator_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.103.2 Member Function Documentation

5.103.2.1 mha_spec_t * fshift_hilbert::frequency_translator_t::process (mha_spec_t * *s*)

5.103.2.2 void fshift_hilbert::frequency_translator_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< hilbert_shifter_t >** (p. 730).

5.103.2.3 void fshift_hilbert::frequency_translator_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< hilbert_shifter_t >** (p. 731).

5.103.2.4 void fshift_hilbert::frequency_translator_t::update ( ) [private]

5.103.3 Member Data Documentation

5.103.3.1 MHAEvents::patchbay_t<frequency_translator_t> fshift_hilbert::frequency_translator_t::patchbay [private]

5.103.3.2 MHAParser::vfloat_t fshift_hilbert::frequency_translator_t::df [private]

Vector containing the shift frequencies in Hz.

5.103.3.3 MHAParser::float_t fshift_hilbert::frequency_translator_t::fmin [private]

Lower boundary for frequency shifter.

5.103.3.4 MHAParser::float_t fshift_hilbert::frequency_translator_t::fmax [private]

Upper boundary for frequency shifter.

5.103.3.5 MHAParser::int_t fshift_hilbert::frequency_translator_t::irslen [private]

Maximum length of cut off filter response.

5.103.3.6 MHAParser::kw_t fshift_hilbert::frequency_translator_t::phasemode [private]

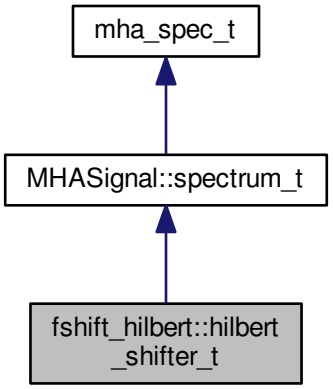
Mode of gain smoothing.

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

- **fshift_hilbert.cpp**

5.104 fshift_hilbert::hilbert_shifter_t Class Reference

Inheritance diagram for fshift_hilbert::hilbert_shifter_t:



Public Member Functions

- **hilbert_shifter_t** (unsigned int fftlen, unsigned int **channels**, **mha_real_t** srate, unsigned int **kmin**, unsigned int **kmax**, std::vector< **mha_real_t** > dphi, unsigned int **frameshift**, unsigned int maxirslen, unsigned int phasemode)
- ~**hilbert_shifter_t** ()
- void **process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::spectrum_t fullspec**

  Part of the spectrum to be frequency shifted.
- **MHASignal::spectrum_t analytic**

  *Analytic signal, defined as a(t)=x(t)+i*H(x(t))*
- **MHASignal::waveform_t shifted**

  The frequency shifted signal in the time domain.
- **MHASignal::spectrum_t mixw_shift**

  Helper varaible containing the coefficients used to split the spectrum.
- **MHASignal::spectrum_t mixw_ref**

  Helper varaible containing the coefficients used to split the spectrum.
- fftw_plan **plan_spec2analytic**

  FFT plan for the transformation of fullspec into the time domain.
- **mha_fft_t mhafft**

  MHA wrapper object for fftw.
- **MHASignal::waveform_t df**

  Vector holding one delta f value for every channel.
- unsigned int **kmin**

  FFT frame that corresponds to f_min.
- unsigned int **kmax**

  FFT frame that corresponds to f_max.
- unsigned int **frameshift**

  Total phase advance within one fragment.
- std::vector< **mha_complex_t** > **delta_phi**

  Phase advance per frame.
- std::vector< **mha_complex_t** > **delta_phi_total**

  Sum of all phase advances.

Additional Inherited Members

5.104.1 Constructor & Destructor Documentation

5.104.1.1 fshift_hilbert::hilbert_shifter_t::hilbert_shifter_t (

unsigned int *fftlen,*
unsigned int *channels,*
mha_real_t *srate,*
unsigned int *kmin,*
unsigned int *kmax,*
std::vector< **mha_real_t** > *dphi,*
unsigned int *frameshift,*
unsigned int *maxirslen,*
unsigned int *phasemode*)

5.104.1.2 fshift_hilbert::hilbert_shifter_t::~hilbert_shifter_t ( )

5.104.2 Member Function Documentation

5.104.2.1 void fshift_hilbert::hilbert_shifter_t::process (

mha_spec_t * *s*)

5.104.3 Member Data Documentation

5.104.3.1 MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::fullspec [private]

Part of the spectrum to be frequency shifted.

5.104.3.2 MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::analytic [private]

Analytic signal, defined as a(t)=x(t)+i∗H(x(t))

5.104.3.3 MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::shifted [private]

The frequency shifted signal in the time domain.

5.104.3.4 MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::mixw_shift [private]

Helper varaible containing the coefficients used to split the spectrum.

Contains 1 for every fft bin to be frequency shifted, 0 for all others

5.104.3.5 MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::mixw_ref [private]

Helper varaible containing the coefficients used to split the spectrum.

Contains 0 for every fft bin to be frequency shifted, 1 for all others

5.104.3.6 fftw_plan fshift_hilbert::hilbert_shifter_t::plan_spec2analytic [private]

FFT plan for the transformation of fullspec into the time domain.

5.104.3.7 mha_fft_t fshift_hilbert::hilbert_shifter_t::mhafft [private]

MHA wrapper object for fftw.

5.104.3.8 MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::df [private]

Vector holding one delta f value for every channel.

5.104.3.9 unsigned int fshift_hilbert::hilbert_shifter_t::kmin [private]

FFT frame that corresponds to f_min.

5.104.3.10 unsigned int fshift_hilbert::hilbert_shifter_t::kmax [private]

FFT frame that corresponds to f_max.

5.104.3.11 unsigned int fshift_hilbert::hilbert_shifter_t::frameshift [private]

Total phase advance within one fragment.

5.104.3.12 std::vector<mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi [private]

Phase advance per frame.

5.104.3.13 std::vector<mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi_total [private]

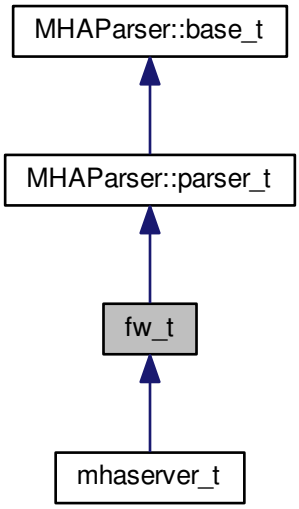
Sum of all phase advances.

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

- **fshift_hilbert.cpp**

5.105 fw_t Class Reference

Inheritance diagram for fw_t:



Public Member Functions

- **fw_t** ()
- ~**fw_t** ()
- bool **exit_request** () const

Protected Attributes

- int **proc_error**
- int **io_error**

Private Types

Private Member Functions

- void **prepare** ()
  preparation for processing
- void **start** ()
  start of processing
- void **stop** ()

stop/pause of processing

- void **release** ()

  release of IO device

- void **quit** ()

  controlled quit

- void **stopped** (int, int)
- void **started** ()
- int **process** (**mha_wave_t** ∗, **mha_wave_t** ∗∗)
- void **exec_fw_command** ()
- void **load_proc_lib** ()
- void **load_io_lib** ()
- void **fw_sleep_cmd** ()
- void **fw_until_cmd** ()
- void **get_input_signal_dimension** ()
- void **async_read** ()
- void **async_poll_msg** ()
- void **get_parserstate** ()

Static Private Member Functions

- static void **stopped** (void ∗h, int proc_err, int io_err)
- static void **started** (void ∗h)
- static int **process** (void ∗h, **mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)

Private Attributes

- **fw_vars_t prepare_vars**
- **MHAParser::int_mon_t nchannels_out**
- **MHAParser::string_t proc_name**
- **MHAParser::string_t io_name**
- **MHAParser::bool_t exit_on_stop**
- **MHAParser::int_t fw_sleep**
- **MHAParser::string_t fw_until**
- **MHAParser::kw_t fw_cmd**
- **MHAParser::string_mon_t parserstate**
- **MHAParser::string_t errorlog**
- **MHAParser::string_t fatallog**
- **MHAParser::vstring_t plugins**
- **MHAParser::vstring_t plugin_paths**
- **MHAParser::bool_t dump_mha**
- **MHAParser::string_t inst_name**

  A variable for naming MHA instances.

- **MHAKernel::algo_comm_class_t ac**
- **PluginLoader::mhapluginloader_t ∗ proc_lib**
- **io_lib_t ∗ io_lib**

- **mhaconfig_t cfin**
- **mhaconfig_t cfout**
- **state_t state**
- bool **b_exit_request**
- **MHAParser::string_mon_t proc_error_string**
- **MHAEvents::patchbay_t< fw_t > patchbay**

Additional Inherited Members

5.105.1 Member Enumeration Documentation

5.105.1.1 enum fw_t::state_t [private]

Enumerator

fw_unprepared

fw_stopped

fw_starting

fw_running

fw_stopping

fw_exiting

5.105.2 Constructor & Destructor Documentation

5.105.2.1 fw_t::fw_t ( )

5.105.2.2 fw_t::~fw_t ( )

5.105.3 Member Function Documentation

5.105.3.1 bool fw_t::exit_request ( ) const [inline]

5.105.3.2 void fw_t::prepare (void) [private]

preparation for processing

5.105.3.3 void fw_t::start ( ) [private]

start of processing

5.105.3.4 void fw_t::stop ( ) [private]

stop/pause of processing

5.105.3.5 void fw_t::release ( ) [private]

release of IO device

5.105.3.6 void fw_t::quit ( ) [private]

controlled quit

5.105.3.7 static void fw_t::stopped (
void * *h,*
int *proc_err,*
int *io_err*) [inline], [static], [private]

5.105.3.8 static void fw_t::started (
void * *h*) [inline], [static], [private]

5.105.3.9 static int fw_t::process (
void * *h,*
mha_wave_t * *sIn,*
mha_wave_t ** *sOut*) [inline], [static], [private]

5.105.3.10 void fw_t::stopped (
int *proc_err,*
int *io_err*) [private]

5.105.3.11 void fw_t::started ( ) [private]

5.105.3.12 int fw_t::process (
mha_wave_t * *s_in,*
mha_wave_t ** *s_out*) [private]

5.105.3.13 void fw_t::exec_fw_command ( ) [private]

5.105.3.14 void fw_t::load_proc_lib ( ) [private]

5.105.3.15 void fw_t::load_io_lib ( ) [private]

5.105.3.16 void fw_t::fw_sleep_cmd ( ) [private]

5.105.3.17 void fw_t::fw_until_cmd ( ) [private]

5.105.3.18 void fw_t::get_input_signal_dimension ( ) [private]

5.105.3.19 void fw_t::async_read ( ) [inline], [private]

5.105.3.20 void fw_t::async_poll_msg ( ) [private]

5.105.3.21 void fw_t::get_parserstate ( ) [private]

5.105.4 Member Data Documentation

5.105.4.1 fw_vars_t fw_t::prepare_vars [private]

5.105.4.2 MHAParser::int_mon_t fw_t::nchannels_out [private]

5.105.4.3 MHAParser::string_t fw_t::proc_name [private]

5.105.4.4 MHAParser::string_t fw_t::io_name [private]

5.105.4.5 MHAParser::bool_t fw_t::exit_on_stop [private]

5.105.4.6 MHAParser::int_t fw_t::fw_sleep [private]

5.105.4.7 MHAParser::string_t fw_t::fw_until [private]

5.105.4.8 MHAParser::kw_t fw_t::fw_cmd [private]

5.105.4.9 MHAParser::string_mon_t fw_t::parserstate [private]

5.105.4.10 MHAParser::string_t fw_t::errorlog [private]

5.105.4.11 MHAParser::string_t fw_t::fatallog [private]

5.105.4.12 MHAParser::vstring_t fw_t::plugins [private]

5.105.4.13 MHAParser::vstring_t fw_t::plugin_paths [private]

5.105.4.14 MHAParser::bool_t fw_t::dump_mha [private]

5.105.4.15 MHAParser::string_t fw_t::inst_name [private]

A variable for naming MHA instances.

5.105.4.16 MHAKernel::algo_comm_class_t fw_t::ac [private]

5.105.4.17 PluginLoader::mhapluginloader_t* fw_t::proc_lib [private]

5.105.4.18 io_lib_t* fw_t::io_lib [private]

5.105.4.19 mhaconfig_t fw_t::cfin [private]

5.105.4.20 mhaconfig_t fw_t::cfout [private]

5.105.4.21 state_t fw_t::state [private]

5.105.4.22 bool fw_t::b_exit_request [private]

5.105.4.23 int fw_t::proc_error [protected]

5.105.4.24 int fw_t::io_error [protected]

5.105.4.25 MHAParser::string_mon_t fw_t::proc_error_string [private]

5.105.4.26 MHAEvents::patchbay_t<fw_t> fw_t::patchbay [private]

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

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.106 fw_vars_t Class Reference

Public Member Functions

- **fw_vars_t** (**MHAParser::parser_t** &)
- void **lock_srate_fragsize** ()
- void **lock_channels** ()
- void **unlock_srate_fragsize** ()
- void **unlock_channels** ()

Public Attributes

- **MHAParser::int_t pinchannels**
- **MHAParser::int_t pfragmentsize**
- **MHAParser::float_t psrate**

5.106.1 Constructor & Destructor Documentation

5.106.1.1 fw_vars_t::fw_vars_t (MHAParser::parser_t & *p*)

5.106.2 Member Function Documentation

5.106.2.1 void fw_vars_t::lock_srate_fragsize ( )

5.106.2.2 void fw_vars_t::lock_channels ( )

5.106.2.3 void fw_vars_t::unlock_srate_fragsize ( )

5.106.2.4 void fw_vars_t::unlock_channels ( )

5.106.3 Member Data Documentation

5.106.3.1 MHAParser::int_t fw_vars_t::pinchannels

5.106.3.2 MHAParser::int_t fw_vars_t::pfragmentsize

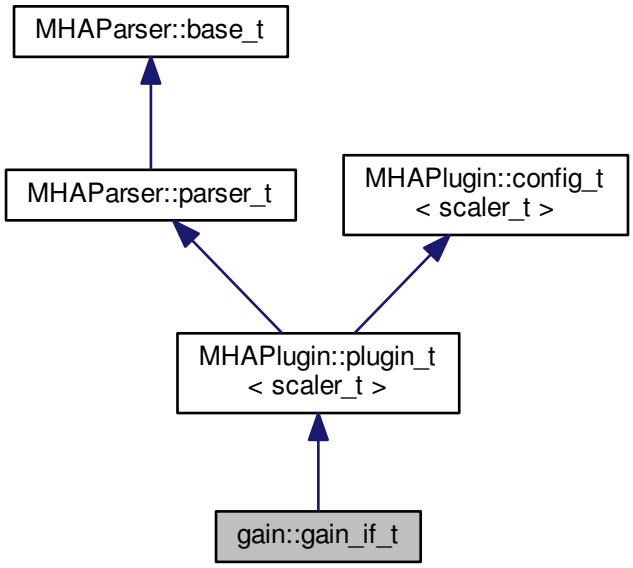
5.106.3.3 MHAParser::float_t fw_vars_t::psrate

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

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.107 gain::gain_if_t Class Reference

Inheritance diagram for gain::gain_if_t:



Public Member Functions

- **gain_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update_gain** ()
- void **update_bbgain** ()
- void **update_minmax** ()

Private Attributes

- **MHAEvents::patchbay_t**< **gain_if_t** > **patchbay**
- **MHAParser::vfloat_t gains**
- **MHAParser::float_t bbgain**
- **MHAParser::float_t vmin**
- **MHAParser::float_t vmax**

Additional Inherited Members

5.107.1 Constructor & Destructor Documentation

5.107.1.1 gain::gain_if_t::gain_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.107.2 Member Function Documentation

5.107.2.1 mha_wave_t * gain::gain_if_t::process (mha_wave_t * *s*)

5.107.2.2 mha_spec_t * gain::gain_if_t::process (mha_spec_t * *s*)

5.107.2.3 void gain::gain_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< scaler_t >** (p. 730).

5.107.2.4 void gain::gain_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< scaler_t >** (p. 731).

5.107.2.5 void gain::gain_if_t::update_gain ( ) [private]

5.107.2.6 void gain::gain_if_t::update_bbgain ( ) [private]

5.107.2.7 void gain::gain_if_t::update_minmax ( ) [private]

5.107.3 Member Data Documentation

5.107.3.1 MHAEvents::patchbay_t<gain_if_t> gain::gain_if_t::patchbay [private]

5.107.3.2 MHAParser::vfloat_t gain::gain_if_t::gains [private]

5.107.3.3 MHAParser::float_t gain::gain_if_t::bbgain [private]

5.107.3.4 MHAParser::float_t gain::gain_if_t::vmin [private]

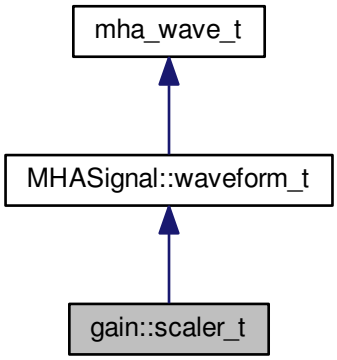
5.107.3.5 MHAParser::float_t gain::gain_if_t::vmax [private]

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

- **gain.cpp**

5.108 gain::scaler_t Class Reference

Inheritance diagram for gain::scaler_t:



Public Member Functions

- **scaler_t** (const unsigned int &**channels**, const **MHAParser::vfloat_t** &gains)

Additional Inherited Members

5.108.1 Constructor & Destructor Documentation

5.108.1.1 gain::scaler_t::scaler_t (const unsigned int & *channels,* const **MHAParser::vfloat_t** & *gains*)

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

- **gain.cpp**

5.109 gtfb_analyzer::gtfb_analyzer_cfg_t Struct Reference

Configuration for Gammatone Filterbank Analyzer.

Public Member Functions

- unsigned **bands** () const
  Each band is split into this number of bands.
- unsigned **channels** () const
  The number of separate audio channels.
- unsigned **frames** () const
  The number of frames in one chunk.
- **mha_complex_t** * **states** (unsigned channel, unsigned band)
  Returns pointer to filter states for that band.
- **gtfb_analyzer_cfg_t** (unsigned ch, unsigned **frames**, unsigned ord, const std::vector< **mha_complex_t** > &_coeff, const std::vector< **mha_complex_t** > &_norm_phase)
  Create a configuration for Gammatone Filterbank Analyzer.
- ~**gtfb_analyzer_cfg_t** ()
- **mha_complex_t** & **cvalue** (unsigned frame, unsigned channel, unsigned band)

Public Attributes

- unsigned **order**
  The order of the gammatone filters.
- std::vector< **mha_complex_t** > **coeff**
  The complex coefficients of the gammatone filter bands.
- std::vector< **mha_complex_t** > **norm_phase**
  Combination of normalization and phase correction factor.
- **mha_wave_t s_out**
  Storage for the (complex) output signal.
- std::vector< **mha_complex_t** > **state**
  Storage for Filter state.

5.109.1 Detailed Description

Configuration for Gammatone Filterbank Analyzer.

5.109.2 Constructor & Destructor Documentation

5.109.2.1 gtfb_analyzer::gtfb_analyzer_cfg_t::gtfb_analyzer_cfg_t (unsigned *ch,* unsigned *frames,* unsigned *ord,* const std::vector< **mha_complex_t** > & *_coeff,* const std::vector< **mha_complex_t** > & *_norm_phase*) [inline]

Create a configuration for Gammatone Filterbank Analyzer.

Parameters

ch	Number of Audio channels.
frames	Number of Audio frames per chunk.
ord	The order of the gammatone filters.
_coeff	Complex gammatone filter coefficients.
_norm_phase	Normalization and phase correction factors.

5.109.2.2 gtfb_analyzer::gtfb_analyzer_cfg_t::~gtfb_analyzer_cfg_t ( ) [inline]

5.109.3 Member Function Documentation

5.109.3.1 unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::bands ( ) const [inline]

Each band is split into this number of bands.

5.109.3.2 unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::channels ( ) const [inline]

The number of separate audio channels.

5.109.3.3 unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::frames ( ) const [inline]

The number of frames in one chunk.

5.109.3.4 mha_complex_t* gtfb_analyzer::gtfb_analyzer_cfg_t::states (unsigned *channel,* unsigned *band*) [inline]

Returns pointer to filter states for that band.

5.109.3.5 mha_complex_t& gtfb_analyzer::gtfb_analyzer_cfg_t::cvalue (unsigned *frame,* unsigned *channel,* unsigned *band*) [inline]

5.109.4 Member Data Documentation

5.109.4.1 unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::order

The order of the gammatone filters.

5.109.4.2 std::vector<mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::coeff

The complex coefficients of the gammatone filter bands.

5.109.4.3 std::vector<mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::norm_phase

Combination of normalization and phase correction factor.

5.109.4.4 mha_wave_t gtfb_analyzer::gtfb_analyzer_cfg_t::s_out

Storage for the (complex) output signal.

Each of the real input audio channels is split into frequency bands with complex time signal output. The split complex time signal is again stored in a **mha_wave_t** (p. 496) buffer. Each complex time signal is stored as adjacent real and imaginary channels. Complex output from one source channel is stored in adjacent complex output channels.

Example: If the input has 2 channels ch0 ch1, and **gtfb_analyzer** (p. 88) splits into 3 bands b0 b1 b2, then the order of output channels in s_out is: ch0_b0_real ch0_b0_imag ch0_b1_real ch0_b1_imag ch0_b2_real ch0_b2_imag ch1_b0_real ch1_b1_imag ch1_b1_real ch1_b1_↩ imag ch1_b2_real ch1_b2_imag

5.109.4.5 std::vector<mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::state

Storage for Filter state.

Holds **channels()** (p. 354) ∗ **bands()** (p. 354) ∗ order complex filter states. Layout↩ : state[(**bands()** (p. 354)∗channel+band)∗order+stage]

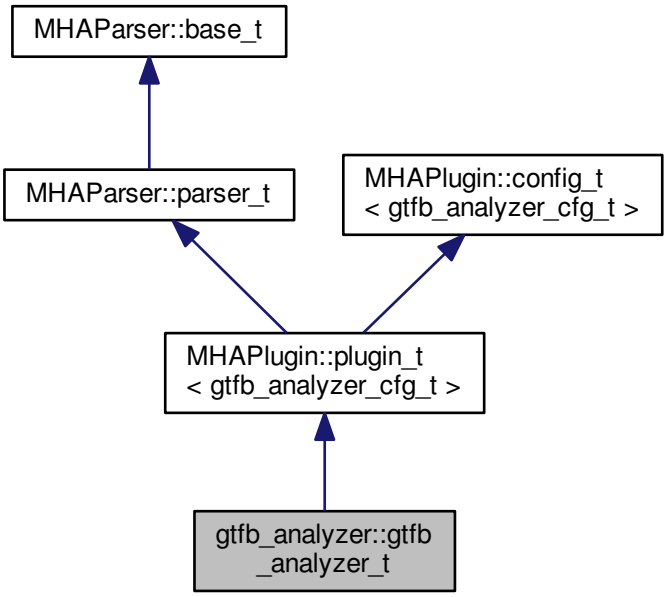
The documentation for this struct was generated from the following file:

- **gtfb_analyzer.cpp**

5.110 gtfb_analyzer::gtfb_analyzer_t Class Reference

Gammatone Filterbank Analyzer Plugin.

Inheritance diagram for gtfb_analyzer::gtfb_analyzer_t:



Public Member Functions

- **gtfb_analyzer_t** (const **algo_comm_t** &, const std::string &thread_name, const std↩ ::string &algo_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **gtfb_analyzer_t** > **patchbay**
- bool **prepared**
- **MHAParser::int_t order**
- **MHAParser::vcomplex_t coeff**
- **MHAParser::vcomplex_t norm_phase**

Additional Inherited Members

5.110.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

5.110.2 Constructor & Destructor Documentation

5.110.2.1 gtfb_analyzer::gtfb_analyzer_t::gtfb_analyzer_t (const algo_comm_t & *iac,* const std::string & *thread_name,* const std::string & *algo_name*)

5.110.3 Member Function Documentation

5.110.3.1 mha_wave_t * gtfb_analyzer::gtfb_analyzer_t::process (mha_wave_t * *s*)

5.110.3.2 void gtfb_analyzer::gtfb_analyzer_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >** (p. 730).

5.110.3.3 void gtfb_analyzer::gtfb_analyzer_t::update_cfg (void) [private]

5.110.4 Member Data Documentation

5.110.4.1 MHAEvents::patchbay_t<gtfb_analyzer_t> gtfb_analyzer::gtfb_analyzer_t::patchbay [private]

5.110.4.2 bool gtfb_analyzer::gtfb_analyzer_t::prepared [private]

5.110.4.3 MHAParser::int_t gtfb_analyzer::gtfb_analyzer_t::order [private]

5.110.4.4 MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t::coeff [private]

5.110.4.5 MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t::norm_phase [private]

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

- **gtfb_analyzer.cpp**

5.111 hanning_ramps_t Class Reference

Public Member Functions

- **hanning_ramps_t** (unsigned int, unsigned int)
- ∼**hanning_ramps_t** ()
- void **operator()** (**MHASignal::waveform_t** &)

Private Attributes

- unsigned int **len_a**
- unsigned int **len_b**
- **mha_real_t** ∗ **ramp_a**
- **mha_real_t** ∗ **ramp_b**

5.111.1 Constructor & Destructor Documentation

5.111.1.1 hanning_ramps_t::hanning_ramps_t (unsigned int *la,* unsigned int *lb*)

5.111.1.2 hanning_ramps_t::∼hanning_ramps_t (void)

5.111.2 Member Function Documentation

5.111.2.1 void hanning_ramps_t::operator() (MHASignal::waveform_t & *b*)

5.111.3 Member Data Documentation

5.111.3.1 unsigned int hanning_ramps_t::len_a [private]

5.111.3.2 unsigned int hanning_ramps_t::len_b [private]

5.111.3.3 mha_real_t∗ hanning_ramps_t::ramp_a [private]

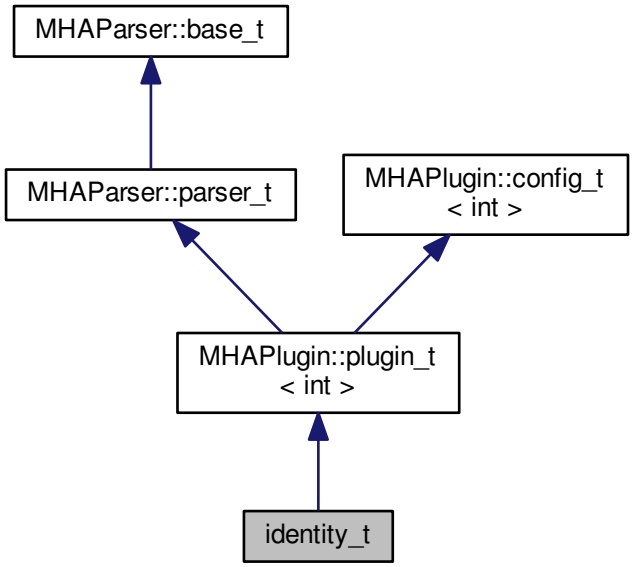
5.111.3.4 mha_real_t∗ hanning_ramps_t::ramp_b [private]

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

- **spec2wave.cpp**

5.112 identity_t Class Reference

Inheritance diagram for identity_t:



Public Member Functions

- **identity_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Additional Inherited Members

5.112.1 Constructor & Destructor Documentation

5.112.1.1 identity_t::identity_t (
const **algo_comm_t** & *iac,*
const **std::string** & ,
const **std::string** &)

5.112.2 Member Function Documentation

5.112.2.1 mha_wave_t * identity_t::process (mha_wave_t * *s*)

5.112.2.2 mha_spec_t * identity_t::process (mha_spec_t * *s*)

5.112.2.3 void identity_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.112.2.4 void identity_t::release (void) [virtual]

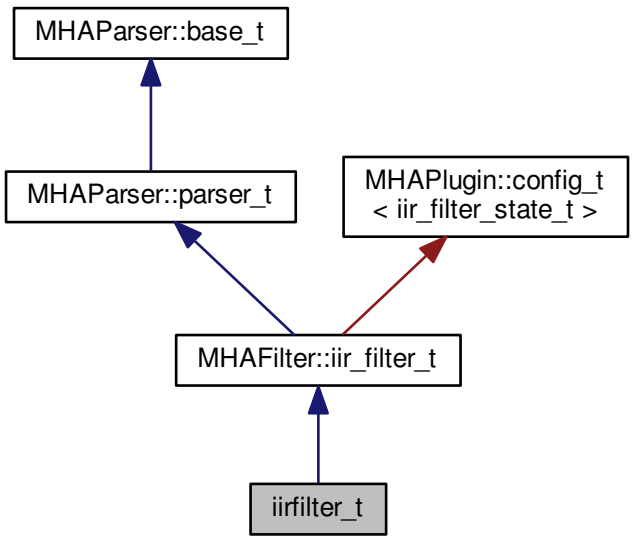
Reimplemented from **MHAPlugin::plugin_t< int >** (p. 731).

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

- **identity.cpp**

5.113 iirfilter_t Class Reference

Inheritance diagram for iirfilter_t:



Public Member Functions

- **iirfilter_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare_** (**mhaconfig_t** &)
- void **release_** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Additional Inherited Members

5.113.1 Constructor & Destructor Documentation

5.113.1.1 iirfilter_t::iirfilter_t (
const **algo_comm_t** & *ac,*
const std::string & *th,*
const std::string & *al*)

5.113.2 Member Function Documentation

5.113.2.1 void iirfilter_t::prepare_ (
mhaconfig_t & *tf*)

5.113.2.2 void iirfilter_t::release_ ( ) [inline]

5.113.2.3 mha_wave_t ∗ iirfilter_t::process (
mha_wave_t ∗ *s*)

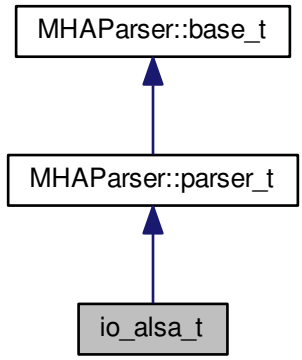
The documentation for this class was generated from the following file:

- **iirfilter.cpp**

5.114 io_alsa_t Class Reference

MHA IO interface class for ALSA IO.

Inheritance diagram for io_alsa_t:



Public Member Functions

- **io_alsa_t** (unsigned int fragsize, float samplerate, **IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)

  Constructor, receives the callback handles to interact with the MHA framework.
- template<typename T = void>
  void **prepare** (int, int)

  Called after the framework has perpared the processing plugins and the number of input and output channels are fixed.
- void **release** ()

  MHA framework leaves prepared state.
- void **start** ()

  MHA framework calls this function when signal processing should start.
- void **stop** ()

  MHA framework calls this function when signal processing should stop.
- template<>
  void **prepare** (int nch_in, int nch_out)

Static Public Member Functions

- static void ∗ **thread_start** (void ∗h)

  MHAIOAlsa uses a separate thread that calls the alsa read and write functions to read and write audio samples, these functions are blocking until samples can be read or written.

Private Member Functions

- void **process** ()

Private Attributes

- bool **b_process**
- unsigned int **fw_fragsize**
- unsigned int **fw_samplerate**
- **IOProcessEvent_t proc_event**
- void ∗ **proc_handle**
- **IOStartedEvent_t start_event**
- void ∗ **start_handle**
- **IOStoppedEvent_t stop_event**
- void ∗ **stop_handle**
- **alsa_base_t** ∗ **dev_in**
- **alsa_base_t** ∗ **dev_out**
- pthread_t **proc_thread**
- **alsa_dev_par_parser_t p_in**
- **alsa_dev_par_parser_t p_out**
- **MHAParser::bool_t pcmlink**
- **MHAParser::int_t priority**
- **MHAParser::kw_t format**
- **MHAParser::int_mon_t alsa_start_counter**
- **MHAEvents::patchbay_t**< **io_alsa_t** > **patchbay**

Additional Inherited Members

5.114.1 Detailed Description

MHA IO interface class for ALSA IO.

5.114.2 Constructor & Destructor Documentation

5.114.2.1 io_alsa_t::io_alsa_t (

unsigned int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

Constructor, receives the callback handles to interact with the MHA framework.

5.114.3 Member Function Documentation

5.114.3.1 template<typename T > void io_alsa_t::prepare (

int *nch_in,*
int *nch_out*)

Called after the framework has perpared the processing plugins and the number of input and output channels are fixed.

open pcm streams

5.114.3.2 void io_alsa_t::release (

void)

MHA framework leaves prepared state.

5.114.3.3 void io_alsa_t::start ( )

MHA framework calls this function when signal processing should start.

5.114.3.4 void io_alsa_t::stop ( )

MHA framework calls this function when signal processing should stop.

5.114.3.5 void * io_alsa_t::thread_start (void * *h*) [static]

MHAIOAlsa uses a separate thread that calls the alsa read and write functions to read and write audio samples, these functions are blocking until samples can be read or written.

This is the start function of that thread.

5.114.3.6 void io_alsa_t::process (void) [private]

5.114.3.7 template<> void io_alsa_t::prepare (int *nch_in,* int *nch_out*)

5.114.4 Member Data Documentation

5.114.4.1 bool io_alsa_t::b_process [private]

5.114.4.2 unsigned int io_alsa_t::fw_fragsize [private]

5.114.4.3 unsigned int io_alsa_t::fw_samplerate [private]

5.114.4.4 IOProcessEvent_t io_alsa_t::proc_event [private]

5.114.4.5 void* io_alsa_t::proc_handle [private]

5.114.4.6 IOStartedEvent_t io_alsa_t::start_event [private]

5.114.4.7 void* io_alsa_t::start_handle [private]

5.114.4.8 IOStoppedEvent_t io_alsa_t::stop_event [private]

5.114.4.9 void* io_alsa_t::stop_handle [private]

5.114.4.10 alsa_base_t* io_alsa_t::dev_in [private]

5.114.4.11 alsa_base_t* io_alsa_t::dev_out [private]

5.114.4.12 pthread_t io_alsa_t::proc_thread [private]

5.114.4.13 alsa_dev_par_parser_t io_alsa_t::p_in [private]

5.114.4.14 alsa_dev_par_parser_t io_alsa_t::p_out [private]

5.114.4.15 MHAParser::bool_t io_alsa_t::pcmlink [private]

5.114.4.16 MHAParser::int_t io_alsa_t::priority [private]

5.114.4.17 MHAParser::kw_t io_alsa_t::format [private]

5.114.4.18 MHAParser::int_mon_t io_alsa_t::alsa_start_counter [private]

5.114.4.19 MHAEvents::patchbay_t<io_alsa_t> io_alsa_t::patchbay [private]

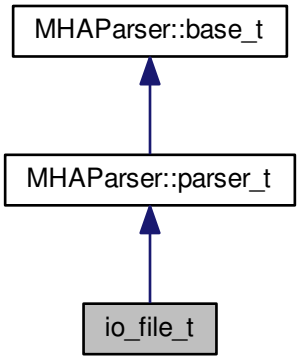
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.115 io_file_t Class Reference

File IO.

Inheritance diagram for io_file_t:



Public Member Functions

- **io_file_t** (int **fragsize**, float **samplerate**, **IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)
- ∼**io_file_t** ()
- void **prepare** (int, int)

  Allocate buffers, activate FILE client and install internal ports.
- void **start** ()
- void **stop** ()
- void **release** ()

  Remove FILE client and deallocate internal ports and buffers.

Private Member Functions

- void **stopped** (int, int)
- void **setlock** (bool locked)

  lock or unlock all parser variables.

Private Attributes

- int **fragsize**
- float **samplerate**
- int **nchannels_in**
- int **nchannels_file_in**
- int **nchannels_out**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- **MHAParser::string_t filename_input**
- **MHAParser::string_t filename_output**
- **MHAParser::kw_t output_sample_format**
- **MHAParser::int_t startsample**
- **MHAParser::int_t length**
- **MHAParser::bool_t strict_channel_match**
- **MHAParser::bool_t strict_srate_match**
- **MHASignal::waveform_t * s_in**
- **MHASignal::waveform_t * s_file_in**
- **mha_wave_t * s_out**
- bool **b_prepared**
- SNDFILE * **sf_in**
- SNDFILE * **sf_out**
- SF_INFO **sfinf_in**
- SF_INFO **sfinf_out**
- sf_count_t **total_read**

Additional Inherited Members

5.115.1 Detailed Description

File IO.

5.115.2 Constructor & Destructor Documentation

5.115.2.1 io_file_t::io_file_t (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

5.115.2.2 io_file_t::~io_file_t ( )

5.115.3 Member Function Documentation

5.115.3.1 void io_file_t::prepare (int *nch_in,* int *nch_out*)

Allocate buffers, activate FILE client and install internal ports.

5.115.3.2 void io_file_t::start ( )

5.115.3.3 void io_file_t::stop ( )

5.115.3.4 void io_file_t::release (void)

Remove FILE client and deallocate internal ports and buffers.

5.115.3.5 void io_file_t::stopped (int *proc_err,* int *io_err*) [private]

5.115.3.6 void io_file_t::setlock (bool *locked*) [private]

lock or unlock all parser variables.

Used in prepare/release.

Parameters

locked	When true, locks. When false, unlocks.

5.115.4 Member Data Documentation

5.115.4.1 int io_file_t::fragsize [private]

5.115.4.2 float io_file_t::samplerate [private]

5.115.4.3 int io_file_t::nchannels_in [private]

5.115.4.4 int io_file_t::nchannels_file_in [private]

5.115.4.5 int io_file_t::nchannels_out [private]

5.115.4.6 IOProcessEvent_t io_file_t::proc_event [private]

5.115.4.7 void* io_file_t::proc_handle [private]

5.115.4.8 IOStartedEvent_t io_file_t::start_event [private]

5.115.4.9 void* io_file_t::start_handle [private]

5.115.4.10 IOStoppedEvent_t io_file_t::stop_event [private]

5.115.4.11 void* io_file_t::stop_handle [private]

5.115.4.12 MHAParser::string_t io_file_t::filename_input [private]

5.115.4.13 MHAParser::string_t io_file_t::filename_output [private]

5.115.4.14 MHAParser::kw_t io_file_t::output_sample_format [private]

5.115.4.15 MHAParser::int_t io_file_t::startsample [private]

5.115.4.16 MHAParser::int_t io_file_t::length [private]

5.115.4.17 MHAParser::bool_t io_file_t::strict_channel_match [private]

5.115.4.18 MHAParser::bool_t io_file_t::strict_srate_match [private]

5.115.4.19 MHASignal::waveform_t* io_file_t::s_in [private]

5.115.4.20 MHASignal::waveform_t* io_file_t::s_file_in [private]

5.115.4.21 mha_wave_t* io_file_t::s_out [private]

5.115.4.22 bool io_file_t::b_prepared [private]

5.115.4.23 SNDFILE* io_file_t::sf_in [private]

5.115.4.24 SNDFILE* io_file_t::sf_out [private]

5.115.4.25 SF_INFO io_file_t::sfinf_in [private]

5.115.4.26 SF_INFO io_file_t::sfinf_out [private]

5.115.4.27 sf_count_t io_file_t::total_read [private]

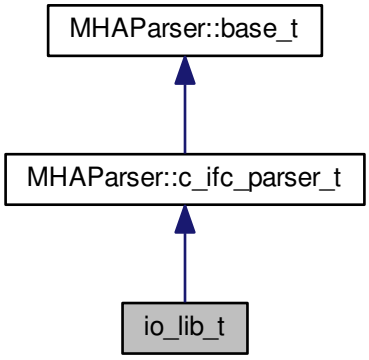
The documentation for this class was generated from the following file:

- **MHAIOFile.cpp**

5.116 io_lib_t Class Reference

Class for loading MHA sound IO module.

Inheritance diagram for io_lib_t:



Public Member Functions

- **io_lib_t** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void ∗proc_↩ handle, **IOStartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_↩ event, void ∗stop_handle, std::string libname)

  load and initialize MHA sound io module.
- ∼**io_lib_t** ()

  Deinitialize and unload this MHA sound io module.
- void **prepare** (unsigned int inch, unsigned int outch)

  Prepare the sound io module.
- void **start** ()

  Tell the sound io module to start sound processing.
- void **stop** ()
- void **release** ()
- std::string **lib_str_error** (int err)

Protected Member Functions

- void **test_error** ()

Protected Attributes

- int **lib_err**
- **dynamiclib_t lib_handle**
- void * **lib_data**
- **IOInit_t IOInit_cb**
- **IOPrepare_t IOPrepare_cb**
- **IOStart_t IOStart_cb**
- **IOStop_t IOStop_cb**
- **IORelease_t IORelease_cb**
- **IOSetVar_t IOSetVar_cb**
- **IOStrError_t IOStrError_cb**
- **IODestroy_t IODestroy_cb**

Additional Inherited Members

5.116.1 Detailed Description

Class for loading MHA sound IO module.

5.116.2 Constructor & Destructor Documentation

5.116.2.1 io_lib_t::io_lib_t (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
std::string *libname*)

load and initialize MHA sound io module.

5.116.2.2 io_lib_t::~io_lib_t ( )

Deinitialize and unload this MHA sound io module.

5.116.3 Member Function Documentation

5.116.3.1 void io_lib_t::prepare (

unsigned int *inch,*
unsigned int *outch*)

Prepare the sound io module.

After preparation, the sound io module may start the sound processing at any time (external trigger). When the sound processing is started, the sound io module will call the start_event callback.

Parameters

inch	number of input channels
outch	number of output channels

5.116.3.2 void io_lib_t::start ( )

Tell the sound io module to start sound processing.

Some io modules need this, for others that wait for external events this method might do nothing.

5.116.3.3 void io_lib_t::stop ( )

5.116.3.4 void io_lib_t::release ( )

5.116.3.5 std::string io_lib_t::lib_str_error (int *err*)

5.116.3.6 void io_lib_t::test_error ( ) [protected]

5.116.4 Member Data Documentation

5.116.4.1 int io_lib_t::lib_err [protected]

5.116.4.2 dynamiclib_t io_lib_t::lib_handle [protected]

5.116.4.3 void* io_lib_t::lib_data [protected]

5.116.4.4 IOInit_t io_lib_t::IOInit_cb [protected]

5.116.4.5 IOPrepare_t io_lib_t::IOPrepare_cb [protected]

5.116.4.6 IOStart_t io_lib_t::IOStart_cb [protected]

5.116.4.7 IOStop_t io_lib_t::IOStop_cb [protected]

5.116.4.8 IORelease_t io_lib_t::IORelease_cb [protected]

5.116.4.9 IOSetVar_t io_lib_t::IOSetVar_cb [protected]

5.116.4.10 IOStrError_t io_lib_t::IOStrError_cb [protected]

5.116.4.11 IODestroy_t io_lib_t::IODestroy_cb [protected]

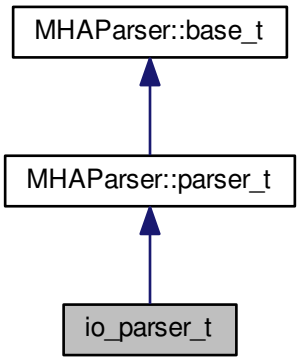
The documentation for this class was generated from the following files:

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.117 io_parser_t Class Reference

Main class for Parser IO.

Inheritance diagram for io_parser_t:



Public Member Functions

- **io_parser_t** (unsigned int **fragsize**, **IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)
- ∼**io_parser_t** ()
- void **prepare** (int, int)

  Allocate buffers, activate JACK client and install internal ports.
- void **start** ()
- void **stop** ()
- void **release** ()

  Remove JACK client and deallocate internal ports and buffers.

Private Member Functions

- void **stopped** (int, int)
- void **started** ()
- void **process_frame** ()

Private Attributes

- unsigned int **fragsize**
- unsigned int **nchannels_in**
- unsigned int **nchannels_out**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- **MHAParser::mfloat_t input**
- **MHAParser::mfloat_mon_t output**
- **MHASignal::waveform_t * s_in**
- **mha_wave_t * s_out**
- bool **b_fw_started**
- bool **b_stopped**
- bool **b_prepared**
- bool **b_starting**
- **MHAEvents::patchbay_t< io_parser_t > patchbay**

Additional Inherited Members

5.117.1 Detailed Description

Main class for Parser IO.

5.117.2 Constructor & Destructor Documentation

5.117.2.1 io_parser_t::io_parser_t (

unsigned int *fragsize,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

5.117.2.2 io_parser_t::~io_parser_t ( )

5.117.3 Member Function Documentation

5.117.3.1 void io_parser_t::prepare (

int *nch_in,*
int *nch_out*)

Allocate buffers, activate JACK client and install internal ports.

5.117.3.2 void io_parser_t::start ( )

5.117.3.3 void io_parser_t::stop ( )

5.117.3.4 void io_parser_t::release (void)

Remove JACK client and deallocate internal ports and buffers.

5.117.3.5 void io_parser_t::stopped (int *proc_err,* int *io_err*) [private]

5.117.3.6 void io_parser_t::started ( ) [private]

5.117.3.7 void io_parser_t::process_frame ( ) [private]

5.117.4 Member Data Documentation

5.117.4.1 unsigned int io_parser_t::fragsize [private]

5.117.4.2 unsigned int io_parser_t::nchannels_in [private]

5.117.4.3 unsigned int io_parser_t::nchannels_out [private]

5.117.4.4 IOProcessEvent_t io_parser_t::proc_event [private]

5.117.4.5 void* io_parser_t::proc_handle [private]

5.117.4.6 IOStartedEvent_t io_parser_t::start_event [private]

5.117.4.7 void* io_parser_t::start_handle [private]

5.117.4.8 IOStoppedEvent_t io_parser_t::stop_event [private]

5.117.4.9 void* io_parser_t::stop_handle [private]

5.117.4.10 MHAParser::mfloat_t io_parser_t::input [private]

5.117.4.11 MHAParser::mfloat_mon_t io_parser_t::output [private]

5.117.4.12 MHASignal::waveform_t* io_parser_t::s_in [private]

5.117.4.13 mha_wave_t* io_parser_t::s_out [private]

5.117.4.14 bool io_parser_t::b_fw_started [private]

5.117.4.15 bool io_parser_t::b_stopped [private]

5.117.4.16 bool io_parser_t::b_prepared [private]

5.117.4.17 bool io_parser_t::b_starting [private]

5.117.4.18 MHAEvents::patchbay_t<io_parser_t> io_parser_t::patchbay [private]

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

- **MHAIOParser.cpp**

5.118 io_tcp_fwcb_t Class Reference

TCP sound-io library's interface to the framework callbacks.

Public Member Functions

- **io_tcp_fwcb_t** (**IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)

  Constructor stores framework handles and initializes error numbers to 0.
- virtual ∼**io_tcp_fwcb_t** ()

  Do-nothing destructor.
- virtual void **start** ()

  Call the framework's start callback.
- virtual int **process** (**mha_wave_t** ∗sIn, **mha_wave_t** ∗&sOut)

  Call the frameworks processing callback.
- virtual void **set_errnos** (int **proc_err**, int **io_err**)

  Save error numbers to use during.
- virtual void **stop** ()

  Call the frameworks stop callback.

Private Attributes

- **IOProcessEvent_t proc_event**

  Pointer to signal processing callback function.
- **IOStartedEvent_t start_event**

  Pointer to start notification callback function.
- **IOStoppedEvent_t stop_event**

  Pointer to stop notification callback function.
- void ∗ **proc_handle**

  Handles belonging to framework.
- void ∗ **start_handle**
- void ∗ **stop_handle**
- int **proc_err**

  Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.
- int **io_err**

5.118.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

5.118.2 Constructor & Destructor Documentation

5.118.2.1 io_tcp_fwcb_t::io_tcp_fwcb_t (

IOProcessEvent_t ***proc_event,***
void ∗ ***proc_handle,***
IOStartedEvent_t ***start_event,***
void ∗ ***start_handle,***
IOStoppedEvent_t ***stop_event,***
void ∗ ***stop_handle***)

Constructor stores framework handles and initializes error numbers to 0.

5.118.2.2 virtual io_tcp_fwcb_t::∼io_tcp_fwcb_t ( ) [inline], [virtual]

Do-nothing destructor.

5.118.3 Member Function Documentation

5.118.3.1 void io_tcp_fwcb_t::start ( ) [virtual]

Call the framework's start callback.

5.118.3.2 int io_tcp_fwcb_t::process (

mha_wave_t ∗ ***sIn,***
mha_wave_t ∗**&** ***sOut***) [virtual]

Call the frameworks processing callback.

Parameters

sIn	The input sound data just received from TCP.
sOut	A pointer to output sound data. Will point to the output sound data storage when the callback finishes.

Returns

Status, an error number from the signal processing callback. If this is != 0, then the connection should be closed.

5.118.3.3 void io_tcp_fwcb_t::set_errnos (

int ***proc_err,***
int ***io_err***) [virtual]

Save error numbers to use during.

See also

stop (p. 377)

Parameters

proc_err	The error number from the

See also

process (p. 376) callback.

Parameters

io_err	The error number from the io library itself.

5.118.3.4 void io_tcp_fwcb_t::stop ( ) [virtual]

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

set_errnos (p. 376).

5.118.4 Member Data Documentation

5.118.4.1 IOProcessEvent_t io_tcp_fwcb_t::proc_event [private]

Pointer to signal processing callback function.

5.118.4.2 IOStartedEvent_t io_tcp_fwcb_t::start_event [private]

Pointer to start notification callback function.

Called when a new TCP connection is established or the user issues the start command while there is a connection.

5.118.4.3 IOStoppedEvent_t io_tcp_fwcb_t::stop_event [private]

Pointer to stop notification callback function.

Called when the connection is closed.

5.118.4.4 void* io_tcp_fwcb_t::proc_handle [private]

Handles belonging to framework.

5.118.4.5 void * io_tcp_fwcb_t::start_handle [private]

5.118.4.6 void * io_tcp_fwcb_t::stop_handle [private]

5.118.4.7 int io_tcp_fwcb_t::proc_err [private]

Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.

MHAIOTCP is notified by the server when the connection has been taken down, and calls

See also

stop (p. 377) from that callback. Within **stop** (p. 377), these error numbers are read again and transmitted to the framework.

5.118.4.8 int io_tcp_fwcb_t::io_err [private]

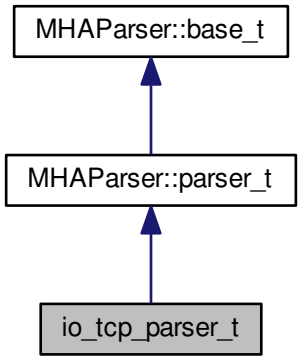
The documentation for this class was generated from the following file:

- **MHAIOTCP.cpp**

5.119 io_tcp_parser_t Class Reference

The parser interface of the IOTCP library.

Inheritance diagram for io_tcp_parser_t:



Public Member Functions

- virtual const std::string & **get_local_address** () const

  Read parser variable local_address, this is the address of the network interface that should listen for incoming connections.
- virtual unsigned short **get_local_port** () const

  Read parser variable local_port, this is the TCP port that should be used for incoming connections.
- virtual void **set_local_port** (unsigned short port)

  Set parser variable local_port.
- virtual bool **get_server_port_open** () const

  Return the status of the server port as it is known to the parser.
- virtual void **set_server_port_open** (bool open)

  Inform the parser of the current status of the server socket.
- virtual bool **get_connected** () const

  Return the parser's knowledge concerning wether there currently exists an established sound data TCP connection or not.
- virtual void **set_connected** (bool **connected**)

  Inform the parser about the existance of a sound data connection.
- virtual void **set_new_peer** (unsigned short port, const std::string &host)

  Set parser monitor variables peer_port and peer_address, and calls set_connected(true).
- **io_tcp_parser_t** ()

  Constructor initializes parser variables.
- virtual ~**io_tcp_parser_t** ()

  Do-nothing destructor.
- virtual void **debug** (const std::string &message)

Private Attributes

- **MHAParser::string_t local_address**

  Lets the user set the local network interface to listen on.
- **MHAParser::int_t local_port**

  Lets the user choose the local tcp port to listen on.
- **MHAParser::int_mon_t server_port_open**

  Indicates wether the TCP server socket is currently open.
- **MHAParser::int_mon_t connected**

  Indicator if there currently is a sound data connection over TCP.
- **MHAParser::string_mon_t peer_address**

  Display the ip address of the currently connected sound data client.
- **MHAParser::int_mon_t peer_port**

  Display the tcp port used by the current sound data client.
- **MHAParser::string_t debug_filename**

  filename to write debugging info to (if non-empty)
- FILE * **debug_file**

  file handle to write debugging info to

Additional Inherited Members

5.119.1 Detailed Description

The parser interface of the IOTCP library.

5.119.2 Constructor & Destructor Documentation

5.119.2.1 io_tcp_parser_t::io_tcp_parser_t ( )

Constructor initializes parser variables.

5.119.2.2 virtual io_tcp_parser_t::~io_tcp_parser_t ( ) [inline], [virtual]

Do-nothing destructor.

5.119.3 Member Function Documentation

5.119.3.1 virtual const std::string& io_tcp_parser_t::get_local_address ( ) const [inline], [virtual]

Read parser variable local_address, this is the address of the network interface that should listen for incoming connections.

Returns

A string containing the address of the local interface as it was set by the user.

5.119.3.2 unsigned short io_tcp_parser_t::get_local_port ( ) const [virtual]

Read parser variable local_port, this is the TCP port that should be used for incoming connections.

Returns

The local tcp port to listen on as it was chosen by the user. The port number is between MIN_TCP_PORT and MAX_TCP_PORT.

5.119.3.3 void io_tcp_parser_t::set_local_port (unsigned short *port*) [virtual]

Set parser variable local_port.

This is needed when it was set to 0 before: In this case, the OS chooses a free port for the TCP server socket, and the port that it chose has to be published to the user via the parser interface.

Parameters

port	The TCP port number that is currently used. In the range [MIN_TCP_PORT, MAX_TCP_PORT], excluding 0.

Precondition

get_local_port() (p. 380) currently returns 0.

5.119.3.4 bool io_tcp_parser_t::get_server_port_open ( ) const [virtual]

Return the status of the server port as it is known to the parser.

Returns

false after initialization, or the value most recently set via

See also

set_server_port_open (p. 381).

5.119.3.5 void io_tcp_parser_t::set_server_port_open (bool *open*) [virtual]

Inform the parser of the current status of the server socket.

Parameters

open	Indicates wether the server socket has just been opened or closed.

Precondition

open may only have the value true if **get_server_port_open()** (p. 381) currently returns false.

Postcondition

See also

get_server_port_open (p. 381) returns the **value** (p. 53) of open.

5.119.3.6 bool io_tcp_parser_t::get_connected ( ) const [virtual]

Return the parser's knowledge concerning wether there currently exists an established sound data TCP connection or not.

Returns

false after initialization, or the value most recently set via

See also

set_connected (p. 382).

5.119.3.7 void io_tcp_parser_t::set_connected (bool *connected*) [virtual]

Inform the parser about the existance of a sound data connection.

Parameters

connected	Indicates wether there currently is a connection or not.

Precondition

connected must not have the same value that is currently returned by

See also

get_connected (p. 382).

Postcondition

See also

get_connected (p. 382) returns the **value** (p. 53) of open.

5.119.3.8 void io_tcp_parser_t::set_new_peer (unsigned short *port,* const std::string & *host*) [virtual]

Set parser monitor variables peer_port and peer_address, and calls set_connected(true).

This method should be called when a new connection is established.

Parameters

port	The TCP port number used by the peer.
host	The Internet host where the peer is located.

Precondition

See also

get_connected (p. 382) currently returns false.

Postcondition

See also

get_connected (p. 382) returns true.

5.119.3.9 virtual void io_tcp_parser_t::debug (const std::string & *message*) [inline], [virtual]

5.119.4 Member Data Documentation

5.119.4.1 MHAParser::string_t io_tcp_parser_t::local_address [private]

Lets the user set the local network interface to listen on.

5.119.4.2 MHAParser::int_t io_tcp_parser_t::local_port [private]

Lets the user choose the local tcp port to listen on.

5.119.4.3 MHAParser::int_mon_t io_tcp_parser_t::server_port_open [private]

Indicates wether the TCP server socket is currently open.

5.119.4.4 MHAParser::int_mon_t io_tcp_parser_t::connected [private]

Indicator if there currently is a sound data connection over TCP.

5.119.4.5 MHAParser::string_mon_t io_tcp_parser_t::peer_address [private]

Display the ip address of the currently connected sound data client.

5.119.4.6 MHAParser::int_mon_t io_tcp_parser_t::peer_port [private]

Display the tcp port used by the current sound data client.

5.119.4.7 MHAParser::string_t io_tcp_parser_t::debug_filename [private]

filename to write debugging info to (if non-empty)

5.119.4.8 FILE* io_tcp_parser_t::debug_file [private]

file handle to write debugging info to

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

- **MHAIOTCP.cpp**

5.120 io_tcp_sound_t Class Reference

Sound data handling of io tcp library.

Classes

- union **float_union**
  This union helps in conversion of floats from host byte order to network byte order and back again.

Public Member Functions

- **io_tcp_sound_t** (int **fragsize**, float **samplerate**)
  Initialize sound data handling.
- virtual ~**io_tcp_sound_t** ()
  Do-nothing destructor.
- virtual void **prepare** (int **num_inchannels**, int **num_outchannels**)
  Called during prepare, sets number of audio channels and allocates sound data storage.
- virtual void **release** ()
  Called during release.
- virtual int **chunkbytes_in** () const
  Number of bytes that constitute one input sound chunk.
- virtual std::string **header** () const
  Create the tcp sound header lines.
- virtual **mha_wave_t** * **ntoh** (const std::string &data)
  *Copy data received from tcp into **mha_wave_t** (p. 496) structure.*
- virtual std::string **hton** (const **mha_wave_t** *s_out)
  Copy sound data from the output sound structure to a string.

Static Private Member Functions

- static void **check_sound_data_type** ()
  Check if mha_real_t is a usable 32-bit floating point type.

Private Attributes

- int **fragsize**
  Number of sound samples in each channel expected and returned from processing callback.
- float **samplerate**
  Sampling rate.
- int **num_inchannels**
  Number of input channels.
- int **num_outchannels**
- **MHASignal::waveform_t** ∗ **s_in**
  Storage for input signal.

5.120.1 Detailed Description

Sound data handling of io tcp library.

5.120.2 Constructor & Destructor Documentation

5.120.2.1 io_tcp_sound_t::io_tcp_sound_t (int *fragsize,* float *samplerate*)

Initialize sound data handling.

Checks sound data type by calling

See also

check_sound_data_type (p. 386).

Parameters

fragsize	Number of sound samples in each channel expected and returned from processing callback.
samplerate	Number of samples per second in each channel.

5.120.2.2 virtual io_tcp_sound_t::~io_tcp_sound_t ( ) [inline], [virtual]

Do-nothing destructor.

5.120.3 Member Function Documentation

5.120.3.1 void io_tcp_sound_t::check_sound_data_type ( ) [static], [private]

Check if mha_real_t is a usable 32-bit floating point type.

Exceptions

MHA_Error *(p. 445)*	if mha_real_t is not compatible to 32-bit float.

5.120.3.2 void io_tcp_sound_t::prepare (int *num_inchannels*, int *num_outchannels*) [virtual]

Called during prepare, sets number of audio channels and allocates sound data storage.

Parameters

num_inchannels	Number of input audio channels.
num_outchannels	Number of output audio channels.

5.120.3.3 void io_tcp_sound_t::release (void) [virtual]

Called during release.

Deletes sound data storage.

5.120.3.4 int io_tcp_sound_t::chunkbytes_in ( ) const [virtual]

Number of bytes that constitute one input sound chunk.

Returns

Number of bytes to read from TCP connection before invoking signal processing.

5.120.3.5 std::string io_tcp_sound_t::header ( ) const [virtual]

Create the tcp sound header lines.

5.120.3.6 mha_wave_t * io_tcp_sound_t::ntoh (const std::string & *data*) [virtual]

Copy data received from tcp into **mha_wave_t** (p. 496) structure.

Doing network-to-host byte order swapping in the process.

Parameters

data	One chunk (

See also

chunkbytes_in (p. 386)) of sound data to process.

Returns

Pointer to the sound data storage.

5.120.3.7 std::string io_tcp_sound_t::hton (const mha_wave_t * *s_out*) [virtual]

Copy sound data from the output sound structure to a string.

Doing host-to-network byte order swapping while at it.

Parameters

s_out	Pointer to the storage of the sound to put out.

Returns

The sound data in network byte order.

5.120.4 Member Data Documentation

5.120.4.1 int io_tcp_sound_t::fragsize [private]

Number of sound samples in each channel expected and returned from processing callback.

5.120.4.2 float io_tcp_sound_t::samplerate [private]

Sampling rate.

Number of samples per second in each channel.

5.120.4.3 int io_tcp_sound_t::num_inchannels [private]

Number of input channels.

Number of channels expected from and returned by signal processing callback.

5.120.4.4 int io_tcp_sound_t::num_outchannels [private]

5.120.4.5 MHASignal::waveform_t* io_tcp_sound_t::s_in [private]

Storage for input signal.

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

- **MHAIOTCP.cpp**

5.121 io_tcp_sound_t::float_union Union Reference

This union helps in conversion of floats from host byte order to network byte order and back again.

Public Attributes

- float **f**
- unsigned int **i**
- char **c** [4]

5.121.1 Detailed Description

This union helps in conversion of floats from host byte order to network byte order and back again.

5.121.2 Member Data Documentation

5.121.2.1 float io_tcp_sound_t::float_union::f

5.121.2.2 unsigned int io_tcp_sound_t::float_union::i

5.121.2.3 char io_tcp_sound_t::float_union::c[4]

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

- **MHAIOTCP.cpp**

5.122 io_tcp_t Class Reference

The tcp sound io library.

Public Member Functions

- **io_tcp_t** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↩ handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↩ event, void *stop_handle)
- void **prepare** (int num_inchannels, int num_outchannels)

  Allocate server socket and start thread waiting for sound data exchange.
- void **start** ()

  Call frameworks start callback if there is a sound data connection at the moment.
- void **stop** ()

  Close the current connection if there is one.
- void **release** ()

  Close the current connection and close the server socket.
- virtual void **accept_loop** ()

  IO thread executes this method.
- virtual void **connection_loop** (**MHA_TCP::Connection** *c)

  IO thread executes this method for each connection.
- virtual void **parse** (const char *cmd, char *retval, unsigned int len)

  Parser interface.
- virtual ~**io_tcp_t** ()

Private Attributes

- **io_tcp_parser_t parser**
- **io_tcp_sound_t sound**
- **io_tcp_fwcb_t fwcb**
- **MHA_TCP::Server * server**
- **MHA_TCP::Thread * thread**
- **MHA_TCP::Async_Notify notify_start**
- **MHA_TCP::Async_Notify notify_stop**
- **MHA_TCP::Async_Notify notify_release**

5.122.1 Detailed Description

The tcp sound io library.

5.122.2 Constructor & Destructor Documentation

5.122.2.1 io_tcp_t::io_tcp_t (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

5.122.2.2 virtual io_tcp_t::~io_tcp_t ( ) [inline], [virtual]

5.122.3 Member Function Documentation

5.122.3.1 void io_tcp_t::prepare (

int *num_inchannels,*
int *num_outchannels*)

Allocate server socket and start thread waiting for sound data exchange.

prepare opens the tcp server socket and starts the io thread that listens for audio data on the tcp socket after doing some sanity checks

5.122.3.2 void io_tcp_t::start ( )

Call frameworks start callback if there is a sound data connection at the moment.

5.122.3.3 void io_tcp_t::stop ( )

Close the current connection if there is one.

stop IO thread

5.122.3.4 void io_tcp_t::release (

void)

Close the current connection and close the server socket.

Stop IO thread and close server socket.

5.122.3.5 void io_tcp_t::accept_loop ( ) [virtual]

IO thread executes this method.

5.122.3.6 void io_tcp_t::connection_loop (

MHA_TCP::Connection * *c*) [virtual]

IO thread executes this method for each connection.

Parameters

c	pointer to connection. connection_loop deletes connection before exiting.

5.122.3.7 virtual void io_tcp_t::parse (const char * *cmd,* char * *retval,* unsigned int *len*) [inline], [virtual]

Parser interface.

5.122.4 Member Data Documentation

5.122.4.1 io_tcp_parser_t io_tcp_t::parser [private]

5.122.4.2 io_tcp_sound_t io_tcp_t::sound [private]

5.122.4.3 io_tcp_fwcb_t io_tcp_t::fwcb [private]

5.122.4.4 MHA_TCP::Server* io_tcp_t::server [private]

5.122.4.5 MHA_TCP::Thread* io_tcp_t::thread [private]

5.122.4.6 MHA_TCP::Async_Notify io_tcp_t::notify_start [private]

5.122.4.7 MHA_TCP::Async_Notify io_tcp_t::notify_stop [private]

5.122.4.8 MHA_TCP::Async_Notify io_tcp_t::notify_release [private]

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

- **MHAIOTCP.cpp**

5.123 latex_doc_t Class Reference

Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUME↩ NTATION macro.

Public Member Functions

- **latex_doc_t** (const std::string &**plugname**, const std::string &**plugin_macro**)

  Constructor loads the plugin into this process.
- std::string **get_latex_doc** ()

  This method accesses the compiled-in contents of the MHAPLUGIN_DOCUMENTATION macro and the exported interface functions of the loaded plugin to produce latex docuementation for the plugin.
- std::string **get_main_category** () const
- std::vector< std::string > **get_categories** () const

Private Member Functions

- std::string **strdom** (**mha_domain_t** d) const
- std::string **get_ac** (**MHAKernel::algo_comm_class_t** &**ac**, std::string txt) const
- std::string **parsername** (std::string s) const
- std::string **get_parser_var** (**MHAParser::base_t** *p, std::string name) const
- std::string **get_parser_tab** (**MHAParser::base_t** *p, const std::string &prefix, const std::string &latex_macro) const

Private Attributes

- const std::string **plugname**
- const std::string **latex_plugname**
- **MHAKernel::algo_comm_class_t ac**
- **PluginLoader::mhapluginloader_t loader**
- const std::string **plugin_macro**

5.123.1 Detailed Description

Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUMENTATION macro.

5.123.2 Constructor & Destructor Documentation

5.123.2.1 latex_doc_t::latex_doc_t (const std::string & *plugname,* const std::string & *plugin_macro*)

Constructor loads the plugin into this process.

Parameters

plugname	Name of the MHA plugin to process
plugin_macro	name of the LaTeX section macro that documents a single plugin (e.g. "section", "subsection", "subsubsection", ...)

5.123.3 Member Function Documentation

5.123.3.1 std::string latex_doc_t::get_latex_doc ( )

This method accesses the compiled-in contents of the MHAPLUGIN_DOCUMENTATION macro and the exported interface functions of the loaded plugin to produce latex docuementation for the plugin.

It tentatively prepares the plugin for processing and checks the AC variables registered by the plugin.

Returns

the complete latex documentation for this plugin

5.123.3.2 std::string latex_doc_t::get_main_category ( ) const

Returns

the first word of the categories string in the MHAPLUGIN_DOCUMENTATION macro

5.123.3.3 std::vector< std::string > latex_doc_t::get_categories ( ) const

Returns

a vector of all words in the categories string in the MHAPLUGIN_DOCUMENTATION macro

5.123.3.4 std::string latex_doc_t::strdom (mha_domain_t *d*) const [private]

5.123.3.5 std::string latex_doc_t::get_ac (MHAKernel::algo_comm_class_t & *ac,* std::string *txt*) const [private]

5.123.3.6 std::string latex_doc_t::parsername (std::string *s*) const [private]

5.123.3.7 std::string latex_doc_t::get_parser_var (MHAParser::base_t * *p,* std::string *name*) const [private]

5.123.3.8 std::string latex_doc_t::get_parser_tab (MHAParser::base_t * *p,* const std::string & *prefix,* const std::string & *latex_macro*) const [private]

5.123.4 Member Data Documentation

5.123.4.1 const std::string latex_doc_t::plugname [private]

5.123.4.2 const std::string latex_doc_t::latex_plugname [private]

5.123.4.3 MHAKernel::algo_comm_class_t latex_doc_t::ac [private]

5.123.4.4 PluginLoader::mhapluginloader_t latex_doc_t::loader [private]

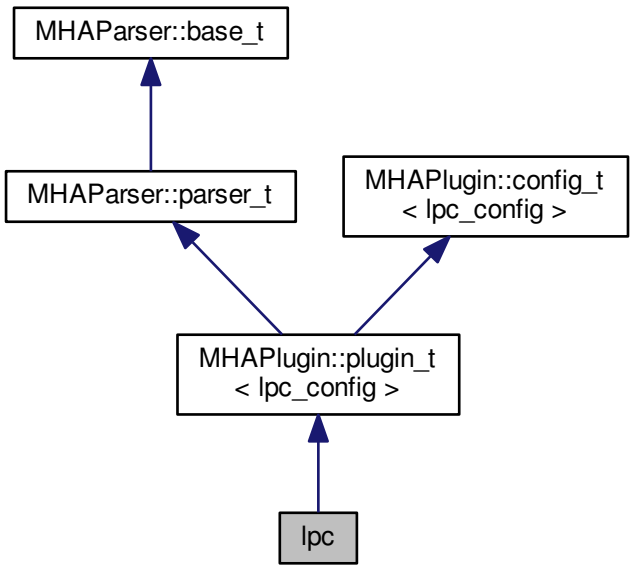
5.123.4.5 const std::string latex_doc_t::plugin_macro [private]

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

- **generatemhaplugindoc.cpp**

5.124 lpc Class Reference

Inheritance diagram for lpc:



Public Member Functions

- **lpc** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &**algo_name**)
  Constructs our plugin.
- ∼**lpc** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation.
- void **release** (void)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- std::string **algo_name**
- **MHAParser::int_t lpc_order**
- **MHAParser::int_t lpc_buffer_size**
- **MHAParser::bool_t shift**
- **MHAParser::int_t comp_each_iter**
- **MHAParser::bool_t norm**
- **MHAEvents::patchbay_t< lpc > patchbay**

Additional Inherited Members

5.124.1 Constructor & Destructor Documentation

5.124.1.1 lpc::lpc (

algo_comm_t & ***ac,***
const std::string & ***chain_name,***
const std::string & ***algo_name*** **)**

Constructs our plugin.

5.124.1.2 lpc::~lpc ( )

5.124.2 Member Function Documentation

5.124.2.1 mha_wave_t * lpc::process (

mha_wave_t * ***signal*** **)**

Checks for the most recent configuration and defers processing to it.

5.124.2.2 void lpc::prepare (

mhaconfig_t & ***signal_info*** **)** `[virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< lpc_config >** (p. 730).

5.124.2.3 void lpc::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< lpc_config >** (p. 731).

5.124.2.4 void lpc::update_cfg (void) [private]

5.124.3 Member Data Documentation

5.124.3.1 std::string lpc::algo_name [private]

5.124.3.2 MHAParser::int_t lpc::lpc_order [private]

5.124.3.3 MHAParser::int_t lpc::lpc_buffer_size [private]

5.124.3.4 MHAParser::bool_t lpc::shift [private]

5.124.3.5 MHAParser::int_t lpc::comp_each_iter [private]

5.124.3.6 MHAParser::bool_t lpc::norm [private]

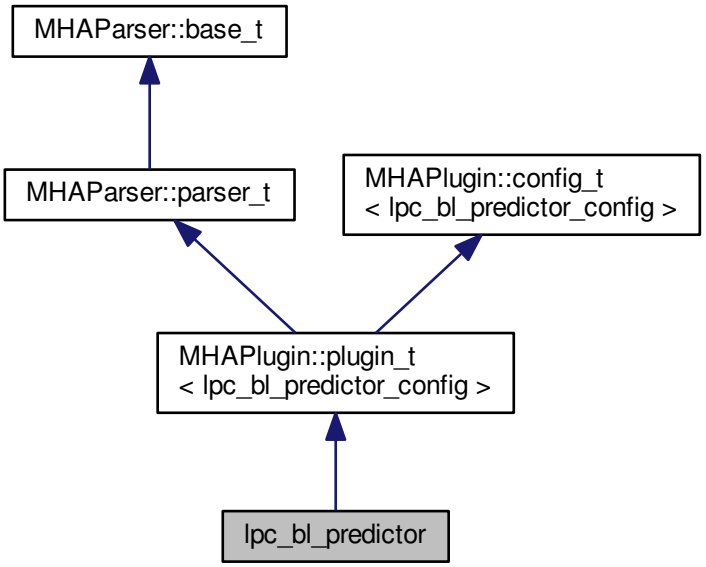
5.124.3.7 MHAEvents::patchbay_t<lpc> lpc::patchbay [private]

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

- **lpc.h**
- **lpc.cpp**

5.125 lpc_bl_predictor Class Reference

Inheritance diagram for lpc_bl_predictor:



Public Member Functions

- **lpc_bl_predictor** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
  Constructs our plugin.
- ∼**lpc_bl_predictor** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t lpc_order**
- **MHAParser::string_t name_kappa**
- **MHAParser::string_t name_lpc_f**
- **MHAParser::string_t name_lpc_b**
- **MHAParser::string_t name_f**
- **MHAParser::string_t name_b**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< lpc_bl_predictor > patchbay**

Additional Inherited Members

5.125.1 Constructor & Destructor Documentation

5.125.1.1 lpc_bl_predictor::lpc_bl_predictor (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.125.1.2 lpc_bl_predictor::~lpc_bl_predictor ( )

5.125.2 Member Function Documentation

5.125.2.1 mha_wave_t * lpc_bl_predictor::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.125.2.2 void lpc_bl_predictor::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< lpc_bl_predictor_config >** (p. 730).

5.125.2.3 void lpc_bl_predictor::release (void) [inline],[virtual]

Reimplemented from **MHAPlugin::plugin_t**< **lpc_bl_predictor_config** > (p. 731).

5.125.2.4 void lpc_bl_predictor::update_cfg (void) [private]

5.125.3 Member Data Documentation

5.125.3.1 MHAParser::int_t lpc_bl_predictor::lpc_order

5.125.3.2 MHAParser::string_t lpc_bl_predictor::name_kappa

5.125.3.3 MHAParser::string_t lpc_bl_predictor::name_lpc_f

5.125.3.4 MHAParser::string_t lpc_bl_predictor::name_lpc_b

5.125.3.5 MHAParser::string_t lpc_bl_predictor::name_f

5.125.3.6 MHAParser::string_t lpc_bl_predictor::name_b

5.125.3.7 MHAEvents::patchbay_t<lpc_bl_predictor> lpc_bl_predictor::patchbay [private]

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

- **lpc_bl_predictor.h**
- **lpc_bl_predictor.cpp**

5.126 lpc_bl_predictor_config Class Reference

Public Member Functions

- **lpc_bl_predictor_config** (**algo_comm_t** &iac, const **mhaconfig_t** in_cfg, **lpc_bl_predictor** *_lpc)
- ~**lpc_bl_predictor_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Attributes

- **algo_comm_t ac**
- **MHA_AC::waveform_t f_est**
- **MHA_AC::waveform_t b_est**
- **MHASignal::waveform_t forward**
- **MHASignal::waveform_t backward**
- int **lpc_order**
- std::string **name_km**
- std::string **name_f**
- std::string **name_b**
- **mha_wave_t km**
- **mha_wave_t s_f**
- **mha_wave_t s_b**

5.126.1 Constructor & Destructor Documentation

5.126.1.1 lpc_bl_predictor_config::lpc_bl_predictor_config (algo_comm_t & *iac,* const mhaconfig_t *in_cfg,* lpc_bl_predictor * *_lpc*)

5.126.1.2 lpc_bl_predictor_config::~lpc_bl_predictor_config ( )

5.126.2 Member Function Documentation

5.126.2.1 mha_wave_t * lpc_bl_predictor_config::process (mha_wave_t * *wave*)

5.126.3 Member Data Documentation

5.126.3.1 algo_comm_t lpc_bl_predictor_config::ac [private]

5.126.3.2 MHA_AC::waveform_t lpc_bl_predictor_config::f_est [private]

5.126.3.3 MHA_AC::waveform_t lpc_bl_predictor_config::b_est [private]

5.126.3.4 MHASignal::waveform_t lpc_bl_predictor_config::forward [private]

5.126.3.5 MHASignal::waveform_t lpc_bl_predictor_config::backward [private]

5.126.3.6 int lpc_bl_predictor_config::lpc_order [private]

5.126.3.7 std::string lpc_bl_predictor_config::name_km [private]

5.126.3.8 std::string lpc_bl_predictor_config::name_f [private]

5.126.3.9 std::string lpc_bl_predictor_config::name_b [private]

5.126.3.10 mha_wave_t lpc_bl_predictor_config::km [private]

5.126.3.11 mha_wave_t lpc_bl_predictor_config::s_f [private]

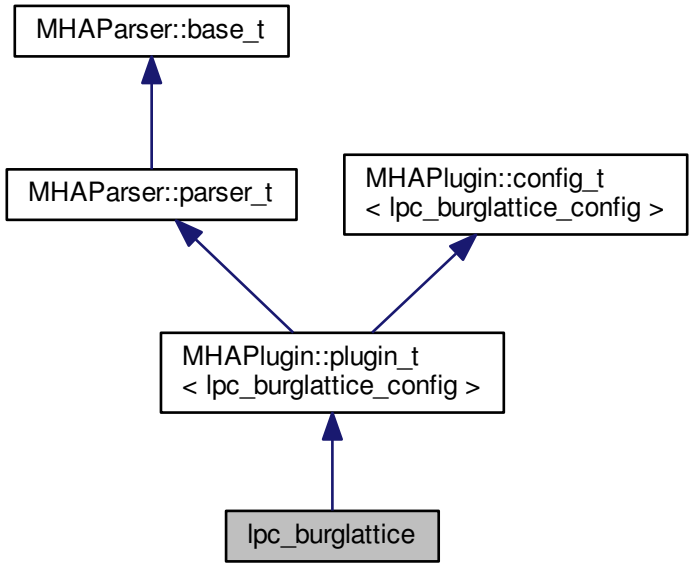
5.126.3.12 mha_wave_t lpc_bl_predictor_config::s_b [private]

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

- **lpc_bl_predictor.h**
- **lpc_bl_predictor.cpp**

5.127 lpc_burglattice Class Reference

Inheritance diagram for lpc_burglattice:



Public Member Functions

- **lpc_burglattice** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  Constructs our plugin.
- ∼**lpc_burglattice** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t lpc_order**
- **MHAParser::string_t name_kappa**
- **MHAParser::string_t name_f**
- **MHAParser::string_t name_b**
- **MHAParser::float_t lambda**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **lpc_burglattice** > **patchbay**

Additional Inherited Members

5.127.1 Constructor & Destructor Documentation

5.127.1.1 lpc_burglattice::lpc_burglattice (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.127.1.2 lpc_burglattice::~lpc_burglattice ( )

5.127.2 Member Function Documentation

5.127.2.1 mha_wave_t * lpc_burglattice::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.127.2.2 void lpc_burglattice::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< lpc_burglattice_config >** (p. 730).

5.127.2.3 void lpc_burglattice::release (void) [inline],[virtual]

Reimplemented from **MHAPlugin::plugin_t< lpc_burglattice_config >** (p. 731).

5.127.2.4 void lpc_burglattice::update_cfg (void) [private]

5.127.3 Member Data Documentation

5.127.3.1 MHAParser::int_t lpc_burglattice::lpc_order

5.127.3.2 MHAParser::string_t lpc_burglattice::name_kappa

5.127.3.3 MHAParser::string_t lpc_burglattice::name_f

5.127.3.4 MHAParser::string_t lpc_burglattice::name_b

5.127.3.5 MHAParser::float_t lpc_burglattice::lambda

5.127.3.6 MHAEvents::patchbay_t<lpc_burglattice> lpc_burglattice::patchbay [private]

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

- **lpc_burg-lattice.h**
- **lpc_burg-lattice.cpp**

5.128 lpc_burglattice_config Class Reference

Public Member Functions

- **lpc_burglattice_config** (**algo_comm_t** &iac, const **mhaconfig_t** in_cfg, **lpc_burglattice** ∗_lpc)
- ∼**lpc_burglattice_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- **algo_comm_t ac**
- **MHASignal::waveform_t forward**
- **MHASignal::waveform_t backward**
- **MHASignal::waveform_t kappa**
- **MHA_AC::waveform_t kappa_block**
- **MHASignal::waveform_t dm**
- **MHASignal::waveform_t nm**
- **mha_real_t lambda**
- int **lpc_order**
- std::string **name_f**
- std::string **name_b**
- **mha_wave_t s_f**
- **mha_wave_t s_b**

5.128.1 Constructor & Destructor Documentation

5.128.1.1 lpc_burglattice_config::lpc_burglattice_config (algo_comm_t & *iac,* const mhaconfig_t *in_cfg,* lpc_burglattice ∗ *_lpc*)

5.128.1.2 lpc_burglattice_config::∼lpc_burglattice_config ( )

5.128.2 Member Function Documentation

5.128.2.1 mha_wave_t ∗ lpc_burglattice_config::process (mha_wave_t ∗ *wave*)

5.128.3 Member Data Documentation

5.128.3.1 algo_comm_t lpc_burglattice_config::ac [private]

5.128.3.2 MHASignal::waveform_t lpc_burglattice_config::forward [private]

5.128.3.3 MHASignal::waveform_t lpc_burglattice_config::backward [private]

5.128.3.4 MHASignal::waveform_t lpc_burglattice_config::kappa [private]

5.128.3.5 MHA_AC::waveform_t lpc_burglattice_config::kappa_block [private]

5.128.3.6 MHASignal::waveform_t lpc_burglattice_config::dm [private]

5.128.3.7 MHASignal::waveform_t lpc_burglattice_config::nm [private]

5.128.3.8 mha_real_t lpc_burglattice_config::lambda [private]

5.128.3.9 int lpc_burglattice_config::lpc_order [private]

5.128.3.10 std::string lpc_burglattice_config::name_f [private]

5.128.3.11 std::string lpc_burglattice_config::name_b [private]

5.128.3.12 mha_wave_t lpc_burglattice_config::s_f [private]

5.128.3.13 mha_wave_t lpc_burglattice_config::s_b [private]

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

- **lpc_burg-lattice.h**
- **lpc_burg-lattice.cpp**

5.129 lpc_config Class Reference

Public Member Functions

- **lpc_config** (**algo_comm_t** &ac, const **mhaconfig_t** in_cfg, std::string &algo_name, unsigned int _order, unsigned int _lpc_buffer_size, bool _shift, unsigned int _comp_each↩ _iter, bool _norm)
- ∼**lpc_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **insert** ()

Private Attributes

- bool **norm**
- bool **shift**
- unsigned int **comp_each_iter**
- unsigned int **order**
- unsigned int **lpc_buffer_size**
- unsigned int **N**
- unsigned int **comp_iter**
- **mha_wave_t sample**
- std::vector< **mha_real_t** > **R**
- std::vector< **mha_real_t** > **A**
- **MHASignal::ringbuffer_t inwave**
- **MHA_AC::waveform_t lpc_out**
- **MHA_AC::waveform_t corr_out**

5.129.1 Constructor & Destructor Documentation

5.129.1.1 lpc_config::lpc_config (
algo_comm_t & *ac,*
const **mhaconfig_t** *in_cfg,*
std::string & *algo_name,*
unsigned int *_order,*
unsigned int *_lpc_buffer_size,*
bool *_shift,*
unsigned int *_comp_each_iter,*
bool *_norm*)

5.129.1.2 lpc_config::~lpc_config ( )

5.129.2 Member Function Documentation

5.129.2.1 mha_wave_t * lpc_config::process (
mha_wave_t * *wave*)

5.129.2.2 void lpc_config::insert ( )

5.129.3 Member Data Documentation

5.129.3.1 bool lpc_config::norm [private]

5.129.3.2 bool lpc_config::shift [private]

5.129.3.3 unsigned int lpc_config::comp_each_iter [private]

5.129.3.4 unsigned int lpc_config::order [private]

5.129.3.5 unsigned int lpc_config::lpc_buffer_size [private]

5.129.3.6 unsigned int lpc_config::N [private]

5.129.3.7 unsigned int lpc_config::comp_iter [private]

5.129.3.8 mha_wave_t lpc_config::sample [private]

5.129.3.9 std::vector<mha_real_t> lpc_config::R [private]

5.129.3.10 std::vector<mha_real_t> lpc_config::A [private]

5.129.3.11 MHASignal::ringbuffer_t lpc_config::inwave [private]

5.129.3.12 MHA_AC::waveform_t lpc_config::lpc_out [private]

5.129.3.13 MHA_AC::waveform_t lpc_config::corr_out [private]

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

- **lpc.h**
- **lpc.cpp**

5.130 matrixmixer::cfg_t Class Reference

Public Member Functions

- **cfg_t** (std::vector< std::vector< float > > imixer, unsigned int ci, unsigned int co, unsigned int fragsize, unsigned int nfft)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::waveform_t m**
- **MHASignal::waveform_t wout**
- **MHASignal::spectrum_t sout**

5.130.1 Constructor & Destructor Documentation

5.130.1.1 cfg_t::cfg_t (

std::vector< std::vector< float > > *imixer,*
unsigned int *ci,*
unsigned int *co,*
unsigned int *fragsize,*
unsigned int *nfft*)

5.130.2 Member Function Documentation

5.130.2.1 mha_wave_t * cfg_t::process (

mha_wave_t * *s*)

5.130.2.2 mha_spec_t * cfg_t::process (

mha_spec_t * *s*)

5.130.3 Member Data Documentation

5.130.3.1 MHASignal::waveform_t matrixmixer::cfg_t::m [private]

5.130.3.2 MHASignal::waveform_t matrixmixer::cfg_t::wout [private]

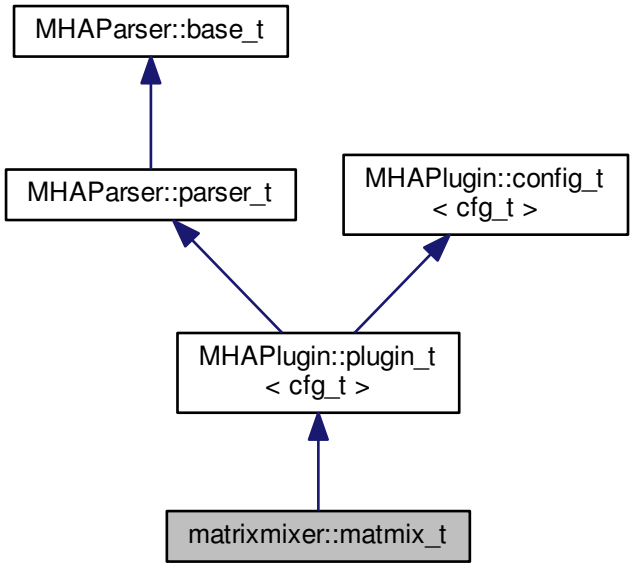
5.130.3.3 MHASignal::spectrum_t matrixmixer::cfg_t::sout [private]

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

- **matrixmixer.cpp**

5.131 matrixmixer::matmix_t Class Reference

Inheritance diagram for matrixmixer::matmix_t:



Public Member Functions

- **matmix_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update_m** ()

Private Attributes

- **MHAEvents::patchbay_t**< **matmix_t** > **patchbay**
- **MHAParser::mfloat_t mixer**
- unsigned int **ci**
- unsigned int **co**

Additional Inherited Members

5.131.1 Constructor & Destructor Documentation

5.131.1.1 matrixmixer::matmix_t::matmix_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.131.2 Member Function Documentation

5.131.2.1 void matrixmixer::matmix_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 730).

5.131.2.2 mha_wave_t * matrixmixer::matmix_t::process (mha_wave_t * *s*)

5.131.2.3 mha_spec_t * matrixmixer::matmix_t::process (mha_spec_t * *s*)

5.131.2.4 void matrixmixer::matmix_t::update_m (void) [private]

5.131.3 Member Data Documentation

5.131.3.1 MHAEvents::patchbay_t<matmix_t> matrixmixer::matmix_t::patchbay [private]

5.131.3.2 MHAParser::mfloat_t matrixmixer::matmix_t::mixer [private]

5.131.3.3 unsigned int matrixmixer::matmix_t::ci [private]

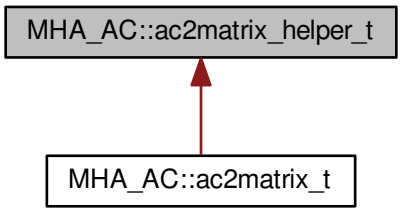
5.131.3.4 unsigned int matrixmixer::matmix_t::co [private]

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

- **matrixmixer.cpp**

5.132 MHA_AC::ac2matrix_helper_t Class Reference

Inheritance diagram for MHA_AC::ac2matrix_helper_t:



Public Member Functions

- **ac2matrix_helper_t** (**algo_comm_t**, const std::string &)
- void **getvar** ()

Public Attributes

- **algo_comm_t ac**
- std::string **name**
- std::string **username**
- **MHASignal::uint_vector_t size**
- bool **is_complex**

Protected Attributes

- **comm_var_t acvar**

5.132.1 Constructor & Destructor Documentation

5.132.1.1 MHA_AC::ac2matrix_helper_t::ac2matrix_helper_t (algo_comm_t *iac,* const std::string & *iname*)

5.132.2 Member Function Documentation

5.132.2.1 void MHA_AC::ac2matrix_helper_t::getvar ( )

5.132.3 Member Data Documentation

5.132.3.1 algo_comm_t MHA_AC::ac2matrix_helper_t::ac

5.132.3.2 std::string MHA_AC::ac2matrix_helper_t::name

5.132.3.3 std::string MHA_AC::ac2matrix_helper_t::username

5.132.3.4 MHASignal::uint_vector_t MHA_AC::ac2matrix_helper_t::size

5.132.3.5 bool MHA_AC::ac2matrix_helper_t::is_complex

5.132.3.6 comm_var_t MHA_AC::ac2matrix_helper_t::acvar [protected]

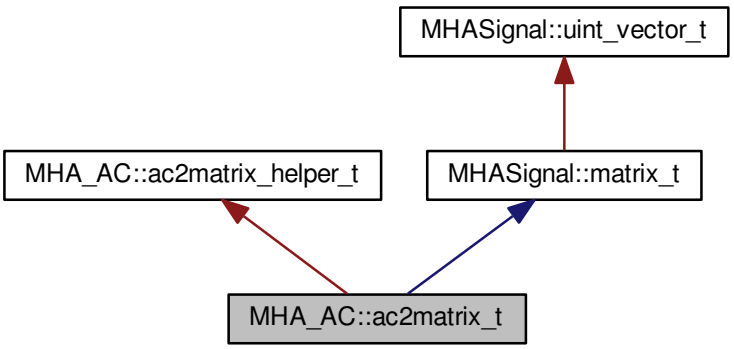
The documentation for this class was generated from the following files:

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.133 MHA_AC::ac2matrix_t Class Reference

Copy AC variable to a matrix.

Inheritance diagram for MHA_AC::ac2matrix_t:



Public Member Functions

- **ac2matrix_t** (**algo_comm_t ac**, const std::string &**name**)
  Constructor.
- void **update** ()
  Update contents of the matrix from the AC space.
- const std::string & **getname** () const
  Return name of AC variable/matrix.
- const std::string & **getusername** () const
  Return user specified name of AC variable/matrix.
- void **insert** (**algo_comm_t ac**)
  Insert matrix into an AC space (other than source AC space)

Additional Inherited Members

5.133.1 Detailed Description

Copy AC variable to a matrix.

This class constructs a matrix of same size as an AC variable and can copy the AC variable to itself. The **update()** (p. 414) function is real-time safe.

5.133.2 Constructor & Destructor Documentation

5.133.2.1 MHA_AC::ac2matrix_t::ac2matrix_t (algo_comm_t *ac*, const std::string & *name*)

Constructor.

Parameters

ac	AC handle
name	Name of AC variable to be copied

5.133.3 Member Function Documentation

5.133.3.1 void MHA_AC::ac2matrix_t::update ( )

Update contents of the matrix from the AC space.

This function is real-time safe. The copy operation performance is of the order of the number of elements in the matrix.

5.133.3.2 const std::string& MHA_AC::ac2matrix_t::getname ( ) const [inline]

Return name of AC variable/matrix.

5.133.3.3 const std::string& MHA_AC::ac2matrix_t::getusername ( ) const [inline]

Return user specified name of AC variable/matrix.

5.133.3.4 void MHA_AC::ac2matrix_t::insert (algo_comm_t *ac*)

Insert matrix into an AC space (other than source AC space)

Parameters

ac	AC space handle to insert data

Note

The AC variable data buffer points to the data of the matrix. Modifications of the AC variable directly modify the data of the matrix; after deletion of the matrix, the data buffer is invalid.

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.134 MHA_AC::acspace2matrix_t Class Reference

Copy all or a subset of all numeric AC variables into an array of matrixes.

Public Member Functions

- **acspace2matrix_t** (**algo_comm_t** ac, const std::vector< std::string > &names)
  Constructor.
- **acspace2matrix_t** (const **MHA_AC::acspace2matrix_t** &src)
  Constructor with initialization from an instance.
- ~**acspace2matrix_t** ()
- **MHA_AC::acspace2matrix_t** & **operator=** (const **MHA_AC::acspace2matrix_t** &src)
  Copy all contents (deep copy).
- **MHA_AC::ac2matrix_t** & **operator[ ]** (unsigned int k)
  Access operator.

- const **MHA_AC::ac2matrix_t** & **operator[ ]** (unsigned int k) const

  Constant access operator.
- void **update** ()

  Update function.
- unsigned int **size** () const

  Number of matrixes in AC space.
- unsigned int **frame** () const

  Actual frame number.
- void **insert** (**algo_comm_t** ac)

  Insert AC space copy into an AC space (other than source AC space)

Private Attributes

- unsigned int **len**
- **MHA_AC::ac2matrix_t** ∗∗ **data**
- unsigned int **frameno**

5.134.1 Detailed Description

Copy all or a subset of all numeric AC variables into an array of matrixes.

5.134.2 Constructor & Destructor Documentation

5.134.2.1 MHA_AC::acspace2matrix_t::acspace2matrix_t (algo_comm_t *ac*, const std::vector< std::string > & *names*)

Constructor.

Scan all given AC variables and allocate corresponding matrixes.

Parameters

ac	AC handle.
names	Names of AC variables, or empty for all.

5.134.2.2 MHA_AC::acspace2matrix_t::acspace2matrix_t (const MHA_AC::acspace2matrix_t & *src*)

Constructor with initialization from an instance.

Parameters

src	Instance to be copied.

5.134.2.3 MHA_AC::acspace2matrix_t::~acspace2matrix_t ( )

5.134.3 Member Function Documentation

5.134.3.1 MHA_AC::acspace2matrix_t & MHA_AC::acspace2matrix_t::operator= (const MHA_AC::acspace2matrix_t & *src*)

Copy all contents (deep copy).

Parameters

src	Array of matrixes to be copied.

5.134.3.2 MHA_AC::ac2matrix_t& MHA_AC::acspace2matrix_t::operator[ ] (unsigned int *k*) [inline]

Access operator.

Parameters

k	index into array; should not exceed **size()** (p. 418)-1.

Return values

Reference	to matrix.

5.134.3.3 const MHA_AC::ac2matrix_t& MHA_AC::acspace2matrix_t::operator[ ] (unsigned int *k*) const [inline]

Constant access operator.

Parameters

k	index into array; should not exceed **size()** (p. 418)-1.

Return values

Constant	reference to matrix.

5.134.3.4 void MHA_AC::acspace2matrix_t::update ( ) [inline]

Update function.

This function updates all matrixes from their corresponding AC variables. It can be called from the MHA Framework prepare function or in the processing callback.

5.134.3.5 unsigned int MHA_AC::acspace2matrix_t::size ( ) const [inline]

Number of matrixes in AC space.

5.134.3.6 unsigned int MHA_AC::acspace2matrix_t::frame ( ) const [inline]

Actual frame number.

5.134.3.7 void MHA_AC::acspace2matrix_t::insert (algo_comm_t *ac*)

Insert AC space copy into an AC space (other than source AC space)

Parameters

ac	AC space handle to insert data

5.134.4 Member Data Documentation

5.134.4.1 unsigned int MHA_AC::acspace2matrix_t::len [private]

5.134.4.2 MHA_AC::ac2matrix_t** MHA_AC::acspace2matrix_t::data [private]

5.134.4.3 unsigned int MHA_AC::acspace2matrix_t::frameno [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.135 MHA_AC::double_t Class Reference

Insert a double precision floating point variable into the AC space.

Public Member Functions

- **double_t** (**algo_comm_t**, std::string, double=0)
- ~**double_t** ()

Public Attributes

- double **data**
  Floating point value variable.

Private Attributes

- **algo_comm_t ac**

5.135.1 Detailed Description

Insert a double precision floating point variable into the AC space.

The variable is automatically removed on destruction.

5.135.2 Constructor & Destructor Documentation

5.135.2.1 MHA_AC::double_t::double_t (algo_comm_t *iac,* std::string *n,* double *v* = 0)

5.135.2.2 MHA_AC::double_t::~double_t ( )

5.135.3 Member Data Documentation

5.135.3.1 double MHA_AC::double_t::data

Floating point value variable.

5.135.3.2 algo_comm_t MHA_AC::double_t::ac [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.136 MHA_AC::float_t Class Reference

Insert a float point variable into the AC space.

Public Member Functions

- **float_t** (**algo_comm_t**, std::string, float=0)
  Constructor.
- ~**float_t** ()

Public Attributes

- float **data**
  Floating point value variable.

Private Attributes

- **algo_comm_t ac**

5.136.1 Detailed Description

Insert a float point variable into the AC space.

The variable is automatically removed on destruction.

5.136.2 Constructor & Destructor Documentation

5.136.2.1 MHA_AC::float_t::float_t (algo_comm_t *iac,* std::string *n,* float *v =* 0)

Constructor.

5.136.2.2 MHA_AC::float_t::~float_t ( )

5.136.3 Member Data Documentation

5.136.3.1 float MHA_AC::float_t::data

Floating point value variable.

5.136.3.2 algo_comm_t MHA_AC::float_t::ac [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.137 MHA_AC::int_t Class Reference

Insert a integer variable into the AC space.

Public Member Functions

- **int_t** (**algo_comm_t**, std::string, int=0)
- ~**int_t** ()

Public Attributes

- int **data**
  Integer value variable.

Private Attributes

- **algo_comm_t ac**

5.137.1 Detailed Description

Insert a integer variable into the AC space.

The variable is automatically removed on destruction.

5.137.2 Constructor & Destructor Documentation

5.137.2.1 MHA_AC::int_t::int_t (algo_comm_t *iac,* std::string *n,* int *v* = 0)

5.137.2.2 MHA_AC::int_t::~int_t ( )

5.137.3 Member Data Documentation

5.137.3.1 int MHA_AC::int_t::data

Integer value variable.

5.137.3.2 algo_comm_t MHA_AC::int_t::ac [private]

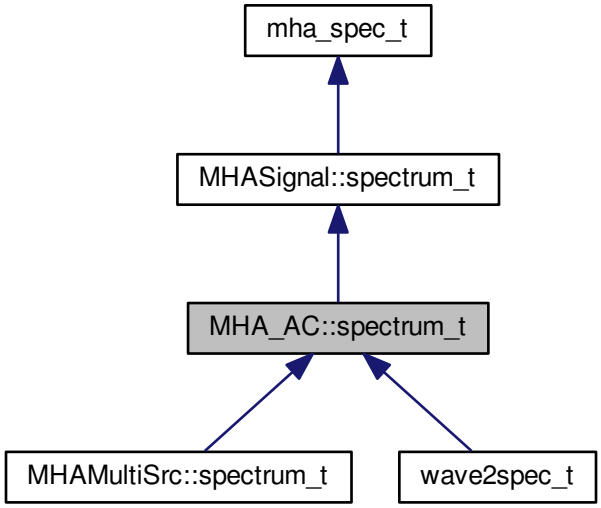
The documentation for this class was generated from the following files:

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.138 MHA_AC::spectrum_t Class Reference

Insert a **MHASignal::spectrum_t** (p. 800) class into the AC space.

Inheritance diagram for MHA_AC::spectrum_t:



Public Member Functions

- **spectrum_t** (**algo_comm_t ac**, std::string **name**, unsigned int bins, unsigned int **channels**, bool insert_now)
  Create the AC variable.
- ~**spectrum_t** ()
- void **insert** ()
  Insert AC variable into AC space.

Protected Attributes

- **algo_comm_t ac**
- std::string **name**

Additional Inherited Members

5.138.1 Detailed Description

Insert a **MHASignal::spectrum_t** (p. 800) class into the AC space.

The variable is automatically removed on destruction.

5.138.2 Constructor & Destructor Documentation

5.138.2.1 MHA_AC::spectrum_t::spectrum_t (algo_comm_t *ac*, std::string *name*, unsigned int *bins*, unsigned int *channels*, bool *insert_now*)

Create the AC variable.

Parameters

ac	AC handle
name	Name of variable in AC space
bins	Number of FFT bins in the **waveform_t** (p. 425) class
channels	Number of audio channels in the **waveform_t** (p. 425) class
insert_now	Insert implicitely in the constructor (true) or explicitely in the **insert()** (p. 423) function (false)

5.138.2.2 MHA_AC::spectrum_t::~spectrum_t (void) [virtual]

Reimplemented from **MHASignal::spectrum_t** (p. 802).

5.138.3 Member Function Documentation

5.138.3.1 void MHA_AC::spectrum_t::insert ( )

Insert AC variable into AC space.

5.138.4 Member Data Documentation

5.138.4.1 algo_comm_t MHA_AC::spectrum_t::ac [protected]

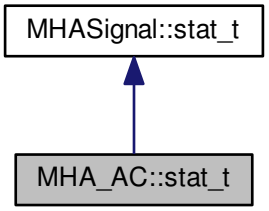
5.138.4.2 std::string MHA_AC::spectrum_t::name [protected]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.139 MHA_AC::stat_t Class Reference

Inheritance diagram for MHA_AC::stat_t:



Public Member Functions

- **stat_t** (**algo_comm_t** ac, const std::string &name, const unsigned int &frames, const unsigned int &**channels**, bool insert_now)
- void **update** ()
- void **insert** ()

Private Attributes

- **MHA_AC::waveform_t mean**
- **MHA_AC::waveform_t std**

5.139.1 Constructor & Destructor Documentation

5.139.1.1 MHA_AC::stat_t::stat_t (algo_comm_t *ac*, const std::string & *name*, const unsigned int & *frames*, const unsigned int & *channels*, bool *insert_now*)

5.139.2 Member Function Documentation

5.139.2.1 void MHA_AC::stat_t::update ( )

5.139.2.2 void MHA_AC::stat_t::insert ( )

5.139.3 Member Data Documentation

5.139.3.1 MHA_AC::waveform_t MHA_AC::stat_t::mean [private]

5.139.3.2 MHA_AC::waveform_t MHA_AC::stat_t::std [private]

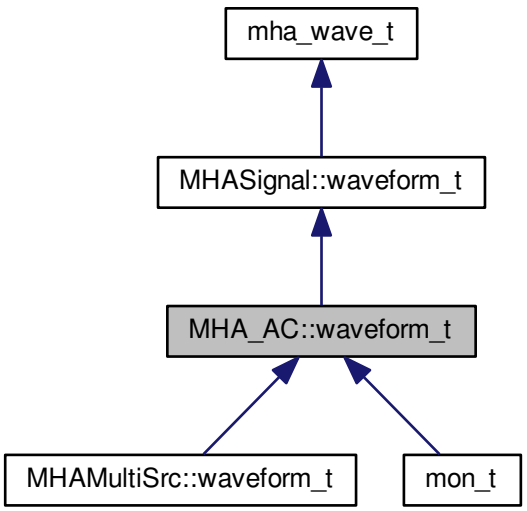
The documentation for this class was generated from the following files:

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.140 MHA_AC::waveform_t Class Reference

Insert a **MHASignal::waveform_t** (p. 812) class into the AC space.

Inheritance diagram for MHA_AC::waveform_t:



Public Member Functions

- **waveform_t** (**algo_comm_t ac**, std::string **name**, unsigned int frames, unsigned int **channels**, bool insert_now)

  Create the AC variable.
- ~**waveform_t** ()
- void **insert** ()

  Insert AC variable into AC space.

Protected Attributes

- **algo_comm_t ac**
- std::string **name**

Additional Inherited Members

5.140.1 Detailed Description

Insert a **MHASignal::waveform_t** (p. 812) class into the AC space.

The variable is automatically removed on destruction.

5.140.2 Constructor & Destructor Documentation

5.140.2.1 MHA_AC::waveform_t::waveform_t (algo_comm_t *ac*, std::string *name*, unsigned int *frames*, unsigned int *channels*, bool *insert_now*)

Create the AC variable.

Parameters

ac	AC handle
name	Name of variable in AC space
frames	Number of frames in the **waveform_t** (p. 425) class
channels	Number of audio channels in the **waveform_t** (p. 425) class
insert_now	Insert implicitely in the constructor (true) or explicitely in the **insert()** (p. 427) function (false)

5.140.2.2 MHA_AC::waveform_t::~waveform_t (void) [virtual]

Reimplemented from **MHASignal::waveform_t** (p. 815).

5.140.3 Member Function Documentation

5.140.3.1 void MHA_AC::waveform_t::insert ( )

Insert AC variable into AC space.

5.140.4 Member Data Documentation

5.140.4.1 algo_comm_t MHA_AC::waveform_t::ac [protected]

5.140.4.2 std::string MHA_AC::waveform_t::name [protected]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.141 mha_audio_descriptor_t Struct Reference

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 504)).

Public Attributes

- unsigned int **n_samples**
  Number of samples.
- unsigned int **n_channels**
  Number of audio channels.
- unsigned int **n_freqs**
  Number of frequency bands.
- unsigned int **is_complex**
  Flag about sample type.
- **mha_real_t dt**
  Time distance between samples (only equidistant samples allowed)
- **mha_real_t** * **cf**
  Center frequencies of frequency bands.
- **mha_real_t** * **chdir**
  Hint on source direction of channel, values below zero is left, values above zero is right, zero means unknown.

5.141.1 Detailed Description

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 504)).

5.141.2 Member Data Documentation

5.141.2.1 unsigned int mha_audio_descriptor_t::n_samples

Number of samples.

5.141.2.2 unsigned int mha_audio_descriptor_t::n_channels

Number of audio channels.

5.141.2.3 unsigned int mha_audio_descriptor_t::n_freqs

Number of frequency bands.

5.141.2.4 unsigned int mha_audio_descriptor_t::is_complex

Flag about sample type.

5.141.2.5 mha_real_t mha_audio_descriptor_t::dt

Time distance between samples (only equidistant samples allowed)

5.141.2.6 mha_real_t* mha_audio_descriptor_t::cf

Center frequencies of frequency bands.

5.141.2.7 mha_real_t* mha_audio_descriptor_t::chdir

Hint on source direction of channel, values below zero is left, values above zero is right, zero means unknown.

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

- **mha.hh**

5.142 mha_audio_t Struct Reference

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465)).

Public Attributes

- **mha_audio_descriptor_t descriptor**
  Dimension and description of the data.
- **mha_real_t ∗ rdata**
  *Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 428) is unset.*
- **mha_complex_t ∗ cdata**
  *Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 428) is set.*

5.142.1 Detailed Description

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465)).

The data alignment is $(t_0, c_0, f_0), (t_0, c_0, f_1), \ldots, (t_0, c_0, f_{freqs}), (t_0, c_1, f_0), \ldots$. This allows a direct cast of the current **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465) data pointers into corresponding **mha_audio_t** (p. 428) objects.

5.142.2 Member Data Documentation

5.142.2.1 mha_audio_descriptor_t mha_audio_t::descriptor

Dimension and description of the data.

5.142.2.2 mha_real_t∗ mha_audio_t::rdata

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 428) is unset.

5.142.2.3 mha_complex_t∗ mha_audio_t::cdata

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 428) is set.

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

- **mha.hh**

5.143 mha_channel_info_t Struct Reference

Channel information structure.

Public Attributes

- int **id**
  channel id
- char **idstr** [32]
  channel id
- unsigned int **side**
  side (left/right)
- **mha_direction_t dir**
  source direction
- **mha_real_t peaklevel**
  Peak level corresponds to this SPL (dB) level.

5.143.1 Detailed Description

Channel information structure.

5.143.2 Member Data Documentation

5.143.2.1 int mha_channel_info_t::id

channel id

5.143.2.2 char mha_channel_info_t::idstr[32]

channel id

5.143.2.3 unsigned int mha_channel_info_t::side

side (left/right)

5.143.2.4 mha_direction_t mha_channel_info_t::dir

source direction

5.143.2.5 mha_real_t mha_channel_info_t::peaklevel

Peak level corresponds to this SPL (dB) level.

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

- **mha.hh**

5.144 mha_complex_t Struct Reference

Type for complex floating point values.

Public Attributes

- **mha_real_t re**
  Real part.
- **mha_real_t im**
  Imaginary part.

5.144.1 Detailed Description

Type for complex floating point values.

5.144.2 Member Data Documentation

5.144.2.1 mha_real_t mha_complex_t::re

Real part.

5.144.2.2 mha_real_t mha_complex_t::im

Imaginary part.

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

- **mha.hh**

5.145 mha_complex_test_array_t Struct Reference

Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 431) c[] to an array of mha_real_t r[] with r[0] == c[0].re r[1] == c[0].im r[2] == c[1].re ...

Public Attributes

- **mha_complex_t c** [2]

5.145.1 Detailed Description

Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 431) c[] to an array of mha_real_t r[] with r[0] == c[0].re r[1] == c[0].im r[2] == c[1].re ...

Check these expectations in static asserts.

5.145.2 Member Data Documentation

5.145.2.1 mha_complex_t mha_complex_test_array_t::c[2]

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

- **mha.hh**

5.146 mha_dblbuf_t< FIFO > Class Template Reference

The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.

Public Types

- typedef FIFO::value_type **value_type**
  The datatype exchanged by the FIFO and this doublebuffer.

Public Member Functions

- virtual unsigned **get_inner_size** () const
- virtual unsigned **get_outer_size** () const
- virtual unsigned **get_delay** () const
- virtual unsigned **get_fifo_size** () const
- virtual unsigned **get_input_channels** () const
- virtual unsigned **get_output_channels** () const
- virtual unsigned **get_input_fifo_fill_count** () const
- virtual unsigned **get_output_fifo_fill_count** () const
- virtual unsigned **get_input_fifo_space** () const
- virtual unsigned **get_output_fifo_space** () const
- virtual **MHA_Error** * **get_inner_error** () const
- virtual void **provoke_inner_error** (const **MHA_Error** &)
- virtual void **provoke_outer_error** (const **MHA_Error** &)

- **mha_dblbuf_t** (unsigned **outer_size**, unsigned **inner_size**, unsigned **delay**, unsigned **input_channels**, unsigned **output_channels**, const **value_type** &delay_data)

  Constructor creates FIFOs with specified delay.
- virtual ~**mha_dblbuf_t** ()
- virtual void **process** (const **value_type** *input_signal, **value_type** *output_signal, unsigned count)

  The outer process has to call this method to propagate the input signal to the inner process, and receives back the output signal.
- virtual void **input** (**value_type** *input_signal)

  The inner process has to call this method to receive its input signal.
- virtual void **output** (const **value_type** *output_signal)

  The outer process has to call this method to deliver its output signal.

Private Attributes

- unsigned **outer_size**

  The block size used by the outer process.
- unsigned **inner_size**

  The block size used by the inner process.
- unsigned **delay**

  The delay introduced by bidirectional buffer size adaptation.
- unsigned **fifo_size**

  The size of each of the FIFOs.
- unsigned **input_channels**

  The number of input channels.
- unsigned **output_channels**

  The number of output channels.
- FIFO **input_fifo**

  The FIFO for transporting the input signal from the outer process to the inner process.
- FIFO **output_fifo**

  The FIFO for transporting the output signal from the inner process to the outer process.
- **MHA_Error** * **inner_error**

  Owned copy of exception to be thrown in inner thread.
- **MHA_Error** * **outer_error**

  Owned copy of exception to be thrown in outer thread.

5.146.1 Detailed Description

template<class FIFO>
class mha_dblbuf_t< FIFO >

The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.

This class introduces the channels concept. Input and output may have different channel counts.

5.146.2 Member Typedef Documentation

5.146.2.1 template<class FIFO > typedef FIFO::value_type mha_dblbuf_t< FIFO >::value_type

The datatype exchanged by the FIFO and this doublebuffer.

5.146.3 Constructor & Destructor Documentation

5.146.3.1 template<class FIFO > mha_dblbuf_t< FIFO >::mha_dblbuf_t (unsigned *outer_size,* unsigned *inner_size,* unsigned *delay,* unsigned *input_channels,* unsigned *output_channels,* const value_type & *delay_data*)

Constructor creates FIFOs with specified delay.

Warning

The doublebuffer may block or raise an exception if the delay is too small. To avoid this, the delay should be

$$delay >= (inner\_size - gcd(inner\_size, outer\_size))$$

.

Parameters

outer_size	The block size used by the outer process.
inner_size	The block size used by the inner process.
delay	The total delay
input_channels	Number of input channels
output_channels	Number of output channels
delay_data	The delay consists of copies of this value.

5.146.3.2 template<class FIFO > mha_dblbuf_t< FIFO >::~mha_dblbuf_t ( ) [virtual]

5.146.4 Member Function Documentation

5.146.4.1 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_inner_size ( ) const [inline], [virtual]

5.146.4.2 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_outer_size ( ) const [inline], [virtual]

5.146.4.3 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_delay ( ) const [inline], [virtual]

5.146.4.4 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_fifo_size ( ) const [inline], [virtual]

5.146.4.5 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_channels ( ) const [inline], [virtual]

5.146.4.6 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_channels ( ) const [inline], [virtual]

5.146.4.7 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_fifo_fill_count ( ) const [inline], [virtual]

5.146.4.8 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_fifo_fill_count ( ) const [inline], [virtual]

5.146.4.9 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_fifo_space ( ) const [inline], [virtual]

5.146.4.10 template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_fifo_space ( ) const [inline], [virtual]

5.146.4.11 template<class FIFO > virtual MHA_Error* mha_dblbuf_t< FIFO >::get_inner_error ( ) const [inline], [virtual]

5.146.4.12 template<class FIFO > void mha_dblbuf_t< FIFO >::provoke_inner_error (const MHA_Error & *error*) [virtual]

5.146.4.13 template<class FIFO > void mha_dblbuf_t< FIFO >::provoke_outer_error (const MHA_Error & *error*) [virtual]

5.146.4.14 template<class FIFO > void mha_dblbuf_t< FIFO >::process (const value_type * *input_signal,* value_type * *output_signal,* unsigned *count*) [virtual]

The outer process has to call this method to propagate the input signal to the inner process, and receives back the output signal.

Parameters

input_signal	Pointer to the input signal array.
output_signal	Pointer to the output signal array.
count	The number of data instances provided and expected, lower or equal to inner_size given to constructor.

Exceptions

MHA_Error *(p. 445)*	When count is > outer_size as given to constructor or the underlying fifo implementation detects an error.

5.146.4.15 template<class FIFO > void mha_dblbuf_t< FIFO >::input (value_type * *input_signal*) [virtual]

The inner process has to call this method to receive its input signal.

Parameters

input_signal	Array where the doublebuffer can store the signal.

Exceptions

MHA_Error *(p. 445)*	When the underlying fifo implementation detects an error.

5.146.4.16 template<class FIFO > void mha_dblbuf_t< FIFO >::output (const value_type * *output_signal*) [virtual]

The outer process has to call this method to deliver its output signal.

Parameters

output_signal	Array from which doublebuffer reads outputsignal.

Exceptions

MHA_Error *(p. 445)*	When the underlying fifo implementation detects an error.

5.146.5 Member Data Documentation

5.146.5.1 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::outer_size [private]

The block size used by the outer process.

5.146.5.2 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::inner_size [private]

The block size used by the inner process.

5.146.5.3 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::delay [private]

The delay introduced by bidirectional buffer size adaptation.

5.146.5.4 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::fifo_size [private]

The size of each of the FIFOs.

5.146.5.5 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::input_channels [private]

The number of input channels.

5.146.5.6 template<class FIFO > unsigned mha_dblbuf_t< FIFO >::output_channels [private]

The number of output channels.

5.146.5.7 template<class FIFO > FIFO mha_dblbuf_t< FIFO >::input_fifo [private]

The FIFO for transporting the input signal from the outer process to the inner process.

5.146.5.8 template<class FIFO > FIFO mha_dblbuf_t< FIFO >::output_fifo [private]

The FIFO for transporting the output signal from the inner process to the outer process.

5.146.5.9 template<class FIFO > MHA_Error* mha_dblbuf_t< FIFO >::inner_error [private]

Owned copy of exception to be thrown in inner thread.

5.146.5.10 template<class FIFO > MHA_Error* mha_dblbuf_t< FIFO >::outer_error [private]

Owned copy of exception to be thrown in outer thread.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

5.147 mha_direction_t Struct Reference

Channel source direction structure.

Public Attributes

- **mha_real_t azimuth**
  azimuth in radiants
- **mha_real_t elevation**
  elevation in radiants
- **mha_real_t distance**
  distance in meters

5.147.1 Detailed Description

Channel source direction structure.

5.147.2 Member Data Documentation

5.147.2.1 mha_real_t mha_direction_t::azimuth

azimuth in radiants

5.147.2.2 mha_real_t mha_direction_t::elevation

elevation in radiants

5.147.2.3 mha_real_t mha_direction_t::distance

distance in meters

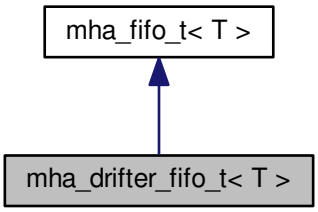
The documentation for this struct was generated from the following file:

- **mha.hh**

5.148 mha_drifter_fifo_t< T > Class Template Reference

A FIFO class for blocksize adaptation without Synchronization.

Inheritance diagram for mha_drifter_fifo_t< T >:



Public Member Functions

- virtual void **write** (const T *data, unsigned count)

  write data to fifo
- virtual void **read** (T ***buf**, unsigned count)

  Read data from fifo.
- virtual unsigned **get_fill_count** () const

  *Return fill_count, adding **mha_drifter_fifo_t<T>::startup_zeros** (p. 445) to the number of samples actually in the fifo's buffer.*
- virtual unsigned **get_available_space** () const

  *Return available space, subtracting number of **mha_drifter_fifo_t<T>::startup_zeros** (p. 445) from the available_space actually present in the fifo's buffer.*
- virtual unsigned **get_des_fill_count** () const

  The desired fill count of this fifo.
- virtual unsigned **get_min_fill_count** () const

  The minimum fill count of this fifo.
- virtual void **stop** ()

  *Called by **mha_drifter_fifo_t<T>::read** (p. 441) or **mha_drifter_fifo_t<T>::write** (p. 441) when their xrun in succession counter exceeds its limit.*
- virtual void **starting** ()

  *Called by **mha_drifter_fifo_t<T>::read** (p. 441) or **mha_drifter_fifo_t<T>::write** (p. 441) when the respective flag (**mha_drifter_fifo_t<T>::reader_started** (p. 443) or **mha_drifter_fifo_t<T>::writer_started** (p. 443)) is about to be toggled from false to true.*
- **mha_drifter_fifo_t** (unsigned min_fill_count, unsigned **desired_fill_count**, unsigned **max_fill_count**)

  Create drifter FIFO.
- **mha_drifter_fifo_t** (unsigned min_fill_count, unsigned **desired_fill_count**, unsigned **max_fill_count**, const T &t)

  Create drifter FIFO where all (initially unused) copies of T are initialized as copies of t.

Private Attributes

- const unsigned **minimum_fill_count**

  The minimum fill count of this fifo.
- const unsigned **desired_fill_count**

  The desired fill count of the fifo.
- bool **writer_started**

  Flag set to true when write is called the first time.
- bool **reader_started**

  Flag set to true when read is called for the first time.
- unsigned **writer_xruns_total**

  The number of xruns seen by the writer since object instantiation.
- unsigned **reader_xruns_total**

  The number of xruns seen by the reader since object instantiation.
- unsigned **writer_xruns_since_start**

  The number of xruns seen by the writer since the last start of processing.
- unsigned **reader_xruns_since_start**

  The number of xruns seen by the reader since the last start of processing.
- unsigned **writer_xruns_in_succession**

  The number of xruns seen by the writer in succession.
- unsigned **reader_xruns_in_succession**

  The number of xruns seen by the reader in succession.
- unsigned **maximum_writer_xruns_in_succession_before_stop**

  A limit to the number of xruns seen in succession during write before the data transmission through the FIFO is stopped.
- unsigned **maximum_reader_xruns_in_succession_before_stop**

  A limit to the number of xruns seen in succession during read before the data transmission through the FIFO is stopped.
- **mha_fifo_t**< T >::**value_type null_data**

  The value used in place of missing data.
- unsigned **startup_zeros**

  *When processing starts, that is when both **mha_drifter_fifo_t<T>::reader_started** (p. 443) and **mha_drifter_fifo_t<T>::writer_started** (p. 443) are true, then first **mha_drifter_fifo_t<T>::desired_fill_count** (p. 443) instances of **mha_drifter_fifo_t<T>::null_data** (p. 444) are delivered to the reader.*

Additional Inherited Members

5.148.1 Detailed Description

template<class T>
class mha_drifter_fifo_t< T >

A FIFO class for blocksize adaptation without Synchronization.

Features: delay concept (desired, minimum and maximum delay), drifting support by throwing away data or inserting zeroes.

5.148.2 Constructor & Destructor Documentation

5.148.2.1 template<class T > mha_drifter_fifo_t< T >::mha_drifter_fifo_t (unsigned *min_fill_count,* unsigned *desired_fill_count,* unsigned *max_fill_count*)

Create drifter FIFO.

5.148.2.2 template<class T > mha_drifter_fifo_t< T >::mha_drifter_fifo_t (unsigned *min_fill_count,* unsigned *desired_fill_count,* unsigned *max_fill_count,* const T & *t*)

Create drifter FIFO where all (initially unused) copies of T are initialized as copies of t.

5.148.3 Member Function Documentation

5.148.3.1 template<class T > void mha_drifter_fifo_t< T >::write (const T * *data,* unsigned *count*) [virtual]

write data to fifo

Sets **writer_started** (p. 443) to true.

When processing has started, i.e. both **reader_started** (p. 443) and **writer_started** (p. 443) are true, write specified ammount of data to the fifo. If there is not enough space available, then the exceeding data is lost and the writer xrun counters are increased.

Processing is stopped when **writer_xruns_in_succession** (p. 444) exceeds **maximum_writer_xruns_in_succession_before_stop** (p. 444).

Parameters

data	Pointer to source data.
count	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 454).

5.148.3.2 template<class T > void mha_drifter_fifo_t< T >::read (T * *buf,* unsigned *count*) [virtual]

Read data from fifo.

Sets **reader_started** (p. 443) to true.

When processing has started, i.e. both **reader_started** (p. 443) and **writer_started** (p. 443) are true, then read specified ammount of data from the fifo. As long as **startup_zeros** (p. 445) is > 0, **null_data** (p. 444) is delivered to the reader and **startup_zeros** (p. 445) is diminished. Only when **startup_zeros** (p. 445) has reached 0, data is actually read from the fifo's buffer.

If the read would cause the fifo's fill count to drop below **minimum_fill_count** (p. 443), then only so much data are read that **minimum_fill_count** (p. 443) entries remain in the fifo, the missing data is replaced with **null_data** (p. 444), and the reader xrun counters are increased.

Processing is stopped when **reader_xruns_in_succession** (p. 444) exceeds **maximum_↩ reader_xruns_in_succession_before_stop** (p. 444).

Parameters

buf	Pointer to the target buffer
count	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 455).

5.148.3.3 template<class T > unsigned mha_drifter_fifo_t< T >::get_fill_count ( ) const [virtual]

Return fill_count, adding **mha_drifter_fifo_t<T>::startup_zeros** (p. 445) to the number of samples actually in the fifo's buffer.

Reimplemented from **mha_fifo_t< T >** (p. 455).

5.148.3.4 template<class T > unsigned mha_drifter_fifo_t< T >::get_available_space ( ) const [virtual]

Return available space, subtracting number of **mha_drifter_fifo_t<T>::startup_zeros** (p. 445) from the available_space actually present in the fifo's buffer.

TODO: uncertain if this is a good idea.

Reimplemented from **mha_fifo_t< T >** (p. 455).

5.148.3.5 template<class T > virtual unsigned mha_drifter_fifo_t< T >::get_des_fill_count ( ) const [inline], [virtual]

The desired fill count of this fifo.

5.148.3.6 template<class T > virtual unsigned mha_drifter_fifo_t< T >::get_min_fill_count ( ) const [inline], [virtual]

The minimum fill count of this fifo.

5.148.3.7 template<class T > void mha_drifter_fifo_t< T >::stop ( ) [virtual]

Called by **mha_drifter_fifo_t<T>::read** (p. 441) or **mha_drifter_fifo_t<T>::write** (p. 441) when their xrun in succession counter exceeds its limit.

Called by **read** (p. 441) or **write** (p. 441) when their xrun in succession counter exceeds its limit.

May also be called explicitly.

5.148.3.8 template<class T > void mha_drifter_fifo_t< T >::starting ( ) [virtual]

Called by **mha_drifter_fifo_t<T>::read** (p. 441) or **mha_drifter_fifo_t<T>::write** (p. 441) when the respective flag (**mha_drifter_fifo_t<T>::reader_started** (p. 443) or **mha_drifter_↩ fifo_t<T>::writer_started** (p. 443)) is about to be toggled from false to true.

The fifo's buffer is emptied, this method resets **startup_zeros** (p. 445) to **desired_fill_count** (p. 443), and it also resets **reader_xruns_since_start** (p. 444) and **writer_xruns_since_start** (p. 444) to 0.

5.148.4 Member Data Documentation

5.148.4.1 template<class T > const unsigned mha_drifter_fifo_t< T >::minimum_fill_count [private]

The minimum fill count of this fifo.

5.148.4.2 template<class T > const unsigned mha_drifter_fifo_t< T >::desired_fill_count [private]

The desired fill count of the fifo.

The fifo is initialized with this ammount of data when data transmission starts.

5.148.4.3 template<class T > bool mha_drifter_fifo_t< T >::writer_started [private]

Flag set to true when write is called the first time.

5.148.4.4 template<class T > bool mha_drifter_fifo_t< T >::reader_started [private]

Flag set to true when read is called for the first time.

5.148.4.5 template<class T > unsigned mha_drifter_fifo_t< T >::writer_xruns_total [private]

The number of xruns seen by the writer since object instantiation.

5.148.4.6 template<class T > unsigned mha_drifter_fifo_t< T >::reader_xruns_total [private]

The number of xruns seen by the reader since object instantiation.

5.148.4.7 template<class T > unsigned mha_drifter_fifo_t< T >::writer_xruns_since_start [private]

The number of xruns seen by the writer since the last start of processing.

5.148.4.8 template<class T > unsigned mha_drifter_fifo_t< T >::reader_xruns_since_start [private]

The number of xruns seen by the reader since the last start of processing.

5.148.4.9 template<class T > unsigned mha_drifter_fifo_t< T >::writer_xruns_in_succession [private]

The number of xruns seen by the writer in succession.

Reset to 0 every time a write succeeds without xrun.

5.148.4.10 template<class T > unsigned mha_drifter_fifo_t< T >::reader_xruns_in_succession [private]

The number of xruns seen by the reader in succession.

Reset to 0 every time a read succeeds without xrun.

5.148.4.11 template<class T > unsigned mha_drifter_fifo_t< T >::maximum_writer_xruns_in_succession_before_stop [private]

A limit to the number of xruns seen in succession during write before the data transmission through the FIFO is stopped.

5.148.4.12 template<class T > unsigned mha_drifter_fifo_t< T >::maximum_reader_xruns_in_succession_before_stop [private]

A limit to the number of xruns seen in succession during read before the data transmission through the FIFO is stopped.

5.148.4.13 template<class T > mha_fifo_t<T>::value_type mha_drifter_fifo_t< T >::null_data [private]

The value used in place of missing data.

5.148.4.14 template<class T > unsigned mha_drifter_fifo_t< T >::startup_zeros [private]

When processing starts, that is when both **mha_drifter_fifo_t<T>::reader_started** (p. 443) and **mha_drifter_fifo_t<T>::writer_started** (p. 443) are true, then first **mha_drifter_fifo_t<T>::desired_fill_count** (p. 443) instances of **mha_drifter_fifo_t<T>::null_data** (p. 444) are delivered to the reader.

These **null_data** (p. 444) instances are not transmitted through the fifo because filling the fifo with enough **null_data** (p. 444) might not be realtime safe and this filling has to be initiated by **starting** (p. 443) or **stop** (p. 443) (this implementation: **starting** (p. 443)) which are be called with realtime constraints.

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

- **mha_fifo.h**

5.149 MHA_Error Class Reference

Error reporting exception class.

Inherits exception.

Public Member Functions

- **MHA_Error** (const char *file, int line, const char *fmt,...)
  *Create an instance of a **MHA_Error** (p. 445).*
- **MHA_Error** (const **MHA_Error** &)
- **MHA_Error** & **operator=** (const **MHA_Error** &)
- ~**MHA_Error** () throw ()
- const char * **get_msg** () const
  Return the error message without source position.
- const char * **get_longmsg** () const
  Return the error message with source position.
- const char * **what** () const throw ()
  overwrite std::execption::what()

Private Attributes

- char * **msg**
- char * **longmsg**

5.149.1 Detailed Description

Error reporting exception class.

This class is used for error handling in the openMHA. It is used by the openMHA kernel and by the openMHA toolbox library. Please note that exceptions should not be used accross ANSI-C interfaces. It is necessary to catch exceptions within the library.

The **MHA_Error** (p. 445) class holds source file name, line number and an error message.

5.149.2 Constructor & Destructor Documentation

5.149.2.1 MHA_Error::MHA_Error (const char * *s_file,* int *l,* const char * *fmt,* ...)

Create an instance of a **MHA_Error** (p. 445).

Parameters

s_file	source file name (**FILE**)
l	source line (**LINE**)
fmt	format string for error message (as in printf)

5.149.2.2 MHA_Error::MHA_Error (const MHA_Error & *p*)

5.149.2.3 MHA_Error::~MHA_Error ( ) throw)

5.149.3 Member Function Documentation

5.149.3.1 MHA_Error & MHA_Error::operator= (const MHA_Error & *p*)

5.149.3.2 const char* MHA_Error::get_msg ( ) const [inline]

Return the error message without source position.

5.149.3.3 const char* MHA_Error::get_longmsg ( ) const [inline]

Return the error message with source position.

5.149.3.4 const char* MHA_Error::what ( ) const throw) [inline]

overwrite std::execption::what()

5.149.4 Member Data Documentation

5.149.4.1 char* MHA_Error::msg [private]

5.149.4.2 char* MHA_Error::longmsg [private]

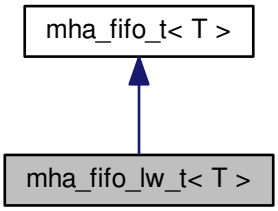
The documentation for this class was generated from the following files:

- **mha_error.hh**
- **mha_error.cpp**

5.150 mha_fifo_lw_t< T > Class Template Reference

This FIFO uses locks to synchronize access.

Inheritance diagram for mha_fifo_lw_t< T >:



Public Member Functions

- virtual void **write** (const T *data, unsigned count)
  write specified ammount of data to the fifo.
- virtual void **read** (T ***buf**, unsigned count)
  read data from fifo.
- **mha_fifo_lw_t** (unsigned **max_fill_count**)
  Create FIFO with fixed buffer size.
- virtual ~**mha_fifo_lw_t** ()
  release synchronization object
- virtual void **set_error** (unsigned index, **MHA_Error** ***error**)
  Process waiting for more data or space should bail out, throwing this error.

Private Attributes

- **mha_fifo_thread_platform_t * sync**
  platform specific thread synchronization
- **MHA_Error * error** [2]
  If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

Additional Inherited Members

5.150.1 Detailed Description

template<class T>
class mha_fifo_lw_t< T >

This FIFO uses locks to synchronize access.

Reading and writing can block until the operation can be executed.

5.150.2 Constructor & Destructor Documentation

5.150.2.1 template<class T > mha_fifo_lw_t< T >::mha_fifo_lw_t (unsigned *max_fill_count*) [explicit]

Create FIFO with fixed buffer size.

5.150.2.2 template<class T > mha_fifo_lw_t< T >::~mha_fifo_lw_t ( ) [virtual]

release synchronization object

5.150.3 Member Function Documentation

5.150.3.1 template<class T > void mha_fifo_lw_t< T >::write (const T * *data,* unsigned *count*) [virtual]

write specified ammount of data to the fifo.

If there is not enough space, then wait for more space.

Parameters

data	Pointer to source data.
count	Number of instances to copy.

Exceptions

MHA_Error *(p. 445)*	when detecting a deadlock situation.

Reimplemented from **mha_fifo_t< T >** (p. 454).

5.150.3.2 template<class T > void mha_fifo_lw_t< T >::read (T * *buf*, unsigned *count*) [virtual]

read data from fifo.

If there is not enough data, then wait for more data.

Parameters

buf	Pointer to the target buffer.
count	Number of instances to copy.

Exceptions

MHA_Error *(p. 445)*	when detecting a deadlock situation.

Reimplemented from **mha_fifo_t< T >** (p. 455).

5.150.3.3 template<class T > void mha_fifo_lw_t< T >::set_error (unsigned *index*, MHA_Error * *error*) [virtual]

Process waiting for more data or space should bail out, throwing this error.

Parameters

index	Use 0 for terminating reader, 1 for terminating writer.
error	**MHA_Error** (p. 445) to be thrown

5.150.4 Member Data Documentation

5.150.4.1 template<class T > mha_fifo_thread_platform_t* mha_fifo_lw_t< T >::sync [private]

platform specific thread synchronization

5.150.4.2 template<class T > MHA_Error* mha_fifo_lw_t< T >::error[2] [private]

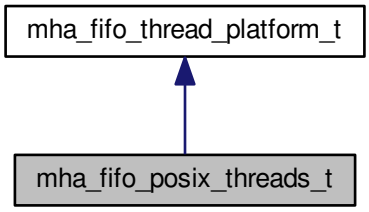
If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

5.151 mha_fifo_posix_threads_t Class Reference

Inheritance diagram for mha_fifo_posix_threads_t:



Public Member Functions

- **mha_fifo_posix_threads_t** ()
- virtual void **aquire_mutex** ()
- virtual void **release_mutex** ()
- virtual void **wait_for_decrease** ()
- virtual void **wait_for_increase** ()
- virtual void **increment** ()
- virtual void **decrement** ()
- virtual ~**mha_fifo_posix_threads_t** ()

Private Attributes

- pthread_mutex_t **mutex**
- pthread_cond_t **decrease_condition**
- pthread_cond_t **increase_condition**

5.151.1 Constructor & Destructor Documentation

5.151.1.1 mha_fifo_posix_threads_t::mha_fifo_posix_threads_t ( ) [inline]

5.151.1.2 virtual mha_fifo_posix_threads_t::~mha_fifo_posix_threads_t ( ) [inline], [virtual]

5.151.2 Member Function Documentation

5.151.2.1 virtual void mha_fifo_posix_threads_t::aquire_mutex ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 459).

5.151.2.2 virtual void mha_fifo_posix_threads_t::release_mutex ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 459).

5.151.2.3 virtual void mha_fifo_posix_threads_t::wait_for_decrease ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 459).

5.151.2.4 virtual void mha_fifo_posix_threads_t::wait_for_increase ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 459).

5.151.2.5 virtual void mha_fifo_posix_threads_t::increment ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 459).

5.151.2.6 virtual void mha_fifo_posix_threads_t::decrement ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 460).

5.151.3 Member Data Documentation

5.151.3.1 pthread_mutex_t mha_fifo_posix_threads_t::mutex [private]

5.151.3.2 pthread_cond_t mha_fifo_posix_threads_t::decrease_condition [private]

5.151.3.3 pthread_cond_t mha_fifo_posix_threads_t::increase_condition [private]

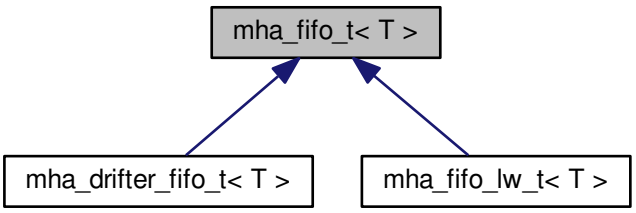
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.152 mha_fifo_t< T > Class Template Reference

A FIFO class for blocksize adaptation Synchronization: None.

Inheritance diagram for mha_fifo_t< T >:



Public Types

- typedef T **value_type**

  The data type exchanged by this fifo.

Public Member Functions

- virtual void **write** (const T *data, unsigned count)
  write specified ammount of data to the fifo.
- virtual void **read** (T ***buf**, unsigned count)
  read data from fifo
- virtual unsigned **get_fill_count** () const
  Read-only access to fill_count.
- virtual unsigned **get_available_space** () const
  Read-only access to available_space.
- virtual unsigned **get_max_fill_count** () const
  The capacity of this fifo.
- **mha_fifo_t** (unsigned **max_fill_count**)
  Create FIFO with fixed buffer size.
- **mha_fifo_t** (unsigned **max_fill_count**, const T &t)
  Create FIFO with fixed buffer size, where all (initially unused) copies of T are initialized as copies of t.
- **mha_fifo_t** (const **mha_fifo_t** &src)
  Copy constructor.
- virtual ~**mha_fifo_t** ()
  Destroy FIFO.
- **mha_fifo_t**< T > & **operator=** (const **mha_fifo_t**< T > &)
  Assignment operator.

Protected Member Functions

- void **clear** ()
  Empty the fifo at once.

Private Attributes

- const unsigned **max_fill_count**
  The maximum fill count of this FIFO.
- T * **buf**
  The memory allocated to store the data.
- T * **write_ptr**
  points to location where to write next
- const T * **read_ptr**
  points to location where to read next
- bool **buf_uses_placement_new**
  wether buf was allocated using placement new or array new.

5.152.1 Detailed Description

template<class T>
class mha_fifo_t< T >

A FIFO class for blocksize adaptation Synchronization: None.

Use external synchronisation or synchronization in inheriting class.

5.152.2 Member Typedef Documentation

5.152.2.1 template<class T> typedef T mha_fifo_t< T >::value_type

The data type exchanged by this fifo.

5.152.3 Constructor & Destructor Documentation

5.152.3.1 template<class T > mha_fifo_t< T >::mha_fifo_t (unsigned *max_fill_count*) [explicit]

Create FIFO with fixed buffer size.

5.152.3.2 template<class T> mha_fifo_t< T >::mha_fifo_t (unsigned *max_fill_count,* const T & *t*)

Create FIFO with fixed buffer size, where all (initially unused) copies of T are initialized as copies of t.

5.152.3.3 template<class T> mha_fifo_t< T >::mha_fifo_t (const mha_fifo_t< T > & *src*)

Copy constructor.

5.152.3.4 template<class T > mha_fifo_t< T >::~mha_fifo_t ( ) [virtual]

Destroy FIFO.

5.152.4 Member Function Documentation

5.152.4.1 template<class T> void mha_fifo_t< T >::write (const T * *data,* unsigned *count*) [virtual]

write specified ammount of data to the fifo.

Parameters

data	Pointer to source data.
count	Number of instances to copy

Exceptions

MHA_Error *(p. 445)*	when there is not enough space available.

Reimplemented in **mha_fifo_lw_t< T >** (p. 448), and **mha_drifter_fifo_t< T >** (p. 441).

5.152.4.2 template<class T> void mha_fifo_t< T >::read (T * *buf*, unsigned *count*) [virtual]

read data from fifo

Parameters

buf	Pointer to the target buffer
count	Number of instances to copy

Exceptions

MHA_Error *(p. 445)*	when there is not enough data available.

Reimplemented in **mha_fifo_lw_t< T >** (p. 449), and **mha_drifter_fifo_t< T >** (p. 441).

5.152.4.3 template<class T > unsigned mha_fifo_t< T >::get_fill_count ( ) const [virtual]

Read-only access to fill_count.

Reimplemented in **mha_drifter_fifo_t< T >** (p. 442).

5.152.4.4 template<class T > unsigned mha_fifo_t< T >::get_available_space ( ) const [virtual]

Read-only access to available_space.

Reimplemented in **mha_drifter_fifo_t< T >** (p. 442).

5.152.4.5 template<class T> virtual unsigned mha_fifo_t< T >::get_max_fill_count ( ) const [inline], [virtual]

The capacity of this fifo.

5.152.4.6 template<class T> mha_fifo_t< T > & mha_fifo_t< T >::operator= (const mha_fifo_t< T > & *src*)

Assignment operator.

5.152.4.7 template<class T> void mha_fifo_t< T >::clear ( ) [inline], [protected]

Empty the fifo at once.

Should be called by the reader, or when the reader is inactive.

5.152.5 Member Data Documentation

5.152.5.1 template<class T> const unsigned mha_fifo_t< T >::max_fill_count [private]

The maximum fill count of this FIFO.

5.152.5.2 template<class T> T* mha_fifo_t< T >::buf [private]

The memory allocated to store the data.

max_fill_count + 1 locations are allocated: At least one location is always unused, because we have max_fill_count + 1 possible fillcounts [0:max_fill_count] that we need to distinguish.

5.152.5.3 template<class T> T* mha_fifo_t< T >::write_ptr [private]

points to location where to write next

5.152.5.4 template<class T> const T* mha_fifo_t< T >::read_ptr [private]

points to location where to read next

5.152.5.5 template<class T> bool mha_fifo_t< T >::buf_uses_placement_new [private]

wether buf was allocated using placement new or array new.

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

- **mha_fifo.h**

5.153 mha_fifo_thread_guard_t Class Reference

Simple Mutex Guard Class.

Public Member Functions

- **mha_fifo_thread_guard_t** (**mha_fifo_thread_platform_t** ∗**sync**)
- ∼**mha_fifo_thread_guard_t** ()

Private Attributes

- **mha_fifo_thread_platform_t** ∗ **sync**

5.153.1 Detailed Description

Simple Mutex Guard Class.

5.153.2 Constructor & Destructor Documentation

5.153.2.1 mha_fifo_thread_guard_t::mha_fifo_thread_guard_t (mha_fifo_thread_platform_t ∗ *sync*) [inline]

5.153.2.2 mha_fifo_thread_guard_t::∼mha_fifo_thread_guard_t ( ) [inline]

5.153.3 Member Data Documentation

5.153.3.1 mha_fifo_thread_platform_t∗ mha_fifo_thread_guard_t::sync [private]

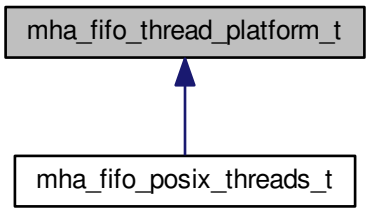
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.154 mha_fifo_thread_platform_t Class Reference

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Inheritance diagram for mha_fifo_thread_platform_t:



Public Member Functions

- virtual void **aquire_mutex** ()=0
  Calling thread waits until it aquires the lock.
- virtual void **release_mutex** ()=0
  Calling thread releases the lock.
- virtual void **wait_for_decrease** ()=0
  Calling producer thread must own the lock.
- virtual void **wait_for_increase** ()=0
  Calling consumer thread must own the lock.
- virtual void **increment** ()=0
  To be called by producer thread after producing.
- virtual void **decrement** ()=0
  To be called by consumer thread after consuming.
- virtual ∼**mha_fifo_thread_platform_t** ()
  Make destructor virtual.
- **mha_fifo_thread_platform_t** ()
  Make default constructor accessible.

Private Member Functions

- **mha_fifo_thread_platform_t** (const **mha_fifo_thread_platform_t** &)
- **mha_fifo_thread_platform_t** & **operator=** (const **mha_fifo_thread_platform_t** &)

5.154.1 Detailed Description

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Works only with single producer and single consumer.

5.154.2 Constructor & Destructor Documentation

5.154.2.1 virtual mha_fifo_thread_platform_t::∼mha_fifo_thread_platform_t ( ) [inline], [virtual]

Make destructor virtual.

5.154.2.2 mha_fifo_thread_platform_t::mha_fifo_thread_platform_t (const mha_fifo_thread_platform_t &) [private]

5.154.2.3 mha_fifo_thread_platform_t::mha_fifo_thread_platform_t ( ) [inline]

Make default constructor accessible.

5.154.3 Member Function Documentation

5.154.3.1 virtual void mha_fifo_thread_platform_t::aquire_mutex ( ) [pure virtual]

Calling thread waits until it aquires the lock.

Must not be called when the lock is already aquired.

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.2 virtual void mha_fifo_thread_platform_t::release_mutex ( ) [pure virtual]

Calling thread releases the lock.

May only be called when lock is owned.

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.3 virtual void mha_fifo_thread_platform_t::wait_for_decrease ( ) [pure virtual]

Calling producer thread must own the lock.

Method releases lock, and waits for consumer thread to call decrease(). Then reaquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.4 virtual void mha_fifo_thread_platform_t::wait_for_increase ( ) [pure virtual]

Calling consumer thread must own the lock.

Method releases lock, and waits for producer thread to call increase(). Then reaquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.5 virtual void mha_fifo_thread_platform_t::increment ( ) [pure virtual]

To be called by producer thread after producing.

Producer thread needs to own the lock to call this method.

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.6 virtual void mha_fifo_thread_platform_t::decrement ( ) [pure virtual]

To be called by consumer thread after consuming.

Consumer thread needs to own the lock to call this method.

Implemented in **mha_fifo_posix_threads_t** (p. 451).

5.154.3.7 mha_fifo_thread_platform_t& mha_fifo_thread_platform_t::operator= (const mha_fifo_thread_platform_t &) [private]

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

- **mha_fifo.h**

5.155 mha_real_test_array_t Struct Reference

Public Attributes

- **mha_real_t r** [4]

5.155.1 Member Data Documentation

5.155.1.1 mha_real_t mha_real_test_array_t::r[4]

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

- **mha.hh**

5.156 mha_rt_fifo_element_t< T > Class Template Reference

Object wrapper for **mha_rt_fifo_t** (p. 462).

Public Member Functions

- **mha_rt_fifo_element_t** (T *__data__)
  Constructor.
- ~**mha_rt_fifo_element_t** ()

Public Attributes

- **mha_rt_fifo_element_t**< T > * **next**
  Pointer to next fifo element. NULL for the last (newest) fifo element.
- bool **abandonned**
  Indicates that this element will no longer be used and may be deleted.
- T * **data**
  Pointer to user data.

5.156.1 Detailed Description

template<**class T**>
class mha_rt_fifo_element_t< **T** >

Object wrapper for **mha_rt_fifo_t** (p. 462).

5.156.2 Constructor & Destructor Documentation

5.156.2.1 template<class T > mha_rt_fifo_element_t< T >::mha_rt_fifo_element_t (T * *data*) [inline]

Constructor.

This element assumes ownership of user data.

Parameters

data	User data. Has to be allocated on the heap with standard operator new, because it will be deleted in this element's destructor.

5.156.2.2 template<class T > mha_rt_fifo_element_t< T >::~mha_rt_fifo_element_t ( ) [inline]

5.156.3 Member Data Documentation

5.156.3.1 template<class T > mha_rt_fifo_element_t<T>* mha_rt_fifo_element_t< T >::next

Pointer to next fifo element. NULL for the last (newest) fifo element.

5.156.3.2 template<class T > bool mha_rt_fifo_element_t< T >::abandonned

Indicates that this element will no longer be used and may be deleted.

5.156.3.3 template<class T > T* mha_rt_fifo_element_t< T >::data

Pointer to user data.

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

- **mha_fifo.h**

5.157 mha_rt_fifo_t< T > Class Template Reference

Template class for thread safe, half real time safe fifo without explixit locks.

Public Member Functions

- **mha_rt_fifo_t** ()
  Construct empty fifo.
- ~**mha_rt_fifo_t** ()
  Destructor will delete all data currently in the fifo.
- T * **poll** ()
  Retrieve the latest element in the Fifo.
- T * **poll_1** ()
  Retrieve the next element in the Fifo, if there is one, and mark the previous element as abandonned.
- void **push** (T *data)
  Add element to the Fifo.

Private Member Functions

- void **remove_abandonned** ()
  Deletes abandonned elements.
- void **remove_all** ()
  Deletes all elements.

Private Attributes

- **mha_rt_fifo_element_t**< T > * **root**
  The first element in the fifo. Deleting elements starts here.
- **mha_rt_fifo_element_t**< T > * **current**
  *The element most recently returned by **poll*** (p. 463) *or **poll_1*** (p. 464)*.*

5.157.1 Detailed Description

template<class T>
class mha_rt_fifo_t< T >

Template class for thread safe, half real time safe fifo without explixit locks.

Reading from this fifo is realtime safe, writing to it is not. This fifo is designed for objects that were constructed on the heap. It assumes ownership of these objects and calls delete on them when they are no longer used. Objects remain inside the Fifo while being used by the reader.

A new fifo element is inserted by using **push** (p. 464). The push operation is not real time safe, it allocates and deallocates memory. The latest element is retrieved by calling **poll** (p. 463). This operation will skip fifo elements if more than one **push** (p. 464) has been occured since the last poll. To avoid skipping, call the **poll_1** (p. 464) operation instead.

5.157.2 Constructor & Destructor Documentation

5.157.2.1 template<class T > mha_rt_fifo_t< T >::mha_rt_fifo_t ( ) [inline]

Construct empty fifo.

5.157.2.2 template<class T > mha_rt_fifo_t< T >::~mha_rt_fifo_t ( ) [inline]

Destructor will delete all data currently in the fifo.

5.157.3 Member Function Documentation

5.157.3.1 template<class T > T* mha_rt_fifo_t< T >::poll ( ) [inline]

Retrieve the latest element in the Fifo.

Will skip fifo elements if more than one element has been added since last poll invocation. Will return the same element as on last call if no elements have been added in the mean time. Marks former elements as abandonned.

Returns

The latest element in this Fifo. Returns NULL if the Fifo is empty.

5.157.3.2 template<class T > T* mha_rt_fifo_t< T >::poll_1 ( ) [inline]

Retrieve the next element in the Fifo, if there is one, and mark the previous element as abandonned.

Else, if there is no newer element, returns the same element as on last **poll()** (p. 463) or **poll_1()** (p. 464) invocation.

Returns

The next element in this Fifo, if there is one, or the same as before. Returns NULL if the Fifo is empty.

5.157.3.3 template<class T > void mha_rt_fifo_t< T >::push (T * *data*) [inline]

Add element to the Fifo.

Deletes abandonned elements in the fifo.

Parameters

data	The new user data to place at the end of the fifo. After this invocation, the fifo is the owner of this object and will delete it when it is no longer used. data must have been allocated on the heap with standard operator new.

5.157.3.4 template<class T > void mha_rt_fifo_t< T >::remove_abandonned ( ) [inline], [private]

Deletes abandonned elements.

5.157.3.5 template<class T > void mha_rt_fifo_t< T >::remove_all ( ) [inline], [private]

Deletes all elements.

5.157.4 Member Data Documentation

5.157.4.1 template<class T > mha_rt_fifo_element_t<T>* mha_rt_fifo_t< T >::root [private]

The first element in the fifo. Deleting elements starts here.

5.157.4.2 template<class T > mha_rt_fifo_element_t<T>* mha_rt_fifo_t< T >::current [private]

The element most recently returned by **poll** (p. 463) or **poll_1** (p. 464).

Searching for new elements starts here.

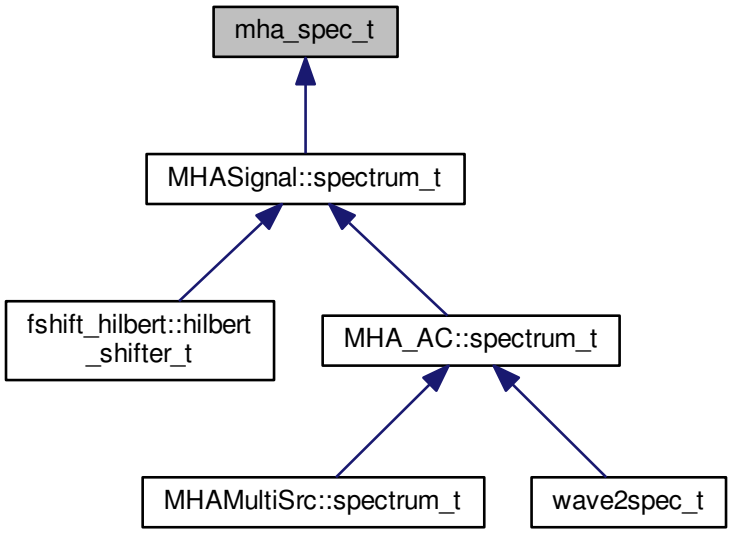
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.158 mha_spec_t Struct Reference

Spectrum signal structure.

Inheritance diagram for mha_spec_t:



Public Attributes

- **mha_complex_t * buf**
  signal buffer
- unsigned int **num_channels**
  number of channels
- unsigned int **num_frames**
  number of frames in each channel
- **mha_channel_info_t * channel_info**
  detailed channel description

5.158.1 Detailed Description

Spectrum signal structure.

This structure contains the short time fourier transform output of the windowed input signal. The member `num_frames` describes the number of frequency bins in each channel. For an even FFT length N, this is $N/2+1$. With odd FFT lengths, it is $(N+1)/2$. The imaginary part of the first bin is zero. For even FFT lengths, also the imaginary part at the Nyquist frequency is zero.

buf[k].re	*Re(0)*	*Re(1)*	*Re(2)*	*Re(3)*	*Re(4)*	···	*Re(n/2−1)*	*Re(n/2)*
buf[k].im		*Im(1)*	*Im(2)*	*Im(3)*	*Im(4)*	···	*Im(n/2−1)*	
k	*0*	*1*	*2*	*3*	*4*		*n/2−1*	*n/2*

Figure 4 Data order of FFT spectrum.

5.158.2 Member Data Documentation

5.158.2.1 mha_complex_t* mha_spec_t::buf

signal buffer

5.158.2.2 unsigned int mha_spec_t::num_channels

number of channels

5.158.2.3 unsigned int mha_spec_t::num_frames

number of frames in each channel

5.158.2.4 mha_channel_info_t* mha_spec_t::channel_info

detailed channel description

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

- **mha.hh**

5.159 mha_stash_environment_variable_t Class Reference

This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.

Public Member Functions

- **mha_stash_environment_variable_t** (const std::string &**variable_name**, const std↩ ::string &new_content)
- ∼**mha_stash_environment_variable_t** ()

Private Attributes

- const bool **existed_before**

  Flag indicates if the environment variable existed before constructor.
- const std::string **variable_name**

  Name of environment variable.
- const std::string **original_content**

  Content of environment variable before constructor executed.

5.159.1 Detailed Description

This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.

Can be used for testing functionality related to environment variables.

Todo Move to collection of unit-test support classes when we have one.

5.159.2 Constructor & Destructor Documentation

5.159.2.1 mha_stash_environment_variable_t::mha_stash_environment_variable_t (const std::string & *variable_name,* const std::string & *new_content*) [inline]

5.159.2.2 mha_stash_environment_variable_t::∼mha_stash_environment_variable_t ( ) [inline]

5.159.3 Member Data Documentation

5.159.3.1 const bool mha_stash_environment_variable_t::existed_before [private]

Flag indicates if the environment variable existed before constructor.

5.159.3.2 const std::string mha_stash_environment_variable_t::variable_name [private]

Name of environment variable.

5.159.3.3 const std::string mha_stash_environment_variable_t::original_content [private]

Content of environment variable before constructor executed.

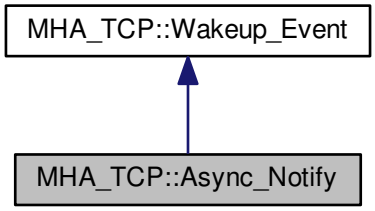
The documentation for this class was generated from the following file:

- **mha_os.h**

5.160 MHA_TCP::Async_Notify Class Reference

Portable Multiplexable cross-thread notification.

Inheritance diagram for MHA_TCP::Async_Notify:



Public Member Functions

- **Async_Notify** ()
- virtual void **reset** ()
- virtual void **set** ()
- virtual ~**Async_Notify** ()

Private Attributes

- int **pipe** [2]

Additional Inherited Members

5.160.1 Detailed Description

Portable Multiplexable cross-thread notification.

5.160.2 Constructor & Destructor Documentation

5.160.2.1 Async_Notify::Async_Notify ( )

5.160.2.2 Async_Notify::~Async_Notify ( ) [virtual]

5.160.3 Member Function Documentation

5.160.3.1 void Async_Notify::reset ( ) [virtual]

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 495).

5.160.3.2 void Async_Notify::set ( ) [virtual]

5.160.4 Member Data Documentation

5.160.4.1 int MHA_TCP::Async_Notify::pipe[2] [private]

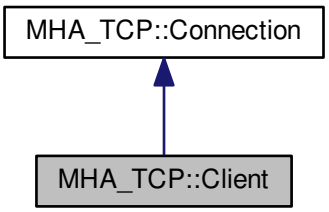
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.161 MHA_TCP::Client Class Reference

A portable class for a tcp client connections.

Inheritance diagram for MHA_TCP::Client:



Public Member Functions

- **Client** (const std::string &host, unsigned short port)

  Constructor connects to host, port via TCP.
- **Client** (const std::string &host, unsigned short port, **Timeout_Watcher** &timeout_↩ watcher)

  Constructor connects to host, port via TCP, using a timeout.

Additional Inherited Members

5.161.1 Detailed Description

A portable class for a tcp client connections.

5.161.2 Constructor & Destructor Documentation

5.161.2.1 Client::Client (const std::string & *host,* unsigned short *port*)

Constructor connects to host, port via TCP.

Parameters

host	The hostname of the TCP **Server** (p. 481).
port	The port or the TCP **Server** (p. 481).

5.161.2.2 Client::Client (const std::string & *host,* unsigned short *port,* Timeout_Watcher & *timeout_watcher*)

Constructor connects to host, port via TCP, using a timeout.

Parameters

host	The hostname of the TCP **Server** (p. 481).
port	The port or the TCP **Server** (p. 481).
timeout_watcher	an Event watcher that implements a timeout.

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

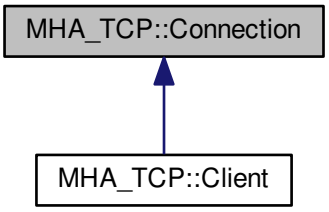
- **mha_tcp.hh**

- **mha_tcp.cpp**

5.162 MHA_TCP::Connection Class Reference

Connection (p. 471) handles Communication between client and server, is used on both sides.

Inheritance diagram for MHA_TCP::Connection:



Public Member Functions

- **Sockread_Event** ∗ **get_read_event** ()
- **Sockwrite_Event** ∗ **get_write_event** ()
- std::string **get_peer_address** ()
  Get peer's IP Address.
- unsigned short **get_peer_port** ()
  Get peer's TCP port.
- **SOCKET get_fd** () const
  Return the (protected) file descriptor of the connection.
- virtual ∼**Connection** ()
  Destructor closes the underlying file descriptor.
- bool **eof** ()
  Checks if the peer has closed the connection.
- bool **can_read_line** (char delim= '\n')
  Checks if a full line of text has arrived by now.
- bool **can_read_bytes** (unsigned howmany)
  Checks if the specified ammount of data can be read.
- std::string **read_line** (char delim= '\n')
  Reads a single line of data from the socket.
- std::string **read_bytes** (unsigned howmany)
  Reads the specified ammount of dat from the socket.
- void **try_write** (const std::string &data="")

Adds data to the internal "outgoing" buffer, and then tries to write as much data from that buffer to the socket as possible without blocking.

- void **write** (const std::string &data="")

  Adds data to the internal "outgoing" buffer, and then writes that that buffer to the socket, regardless of blocking.
- bool **needs_write** ()

  Checks if the internal "outgoing" buffer contains data.
- unsigned **buffered_incoming_bytes** () const

  Returns the number of bytes in the internal "incoming" buffer.
- unsigned **buffered_outgoing_bytes** () const

  Returns the number of bytes in the internal "outgoing" buffer.

Protected Member Functions

- **Connection** (**SOCKET** _fd)

  Create a connection instance from a socket filedescriptor.

Protected Attributes

- **SOCKET fd**

  The file descriptor of the TCP Socket.

Private Member Functions

- void **init_peer_data** ()

  determine peer address and port
- bool **can_sysread** ()

  Determine wether at least 1 byte can be read without blocking.
- bool **can_syswrite** ()

  Determine wether at least 1 byte can be written without blocking.
- std::string **sysread** (unsigned bytes)

  Call the system's read function and try to read bytes.
- std::string **syswrite** (const std::string &data)

  Call the system's write function and try to write all characters in the string data.

Private Attributes

- std::string **outbuf**
- std::string **inbuf**
- **Sockread_Event** * **read_event**
- **Sockwrite_Event** * **write_event**
- bool **closed**
- struct sockaddr_in **peer_addr**

5.162.1 Detailed Description

Connection (p. 471) handles Communication between client and server, is used on both sides.

5.162.2 Constructor & Destructor Documentation

5.162.2.1 MHA_TCP::Connection::Connection (SOCKET *_fd*) [protected]

Create a connection instance from a socket filedescriptor.

Parameters

↩ _↩ *fd*	The file descriptor of the TCP Socket. This file descriptor is closed again in the destructor.

Exceptions

MHA_Error *(p. 445)*	If the file descriptor is < 0.

5.162.2.2 Connection::∼Connection ( ) [virtual]

Destructor closes the underlying file descriptor.

5.162.3 Member Function Documentation

5.162.3.1 void MHA_TCP::Connection::init_peer_data ( ) [private]

determine peer address and port

5.162.3.2 bool Connection::can_sysread ( ) [private]

Determine wether at least 1 byte can be read without blocking.

5.162.3.3 bool Connection::can_syswrite ( ) [private]

Determine wether at least 1 byte can be written without blocking.

5.162.3.4 std::string Connection::sysread (unsigned *bytes*) [private]

Call the system's read function and try to read bytes.

This will block in a situation where can_sysread returns false.

Parameters

bytes	The desired number of characters.

Returns

The characters read from the socket. The result may have fewer characters than specified by bytes. If the result is an empty string, then the socket has been closed by the peer.

5.162.3.5 std::string Connection::syswrite (const std::string & *data*) [private]

Call the system's write function and try to write all characters in the string data.

May write fewer characters, but will at least write one character.

Parameters

data	A string of characters to write to the socket.

Returns

The rest of the characters that have not yet been written.

5.162.3.6 Sockread_Event * Connection::get_read_event ( )

5.162.3.7 Sockwrite_Event * Connection::get_write_event ( )

5.162.3.8 std::string Connection::get_peer_address ( )

Get peer's IP Address.

5.162.3.9 unsigned short Connection::get_peer_port ( )

Get peer's TCP port.

5.162.3.10 SOCKET MHA_TCP::Connection::get_fd ( ) const [inline]

Return the (protected) file descriptor of the connection.

Will be required for SSL.

5.162.3.11 bool Connection::eof ( )

Checks if the peer has closed the connection.

As a side effect, this method fills the internal "incoming" buffer if it was empty and the socket is readable and not eof.

5.162.3.12 bool Connection::can_read_line (char *delim* = '\n')

Checks if a full line of text has arrived by now.

This method reads data from the socket into the internal "incoming" buffer if it can be done without blocking.

Parameters

delim	The line delimiter.

Returns

true if at least one full line of text is present in the internal buffer after this method call, false otherwise.

5.162.3.13 bool Connection::can_read_bytes (unsigned *howmany*)

Checks if the specified ammount of data can be read.

This method reads data from the socket into an internal "incoming" buffer if it can be done without blocking.

Parameters

howmany	The number of bytes that the caller wants to have checked.

Returns

true if at least the specified ammount of data is present in the internal buffer after this method call, false otherwise

5.162.3.14 std::string Connection::read_line (char *delim* = '\n')

Reads a single line of data from the socket.

Blocks if necessary.

Parameters

delim	The line delimiter.

Returns

The string of characters in this line, including the trailing delimiter. The delimiter may be missing if the last line before EOF does not have a delimiter.

5.162.3.15 std::string Connection::read_bytes (unsigned *howmany*)

Reads the specified ammount of dat from the socket.

Blocks if necessary.

Parameters

howmany	The number of bytes to read.

Returns

The string of characters read. The string may be shorter if EOF is encountered.

5.162.3.16 void Connection::try_write (const std::string & *data* = " ")

Adds data to the internal "outgoing" buffer, and then tries to write as much data from that buffer to the socket as possible without blocking.

Parameters

data	data to send over the socket.

5.162.3.17 void Connection::write (const std::string & *data* = " ")

Adds data to the internal "outgoing" buffer, and then writes that that buffer to the socket, regardless of blocking.

Parameters

data	data to send over the socket.

5.162.3.18 bool Connection::needs_write ( )

Checks if the internal "outgoing" buffer contains data.

5.162.3.19 unsigned Connection::buffered_incoming_bytes ( ) const

Returns the number of bytes in the internal "incoming" buffer.

5.162.3.20 unsigned Connection::buffered_outgoing_bytes ( ) const

Returns the number of bytes in the internal "outgoing" buffer.

5.162.4 Member Data Documentation

5.162.4.1 std::string MHA_TCP::Connection::outbuf [private]

5.162.4.2 std::string MHA_TCP::Connection::inbuf [private]

5.162.4.3 Sockread_Event* MHA_TCP::Connection::read_event [private]

5.162.4.4 Sockwrite_Event* MHA_TCP::Connection::write_event [private]

5.162.4.5 bool MHA_TCP::Connection::closed [private]

5.162.4.6 struct sockaddr_in MHA_TCP::Connection::peer_addr [private]

5.162.4.7 SOCKET MHA_TCP::Connection::fd [protected]

The file descriptor of the TCP Socket.

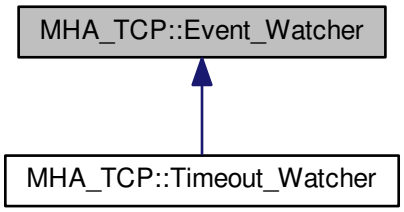
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.163 MHA_TCP::Event_Watcher Class Reference

OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.

Inheritance diagram for MHA_TCP::Event_Watcher:



Public Types

- typedef **std::set**< **Wakeup_Event** * > **Events**
- typedef **std::set**< **Wakeup_Event** * >::**iterator iterator**

Public Member Functions

- void **observe** (**Wakeup_Event** *event)
  Add an event to this observer.
- void **ignore** (**Wakeup_Event** *event)
  Remove an event from this observer.
- **std::set**< **Wakeup_Event** * > **wait** ()
  \ Wait for some event to occur.
- virtual ~**Event_Watcher** ()

Private Attributes

- **std::set**< **Wakeup_Event** * > **events**
  The list of events to watch.

5.163.1 Detailed Description

OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.

5.163.2 Member Typedef Documentation

5.163.2.1 typedef std::set<Wakeup_Event*> MHA_TCP::Event_Watcher::Events

5.163.2.2 typedef std::set<Wakeup_Event*>::iterator MHA_TCP::Event_Watcher::iterator

5.163.3 Constructor & Destructor Documentation

5.163.3.1 Event_Watcher::~Event_Watcher ( ) [virtual]

5.163.4 Member Function Documentation

5.163.4.1 void Event_Watcher::observe (Wakeup_Event * *event*)

Add an event to this observer.

5.163.4.2 void Event_Watcher::ignore (Wakeup_Event * *event*)

Remove an event from this observer.

5.163.4.3 std::set< Wakeup_Event * > Event_Watcher::wait ( )

\ Wait for some event to occur.

Return all events that are ready

5.163.5 Member Data Documentation

5.163.5.1 std::set<Wakeup_Event*> MHA_TCP::Event_Watcher::events [private]

The list of events to watch.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.164 MHA_TCP::OS_EVENT_TYPE Struct Reference

Public Types

Public Attributes

- enum MHA_TCP::OS_EVENT_TYPE:: { ... } **mode**
- union {
  int **fd**
  double **timeout**
  };

5.164.1 Member Enumeration Documentation

5.164.1.1 anonymous enum

Enumerator

R

W

X

T

5.164.2 Member Data Documentation

5.164.2.1 enum { ... } MHA_TCP::OS_EVENT_TYPE::mode

5.164.2.2 int MHA_TCP::OS_EVENT_TYPE::fd

5.164.2.3 double MHA_TCP::OS_EVENT_TYPE::timeout

5.164.2.4 union { ... }

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

- **mha_tcp.hh**

5.165 MHA_TCP::Server Class Reference

Public Member Functions

- **Server** (unsigned short **port**=0, const std::string &**iface**="0.0.0.0")
  Create a TCP server socket.
- **Server** (const std::string &**iface**, unsigned short **port**=0)
  Create a TCP server socket.
- ~**Server** ()
  Close the TCP server socket.
- std::string **get_interface** () const
  Get the name given in the constructor for the network interface.
- unsigned short **get_port** () const
  Get the port that the TCP server socket currently listens to.
- **Sockaccept_Event** * **get_accept_event** ()
  *Produces an event that can be observed by an **Event_Watcher** (p. 478).*
- **Connection** * **accept** ()
  Accept an incoming connection.
- **Connection** * **try_accept** ()
  Accept an incoming connection if it can be done without blocking.

Private Member Functions

- void **initialize** (const std::string &**iface**, unsigned short **port**)

Private Attributes

- sockaddr_in **sock_addr**
- **SOCKET serversocket**
- std::string **iface**
- unsigned short **port**
- **Sockaccept_Event** * **accept_event**

5.165.1 Constructor & Destructor Documentation

5.165.1.1 Server::Server (unsigned short *port* = 0, const std::string & *iface* = "0.0.0.0")

Create a TCP server socket.

Parameters

port	The TCP port to listen to.
iface	The network interface to bind to.

5.165.1.2 Server::Server (const std::string & *iface*, unsigned short *port* = 0)

Create a TCP server socket.

Parameters

port	The TCP port to listen to.
iface	The network interface to bind to.

5.165.1.3 Server::~Server ( )

Close the TCP server socket.

5.165.2 Member Function Documentation

5.165.2.1 void Server::initialize (const std::string & *iface*, unsigned short *port*) [private]

5.165.2.2 std::string Server::get_interface ( ) const

Get the name given in the constructor for the network interface.

5.165.2.3 unsigned short Server::get_port ( ) const

Get the port that the TCP server socket currently listens to.

5.165.2.4 Sockaccept_Event * Server::get_accept_event ( )

Produces an event that can be observed by an **Event_Watcher** (p. 478).

This event signals incoming connections that can be accepted.

5.165.2.5 Connection * Server::accept ( )

Accept an incoming connection.

blocks if necessary.

Returns

The new TCP connection. The connection has to be deleted by the caller.

5.165.2.6 Connection * Server::try_accept ( )

Accept an incoming connection if it can be done without blocking.

Returns

The new TCP connection or 0 if there is no immediate connection. The connection has to be deleted by the caller.

5.165.3 Member Data Documentation

5.165.3.1 sockaddr_in MHA_TCP::Server::sock_addr [private]

5.165.3.2 SOCKET MHA_TCP::Server::serversocket [private]

5.165.3.3 std::string MHA_TCP::Server::iface [private]

5.165.3.4 unsigned short MHA_TCP::Server::port [private]

5.165.3.5 Sockaccept_Event* MHA_TCP::Server::accept_event [private]

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.166 MHA_TCP::sock_init_t Class Reference

Public Member Functions

- **sock_init_t** ()

5.166.1 Constructor & Destructor Documentation

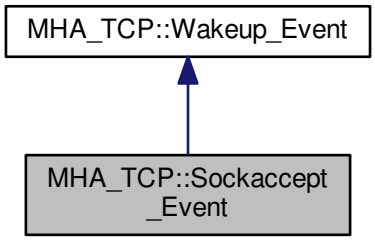
5.166.1.1 MHA_TCP::sock_init_t::sock_init_t ( ) [inline]

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

- **mha_tcp.cpp**

5.167 MHA_TCP::Sockaccept_Event Class Reference

Inheritance diagram for MHA_TCP::Sockaccept_Event:



Public Member Functions

- **Sockaccept_Event** (**SOCKET**)

Additional Inherited Members

5.167.1 Constructor & Destructor Documentation

5.167.1.1 MHA_TCP::Sockaccept_Event::Sockaccept_Event (SOCKET *s*)

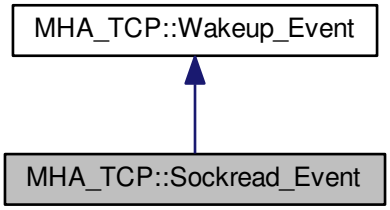
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.168 MHA_TCP::Sockread_Event Class Reference

Watch socket for incoming data.

Inheritance diagram for MHA_TCP::Sockread_Event:



Public Member Functions

- **Sockread_Event** (**SOCKET** s)
  Set socket to watch for.

Additional Inherited Members

5.168.1 Detailed Description

Watch socket for incoming data.

5.168.2 Constructor & Destructor Documentation

5.168.2.1 MHA_TCP::Sockread_Event::Sockread_Event (SOCKET *s*)

Set socket to watch for.

Parameters

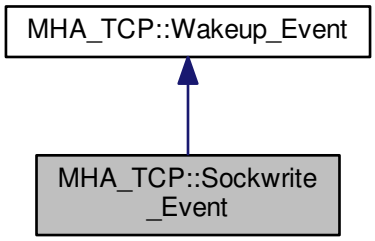
s	The socket to observe incoming data on.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.169 MHA_TCP::Sockwrite_Event Class Reference

Inheritance diagram for MHA_TCP::Sockwrite_Event:



Public Member Functions

- **Sockwrite_Event** (**SOCKET** s)

Additional Inherited Members

5.169.1 Constructor & Destructor Documentation

5.169.1.1 MHA_TCP::Sockwrite_Event::Sockwrite_Event (SOCKET *s*)

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.170 MHA_TCP::Thread Class Reference

A very simple class for portable threads.

Public Types

- typedef void ∗(∗ **thr_f**) (void ∗)

  The thread function signature to use with this class.

Public Member Functions

- **Thread** (**thr_f** func, void ∗**arg**=0)

  Constructor starts a new thread.
- virtual ∼**Thread** ()

  *The destructor should only be called when the **Thread** (p. 486) is finished.*
- virtual void **run** ()

  *The internal method that delegated the new thread to the registered **Thread** (p. 486) function.*

Public Attributes

- **Async_Notify thread_finish_event**

  Event will be triggered when the thread exits.
- enum MHA_TCP::Thread:: { ... } **state**

  The current state of the thread.
- **thr_f thread_func**

  The thread function that the client has registered.
- void ∗ **thread_arg**

  The argument that the client wants to be handed through to the thread function.
- **MHA_Error** ∗ **error**

  *The **MHA_Error** (p. 445) that caused the thread to abort, if any.*

Protected Member Functions

- **Thread** ()

  Default constructor may only be used by derived classes that want to start the thread themselves.

Protected Attributes

- void ∗ **arg**

  The argument for the client's thread function.
- void ∗ **return_value**

  The return value from the client's thread function is stored here When that function returns.

Private Attributes

- pthread_t **thread_handle**
  The posix thread handle.
- pthread_attr_t **thread_attr**
  The posix thread attribute structure.

5.170.1 Detailed Description

A very simple class for portable threads.

5.170.2 Member Typedef Documentation

5.170.2.1 typedef void*(* MHA_TCP::Thread::thr_f) (void *)

The thread function signature to use with this class.

Derive from this class and call protected standard constructor to start threads differently.

5.170.3 Member Enumeration Documentation

5.170.3.1 anonymous enum

The current state of the thread.

Enumerator

PREPARED
RUNNING
FINISHED

5.170.4 Constructor & Destructor Documentation

5.170.4.1 MHA_TCP::Thread::Thread ( ) [protected]

Default constructor may only be used by derived classes that want to start the thread themselves.

5.170.4.2 Thread::Thread (Thread::thr_f *func,* void * *arg* = 0)

Constructor starts a new thread.

Parameters

func	The function to be executed by the thread.
arg	The argument given to pass to the thread function.

5.170.4.3 Thread::~Thread ( ) [virtual]

The destructor should only be called when the **Thread** (p. 486) is finished.

There is preliminary support for forceful thread cancellation in the destructor, but probably not very robust or portable..

5.170.5 Member Function Documentation

5.170.5.1 void Thread::run ( ) [virtual]

The internal method that delegated the new thread to the registered **Thread** (p. 486) function.

5.170.6 Member Data Documentation

5.170.6.1 pthread_t MHA_TCP::Thread::thread_handle [private]

The posix thread handle.

5.170.6.2 pthread_attr_t MHA_TCP::Thread::thread_attr [private]

The posix thread attribute structure.

Required for starting a thread in detached state. Detachment is required to eliminate the need for joining this thread.

5.170.6.3 void* MHA_TCP::Thread::arg [protected]

The argument for the client's thread function.

5.170.6.4 void* MHA_TCP::Thread::return_value [protected]

The return value from the client's thread function is stored here When that function returns.

5.170.6.5 Async_Notify MHA_TCP::Thread::thread_finish_event

Event will be triggered when the thread exits.

5.170.6.6 enum { ... } MHA_TCP::Thread::state

The current state of the thread.

5.170.6.7 thr_f MHA_TCP::Thread::thread_func

The thread function that the client has registered.

5.170.6.8 void* MHA_TCP::Thread::thread_arg

The argument that the client wants to be handed through to the thread function.

5.170.6.9 MHA_Error* MHA_TCP::Thread::error

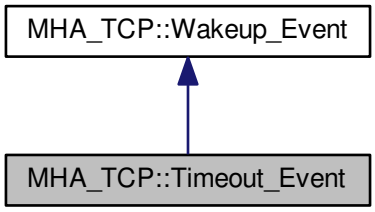
The **MHA_Error** (p. 445) that caused the thread to abort, if any.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.171 MHA_TCP::Timeout_Event Class Reference

Inheritance diagram for MHA_TCP::Timeout_Event:



Public Member Functions

- **Timeout_Event** (double interval)
- virtual **OS_EVENT_TYPE get_os_event** ()

Private Attributes

- double **end_time**

Additional Inherited Members

5.171.1 Constructor & Destructor Documentation

5.171.1.1 Timeout_Event::Timeout_Event (double *interval*)

5.171.2 Member Function Documentation

5.171.2.1 OS_EVENT_TYPE Timeout_Event::get_os_event ( ) [virtual]

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 494).

5.171.3 Member Data Documentation

5.171.3.1 double MHA_TCP::Timeout_Event::end_time [private]

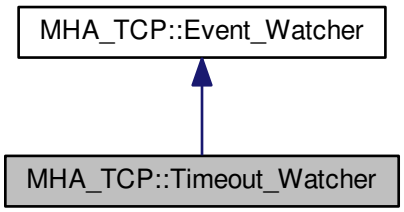
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.172 MHA_TCP::Timeout_Watcher Class Reference

OS-independent event watcher with internal fixed-end-time timeout.

Inheritance diagram for MHA_TCP::Timeout_Watcher:



Public Member Functions

- **Timeout_Watcher** (double interval)
- virtual ~**Timeout_Watcher** ()

Private Attributes

- **Timeout_Event timeout**

Additional Inherited Members

5.172.1 Detailed Description

OS-independent event watcher with internal fixed-end-time timeout.

5.172.2 Constructor & Destructor Documentation

5.172.2.1 Timeout_Watcher::Timeout_Watcher (double *interval*) [explicit]

5.172.2.2 Timeout_Watcher::~Timeout_Watcher ( ) [virtual]

5.172.3 Member Data Documentation

5.172.3.1 Timeout_Event MHA_TCP::Timeout_Watcher::timeout [private]

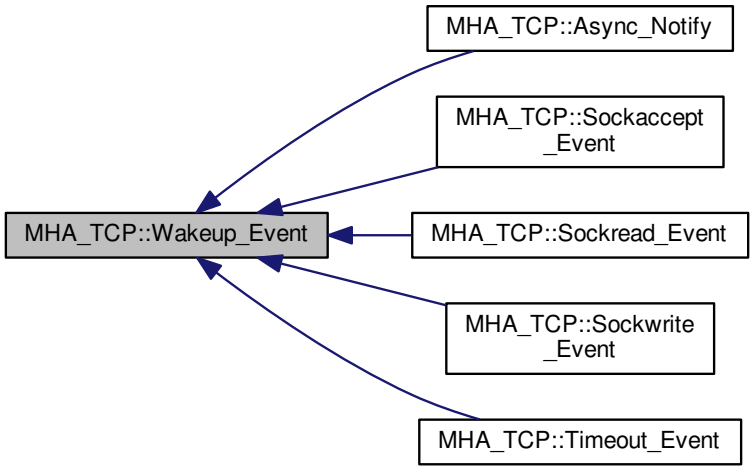
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.173 MHA_TCP::Wakeup_Event Class Reference

A base class for asynchronous wakeup events.

Inheritance diagram for MHA_TCP::Wakeup_Event:



Public Member Functions

- **Wakeup_Event** ()
  Event Constructor.
- virtual void **observed_by** (**Event_Watcher** ∗observer)
  *Called by the **Event_Watcher*** (p. 478) *when this event is added to its list of observed events.*
- virtual void **ignored_by** (**Event_Watcher** ∗observer)
  *Called by the **Event_Watcher*** (p. 478) *when this event is removed from its list of observed events.*
- virtual ∼**Wakeup_Event** ()
  Destructor deregisters from observers.
- virtual **OS_EVENT_TYPE get_os_event** ()
  Get necessary information for the Event Watcher.
- virtual void **reset** ()
  For pure notification events, reset the "signalled" status.
- virtual bool **status** ()
  Query wether the event is in signalled state now.

Protected Attributes

- **OS_EVENT_TYPE os_event**
- bool **os_event_valid**

Private Attributes

- **std::set**< class **Event_Watcher** * > **observers**

  *A list of all **Event_Watcher** (p. 478) instances that this **Wakeup_Event** (p. 493) is observed by (stored here for proper deregistering).*

5.173.1 Detailed Description

A base class for asynchronous wakeup events.

5.173.2 Constructor & Destructor Documentation

5.173.2.1 Wakeup_Event::Wakeup_Event ( )

Event Constructor.

The new event has invalid state.

5.173.2.2 Wakeup_Event::~Wakeup_Event ( ) [virtual]

Destructor deregisters from observers.

5.173.3 Member Function Documentation

5.173.3.1 void Wakeup_Event::observed_by (Event_Watcher * *observer*) [virtual]

Called by the **Event_Watcher** (p. 478) when this event is added to its list of observed events.

5.173.3.2 void Wakeup_Event::ignored_by (Event_Watcher * *observer*) [virtual]

Called by the **Event_Watcher** (p. 478) when this event is removed from its list of observed events.

5.173.3.3 OS_EVENT_TYPE Wakeup_Event::get_os_event ( ) [virtual]

Get necessary information for the Event Watcher.

Reimplemented in **MHA_TCP::Timeout_Event** (p. 491).

5.173.3.4 void Wakeup_Event::reset ( ) [virtual]

For pure notification events, reset the "signalled" status.

Reimplemented in **MHA_TCP::Async_Notify** (p. 469).

5.173.3.5 bool Wakeup_Event::status ( ) [virtual]

Query wether the event is in signalled state now.

5.173.4 Member Data Documentation

5.173.4.1 std::set<class Event_Watcher *> MHA_TCP::Wakeup_Event::observers [private]

A list of all **Event_Watcher** (p. 478) instances that this **Wakeup_Event** (p. 493) is observed by (stored here for proper deregistering).

5.173.4.2 OS_EVENT_TYPE MHA_TCP::Wakeup_Event::os_event [protected]

5.173.4.3 bool MHA_TCP::Wakeup_Event::os_event_valid [protected]

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.174 mha_tictoc_t Struct Reference

Public Attributes

- struct timeval **tv1**
- struct timeval **tv2**
- struct timezone **tz**
- float **t**

5.174.1 Member Data Documentation

5.174.1.1 struct timeval mha_tictoc_t::tv1

5.174.1.2 struct timeval mha_tictoc_t::tv2

5.174.1.3 struct timezone mha_tictoc_t::tz

5.174.1.4 float mha_tictoc_t::t

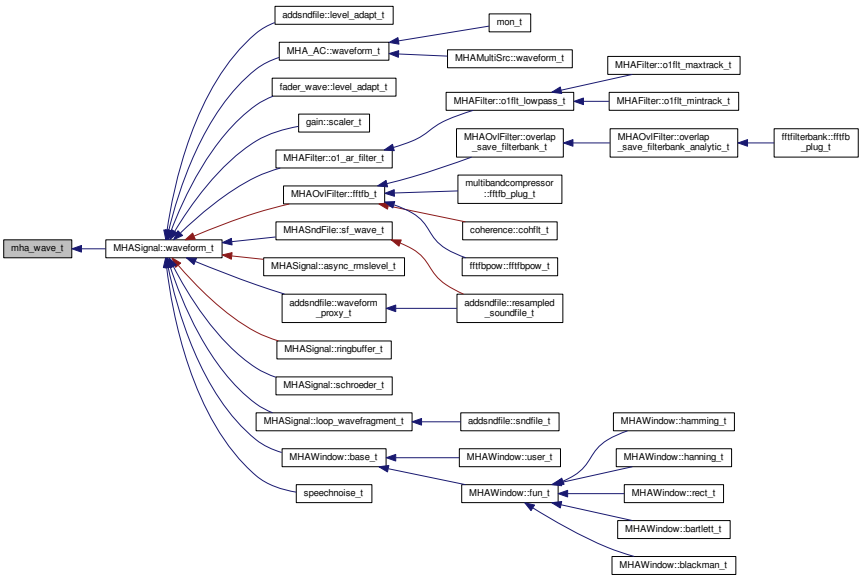
The documentation for this struct was generated from the following file:

- **mha_profiling.h**

5.175 mha_wave_t Struct Reference

Waveform signal structure.

Inheritance diagram for mha_wave_t:



Public Attributes

- **mha_real_t ∗ buf**
  signal buffer
- unsigned int **num_channels**
  number of channels
- unsigned int **num_frames**
  number of frames in each channel
- **mha_channel_info_t ∗ channel_info**
  detailed channel description

5.175.1 Detailed Description

Waveform signal structure.

This structure contains one fragment of a waveform signal. The member num_frames describes the number of audio samples in each audio channel.

The field channel_info must be an array of num_channels entries or NULL.

5.175.2 Member Data Documentation

5.175.2.1 mha_real_t* mha_wave_t::buf

signal buffer

5.175.2.2 unsigned int mha_wave_t::num_channels

number of channels

5.175.2.3 unsigned int mha_wave_t::num_frames

number of frames in each channel

5.175.2.4 mha_channel_info_t* mha_wave_t::channel_info

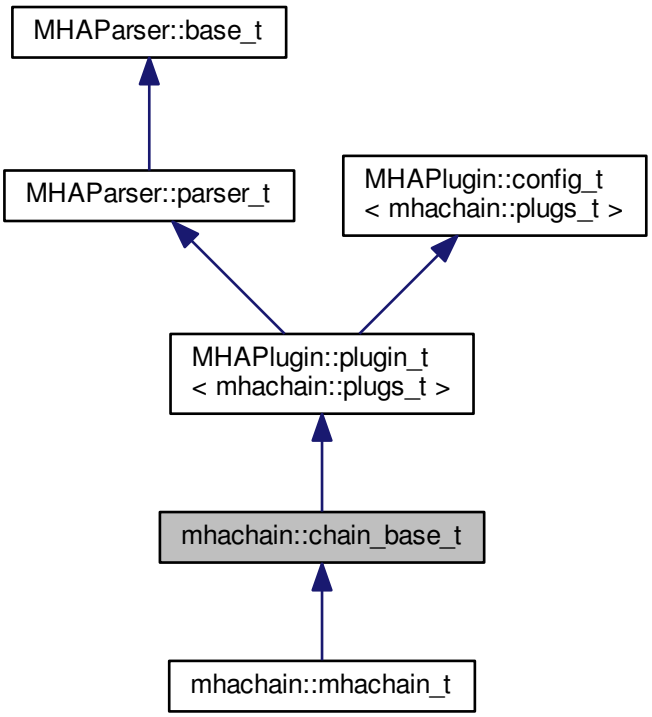
detailed channel description

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

- **mha.hh**

5.176 mhachain::chain_base_t Class Reference

Inheritance diagram for mhachain::chain_base_t:



Public Member Functions

- **chain_base_t** (**algo_comm_t**, const std::string &, const std::string &)
- void **process** (**mha_wave_t** ∗, **mha_wave_t** ∗∗)
- void **process** (**mha_spec_t** ∗, **mha_wave_t** ∗∗)
- void **process** (**mha_wave_t** ∗, **mha_spec_t** ∗∗)
- void **process** (**mha_spec_t** ∗, **mha_spec_t** ∗∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Protected Attributes

- **MHAParser::bool_t bprofiling**
- **MHAParser::vstring_t algos**

Private Member Functions

- void **update** ()

Private Attributes

- std::vector< std::string > **old_algos**
- **MHAEvents::patchbay_t**< **mhachain::chain_base_t** > **patchbay**
- **mhaconfig_t cfin**
- **mhaconfig_t cfout**
- bool **b_prepared**
- std::string **chain**

Additional Inherited Members

5.176.1 Constructor & Destructor Documentation

5.176.1.1 mhachain::chain_base_t::chain_base_t (algo_comm_t *iac,* const std::string & *ichain,* const std::string & *ialgo*)

5.176.2 Member Function Documentation

5.176.2.1 void mhachain::chain_base_t::process (mha_wave_t ∗ *sin,* mha_wave_t ∗∗ *sout*)

5.176.2.2 void mhachain::chain_base_t::process (mha_spec_t * *sin*, mha_wave_t ** *sout*)

5.176.2.3 void mhachain::chain_base_t::process (mha_wave_t * *sin*, mha_spec_t ** *sout*)

5.176.2.4 void mhachain::chain_base_t::process (mha_spec_t * *sin*, mha_spec_t ** *sout*)

5.176.2.5 void mhachain::chain_base_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< mhachain::plugs_t >** (p. 730).

5.176.2.6 void mhachain::chain_base_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< mhachain::plugs_t >** (p. 731).

5.176.2.7 void mhachain::chain_base_t::update ( ) [private]

5.176.3 Member Data Documentation

5.176.3.1 **MHAParser::bool_t** mhachain::chain_base_t::bprofiling [protected]

5.176.3.2 **MHAParser::vstring_t** mhachain::chain_base_t::algos [protected]

5.176.3.3 std::vector<std::string> mhachain::chain_base_t::old_algos [private]

5.176.3.4 **MHAEvents::patchbay_t< mhachain::chain_base_t >** mhachain::chain_base_t::patchbay [private]

5.176.3.5 **mhaconfig_t** mhachain::chain_base_t::cfin [private]

5.176.3.6 **mhaconfig_t** mhachain::chain_base_t::cfout [private]

5.176.3.7 bool mhachain::chain_base_t::b_prepared [private]

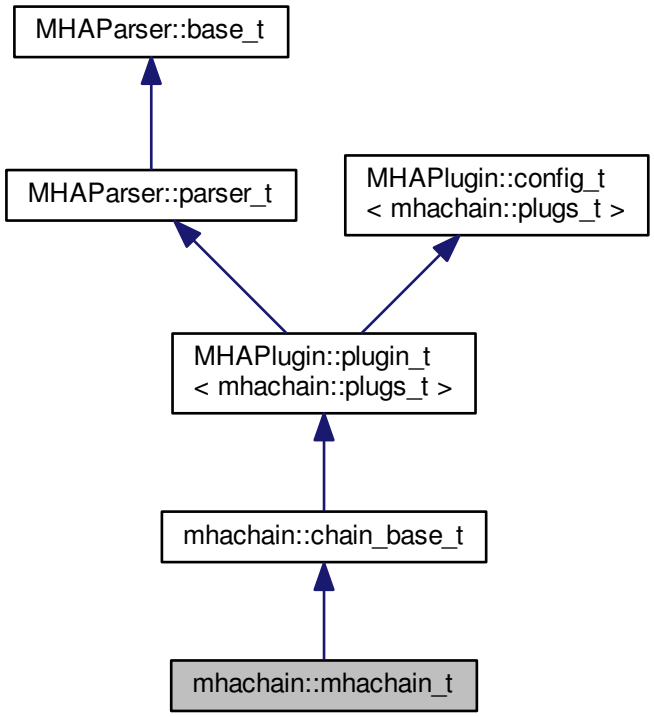
5.176.3.8 std::string mhachain::chain_base_t::chain [private]

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

- **mha_generic_chain.h**
- **mha_generic_chain.cpp**

5.177 mhachain::mhachain_t Class Reference

Inheritance diagram for mhachain::mhachain_t:



Public Member Functions

- **mhachain_t** (**algo_comm_t** iac, const std::string &ichain, const std::string &ialgo)

Additional Inherited Members

5.177.1 Constructor & Destructor Documentation

5.177.1.1 mhachain::mhachain_t::mhachain_t (**algo_comm_t** *iac,* const std::string & *ichain,* const std::string & *ialgo*)

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

- **mhachain.cpp**

5.178 mhachain::plugs_t Class Reference

Public Member Functions

- **plugs_t** (std::vector< std::string > **algos**, **mhaconfig_t** cfin, **mhaconfig_t** cfout, bool do_prepare, **MHAParser::parser_t** &p, **algo_comm_t** iac, std::string ichain, bool use_↩ profiling)
- ~**plugs_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** (**mha_wave_t** *, **mha_spec_t** *, **mha_wave_t** **, **mha_spec_t** **)
- bool **prepared** () const

Private Member Functions

- void **alloc_plugs** (std::vector< std::string > **algos**)
- void **cleanup_plugs** ()
- void **update_proc_load** ()

Private Attributes

- bool **b_prepared**
- std::vector< **PluginLoader::mhapluginloader_t** * > **algos**
- **MHAParser::parser_t** & **parser**
- **algo_comm_t ac**
- std::string **chain**
- **MHAParser::parser_t profiling**
- **MHAParser::vstring_mon_t prof_algos**
- **MHAParser::vfloat_mon_t prof_init**
- **MHAParser::vfloat_mon_t prof_prepare**
- **MHAParser::vfloat_mon_t prof_release**
- **MHAParser::vfloat_mon_t prof_process**
- **MHAParser::float_mon_t prof_process_tt**
- **MHAParser::vfloat_mon_t prof_process_load**
- unsigned int **proc_cnt**
- **mhaconfig_t prof_cfg**
- **MHAEvents::connector_t**< **mhachain::plugs_t** > **prof_load_con**
- **MHAEvents::connector_t**< **mhachain::plugs_t** > **prof_tt_con**
- bool **b_use_profiling**
- **mha_platform_tictoc_t tictoc**

5.178.1 Constructor & Destructor Documentation

5.178.1.1 mhachain::plugs_t::plugs_t (
std::vector< std::string > *algos,*
mhaconfig_t *cfin,*
mhaconfig_t *cfout,*
bool *do_prepare,*
MHAParser::parser_t & *p,*
algo_comm_t *iac,*
std::string *ichain,*
bool *use_profiling*)

5.178.1.2 mhachain::plugs_t::~plugs_t ( )

5.178.2 Member Function Documentation

5.178.2.1 void mhachain::plugs_t::prepare (
mhaconfig_t & *tf*)

5.178.2.2 void mhachain::plugs_t::release (
void)

5.178.2.3 void mhachain::plugs_t::process (
mha_wave_t * *win,*
mha_spec_t * *sin,*
mha_wave_t ** *wout,*
mha_spec_t ** *sout*)

5.178.2.4 bool mhachain::plugs_t::prepared ( ) const [inline]

5.178.2.5 void mhachain::plugs_t::alloc_plugs (
std::vector< std::string > *algos*) [private]

5.178.2.6 void mhachain::plugs_t::cleanup_plugs ( ) [private]

5.178.2.7 void mhachain::plugs_t::update_proc_load ( ) [private]

5.178.3 Member Data Documentation

5.178.3.1 bool mhachain::plugs_t::b_prepared [private]

5.178.3.2 std::vector< **PluginLoader::mhapluginloader_t*** > mhachain::plugs_t::algos [private]

5.178.3.3 **MHAParser::parser_t**& mhachain::plugs_t::parser [private]

5.178.3.4 algo_comm_t mhachain::plugs_t::ac [private]

5.178.3.5 std::string mhachain::plugs_t::chain [private]

5.178.3.6 MHAParser::parser_t mhachain::plugs_t::profiling [private]

5.178.3.7 MHAParser::vstring_mon_t mhachain::plugs_t::prof_algos [private]

5.178.3.8 MHAParser::vfloat_mon_t mhachain::plugs_t::prof_init [private]

5.178.3.9 MHAParser::vfloat_mon_t mhachain::plugs_t::prof_prepare [private]

5.178.3.10 MHAParser::vfloat_mon_t mhachain::plugs_t::prof_release [private]

5.178.3.11 MHAParser::vfloat_mon_t mhachain::plugs_t::prof_process [private]

5.178.3.12 MHAParser::float_mon_t mhachain::plugs_t::prof_process_tt [private]

5.178.3.13 MHAParser::vfloat_mon_t mhachain::plugs_t::prof_process_load [private]

5.178.3.14 unsigned int mhachain::plugs_t::proc_cnt [private]

5.178.3.15 mhaconfig_t mhachain::plugs_t::prof_cfg [private]

5.178.3.16 MHAEvents::connector_t<mhachain::plugs_t> mhachain::plugs_t::prof_load_con [private]

5.178.3.17 MHAEvents::connector_t<mhachain::plugs_t> mhachain::plugs_t::prof_tt_con [private]

5.178.3.18 bool mhachain::plugs_t::b_use_profiling [private]

5.178.3.19 mha_platform_tictoc_t mhachain::plugs_t::tictoc [private]

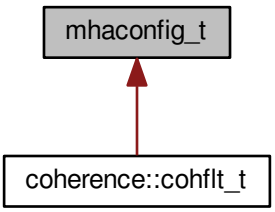
The documentation for this class was generated from the following files:

- **mha_generic_chain.h**
- **mha_generic_chain.cpp**

5.179 mhaconfig_t Struct Reference

MHA prepare configuration structure.

Inheritance diagram for mhaconfig_t:



Public Attributes

- unsigned int **channels**
  Number of audio channels.
- unsigned int **domain**
  Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)
- unsigned int **fragsize**
  Fragment size of waveform data.
- unsigned int **wndlen**
  Window length of spectral data.
- unsigned int **fftlen**
  FFT length of spectral data.
- **mha_real_t srate**
  Sampling rate in Hz.

5.179.1 Detailed Description

MHA prepare configuration structure.

This structure contains information about channel number and domain for input and output signals of a openMHA Plugin. Each plugin can change any of these parameters, e.g. by resampling of the signal. The only limitation is that the callback frequency is fixed (except for the plugins `db` and `dbasync`).

Todo Add information on number of bands and on center frequencies, or replace by **mha_audio_descriptor_t** (p. 427).

5.179.2 Member Data Documentation

5.179.2.1 unsigned int mhaconfig_t::channels

Number of audio channels.

5.179.2.2 unsigned int mhaconfig_t::domain

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.179.2.3 unsigned int mhaconfig_t::fragsize

Fragment size of waveform data.

5.179.2.4 unsigned int mhaconfig_t::wndlen

Window length of spectral data.

5.179.2.5 unsigned int mhaconfig_t::fftlen

FFT length of spectral data.

5.179.2.6 mha_real_t mhaconfig_t::srate

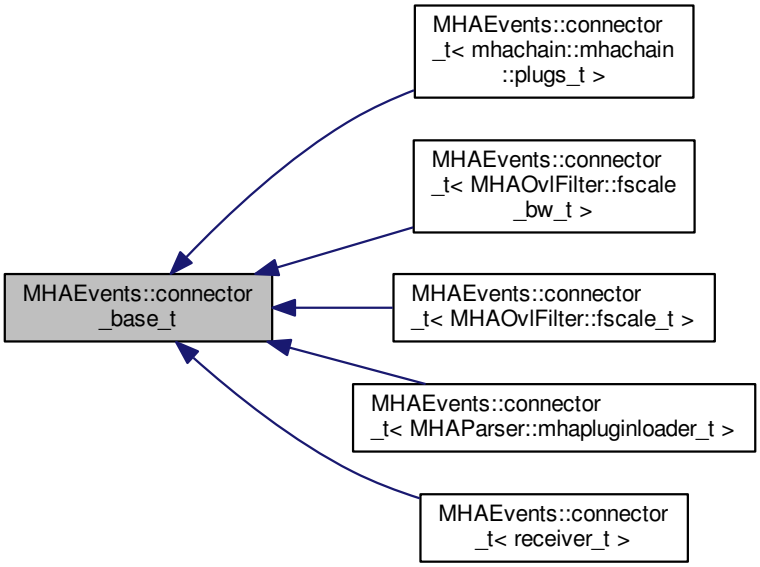
Sampling rate in Hz.

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

- **mha.hh**

5.180 MHAEvents::connector_base_t Class Reference

Inheritance diagram for MHAEvents::connector_base_t:



Public Member Functions

- **connector_base_t** ()
- virtual ~**connector_base_t** ()
- virtual void **emit_event** ()
- virtual void **emit_event** (const std::string &)
- virtual void **emit_event** (const std::string &, unsigned int, unsigned int)
- void **emitter_die** ()

Protected Attributes

- bool **emitter_is_alive**

5.180.1 Constructor & Destructor Documentation

5.180.1.1 MHAEvents::connector_base_t::connector_base_t ( )

5.180.1.2 MHAEvents::connector_base_t::~connector_base_t ( ) [virtual]

5.180.2 Member Function Documentation

5.180.2.1 void MHAEvents::connector_base_t::emit_event ( ) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 509), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 509), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 509), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 509), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 509).

5.180.2.2 void MHAEvents::connector_base_t::emit_event (const std::string &) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 509), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 509), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 509), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 509), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 509).

5.180.2.3 void MHAEvents::connector_base_t::emit_event (const std::string &, unsigned int, unsigned int) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 509), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 509), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 509), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 509), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 509).

5.180.2.4 void MHAEvents::connector_base_t::emitter_die ( )

5.180.3 Member Data Documentation

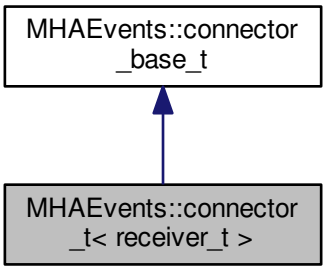
5.180.3.1 bool MHAEvents::connector_base_t::emitter_is_alive [protected]

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

- **mha_event_emitter.h**
- **mha_events.cpp**

5.181 MHAEvents::connector_t< receiver_t > Class Template Reference

Inheritance diagram for MHAEvents::connector_t< receiver_t >:



Public Member Functions

- **connector_t** (**emitter_t** ∗, receiver_t ∗, void(receiver_t::∗)())
- **connector_t** (**emitter_t** ∗, receiver_t ∗, void(receiver_t::∗)(const std::string &))
- **connector_t** (**emitter_t** ∗, receiver_t ∗, void(receiver_t::∗)(const std::string &, unsigned int, unsigned int))
- ∼**connector_t** ()

Private Member Functions

- void **emit_event** ()
- void **emit_event** (const std::string &)
- void **emit_event** (const std::string &, unsigned int, unsigned int)

Private Attributes

- **emitter_t** ∗ **emitter**
- receiver_t ∗ **receiver**
- void(receiver_t::∗ **eventhandler**)()
- void(receiver_t::∗ **eventhandler_s**)(const std::string &)
- void(receiver_t::∗ **eventhandler_suu**)(const std::string &, unsigned int, unsigned int)

Additional Inherited Members

5.181.1 Constructor & Destructor Documentation

5.181.1.1 template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)() rfun)

5.181.1.2 template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &) rfun)

5.181.1.3 template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &, unsigned int, unsigned int) rfun)

5.181.1.4 template<class receiver_t > MHAEvents::connector_t< receiver_t >::~connector_t ( )

5.181.2 Member Function Documentation

5.181.2.1 template<class receiver_t > void MHAEvents::connector_t< receiver_t >::emit_event ( ) [private], [virtual]

Reimplemented from **MHAEvents::connector_base_t** (p. 507).

5.181.2.2 template<class receiver_t > void MHAEvents::connector_t< receiver_t >::emit_event (const std::string & arg) [private], [virtual]

Reimplemented from **MHAEvents::connector_base_t** (p. 507).

5.181.2.3 template<class receiver_t > void MHAEvents::connector_t< receiver_t >::emit_event (const std::string & arg, unsigned int arg2, unsigned int arg3) [private], [virtual]

Reimplemented from **MHAEvents::connector_base_t** (p. 507).

5.181.3 Member Data Documentation

5.181.3.1 template<class receiver_t> emitter_t* MHAEvents::connector_t< receiver_t >::emitter [private]

5.181.3.2 template<class receiver_t> receiver_t* MHAEvents::connector_t< receiver_t >::receiver [private]

5.181.3.3 template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler) () [private]

5.181.3.4 template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_s) (const std::string &) [private]

5.181.3.5 template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_suu) (const std::string &, unsigned int, unsigned int) [private]

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

- **mha_events.h**

5.182 MHAEvents::emitter_t Class Reference

Class for emitting openMHA events.

Public Member Functions

- ~**emitter_t** ()
- void **operator()** ()
  Emit an event without parameter.
- void **operator()** (const std::string &)
  Emit an event with string parameter.
- void **operator()** (const std::string &, unsigned int, unsigned int)
  Emit an event with string parameter and two unsigned int parameters.
- void **connect** (**connector_base_t** *)
- void **disconnect** (**connector_base_t** *)

Private Attributes

- std::list< **connector_base_t** * > **connections**

5.182.1 Detailed Description

Class for emitting openMHA events.

Use the template claas **MHAEvents::patchbay_t** (p. 512) for connecting to an emitter.

5.182.2 Constructor & Destructor Documentation

5.182.2.1 MHAEvents::emitter_t::~emitter_t ( )

5.182.3 Member Function Documentation

5.182.3.1 void MHAEvents::emitter_t::operator() ( )

Emit an event without parameter.

5.182.3.2 void MHAEvents::emitter_t::operator() (const std::string & *arg*)

Emit an event with string parameter.

5.182.3.3 void MHAEvents::emitter_t::operator() (const std::string & *arg,* unsigned int *arg2,* unsigned int *arg3*)

Emit an event with string parameter and two unsigned int parameters.

5.182.3.4 void MHAEvents::emitter_t::connect (connector_base_t * *c*)

5.182.3.5 void MHAEvents::emitter_t::disconnect (connector_base_t * *c*)

5.182.4 Member Data Documentation

5.182.4.1 std::list<connector_base_t*> MHAEvents::emitter_t::connections [private]

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

- **mha_event_emitter.h**
- **mha_events.cpp**

5.183 MHAEvents::patchbay_t< receiver_t > Class Template Reference

Patchbay which connects any event emitter with any member function of the parameter class.

Public Member Functions

- ~**patchbay_t** ()
- void **connect** (**emitter_t** *, receiver_t *, void(receiver_t::*)())
  Connect a receiver member function void (receiver_t::)() with an event emitter.*
- void **connect** (**emitter_t** *, receiver_t *, void(receiver_t::*)(const std::string &))
  Connect a receiver member function void (receiver_t::)(const std::string&) with an event emitter.*
- void **connect** (**emitter_t** *, receiver_t *, void(receiver_t::*)(const std::string &, unsigned int, unsigned int))

Private Attributes

- std::list< **connector_t**< receiver_t > * > **cons**

5.183.1 Detailed Description

template<class receiver_t>
class MHAEvents::patchbay_t< receiver_t >

Patchbay which connects any event emitter with any member function of the parameter class.

The connections created by the **connect()** (p. 512) function are hold until the destructor is called. To avoid access to invalid function pointers, it is required to destruct the patchbay before the receiver, usually by declaring the patchbay as a member of the receiver.

The receiver can be any claas or structure; the event callback can be either a member function without arguments or with const std::string& argument.

5.183.2 Constructor & Destructor Documentation

5.183.2.1 template<class receiver_t > MHAEvents::patchbay_t< receiver_t >::~patchbay_t ( )

5.183.3 Member Function Documentation

5.183.3.1 template<class receiver_t> void MHAEvents::patchbay_t< receiver_t >::connect (emitter_t * *e*, receiver_t * *r*, void(receiver_t::*)() *rfun*)

Connect a receiver member function void (receiver_t::*)() with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

e	Pointer to an event emitter
r	Pointer to the receiver
rfun	Pointer to a member function of the receiver class

5.183.3.2 template<class receiver_t> void MHAEvents::patchbay_t< receiver_t >::connect (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &) rfun)

Connect a receiver member function void (receiver_t::*)(const std::string&) with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

e	Pointer to an event emitter
r	Pointer to the receiver
rfun	Pointer to a member function of the receiver class

5.183.3.3 template<class receiver_t> void MHAEvents::patchbay_t< receiver_t >::connect (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &, unsigned int, unsigned int) rfun)

5.183.4 Member Data Documentation

5.183.4.1 template<class receiver_t> std::list<connector_t<receiver_t>*> MHAEvents::patchbay_t< receiver_t >::cons [private]

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

- **mha_events.h**

5.184 MHAFilter::adapt_filter_param_t Class Reference

Public Member Functions

- **adapt_filter_param_t** (**mha_real_t** imu, bool ierr_in)

Public Attributes

- **mha_real_t mu**
- bool **err_in**

5.184.1 Constructor & Destructor Documentation

5.184.1.1 MHAFilter::adapt_filter_param_t::adapt_filter_param_t (mha_real_t *imu,* bool *ierr_in*)

5.184.2 Member Data Documentation

5.184.2.1 mha_real_t MHAFilter::adapt_filter_param_t::mu

5.184.2.2 bool MHAFilter::adapt_filter_param_t::err_in

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.185 MHAFilter::adapt_filter_state_t Class Reference

Public Member Functions

- **adapt_filter_state_t** (int **ntaps**, int **nchannels**)
- void **filter** (**mha_wave_t** y, **mha_wave_t** e, **mha_wave_t** x, **mha_wave_t** d, **mha_real_t** mu, bool err_in)

Private Attributes

- int **ntaps**
- int **nchannels**
- **MHASignal::waveform_t W**
- **MHASignal::waveform_t X**
- **MHASignal::waveform_t od**
- **MHASignal::waveform_t oy**

5.185.1 Constructor & Destructor Documentation

5.185.1.1 MHAFilter::adapt_filter_state_t::adapt_filter_state_t (int *ntaps*, int *nchannels*)

5.185.2 Member Function Documentation

5.185.2.1 void MHAFilter::adapt_filter_state_t::filter (mha_wave_t *y*, mha_wave_t *e*, mha_wave_t *x*, mha_wave_t *d*, mha_real_t *mu*, bool *err_in*)

5.185.3 Member Data Documentation

5.185.3.1 int MHAFilter::adapt_filter_state_t::ntaps [private]

5.185.3.2 int MHAFilter::adapt_filter_state_t::nchannels [private]

5.185.3.3 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::W [private]

5.185.3.4 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::X [private]

5.185.3.5 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::od [private]

5.185.3.6 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::oy [private]

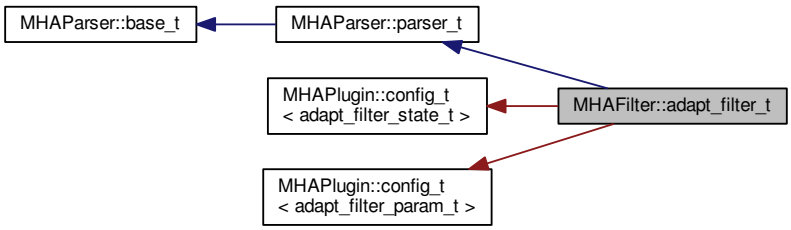
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.186 MHAFilter::adapt_filter_t Class Reference

Adaptive filter.

Inheritance diagram for MHAFilter::adapt_filter_t:



Public Member Functions

- **adapt_filter_t** (std::string)
- void **filter** (**mha_wave_t** y, **mha_wave_t** e, **mha_wave_t** x, **mha_wave_t** d)
- void **set_channelcnt** (unsigned int)

Private Member Functions

- void **update_mu** ()
- void **update_ntaps** ()

Private Attributes

- **MHAParser::float_t mu**
- **MHAParser::int_t ntaps**
- **MHAParser::bool_t err_in**
- **MHAEvents::patchbay_t**< **adapt_filter_t** > **connector**
- unsigned int **nchannels**

Additional Inherited Members

5.186.1 Detailed Description

Adaptive filter.

5.186.2 Constructor & Destructor Documentation

5.186.2.1 MHAFilter::adapt_filter_t::adapt_filter_t (std::string *help*)

5.186.3 Member Function Documentation

5.186.3.1 void MHAFilter::adapt_filter_t::filter (mha_wave_t *y*, mha_wave_t *e*, mha_wave_t *x*, mha_wave_t *d*)

5.186.3.2 void MHAFilter::adapt_filter_t::set_channelcnt (unsigned int *nch*)

5.186.3.3 void MHAFilter::adapt_filter_t::update_mu ( ) [private]

5.186.3.4 void MHAFilter::adapt_filter_t::update_ntaps ( ) [private]

5.186.4 Member Data Documentation

5.186.4.1 MHAParser::float_t MHAFilter::adapt_filter_t::mu [private]

5.186.4.2 MHAParser::int_t MHAFilter::adapt_filter_t::ntaps [private]

5.186.4.3 MHAParser::bool_t MHAFilter::adapt_filter_t::err_in [private]

5.186.4.4 MHAEvents::patchbay_t<adapt_filter_t> MHAFilter::adapt_filter_t::connector [private]

5.186.4.5 unsigned int MHAFilter::adapt_filter_t::nchannels [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.187 MHAFilter::blockprocessing_polyphase_resampling_t Class Reference

A class that does polyphase resampling and takes into account block processing.

Public Member Functions

- **blockprocessing_polyphase_resampling_t** (float source_srate, unsigned source_↩ fragsize, float target_srate, unsigned target_fragsize, float nyquist_ratio, float irslen, unsigned nchannels, bool add_delay)

  Contructs a polyphase resampling filter that can be used for blockprocessing with the given parameters.
- virtual ∼**blockprocessing_polyphase_resampling_t** ()
- void **write** (**mha_wave_t** &signal)

  Write signal to the ringbuffer.
- void **read** (**mha_wave_t** &signal)

  Read resampled signal.
- bool **can_read** () const

  Checks if the resampling ring buffer can produce another output signal block.

Private Attributes

- **polyphase_resampling_t** ∗ **resampling**
- unsigned **fragsize_in**
- unsigned **fragsize_out**
- unsigned **num_channels**

5.187.1 Detailed Description

A class that does polyphase resampling and takes into account block processing.

5.187.2 Constructor & Destructor Documentation

5.187.2.1 MHAFilter::blockprocessing_polyphase_resampling_t::blockprocessing_polyphase_resampling_t (float *source_srate,* unsigned *source_fragsize,* float *target_srate,* unsigned *target_fragsize,* float *nyquist_ratio,* float *irslen,* unsigned *nchannels,* bool *add_delay*)

Contructs a polyphase resampling filter that can be used for blockprocessing with the given parameters.

Parameters

source_srate	Source sampling rate / Hz
source_fragsize	Fragment size of incoming audio blocks / frames at source_srate
target_srate	Target sampling rate / Hz
target_fragsize	Fragment size of produced audio blocks / frames at target_srate
nyquist_ratio	Low pass filter cutoff frequency relative to the nyquist frequency of the smaller of the two sampling rates. Example values: 0.8, 0.9
irslen	Impulse response length used for low pass filtering / s
nchannels	Number of audio channels
add_delay	To avoid underruns, a delay is generally necessary for round trip block size adaptations. It is only necessary to add this delay to one of the two resampling chains. Set this parameter to true for the first resampling object of a round trip pair. It will add the necessary delay, and calculate the size of the ring buffer appropriately, When set to false, only the ringbuffer size will be set sufficiently.

5.187.2.2 virtual MHAFilter::blockprocessing_polyphase_resampling_t::~blockprocessing_polyphase_resampling_t ( ) [inline], [virtual]

5.187.3 Member Function Documentation

5.187.3.1 void MHAFilter::blockprocessing_polyphase_resampling_t::write (mha_wave_t & *signal*)

Write signal to the ringbuffer.

Parameters

signal	input signal in original sampling rate

Exceptions

MHA_Error *(p. 445)*	Raises exception if there is not enough room, if the number of channels does not match, or if the number of frames is not equal to the number specified in the constructor

5.187.3.2 void MHAFilter::blockprocessing_polyphase_resampling_t::read (mha_wave_t & *signal*)

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

signal	buffer to write the resampled signal to.

Exceptions

MHA_Error *(p. 445)*	Raises exception if there is not enough input signal, if the number of channels of frames does not match.

5.187.3.3 bool MHAFilter::blockprocessing_polyphase_resampling_t::can_read ( ) const [inline]

Checks if the resampling ring buffer can produce another output signal block.

5.187.4 Member Data Documentation

5.187.4.1 polyphase_resampling_t* MHAFilter::blockprocessing_polyphase_resampling_t::resampling [private]

5.187.4.2 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_in [private]

5.187.4.3 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_out [private]

5.187.4.4 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::num_channels [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.188 MHAFilter::complex_bandpass_t Class Reference

Complex bandpass filter.

Public Member Functions

- **complex_bandpass_t** (std::vector< **mha_complex_t** > A, std::vector< **mha_complex_t** > B)
  Constructor with filter coefficients (one per channel)
- void **set_state** (**mha_real_t** val)
- void **set_state** (std::vector< **mha_real_t** > val)
- void **set_state** (**mha_complex_t** val)
- void **set_weights** (std::vector< **mha_complex_t** > new_B)
  Allow to modify the input weights at a later stage.
- std::vector< **mha_complex_t** > **get_weights** () const
- void **filter** (const **mha_wave_t** &X, **mha_spec_t** &Y)
  Filter method for real value input.
- void **filter** (const **mha_wave_t** &X, **mha_wave_t** &Yre, **mha_wave_t** &Yim)
  Filter method for real value input.
- void **filter** (const **mha_spec_t** &X, **mha_spec_t** &Y)
  Filter method for complex value input.
- void **filter** (const **mha_wave_t** &Xre, const **mha_wave_t** &Xim, **mha_wave_t** &Yre, **mha_wave_t** &Yim)
  Filter method for complex value input.
- std::string **inspect** () const

Static Public Member Functions

- static std::vector< **mha_complex_t** > **creator_A** (std::vector< **mha_real_t** > cf, std::vector< **mha_real_t** > bw, **mha_real_t** srate, unsigned int order)
- static std::vector< **mha_complex_t** > **creator_B** (std::vector< **mha_complex_t** > A, unsigned int order)

Private Attributes

- std::vector< **mha_complex_t** > **A_**
- std::vector< **mha_complex_t** > **B_**
- std::vector< **mha_complex_t** > **Yn**

5.188.1 Detailed Description

Complex bandpass filter.

5.188.2 Constructor & Destructor Documentation

5.188.2.1 MHAFilter::complex_bandpass_t::complex_bandpass_t (std::vector< mha_complex_t > *A*, std::vector< mha_complex_t > *B*)

Constructor with filter coefficients (one per channel)

Parameters

A	complex filter coefficients, one per band
B	complex weights

5.188.3 Member Function Documentation

5.188.3.1 std::vector< mha_complex_t > MHAFilter::complex_bandpass_t::creator_A (std::vector< mha_real_t > *cf*, std::vector< mha_real_t > *bw*, mha_real_t *srate*, unsigned int *order*) [static]

5.188.3.2 std::vector< mha_complex_t > MHAFilter::complex_bandpass_t::creator_B (std::vector< mha_complex_t > *A*, unsigned int *order*) [static]

5.188.3.3 void MHAFilter::complex_bandpass_t::set_state (mha_real_t *val*)

5.188.3.4 void MHAFilter::complex_bandpass_t::set_state (std::vector< mha_real_t > *val*)

5.188.3.5 void MHAFilter::complex_bandpass_t::set_state (mha_complex_t *val*)

5.188.3.6 void MHAFilter::complex_bandpass_t::set_weights (std::vector< mha_complex_t > *new_B*)

Allow to modify the input weights at a later stage.

5.188.3.7 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::get_weights ( ) const [inline]

5.188.3.8 void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & *X*, mha_spec_t & *Y*) [inline]

Filter method for real value input.

5.188.3.9 void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & *X*, mha_wave_t & *Yre*, mha_wave_t & *Yim*) [inline]

Filter method for real value input.

5.188.3.10 void MHAFilter::complex_bandpass_t::filter (const mha_spec_t & *X*, mha_spec_t & *Y*) [inline]

Filter method for complex value input.

5.188.3.11 void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & *Xre*, const mha_wave_t & *Xim*, mha_wave_t & *Yre*, mha_wave_t & *Yim*) [inline]

Filter method for complex value input.

5.188.3.12 std::string MHAFilter::complex_bandpass_t::inspect ( ) const [inline]

5.188.4 Member Data Documentation

5.188.4.1 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::A_ [private]

5.188.4.2 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::B_ [private]

5.188.4.3 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::Yn [private]

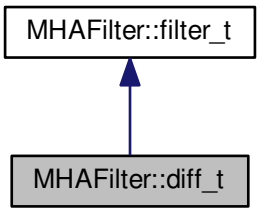
The documentation for this class was generated from the following files:

- **complex_filter.h**
- **complex_filter.cpp**

5.189 MHAFilter::diff_t Class Reference

Differentiator class (non-normalized)

Inheritance diagram for MHAFilter::diff_t:



Public Member Functions

- **diff_t** (unsigned int ch)

Additional Inherited Members

5.189.1 Detailed Description

Differentiator class (non-normalized)

5.189.2 Constructor & Destructor Documentation

5.189.2.1 MHAFilter::diff_t::diff_t (unsigned int *ch*)

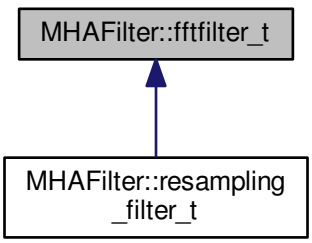
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.190 MHAFilter::fftfilter_t Class Reference

FFT based FIR filter implementation.

Inheritance diagram for MHAFilter::fftfilter_t:



Public Member Functions

- **fftfilter_t** (unsigned int **fragsize**, unsigned int **channels**, unsigned int **fftlen**)
  Constructor.
- ~**fftfilter_t** ()
- void **update_coeffs** (const **mha_wave_t** *pwIRS)
  Update the set of coefficients.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut, const **mha_wave_t** *pwIRS)
  Apply filter with changing coefficients to a waveform fragment.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut)
  Apply filter to waveform fragment, without changing the coefficients.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut, const **mha_spec_t** *psWeights)
  Apply filter with changing coefficients to a waveform fragment.

Private Attributes

- unsigned int **fragsize**
- unsigned int **channels**
- unsigned int **fftlen**
- **MHASignal::waveform_t wInput_fft**
- **mha_wave_t wInput**
- **MHASignal::waveform_t wOutput_fft**
- **mha_wave_t wOutput**
- **MHASignal::spectrum_t sInput**
- **MHASignal::spectrum_t sWeights**
- **MHASignal::waveform_t wIRS_fft**
- **mha_fft_t fft**

5.190.1 Detailed Description

FFT based FIR filter implementation.

The maximal number of coefficients can be FFT length - fragsize + 1.

5.190.2 Constructor & Destructor Documentation

5.190.2.1 MHAFilter::fftfilter_t::fftfilter_t (unsigned int *fragsize,* unsigned int *channels,* unsigned int *fftlen*)

Constructor.

Parameters

fragsize	Number of frames expected in input signal (each cycle).
channels	Number of channels expected in input signal.
fftlen	FFT length of filter.

5.190.2.2 MHAFilter::fftfilter_t::~fftfilter_t ( )

5.190.3 Member Function Documentation

5.190.3.1 void MHAFilter::fftfilter_t::update_coeffs (const mha_wave_t * *pwIRS*)

Update the set of coefficients.

Parameters

pwIRS	Coefficients structure

Note

The number of channels in h must match the number of channels given in the constructor. The filter length is limited to fftlen-fragsize+1 (longer IRS will be shortened).

5.190.3.2 void MHAFilter::fftfilter_t::filter (const mha_wave_t * *pwIn,* mha_wave_t ** *ppwOut,* const mha_wave_t * *pwIRS*)

Apply filter with changing coefficients to a waveform fragment.

Parameters

pw↩ In	Input signal pointer.

Return values

ppwOut	Pointer to output signal pointer, will be set to a valid signal.

Parameters

pwIRS	Pointer to FIR coefficients structure.

5.190.3.3 void MHAFilter::fftfilter_t::filter (const mha_wave_t * pwIn, mha_wave_t ** ppwOut)

Apply filter to waveform fragment, without changing the coefficients.

Parameters

pw↩ In	Input signal pointer.

Return values

ppwOut	Pointer to output signal pointer, will be set to a valid signal

5.190.3.4 void MHAFilter::fftfilter_t::filter (const mha_wave_t * pwIn, mha_wave_t ** ppwOut, const mha_spec_t * psWeights)

Apply filter with changing coefficients to a waveform fragment.

Parameters

pw↩ In	Input signal pointer.

Return values

ppwOut	Pointer to output signal pointer, will be set to a valid signal.

Parameters

psWeights	Pointer to filter weights structure.

5.190.4 Member Data Documentation

5.190.4.1 unsigned int MHAFilter::fftfilter_t::fragsize [private]

5.190.4.2 unsigned int MHAFilter::fftfilter_t::channels [private]

5.190.4.3 unsigned int MHAFilter::fftfilter_t::fftlen [private]

5.190.4.4 MHASignal::waveform_t MHAFilter::fftfilter_t::wInput_fft [private]

5.190.4.5 mha_wave_t MHAFilter::fftfilter_t::wInput [private]

5.190.4.6 MHASignal::waveform_t MHAFilter::fftfilter_t::wOutput_fft [private]

5.190.4.7 mha_wave_t MHAFilter::fftfilter_t::wOutput [private]

5.190.4.8 MHASignal::spectrum_t MHAFilter::fftfilter_t::sInput [private]

5.190.4.9 MHASignal::spectrum_t MHAFilter::fftfilter_t::sWeights [private]

5.190.4.10 MHASignal::waveform_t MHAFilter::fftfilter_t::wIRS_fft [private]

5.190.4.11 mha_fft_t MHAFilter::fftfilter_t::fft [private]

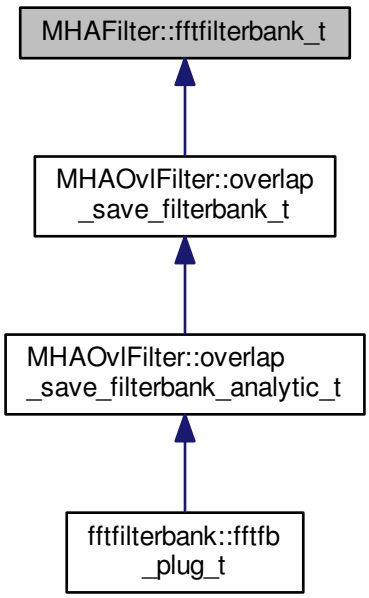
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.191 MHAFilter::fftfilterbank_t Class Reference

FFT based FIR filterbank implementation.

Inheritance diagram for MHAFilter::fftfilterbank_t:



Public Member Functions

- **fftfilterbank_t** (unsigned int **fragsize**, unsigned int **inputchannels**, unsigned int **firchannels**, unsigned int **fftlen**)
  Constructor.
- ∼**fftfilterbank_t** ()
- void **update_coeffs** (const **mha_wave_t** ∗h)
  Update the set of coefficients.
- void **filter** (const **mha_wave_t** ∗s_in, **mha_wave_t** ∗∗s_out, const **mha_wave_t** ∗h)
  Apply filter with changing coefficients to a waveform fragment.
- void **filter** (const **mha_wave_t** ∗s_in, **mha_wave_t** ∗∗s_out)
  Apply filter to waveform fragment, without changing the coefficients.
- const **mha_wave_t** ∗ **get_irs** () const
  Return the current IRS.

Private Attributes

- unsigned int **fragsize**
- unsigned int **inputchannels**
- unsigned int **firchannels**
- unsigned int **outputchannels**
- unsigned int **fftlen**
- **MHASignal::waveform_t hw**
- **MHASignal::spectrum_t Hs**
- **MHASignal::waveform_t xw**
- **MHASignal::spectrum_t Xs**
- **MHASignal::waveform_t yw**
- **MHASignal::spectrum_t Ys**
- **MHASignal::waveform_t yw_temp**
- **MHASignal::waveform_t tail**
- **mha_fft_t fft**

5.191.1 Detailed Description

FFT based FIR filterbank implementation.

This class convolves n input channels with m filter coefficient sets and returns n∗m output channels.

The maximal number of coefficients can be FFT length - fragsize + 1.

5.191.2 Constructor & Destructor Documentation

5.191.2.1 MHAFilter::fftfilterbank_t::fftfilterbank_t (unsigned int *fragsize,* unsigned int *inputchannels,* unsigned int *firchannels,* unsigned int *fftlen*)

Constructor.

Parameters

fragsize	Number of frames expected in input signal (each cycle).
inputchannels	Number of channels expected in input signal.
firchannels	Number of channels expected in FIR filter coefficients (= number of bands).
fftlen	FFT length of filter.

The number of output channels is inputchannels∗firchannels.

5.191.2.2 MHAFilter::fftfilterbank_t::∼fftfilterbank_t ( )

5.191.3 Member Function Documentation

5.191.3.1 void MHAFilter::fftfilterbank_t::update_coeffs (const mha_wave_t ∗ *h*)

Update the set of coefficients.

Parameters

h	Coefficients structure

Note

The number of channels in h must match the number of channels given in the constructor, and the number of frames can not be more than fftlen-fragsize+1.

5.191.3.2 void MHAFilter::fftfilterbank_t::filter (const mha_wave_t ∗ *s_in,* mha_wave_t ∗∗ *s_out,* const mha_wave_t ∗ *h*)

Apply filter with changing coefficients to a waveform fragment.

Parameters

s↩ _in	Input signal pointer.

Return values

s_out	Pointer to output signal pointer, will be set to a valid signal

Parameters

h	FIR coefficients

5.191.3.3 void MHAFilter::fftfilterbank_t::filter (const mha_wave_t * *s_in*, mha_wave_t ** *s_out*)

Apply filter to waveform fragment, without changing the coefficients.

Parameters

s_in	Input signal pointer.

Return values

s_out	Pointer to output signal pointer, will be set to a valid signal

5.191.3.4 const mha_wave_t* MHAFilter::fftfilterbank_t::get_irs ( ) const [inline]

Return the current IRS.

5.191.4 Member Data Documentation

5.191.4.1 unsigned int MHAFilter::fftfilterbank_t::fragsize [private]

5.191.4.2 unsigned int MHAFilter::fftfilterbank_t::inputchannels [private]

5.191.4.3 unsigned int MHAFilter::fftfilterbank_t::firchannels [private]

5.191.4.4 unsigned int MHAFilter::fftfilterbank_t::outputchannels [private]

5.191.4.5 unsigned int MHAFilter::fftfilterbank_t::fftlen [private]

5.191.4.6 MHASignal::waveform_t MHAFilter::fftfilterbank_t::hw [private]

5.191.4.7 MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Hs [private]

5.191.4.8 MHASignal::waveform_t MHAFilter::fftfilterbank_t::xw [private]

5.191.4.9 MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Xs [private]

5.191.4.10 MHASignal::waveform_t MHAFilter::fftfilterbank_t::yw [private]

5.191.4.11 MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Ys [private]

5.191.4.12 MHASignal::waveform_t MHAFilter::fftfilterbank_t::yw_temp [private]

5.191.4.13 MHASignal::waveform_t MHAFilter::fftfilterbank_t::tail [private]

5.191.4.14 mha_fft_t MHAFilter::fftfilterbank_t::fft [private]

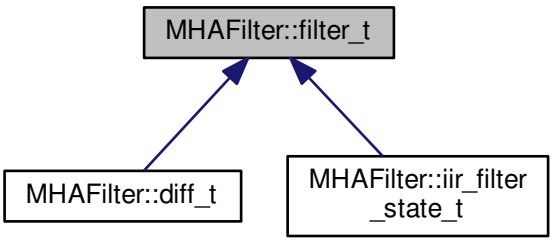
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.192 MHAFilter::filter_t Class Reference

Generic IIR filter class.

Inheritance diagram for MHAFilter::filter_t:



Public Member Functions

- **filter_t** (unsigned int ch, unsigned int lena, unsigned int lenb)
  Constructor.
- **filter_t** (unsigned int ch, const std::vector< **mha_real_t** > &vA, const std::vector< **mha_real_t** > &vB)
  Constructor with initialization of coefficients.
- **filter_t** (const **MHAFilter::filter_t** &src)
- ~**filter_t** ()
- void **filter** (**mha_wave_t** *out, const **mha_wave_t** *in)

Filter all channels in a waveform structure.

- void **filter** (**mha_real_t** ∗dest, const **mha_real_t** ∗src, unsigned int dframes, unsigned int frame_dist, unsigned int channel_dist, unsigned int channel_begin, unsigned int channel_end)

  Filter parts of a waveform structure.
- **mha_real_t filter** (**mha_real_t** x, unsigned int ch)

  Filter one sample.
- unsigned int **get_len_A** () const

  Return length of recursive coefficients.
- unsigned int **get_len_B** () const

  Return length of non-recursive coefficients.

Public Attributes

- double ∗ **A**

  Pointer to recursive coefficients.
- double ∗ **B**

  Pointer to non-recursive coefficients.

Private Attributes

- unsigned int **len_A**
- unsigned int **len_B**
- unsigned int **len**
- unsigned int **channels**
- double ∗ **state**

5.192.1 Detailed Description

Generic IIR filter class.

This class implements a generic multichannel IIR filter. It is realized as direct form II. It can work on any float array or on **mha_wave_t** (p. 496) structs. The filter coefficients can be directly accessed.

Todo Implement a more robust filter form.

5.192.2 Constructor & Destructor Documentation

5.192.2.1 MHAFilter::filter_t::filter_t (unsigned int *ch,* unsigned int *lena,* unsigned int *lenb*)

Constructor.

Parameters

ch	Number of channels
lena	Number of recursive coefficients
lenb	Number of non-recursive coefficients

5.192.2.2 MHAFilter::filter_t::filter_t (unsigned int *ch*, const std::vector< mha_real_t > & *vA*, const std::vector< mha_real_t > & *vB*)

Constructor with initialization of coefficients.

Parameters

ch	Number of channels.
vA	Recursive coefficients.
vB	Non-recursive coefficients.

5.192.2.3 MHAFilter::filter_t::filter_t (const MHAFilter::filter_t & *src*)

5.192.2.4 MHAFilter::filter_t::~filter_t ( )

5.192.3 Member Function Documentation

5.192.3.1 void MHAFilter::filter_t::filter (mha_wave_t * *out*, const mha_wave_t * *in*)

Filter all channels in a waveform structure.

Parameters

out	Output signal
in	Input signal

5.192.3.2 void MHAFilter::filter_t::filter (mha_real_t * dest, const mha_real_t * src, unsigned int dframes, unsigned int frame_dist, unsigned int channel_dist, unsigned int channel_begin, unsigned int channel_end)

Filter parts of a waveform structure.

Parameters

dest	Output signal.
src	Input signal.
dframes	Number of frames to be filtered.
frame_dist	Index distance between frames of one channel
channel_dist	Index distance between audio channels
channel_begin	Number of first channel to be processed
channel_end	Number of last channel to be processed

5.192.3.3 mha_real_t MHAFilter::filter_t::filter (mha_real_t x, unsigned int ch)

Filter one sample.

Parameters

x	Input value
ch	Channel number to use in filter state

5.192.3.4 unsigned int MHAFilter::filter_t::get_len_A ( ) const [inline]

Return length of recursive coefficients.

5.192.3.5 unsigned int MHAFilter::filter_t::get_len_B ( ) const [inline]

Return length of non-recursive coefficients.

5.192.4 Member Data Documentation

5.192.4.1 double* MHAFilter::filter_t::A

Pointer to recursive coefficients.

5.192.4.2 double* MHAFilter::filter_t::B

Pointer to non-recursive coefficients.

5.192.4.3 unsigned int MHAFilter::filter_t::len_A [private]

5.192.4.4 unsigned int MHAFilter::filter_t::len_B [private]

5.192.4.5 unsigned int MHAFilter::filter_t::len [private]

5.192.4.6 unsigned int MHAFilter::filter_t::channels [private]

5.192.4.7 double* MHAFilter::filter_t::state [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.193 MHAFilter::gamma_flt_t Class Reference

Class for gammatone filter.

Public Member Functions

- **gamma_flt_t** (std::vector< **mha_real_t** > cf, std::vector< **mha_real_t** > bw, **mha_real_t** srate, unsigned int order)
  Constructor.
- ~**gamma_flt_t** ()
- void **operator()** (**mha_wave_t** &X, **mha_spec_t** &Y)
  Filter method.
- void **operator()** (**mha_wave_t** &X, **mha_wave_t** &Yre, **mha_wave_t** &Yim)
  Filter method.
- void **operator()** (**mha_wave_t** &Yre, **mha_wave_t** &Yim, unsigned int stage)
  Filter method for specific stage.
- void **phase_correction** (unsigned int desired_delay, unsigned int inchannels)
- void **set_weights** (std::vector< **mha_complex_t** > new_B)
- void **set_weights** (unsigned int stage, std::vector< **mha_complex_t** > new_B)
- std::vector< **mha_complex_t** > **get_weights** () const
- std::vector< **mha_complex_t** > **get_weights** (unsigned int stage) const
- std::vector< **mha_real_t** > **get_resynthesis_gain** () const
- void **reset_state** ()
- const std::vector< **mha_complex_t** > & **get_A** ()
- std::string **inspect** () const

Private Attributes

- std::vector< **mha_complex_t** > **A**
- std::vector< **complex_bandpass_t** > **GF**
- **MHASignal::delay_t** * **delay**
- std::vector< int > **envelope_delay**
- std::vector< **mha_real_t** > **resynthesis_gain**
- std::vector< **mha_real_t** > **cf_**
- std::vector< **mha_real_t** > **bw_**
- **mha_real_t srate_**

5.193.1 Detailed Description

Class for gammatone filter.

5.193.2 Constructor & Destructor Documentation

5.193.2.1 MHAFilter::gamma_flt_t::gamma_flt_t (std::vector< mha_real_t > *cf,* std::vector< mha_real_t > *bw,* mha_real_t *srate,* unsigned int *order*)

Constructor.

Parameters

cf	Center frequency in Hz.
bw	Bandwidth in Hz (same number of entries as in cf).
srate	Sampling frequency in Hz.
order	Filter order.

5.193.2.2 MHAFilter::gamma_flt_t::~gamma_flt_t ( )

5.193.3 Member Function Documentation

5.193.3.1 void MHAFilter::gamma_flt_t::operator() (mha_wave_t & *X,* mha_spec_t & *Y*) [inline]

Filter method.

5.193.3.2 void MHAFilter::gamma_flt_t::operator() (
mha_wave_t & *X,*
mha_wave_t & *Yre,*
mha_wave_t & *Yim*) [inline]

Filter method.

5.193.3.3 void MHAFilter::gamma_flt_t::operator() (
mha_wave_t & *Yre,*
mha_wave_t & *Yim,*
unsigned int *stage*) [inline]

Filter method for specific stage.

5.193.3.4 void MHAFilter::gamma_flt_t::phase_correction (
unsigned int *desired_delay,*
unsigned int *inchannels*)

5.193.3.5 void MHAFilter::gamma_flt_t::set_weights (
std::vector< mha_complex_t > *new_B*)

5.193.3.6 void MHAFilter::gamma_flt_t::set_weights (
unsigned int *stage,*
std::vector< mha_complex_t > *new_B*)

5.193.3.7 std::vector<mha_complex_t> MHAFilter::gamma_flt_t::get_weights ( ) const [inline]

5.193.3.8 std::vector<mha_complex_t> MHAFilter::gamma_flt_t::get_weights (
unsigned int *stage*) const [inline]

5.193.3.9 std::vector<mha_real_t> MHAFilter::gamma_flt_t::get_resynthesis_gain ( ) const [inline]

5.193.3.10 void MHAFilter::gamma_flt_t::reset_state ( )

5.193.3.11 const std::vector<mha_complex_t>& MHAFilter::gamma_flt_t::get_A ( ) [inline]

5.193.3.12 std::string MHAFilter::gamma_flt_t::inspect ( ) const [inline]

5.193.4 Member Data Documentation

5.193.4.1 std::vector<mha_complex_t> MHAFilter::gamma_flt_t::A [private]

5.193.4.2 std::vector<complex_bandpass_t> MHAFilter::gamma_flt_t::GF [private]

5.193.4.3 MHASignal::delay_t* MHAFilter::gamma_flt_t::delay [private]

5.193.4.4 std::vector<int> MHAFilter::gamma_flt_t::envelope_delay [private]

5.193.4.5 std::vector<mha_real_t> MHAFilter::gamma_flt_t::resynthesis_gain [private]

5.193.4.6 std::vector<mha_real_t> MHAFilter::gamma_flt_t::cf_ [private]

5.193.4.7 std::vector<mha_real_t> MHAFilter::gamma_flt_t::bw_ [private]

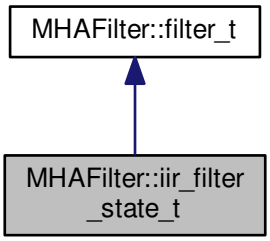
5.193.4.8 mha_real_t MHAFilter::gamma_flt_t::srate_ [private]

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

- **complex_filter.h**
- **complex_filter.cpp**

5.194 MHAFilter::iir_filter_state_t Class Reference

Inheritance diagram for MHAFilter::iir_filter_state_t:



Public Member Functions

- **iir_filter_state_t** (unsigned int **channels**, std::vector< float > cf_A, std::vector< float > cf_B)

Additional Inherited Members

5.194.1 Constructor & Destructor Documentation

5.194.1.1 MHAFilter::iir_filter_state_t::iir_filter_state_t (unsigned int *channels*, std::vector< float > *cf_A*, std::vector< float > *cf_B*)

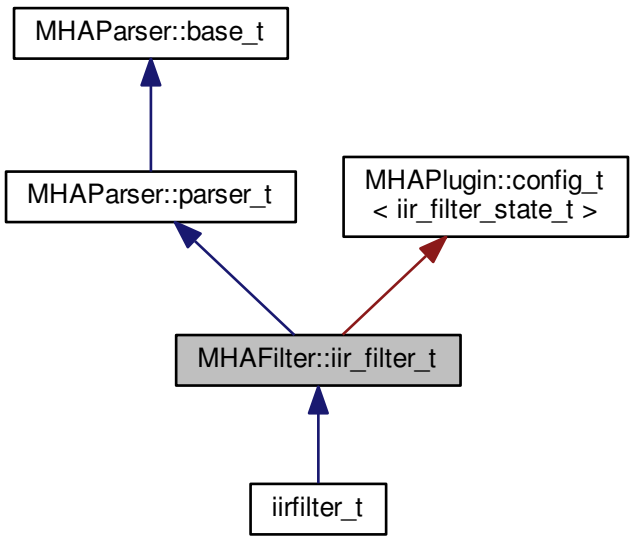
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.195 MHAFilter::iir_filter_t Class Reference

IIR filter class wrapper for integration into parser structure.

Inheritance diagram for MHAFilter::iir_filter_t:



Public Member Functions

- **iir_filter_t** (std::string **help**="IIR **filter** structure", std::string def_A="[1]", std::string def↩ _B="[1]", unsigned int **channels**=1)
  Constructor of the IIR filter.
- void **filter** (**mha_wave_t** ∗y, const **mha_wave_t** ∗x)
  The filter processes the audio signal.
- **mha_real_t filter** (**mha_real_t** x, unsigned int ch)
  Filter a single audio sample.
- void **resize** (unsigned int **channels**)
  Change the number of channels after object creation.

Private Member Functions

- void **update_filter** ()

Private Attributes

- **MHAParser::vfloat_t A**
- **MHAParser::vfloat_t B**
- **MHAEvents::patchbay_t**< **iir_filter_t** > **connector**
- unsigned int **nchannels**

Additional Inherited Members

5.195.1 Detailed Description

IIR filter class wrapper for integration into parser structure.

This class implements an infinite impulse response filter. Since it inherits from **MHAParser**↩ **::parser_t** (p. 688), it can easily be integrated in the openMHA configuration tree. It provides the configuration language variables "A" (vector of recursive filter coefficients) and "B" (vector of non-recursive filter coefficients).

The filter instance reacts to changes in filter coefficients through the openMHA configuration language, and uses the updated coefficients in the next invocation of the filter method.

Update of the coefficients is thread-safe and non-blocking. Simply add this subparser to your parser items and use the "filter" member function. Filter states are reset to all 0 on update.

5.195.2 Constructor & Destructor Documentation

5.195.2.1 MHAFilter::iir_filter_t::iir_filter_t (

std::string *help* = `"IIR` **filter** `structure",`
std::string *def_A* = `"[1]",`
std::string *def_B* = `"[1]",`
unsigned int *channels* = `1` **)**

Constructor of the IIR filter.

Initialises the sub-parser structure and the memory for holding the filter's state.

Parameters

help	The help string for the parser that groups the configuration variables of this filter. Could be used to describe the purpose of this IIR filter.
def_A	The initial value of the vector of the recursive filter coefficients, represented as string.
def_B	The initial value of the vector of the non-recursive filter coefficients, represented as string.
channels	The number of indipendent audio channels to process with this filter. Needed to allocate a state vector for each audio channel.

5.195.3 Member Function Documentation

5.195.3.1 void MHAFilter::iir_filter_t::filter (mha_wave_t * y, const mha_wave_t * x)

The filter processes the audio signal.

All channels in the audio signal are processed using the same filter coefficients. Indipendent state is stored between calls for each audio channel.

Parameters

y	Pointer to output signal holder. The output signal is stored here. Has to have the same signal dimensions as the input signal x. In-place processing (y and x pointing to the same signal holder) is possible.
x	Pointer to input signal holder. Number of channels has to be the same as given to the constructor, or to the **resize** (p. 542) method.

5.195.3.2 mha_real_t MHAFilter::iir_filter_t::filter (mha_real_t x, unsigned int ch)

Filter a single audio sample.

Parameters

x	The single audio sample
ch	Zero-based channel index. Use and change the state of channel ch. ch has to be less than the number of channels given to the constructor or the **resize** (p. 542) method.

Returns

the filtered result sample.

5.195.3.3 void MHAFilter::iir_filter_t::resize (unsigned int *channels*)

Change the number of channels after object creation.

Parameters

channels	The new number of channels. Old filter states are lost.

5.195.3.4 void MHAFilter::iir_filter_t::update_filter ( ) [private]

5.195.4 Member Data Documentation

5.195.4.1 MHAParser::vfloat_t MHAFilter::iir_filter_t::A [private]

5.195.4.2 MHAParser::vfloat_t MHAFilter::iir_filter_t::B [private]

5.195.4.3 MHAEvents::patchbay_t<iir_filter_t> MHAFilter::iir_filter_t::connector [private]

5.195.4.4 unsigned int MHAFilter::iir_filter_t::nchannels [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.196 MHAFilter::iir_ord1_real_t Class Reference

First order recursive filter.

Public Member Functions

- **iir_ord1_real_t** (std::vector< **mha_real_t** > A, std::vector< **mha_real_t** > B)
  Constructor with filter coefficients (one per channel)
- **iir_ord1_real_t** (std::vector< **mha_real_t** > tau, **mha_real_t** srate)
  Constructor for low pass filter (one time constant per channel)
- void **set_state** (**mha_real_t** val)
- void **set_state** (std::vector< **mha_real_t** > val)
- void **set_state** (**mha_complex_t** val)
- **mha_real_t operator()** (unsigned int ch, **mha_real_t** x)
  Filter method for real value input, one element.
- **mha_complex_t operator()** (unsigned int ch, **mha_complex_t** x)
  Filter method for complex input, one element.
- void **operator()** (const **mha_wave_t** &X, **mha_wave_t** &Y)
  Filter method for real value input.
- void **operator()** (const **mha_spec_t** &X, **mha_spec_t** &Y)
  Filter method for complex value input.
- void **operator()** (const **mha_wave_t** &Xre, const **mha_wave_t** &Xim, **mha_wave_t** &Yre, **mha_wave_t** &Yim)
  Filter method for complex value input.

Private Attributes

- std::vector< **mha_real_t** > **A_**
- std::vector< **mha_real_t** > **B_**
- std::vector< **mha_complex_t** > **Yn**

5.196.1 Detailed Description

First order recursive filter.

5.196.2 Constructor & Destructor Documentation

5.196.2.1 MHAFilter::iir_ord1_real_t::iir_ord1_real_t (std::vector< mha_real_t > *A,* std::vector< mha_real_t > *B*)

Constructor with filter coefficients (one per channel)

5.196.2.2 MHAFilter::iir_ord1_real_t::iir_ord1_real_t (std::vector< mha_real_t > *tau,* mha_real_t *srate*)

Constructor for low pass filter (one time constant per channel)

5.196.3 Member Function Documentation

5.196.3.1 void MHAFilter::iir_ord1_real_t::set_state (mha_real_t *val*)

5.196.3.2 void MHAFilter::iir_ord1_real_t::set_state (std::vector< mha_real_t > *val*)

5.196.3.3 void MHAFilter::iir_ord1_real_t::set_state (mha_complex_t *val*)

5.196.3.4 mha_real_t MHAFilter::iir_ord1_real_t::operator() (unsigned int *ch,* mha_real_t *x*) [inline]

Filter method for real value input, one element.

5.196.3.5 mha_complex_t MHAFilter::iir_ord1_real_t::operator() (unsigned int *ch,* mha_complex_t *x*) [inline]

Filter method for complex input, one element.

5.196.3.6 void MHAFilter::iir_ord1_real_t::operator() (const mha_wave_t & *X,* mha_wave_t & *Y*) [inline]

Filter method for real value input.

5.196.3.7 void MHAFilter::iir_ord1_real_t::operator() (const mha_spec_t & *X,* mha_spec_t & *Y*) [inline]

Filter method for complex value input.

5.196.3.8 void MHAFilter::iir_ord1_real_t::operator() (const mha_wave_t & *Xre,* const mha_wave_t & *Xim,* mha_wave_t & *Yre,* mha_wave_t & *Yim*) [inline]

Filter method for complex value input.

5.196.4 Member Data Documentation

5.196.4.1 std::vector<mha_real_t> MHAFilter::iir_ord1_real_t::A_ [private]

5.196.4.2 std::vector<mha_real_t> MHAFilter::iir_ord1_real_t::B_ [private]

5.196.4.3 std::vector<mha_complex_t> MHAFilter::iir_ord1_real_t::Yn [private]

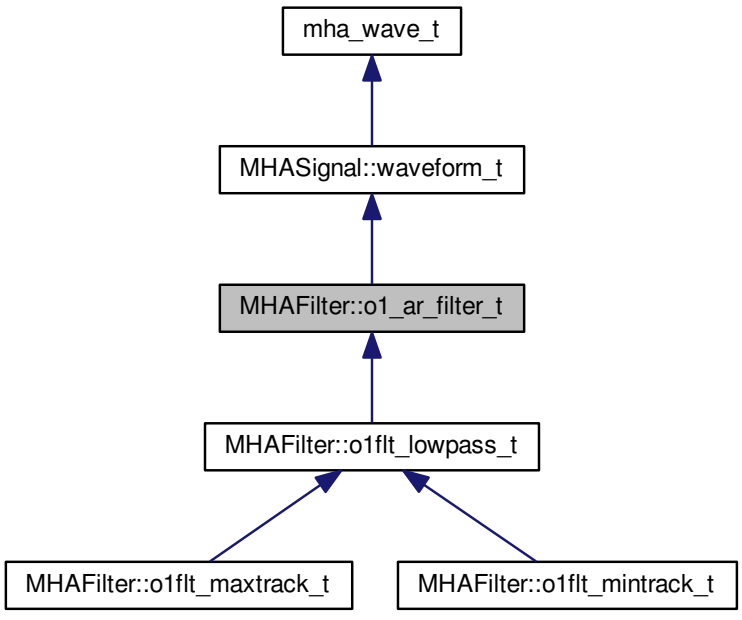
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.197 MHAFilter::o1_ar_filter_t Class Reference

First order attack-release lowpass filter.

Inheritance diagram for MHAFilter::o1_ar_filter_t:



Public Member Functions

- **o1_ar_filter_t** (unsigned int **channels**, **mha_real_t fs**=1.0f, std::vector< mha_real_t > tau_a=std::vector< float >(1, 0.0f), std::vector< mha_real_t > tau_r=std::vector< float >(1, 0.0f))

  Constructor, setting all taus to zero.
- void **set_tau_attack** (unsigned int ch, **mha_real_t** tau)

  Set the attack time constant.
- void **set_tau_release** (unsigned int ch, **mha_real_t** tau)

  Set the release time constant.
- **mha_real_t operator()** (unsigned int ch, **mha_real_t** x)

  Apply filter to value x, using state channel ch.
- void **operator()** (const **mha_wave_t** &in, **mha_wave_t** &out)

  *Apply filter to a **mha_wave_t** (p. 496) data.*

Protected Attributes

- **MHASignal::waveform_t c1_a**
- **MHASignal::waveform_t c2_a**
- **MHASignal::waveform_t c1_r**
- **MHASignal::waveform_t c2_r**
- **mha_real_t fs**

Additional Inherited Members

5.197.1 Detailed Description

First order attack-release lowpass filter.

This filter is the base of first order lowpass filter, maximum tracker and minimum tracker.

5.197.2 Constructor & Destructor Documentation

5.197.2.1 MHAFilter::o1_ar_filter_t::o1_ar_filter_t (unsigned int *channels*, mha_real_t *fs* = 1.0f, std::vector< mha_real_t > *tau_a* = std::vector<float>(1,0.0f), std::vector< mha_real_t > *tau_r* = std::vector<float>(1,0.0f))

Constructor, setting all taus to zero.

The filter state can be accessed through the member functions of **MHASignal::waveform_t** (p. 812).

Parameters

channels	Number of independent filters
fs	Sampling rate (optional, default = 1)
tau_a	Attack time constants (optional, default = 0)
tau_r	Release time constants (optional, default = 0)

5.197.3 Member Function Documentation

5.197.3.1 void MHAFilter::o1_ar_filter_t::set_tau_attack (unsigned int *ch,* mha_real_t *tau*)

Set the attack time constant.

Parameters

ch	Channel number
tau	Time constant

5.197.3.2 void MHAFilter::o1_ar_filter_t::set_tau_release (unsigned int *ch,* mha_real_t *tau*)

Set the release time constant.

Parameters

ch	Channel number
tau	Time constant

5.197.3.3 mha_real_t MHAFilter::o1_ar_filter_t::operator() (unsigned int *ch,* mha_real_t *x*) [inline]

Apply filter to value x, using state channel ch.

Parameters

ch	Cannel number
x	Input value

Returns

Output value

5.197.3.4 void MHAFilter::o1_ar_filter_t::operator() (const mha_wave_t & *in*, mha_wave_t & *out*) [inline]

Apply filter to a **mha_wave_t** (p. 496) data.

Parameters

in	Input signal
out	Output signal

The number of channels must match the number of filter bands.

5.197.4 Member Data Documentation

5.197.4.1 MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c1_a [protected]

5.197.4.2 MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c2_a [protected]

5.197.4.3 MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c1_r [protected]

5.197.4.4 MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c2_r [protected]

5.197.4.5 mha_real_t MHAFilter::o1_ar_filter_t::fs [protected]

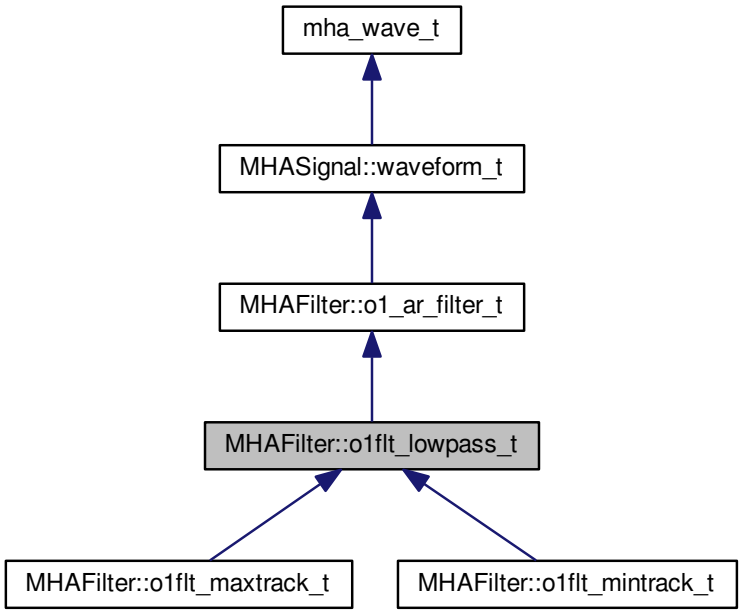
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.198 MHAFilter::o1flt_lowpass_t Class Reference

First order low pass filter.

Inheritance diagram for MHAFilter::o1flt_lowpass_t:



Public Member Functions

- **o1flt_lowpass_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
  Constructor of low pass filter, sets sampling rate and time constants.
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
  change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
  set time constant in all channels to tau
- **mha_real_t get_c1** (unsigned int ch) const
- **mha_real_t get_last_output** (unsigned int ch) const

Additional Inherited Members

5.198.1 Detailed Description

First order low pass filter.

5.198.2 Constructor & Destructor Documentation

5.198.2.1 MHAFilter::o1flt_lowpass_t::o1flt_lowpass_t (const std::vector< mha_real_t > & *tau,* mha_real_t *fs,* mha_real_t *startval* = 0)

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

tau	Vector of time constants
fs	Sampling rate
startval	Initial internal state value

5.198.3 Member Function Documentation

5.198.3.1 void MHAFilter::o1flt_lowpass_t::set_tau (unsigned int *ch,* mha_real_t *tau*)

change the time constant in one channel

5.198.3.2 void MHAFilter::o1flt_lowpass_t::set_tau (mha_real_t *tau*)

set time constant in all channels to tau

5.198.3.3 mha_real_t MHAFilter::o1flt_lowpass_t::get_c1 (unsigned int *ch*) const [inline]

5.198.3.4 mha_real_t MHAFilter::o1flt_lowpass_t::get_last_output (unsigned int *ch*) const [inline]

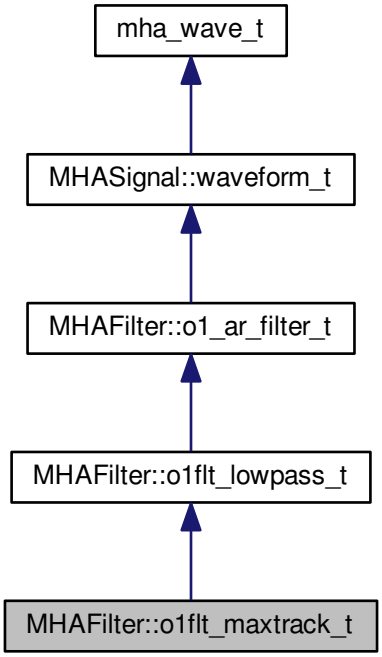
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.199 MHAFilter::o1flt_maxtrack_t Class Reference

First order maximum tracker.

Inheritance diagram for MHAFilter::o1flt_maxtrack_t:



Public Member Functions

- **o1flt_maxtrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
  Constructor of low pass filter, sets sampling rate and time constants.
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
  change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
  set time constant in all channels to tau

Additional Inherited Members

5.199.1 Detailed Description

First order maximum tracker.

5.199.2 Constructor & Destructor Documentation

5.199.2.1 MHAFilter::o1flt_maxtrack_t::o1flt_maxtrack_t (const std::vector< mha_real_t > & *tau,* mha_real_t *fs,* mha_real_t *startval* = 0)

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

tau	Vector of time constants
fs	Sampling rate
startval	Initial internal state value

5.199.3 Member Function Documentation

5.199.3.1 void MHAFilter::o1flt_maxtrack_t::set_tau (unsigned int *ch,* mha_real_t *tau*)

change the time constant in one channel

5.199.3.2 void MHAFilter::o1flt_maxtrack_t::set_tau (mha_real_t *tau*)

set time constant in all channels to tau

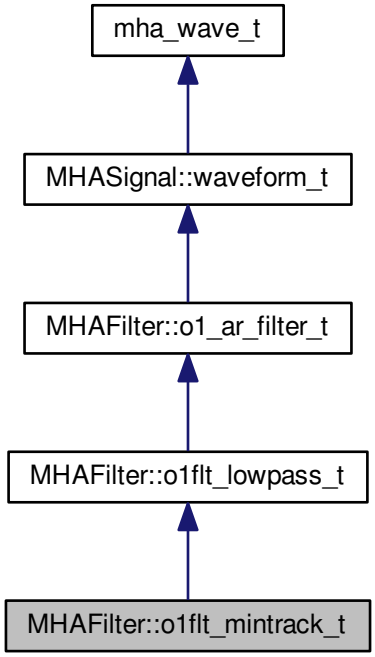
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.200 MHAFilter::o1flt_mintrack_t Class Reference

First order minimum tracker.

Inheritance diagram for MHAFilter::o1flt_mintrack_t:



Public Member Functions

- **o1flt_mintrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
- void **set_tau** (unsigned int ch, **mha_real_t** tau)

  change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)

  set time constant in all channels to tau

Additional Inherited Members

5.200.1 Detailed Description

First order minimum tracker.

5.200.2 Constructor & Destructor Documentation

5.200.2.1 MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (const std::vector< mha_real_t > & *tau,* mha_real_t *fs_,* mha_real_t *startval* = 0)

5.200.3 Member Function Documentation

5.200.3.1 void MHAFilter::o1flt_mintrack_t::set_tau (unsigned int *ch,* mha_real_t *tau*)

change the time constant in one channel

5.200.3.2 void MHAFilter::o1flt_mintrack_t::set_tau (mha_real_t *tau*)

set time constant in all channels to tau

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.201 MHAFilter::partitioned_convolution_t Class Reference

A filter class for partitioned convolution.

Classes

- struct **index_t**
  Bookkeeping class.

Public Member Functions

- **partitioned_convolution_t** (unsigned int **fragsize**, unsigned int **nchannels_in**, unsigned int **nchannels_out**, const **transfer_matrix_t** &transfer)
  Create a new partitioned convolver.
- ~**partitioned_convolution_t** ()
  Free fftw resource allocated in constructor.
- **mha_wave_t** * **process** (const **mha_wave_t** *s_in)
  processing

Public Attributes

- unsigned int **fragsize**
  Audio fragment size, always equal to partition size.
- unsigned int **nchannels_in**
  Number of audio input channels.
- unsigned int **nchannels_out**
  Number of audio output channels.
- unsigned int **output_partitions**
  The maximum number of partitions in any of the impulse responses.
- unsigned int **filter_partitions**
  The total number of non-zero impulse response partitions.
- **MHASignal::waveform_t input_signal_wave**
  Buffer for input signal.
- unsigned int **current_input_signal_buffer_half_index**
  A counter modulo 2.
- **MHASignal::spectrum_t input_signal_spec**
  Buffer for FFT transformed input signal.
- **MHASignal::spectrum_t frequency_response**
  Buffers for frequency response spectra of impulse response partitions.
- std::vector< **index_t** > **bookkeeping**
  Keeps track of input channels, output channels, impulse response partition, and delay.
- std::vector< **MHASignal::spectrum_t** > **output_signal_spec**
  Buffers for FFT transformed output signal.
- unsigned int **current_output_partition_index**
  A counter modulo output_partitions, indexing the "current" output partition.
- **MHASignal::waveform_t output_signal_wave**
  Buffer for the wave output signal.
- **mha_fft_t fft**
  The FFT transformer.

5.201.1 Detailed Description

A filter class for partitioned convolution.

Impulse responses are partitioned into sections of fragment size. Audio signal is convolved with every partition and delayed as needed. Convolution is done according to overlap-save. FFT length used is 2 times fragment size.

5.201.2 Constructor & Destructor Documentation

5.201.2.1 MHAFilter::partitioned_convolution_t::partitioned_convolution_t (unsigned int *fragsize,* unsigned int *nchannels_in,* unsigned int *nchannels_out,* const **transfer_matrix_t** & *transfer*)

Create a new partitioned convolver.

Parameters

fragsize	Audio fragment size, equal to partition size.
nchannels_in	Number of input audio channels.
nchannels_out	Number of output audio channels.
transfer	A sparse matrix of impulse responses.

5.201.2.2 MHAFilter::partitioned_convolution_t::~partitioned_convolution_t ( )

Free fftw resource allocated in constructor.

5.201.3 Member Function Documentation

5.201.3.1 mha_wave_t * MHAFilter::partitioned_convolution_t::process (const mha_wave_t * *s_in*)

processing

5.201.4 Member Data Documentation

5.201.4.1 unsigned int MHAFilter::partitioned_convolution_t::fragsize

Audio fragment size, always equal to partition size.

5.201.4.2 unsigned int MHAFilter::partitioned_convolution_t::nchannels_in

Number of audio input channels.

5.201.4.3 unsigned int MHAFilter::partitioned_convolution_t::nchannels_out

Number of audio output channels.

5.201.4.4 unsigned int MHAFilter::partitioned_convolution_t::output_partitions

The maximum number of partitions in any of the impulse responses.

Determines the size if the delay line.

5.201.4.5 unsigned int MHAFilter::partitioned_convolution_t::filter_partitions

The total number of non-zero impulse response partitions.

5.201.4.6 MHASignal::waveform_t MHAFilter::partitioned_convolution_t::input_signal_wave

Buffer for input signal.

Has nchannels_in channels and fragsize∗2 frames

5.201.4.7 unsigned int MHAFilter::partitioned_convolution_t::current_input_signal_buffer_half_index

A counter modulo 2.

Indicates the buffer half in input signal wave into which to copy the current input signal.

5.201.4.8 MHASignal::spectrum_t MHAFilter::partitioned_convolution_t::input_signal_spec

Buffer for FFT transformed input signal.

Has nchannels_in channels and fragsize+1 frames (fft bins).

5.201.4.9 MHASignal::spectrum_t MHAFilter::partitioned_convolution_t::frequency_response

Buffers for frequency response spectra of impulse response partitions.

Each "channel" contains another partition of some impulse response. The bookkeeping array is used to keep track what to do with these frequency responses. This container has filter_↩ partitions channels and fragsize+1 frames (fft bins).

5.201.4.10 std::vector<index_t> MHAFilter::partitioned_convolution_t::bookkeeping

Keeps track of input channels, output channels, impulse response partition, and delay.

The index into this array is the same as the "channel" index into the frequency_response array. Array has filter_partitions entries.

5.201.4.11 std::vector<MHASignal::spectrum_t> MHAFilter::partitioned_convolution_t::output_↩ signal_spec

Buffers for FFT transformed output signal.

For each array member, Number of channels is equal to nchannels_out, number of frames (fft bins) is equal to fragsize+1. Array size is equal to output_partitions.

5.201.4.12 unsigned int MHAFilter::partitioned_convolution_t::current_output_partition_index

A counter modulo output_partitions, indexing the "current" output partition.

5.201.4.13 MHASignal::waveform_t MHAFilter::partitioned_convolution_t::output_signal_wave

Buffer for the wave output signal.

Number of channels is equal to nchannels_out, number of frames is equal to fragsize

5.201.4.14 mha_fft_t MHAFilter::partitioned_convolution_t::fft

The FFT transformer.

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.202 MHAFilter::partitioned_convolution_t::index_t Struct Reference

Bookkeeping class.

Public Member Functions

- **index_t** (unsigned int src, unsigned int tgt, unsigned int dly)
  Data constructor.
- **index_t** ()
  Default constructor for STL compatibility.

Public Attributes

- unsigned int **source_channel_index**
  The input channel index to apply the current partition to.
- unsigned int **target_channel_index**
  The index of the output channel to which the filter result should go.
- unsigned int **delay**
  The delay (in blocks) of this partition.

5.202.1 Detailed Description

Bookkeeping class.

For each impulse response partition, keeps track of which input to filter, which output channel to filter to, and the delay in blocks. Objects of class Index should be kept in an array with the same indices as the corresponding inpulse response partitions.

5.202.2 Constructor & Destructor Documentation

5.202.2.1 MHAFilter::partitioned_convolution_t::index_t::index_t (unsigned int *src,* unsigned int *tgt,* unsigned int *dly*) [inline]

Data constructor.

Parameters

src	The input channel index to apply the current partition to.
tgt	The index of the output channel to which the filter result should go.
dly	The delay (in blocks) of this partition

5.202.2.2 MHAFilter::partitioned_convolution_t::index_t::index_t ( ) [inline]

Default constructor for STL compatibility.

5.202.3 Member Data Documentation

5.202.3.1 unsigned int MHAFilter::partitioned_convolution_t::index_t::source_channel_index

The input channel index to apply the current partition to.

5.202.3.2 unsigned int MHAFilter::partitioned_convolution_t::index_t::target_channel_index

The index of the output channel to which the filter result should go.

5.202.3.3 unsigned int MHAFilter::partitioned_convolution_t::index_t::delay

The delay (in blocks) of this partition.

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

- **mha_filter.hh**

5.203 MHAFilter::polyphase_resampling_t Class Reference

A class that performs polyphase resampling.

Public Member Functions

- **polyphase_resampling_t** (unsigned n_up, unsigned n_down, **mha_real_t** nyquist_ratio, unsigned n_irs, unsigned n_ringbuffer, unsigned n_channels, unsigned n_prefill)
  Construct a polyphase resampler instance.
- void **write** (**mha_wave_t** &signal)
  Write signal to the ringbuffer.
- void **read** (**mha_wave_t** &signal)
  Read resampled signal.
- unsigned **readable_frames** () const
  Number of frames at target sampling rate that can be produced.

Private Attributes

- unsigned **upsampling_factor**

  Integer upsampling factor.
- unsigned **downsampling_factor**

  Integer downsampling factor.
- unsigned **now_index**

  Index of "now" in the interpolated sampling rate.
- bool **underflow**

  Set to true when an underflow has occurred.
- **MHAWindow::hanning_t impulse_response**

  Contains the impulse response of the lowpass filter needed for anti-aliasing.
- **MHASignal::ringbuffer_t ringbuffer**

  Storage of input signal.

5.203.1 Detailed Description

A class that performs polyphase resampling.

Background information: When resampling from one sampling rate to another, it helps when one sampling rate is a multiple of the other sampling rate: In the case of upsampling, the samples at the original rate are copied to the upsampled signal spread out with a constant number of zero samples between the originally adjacent samples. The signal is then low-pass filtered to avoid frequency aliasing and to fill the zero-samples with interpolated values. In the case of down-sampling, the signal is first low-pass filtered for anti-aliasing, and only every n^{th} sample of the filtered output is used for the signal at the new sample rate. Of course, for finite-impulse-response (FIR) filters this means that only every n^{th} sample needs to be computed.

When resampling from one sampling rate to another where neither is a multiple of the other, the signal first needs to be upsampled to a sampling rate that is a multiple of both (source and target) sampling rates, and then downsampled again to the target sampling rate. Instead of applying two separate lowpass filters directly after each other (one filter for upsampling and another for downsampling), it is sufficient to apply only one low-pass filter, when producing the output at the final target rate, with a cut-off frequency equal to the lower cut-off-frequency of the replaced two low-pass filters. Not filtering to produce a filtered signal already at the common multiple sampling rate has the side effect that this intermediate signal at the common multiple sampling rate keeps its filler zero samples unaltered. These zero samples can be taken advantage of when filtering to produce the output at the target rate: The zeros do not need to be multiplied with their corresponding filter coefficients, because the result is known to be zero again, and this zero product has no effect on the summation operation to compute a target sample at the target rate. To summarize, the following optimization techniques are available:

- The signal does not need to be stored in memory at the interpolation rate. It is sufficient to have the signal available at the source rate and to know where the zeros would be.
- The signal needs to be low-pass-filtered only once.
- The FIR low-pass filtering can take advantage of

- computing only filter outputs for the required samples at the target rate,
- skipping over zero-samples at the interpolation rate.

The procedure that takes advantage of these optimization possibilites is known as polyphase resampling.

This class implements polyphase resampling in this way for a source sampling rate and a target sampling rate that have common multiple, the interpolation sampling rate. Non-rational and drifting sample rates are outside the scope of this resampler.

5.203.2 Constructor & Destructor Documentation

5.203.2.1 MHAFilter::polyphase_resampling_t::polyphase_resampling_t (

unsigned ***n_up,***
unsigned ***n_down,***
mha_real_t ***nyquist_ratio,***
unsigned ***n_irs,***
unsigned ***n_ringbuffer,***
unsigned ***n_channels,***
unsigned ***n_prefill***)

Construct a polyphase resampler instance.

Allocates a ringbuffer with the given capacity *n_ringbuffer*. Client that triggers the constructor must ensure that the capacity *n_ringbuffer* and the delay *n_prefill* are sufficient, i.e. enough old and new samples are always available to compute sufficient samples in using an impulse response of length *n_irs*. Audio block sizes at both sides of the resampler have to be taken into account. Class `MHASignal::blockprocessing_polyphase_resampling_t` takes care of this, and it is recommended to use this class for block-based processing.

Based on *n_up*, *n_down*, *n_irs* and *nyquist_ratio*, a suitable sinc impulse response is computed and windowed with a hanning window to limit its extent.

The actual source sampling rate, target sampling rate, and interpolation sampling rate are not parameters to this constructors, because only their ratios matter.

Parameters

n_up	upsampling factor, ratio between interpolation rate and source rate.
n_down	downsampling factor, ratio between interpolation rate and target rate.
nyquist_ratio	low pass filter cutoff frequency relative to the nyquist frequency of the smaller of the two sampling rates. Example values: E.g. 0.8, 0.9
n_irs	length of impulse response (in samples at interpolation rate)
n_ringbuffer	length of ringbuffer, in samples at source sampling rate
n_channels	audio channels count
n_prefill	Prefill the ringbuffer with this many zero frames in samples at source sampling rate

5.203.3 Member Function Documentation

5.203.3.1 void MHAFilter::polyphase_resampling_t::write (mha_wave_t & *signal*)

Write signal to the ringbuffer.

Signal contained in signal is appended to the audio frames already present in the ringbuffer.

Parameters

signal	input signal in original sampling rate

Exceptions

MHA_Error *(p. 445)*	Raises exception if there is not enough room or if the number of channels does not match.

5.203.3.2 void MHAFilter::polyphase_resampling_t::read (mha_wave_t & *signal*)

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

signal	buffer to write the resampled signal to.

Exceptions

MHA_Error *(p. 445)*	Raises exception if there is not enough input signal or if the number of channels is too high.

5.203.3.3 unsigned MHAFilter::polyphase_resampling_t::readable_frames ( ) const [inline]

Number of frames at target sampling rate that can be produced.

This method only checks for enough future samples present, therefore, this number can be positive and a read operation can still fail if there are not enough past samples present to perform the filtering for the first output sample. This could only happen if the constructor parameters *n_ringbuffer* or *n_prefill* have been chosen too small, because otherwise the method **read** (p. 562) ensures that enough past samples are present to compute the next target sample.

5.203.4 Member Data Documentation

5.203.4.1 unsigned MHAFilter::polyphase_resampling_t::upsampling_factor [private]

Integer upsampling factor.

Interpolation rate divided by source rate.

5.203.4.2 unsigned MHAFilter::polyphase_resampling_t::downsampling_factor [private]

Integer downsampling factor.

Interpolation rate divided by target rate.

5.203.4.3 unsigned MHAFilter::polyphase_resampling_t::now_index [private]

Index of "now" in the interpolated sampling rate.

Todo Index into what? What is the meaning of now?

5.203.4.4 bool MHAFilter::polyphase_resampling_t::underflow [private]

Set to true when an underflow has occurred.

When this is true, then the object can no longer be used. Underflows have to be avoided by clients, e.g. by checking that enough **readable_frames** (p. 562) are present before calling **read** (p. 562)

5.203.4.5 MHAWindow::hanning_t MHAFilter::polyphase_resampling_t::impulse_response [private]

Contains the impulse response of the lowpass filter needed for anti-aliasing.

The impulse response is stored at the interpolation sampling rate. We use an instance of **MHAWindow::hanning_t** (p. 840) here because we are limiting the sinc impulse response with a Hanning window (otherwise the impulse response would extend indefinitely into past and future). And the samples inside an **MHAWindow::hanning_t** (p. 840) can be altered with $*=$, which our constructor does.

5.203.4.6 MHASignal::ringbuffer_t MHAFilter::polyphase_resampling_t::ringbuffer [private]

Storage of input signal.

Part of the polyphase resampling optimization is that apart from the FIR impulse response, nothing is stored at the interpolation rate, saving memory and computation cycles.

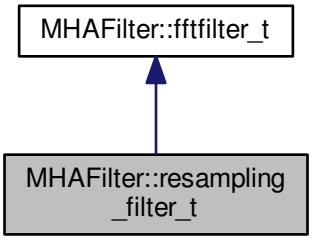
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.204 MHAFilter::resampling_filter_t Class Reference

Hann shaped low pass filter for resampling.

Inheritance diagram for MHAFilter::resampling_filter_t:



Public Member Functions

- **resampling_filter_t** (unsigned int **fftlen**, unsigned int irslen, unsigned int **channels**, unsigned int Nup, unsigned int Ndown, double fCutOff)

  Constructor.

Static Public Member Functions

- static unsigned int **fragsize_validator** (unsigned int **fftlen**, unsigned int irslen)

Private Attributes

- unsigned int **fragsize**

5.204.1 Detailed Description

Hann shaped low pass filter for resampling.

This class uses FFT filter at upsampled rate.

5.204.2 Constructor & Destructor Documentation

5.204.2.1 MHAFilter::resampling_filter_t::resampling_filter_t (

unsigned int ***fftlen,***
unsigned int ***irslen,***
unsigned int ***channels,***
unsigned int ***Nup,***
unsigned int ***Ndown,***
double ***fCutOff***)

Constructor.

Parameters

fftlen	FFT length.
irslen	Length of filter.
channels	Number of channels to be filtered.
Nup	Upsampling ratio.
Ndown	Downsampling ratio.
fCutOff	Cut off frequency (relative to lower Nyquist Frequency)

5.204.3 Member Function Documentation

5.204.3.1 unsigned int MHAFilter::resampling_filter_t::fragsize_validator (unsigned int *fftlen,* unsigned int *irslen*) [static]

5.204.4 Member Data Documentation

5.204.4.1 unsigned int MHAFilter::resampling_filter_t::fragsize [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.205 MHAFilter::smoothspec_t Class Reference

Smooth spectral gains, create a windowed impulse response.

Public Member Functions

- **smoothspec_t** (unsigned int **fftlen**, unsigned int **nchannels**, const **MHAWindow::base_t** &**window**, bool **minphase**, bool linphase_asym=false)
  Constructor.
- void **smoothspec** (const **mha_spec_t** &s_in, **mha_spec_t** &s_out)
  Create a smoothed spectrum.
- void **smoothspec** (**mha_spec_t** &spec)
  Create a smoothed spectrum (in place)
- void **spec2fir** (const **mha_spec_t** &spec, **mha_wave_t** &fir)
  Return FIR coefficients.
- ~**smoothspec_t** ()

Private Member Functions

- void **internal_fir** (const **mha_spec_t** &)

Private Attributes

- unsigned int **fftlen**
- unsigned int **nchannels**
- **MHAWindow::base_t window**
- **MHASignal::waveform_t tmp_wave**
- **MHASignal::spectrum_t tmp_spec**
- **MHASignal::minphase_t** ∗ **minphase**
- bool **_linphase_asym**
- **mha_fft_t fft**

5.205.1 Detailed Description

Smooth spectral gains, create a windowed impulse response.

Spectral gains are smoothed by multiplicating the impulse response with a window function.

If a minimal phase is used, then the original phase is discarded and replaced by the minimal phase function. In this case, the window is applied to the beginning of the inverse Fourier transform of the input spectrum, and the remaining signal set to zero. If the original phase is kept, the window is applied symmetrical arround zero, i.e. to the first and last samples of the inverse Fourier transform of the input spectrum. The **spec2fir()** (p. 567) function creates a causal impulse response by circular shifting the impulse response by half of the window length.

The signal dimensions of the arguments of **smoothspec()** (p. 567) must correspond to the FFT length and number of channels provided in the constructor. The function **spec2fir()** (p. 567) can fill signal structures with more than window length frames.

5.205.2 Constructor & Destructor Documentation

5.205.2.1 MHAFilter::smoothspec_t::smoothspec_t (unsigned int *fftlen,* unsigned int *nchannels,* const **MHAWindow::base_t** & *window,* bool *minphase,* bool *linphase_asym* = `false`)

Constructor.

Parameters

fftlen	FFT length of input spectrum (fftlen/2+1 bins)
nchannels	Number of channels in input spectrum
window	Window used for smoothing
minphase	Use minimal phase (true) or original phase (false)
linphase_asym	Keep phase, but apply full window at beginning of IRS

5.205.2.2 MHAFilter::smoothspec_t::∼smoothspec_t ( )

5.205.3 Member Function Documentation

5.205.3.1 void MHAFilter::smoothspec_t::smoothspec (const mha_spec_t & *s_in,* mha_spec_t & *s_out*)

Create a smoothed spectrum.

Parameters

s_in	Input spectrum

Return values

s_out	Output spectrum

5.205.3.2 void MHAFilter::smoothspec_t::smoothspec (mha_spec_t & *spec*) [inline]

Create a smoothed spectrum (in place)

Parameters

spec	Spectrum to be smoothed.

5.205.3.3 void MHAFilter::smoothspec_t::spec2fir (const mha_spec_t & *spec,* mha_wave_t & *fir*)

Return FIR coefficients.

Parameters

spec	Input spectrum

Return values

fir	FIR coefficients, minimum length is window length

5.205.3.4 void MHAFilter::smoothspec_t::internal_fir (const mha_spec_t & *s_in*) [private]

5.205.4 Member Data Documentation

5.205.4.1 unsigned int MHAFilter::smoothspec_t::fftlen [private]

5.205.4.2 unsigned int MHAFilter::smoothspec_t::nchannels [private]

5.205.4.3 MHAWindow::base_t MHAFilter::smoothspec_t::window [private]

5.205.4.4 MHASignal::waveform_t MHAFilter::smoothspec_t::tmp_wave [private]

5.205.4.5 MHASignal::spectrum_t MHAFilter::smoothspec_t::tmp_spec [private]

5.205.4.6 MHASignal::minphase_t* MHAFilter::smoothspec_t::minphase [private]

5.205.4.7 bool MHAFilter::smoothspec_t::_linphase_asym [private]

5.205.4.8 mha_fft_t MHAFilter::smoothspec_t::fft [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.206 MHAFilter::thirdoctave_analyzer_t Class Reference

Public Member Functions

- **thirdoctave_analyzer_t** (**mhaconfig_t** cfg)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- unsigned int **nbands** ()
- unsigned int **nchannels** ()
- std::vector< **mha_real_t** > **get_cf_hz** ()

Static Public Member Functions

- static std::vector< **mha_real_t** > **cf_generator** (**mhaconfig_t** cfg)
- static std::vector< **mha_real_t** > **bw_generator** (**mhaconfig_t** cfg)
- static std::vector< **mha_real_t** > **dup** (std::vector< **mha_real_t** >, **mhaconfig_t** cfg)

Private Attributes

- **mhaconfig_t cfg_**
- std::vector< **mha_real_t** > **cf**
- **MHAFilter::gamma_flt_t fb**
- **MHASignal::waveform_t out_chunk**
- **MHASignal::waveform_t out_chunk_im**

5.206.1 Constructor & Destructor Documentation

5.206.1.1 MHAFilter::thirdoctave_analyzer_t::thirdoctave_analyzer_t (mhaconfig_t *cfg*)

5.206.2 Member Function Documentation

5.206.2.1 mha_wave_t * MHAFilter::thirdoctave_analyzer_t::process (mha_wave_t * *sln*)

5.206.2.2 unsigned int MHAFilter::thirdoctave_analyzer_t::nbands ( )

5.206.2.3 unsigned int MHAFilter::thirdoctave_analyzer_t::nchannels ( )

5.206.2.4 std::vector< mha_real_t > MHAFilter::thirdoctave_analyzer_t::get_cf_hz ( )

5.206.2.5 std::vector< mha_real_t > MHAFilter::thirdoctave_analyzer_t::cf_generator (mhaconfig_t *cfg*) [static]

5.206.2.6 std::vector< mha_real_t > MHAFilter::thirdoctave_analyzer_t::bw_generator (mhaconfig_t *cfg*) [static]

5.206.2.7 std::vector< mha_real_t > MHAFilter::thirdoctave_analyzer_t::dup (std::vector< mha_real_t > *vec,* mhaconfig_t *cfg*) [static]

5.206.3 Member Data Documentation

5.206.3.1 mhaconfig_t MHAFilter::thirdoctave_analyzer_t::cfg_ [private]

5.206.3.2 std::vector<mha_real_t> MHAFilter::thirdoctave_analyzer_t::cf [private]

5.206.3.3 MHAFilter::gamma_flt_t MHAFilter::thirdoctave_analyzer_t::fb [private]

5.206.3.4 MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk [private]

5.206.3.5 MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk_im [private]

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

- **complex_filter.h**
- **complex_filter.cpp**

5.207 MHAFilter::transfer_function_t Struct Reference

a structure containing a source channel number, a target channel number, and an impulse response.

Public Member Functions

- **transfer_function_t** ()
  Default constructor for STL conformity.
- **transfer_function_t** (unsigned int **source_channel_index**, unsigned int **target_channel_index**, const std::vector< float > &**impulse_response**)
  Data constructor.
- unsigned int **partitions** (unsigned int fragsize) const
  for the given partition size, return the number of partitions of the impulse response.
- unsigned int **non_empty_partitions** (unsigned int fragsize) const
  for the given partition size, return the number of non-empty partitions of the impulse response.
- bool **isempty** (unsigned int fragsize, unsigned int index) const
  checks if the partition contains only zeros

Public Attributes

- unsigned int **source_channel_index**
  Source audio channel index for this transfer function.
- unsigned int **target_channel_index**
  Target audio channel index for this transfer function.
- std::vector< float > **impulse_response**
  Impulse response of transfer from source to target channel.

5.207.1 Detailed Description

a structure containing a source channel number, a target channel number, and an impulse response.

5.207.2 Constructor & Destructor Documentation

5.207.2.1 MHAFilter::transfer_function_t::transfer_function_t ( ) [inline]

Default constructor for STL conformity.

Not used.

5.207.2.2 MHAFilter::transfer_function_t::transfer_function_t (unsigned int *source_channel_index,* unsigned int *target_channel_index,* const std::vector< float > & *impulse_response*)

Data constructor.

Parameters

source_channel_index	Source audio channel index for this transfer function
target_channel_index	Target audio channel index for this transfer function
impulse_response	Impulse response of transfer from source to target channel

5.207.3 Member Function Documentation

5.207.3.1 unsigned int MHAFilter::transfer_function_t::partitions (unsigned int *fragsize*) const [inline]

for the given partition size, return the number of partitions of the impulse response.

Parameters

fragsize	partition size

Returns

number of partitions occupied by the impulse response

5.207.3.2 unsigned int MHAFilter::transfer_function_t::non_empty_partitions (unsigned int *fragsize*) const [inline]

for the given partition size, return the number of non-empty partitions of the impulse response.

Parameters

fragsize	partition size

Returns

the number of non-empty partitions of the impulse response, i.e. partitions containing only zeros are not counted.

5.207.3.3 bool MHAFilter::transfer_function_t::isempty (unsigned int *fragsize,* unsigned int *index*) const [inline]

checks if the partition contains only zeros

Parameters

fragsize	partition size
index	partition index

Returns

true when this partition of the impulse response contains only zeros.

5.207.4 Member Data Documentation

5.207.4.1 unsigned int MHAFilter::transfer_function_t::source_channel_index

Source audio channel index for this transfer function.

5.207.4.2 unsigned int MHAFilter::transfer_function_t::target_channel_index

Target audio channel index for this transfer function.

5.207.4.3 std::vector<float> MHAFilter::transfer_function_t::impulse_response

Impulse response of transfer from source to target channel.

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.208 MHAFilter::transfer_matrix_t Struct Reference

A sparse matrix of transfer function partitionss.

Inherits vector< transfer_function_t >.

Public Member Functions

- std::valarray< unsigned int > **partitions** (unsigned fragsize) const

  *Returns an array of the results of calling the **partitions()** (p. 573) method on every matrix member.*
- std::valarray< unsigned int > **non_empty_partitions** (unsigned int fragsize) const

  *Returns an array of the results of calling the **non_empty_partitions()** (p. 573) method on every matrix member.*

5.208.1 Detailed Description

A sparse matrix of transfer function partitionss.

Each matrix element knows its position in the matrix, so they can be stored as a vector.

5.208.2 Member Function Documentation

5.208.2.1 std::valarray<unsigned int> MHAFilter::transfer_matrix_t::partitions (unsigned *fragsize*) const [inline]

Returns an array of the results of calling the **partitions()** (p. 573) method on every matrix member.

5.208.2.2 std::valarray<unsigned int> MHAFilter::transfer_matrix_t::non_empty_partitions (unsigned int *fragsize*) const [inline]

Returns an array of the results of calling the **non_empty_partitions()** (p. 573) method on every matrix member.

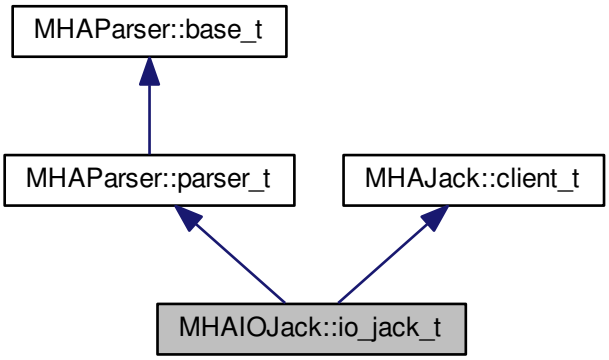
The documentation for this struct was generated from the following file:

- **mha_filter.hh**

5.209 MHAIOJack::io_jack_t Class Reference

Main class for JACK IO.

Inheritance diagram for MHAIOJack::io_jack_t:



Public Member Functions

- **io_jack_t** (unsigned int **fragsize**, float **samplerate**, **IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)
- void **prepare** (int, int)
  Allocate buffers, activate JACK client and install internal ports.
- void **release** ()

Private Member Functions

- void **reconnect_inports** ()
  Connect the input ports when connection variable is accessed.
- void **reconnect_outports** ()
  Connect the output ports when connection variable is accessed.
- void **get_physical_input_ports** ()
- void **get_physical_output_ports** ()
- void **get_all_input_ports** ()
- void **get_all_output_ports** ()
- void **get_delays_in** ()
- void **get_delays_out** ()
- void **read_get_cpu_load** ()
- void **read_get_xruns** ()
- void **read_get_scheduler** ()

Private Attributes

- unsigned int **fw_fragsize**
- float **fw_samplerate**
- **MHAParser::string_t servername**
- **MHAParser::string_t clientname**
- **MHAParser::vstring_t connections_in**
- **MHAParser::vint_mon_t delays_in**
- **MHAParser::vstring_t connections_out**
- **MHAParser::vint_mon_t delays_out**
- **MHAParser::vstring_t portnames_in**
- **MHAParser::vstring_t portnames_out**
- **MHAParser::vstring_mon_t ports_in_physical**
- **MHAParser::vstring_mon_t ports_out_physical**
- **MHAParser::vstring_mon_t ports_in_all**
- **MHAParser::vstring_mon_t ports_out_all**
- **MHAParser::parser_t ports_parser**
- **MHAParser::float_mon_t state_cpuload**
- **MHAParser::int_mon_t state_xruns**
- **MHAParser::int_mon_t state_priority**
- **MHAParser::string_mon_t state_scheduler**
- **MHAParser::parser_t state_parser**
- **MHAEvents::patchbay_t< io_jack_t > patchbay**

Additional Inherited Members

5.209.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

5.209.2 Constructor & Destructor Documentation

5.209.2.1 io_jack_t::io_jack_t (

unsigned int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

5.209.3 Member Function Documentation

5.209.3.1 void io_jack_t::prepare (

int *nch_in,*
int *nch_out*)

Allocate buffers, activate JACK client and install internal ports.

5.209.3.2 void io_jack_t::release (void)

5.209.3.3 void io_jack_t::reconnect_inports ( ) [private]

Connect the input ports when connection variable is accessed.

5.209.3.4 void io_jack_t::reconnect_outports ( ) [private]

Connect the output ports when connection variable is accessed.

5.209.3.5 void io_jack_t::get_physical_input_ports ( ) [private]

5.209.3.6 void io_jack_t::get_physical_output_ports ( ) [private]

5.209.3.7 void io_jack_t::get_all_input_ports ( ) [private]

5.209.3.8 void io_jack_t::get_all_output_ports ( ) [private]

5.209.3.9 void io_jack_t::get_delays_in ( ) [private]

5.209.3.10 void io_jack_t::get_delays_out ( ) [private]

5.209.3.11 void io_jack_t::read_get_cpu_load ( ) [private]

5.209.3.12 void io_jack_t::read_get_xruns ( ) [private]

5.209.3.13 void io_jack_t::read_get_scheduler ( ) [private]

5.209.4 Member Data Documentation

5.209.4.1 unsigned int MHAIOJack::io_jack_t::fw_fragsize [private]

5.209.4.2 float MHAIOJack::io_jack_t::fw_samplerate [private]

5.209.4.3 MHAParser::string_t MHAIOJack::io_jack_t::servername [private]

5.209.4.4 MHAParser::string_t MHAIOJack::io_jack_t::clientname [private]

5.209.4.5 MHAParser::vstring_t MHAIOJack::io_jack_t::connections_in [private]

5.209.4.6 MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_in [private]

5.209.4.7 MHAParser::vstring_t MHAIOJack::io_jack_t::connections_out [private]

5.209.4.8 MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_out [private]

5.209.4.9 MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_in [private]

5.209.4.10 MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_out [private]

5.209.4.11 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_physical [private]

5.209.4.12 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_physical [private]

5.209.4.13 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_all [private]

5.209.4.14 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_all [private]

5.209.4.15 MHAParser::parser_t MHAIOJack::io_jack_t::ports_parser [private]

5.209.4.16 MHAParser::float_mon_t MHAIOJack::io_jack_t::state_cpuload [private]

5.209.4.17 MHAParser::int_mon_t MHAIOJack::io_jack_t::state_xruns [private]

5.209.4.18 MHAParser::int_mon_t MHAIOJack::io_jack_t::state_priority [private]

5.209.4.19 MHAParser::string_mon_t MHAIOJack::io_jack_t::state_scheduler [private]

5.209.4.20 MHAParser::parser_t MHAIOJack::io_jack_t::state_parser [private]

5.209.4.21 MHAEvents::patchbay_t<io_jack_t> MHAIOJack::io_jack_t::patchbay [private]

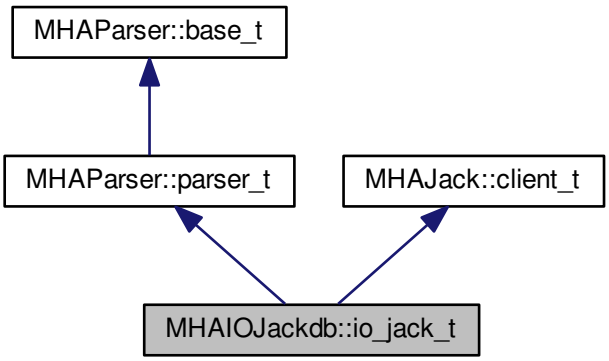
The documentation for this class was generated from the following file:

- **MHAIOJack.cpp**

5.210 MHAIOJackdb::io_jack_t Class Reference

Main class for JACK IO.

Inheritance diagram for MHAIOJackdb::io_jack_t:



Public Member Functions

- **io_jack_t** (unsigned int **fragsize**, float **samplerate**, **IOProcessEvent_t proc_event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IOStoppedEvent_t stop_event**, void ∗**stop_handle**)
- void **prepare** (int, int)
  Allocate buffers, activate JACK client and install internal ports.
- void **release** ()
- bool **fail_on_async_jackerror** () const

Private Member Functions

- int **IOProcessEvent_inner** (**mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- void **reconnect_inports** ()
  Connect the input ports when connection variable is accessed.
- void **reconnect_outports** ()
  Connect the output ports when connection variable is accessed.
- void **get_physical_input_ports** ()
- void **get_physical_output_ports** ()
- void **get_all_input_ports** ()
- void **get_all_output_ports** ()
- void **read_get_cpu_load** ()
- void **read_get_xruns** ()
- void **read_get_scheduler** ()
- void **set_use_jack_transport** ()
- void **set_locate** ()

Static Private Member Functions

- static int **IOProcessEvent_inner** (void *handle, **mha_wave_t** *sIn, **mha_wave_t** **s↩ Out)

Private Attributes

- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- unsigned int **mha_fragsize**
- float **mha_samplerate**
- unsigned int **fragsize_ratio**
- **MHAParser::string_t servername**
- **MHAParser::string_t clientname**
- **MHAParser::vstring_t connections_in**
- **MHAParser::vstring_t connections_out**
- **MHAParser::vstring_t portnames_in**
- **MHAParser::vstring_t portnames_out**
- **MHAParser::bool_t fail_on_async_jackerr**
- **MHAParser::bool_t use_jack_transport**
- **MHAParser::float_t locate**
- **MHAParser::float_mon_t server_srate**
- **MHAParser::int_mon_t server_fragsize**
- **MHAParser::vstring_mon_t ports_in_physical**
- **MHAParser::vstring_mon_t ports_out_physical**
- **MHAParser::vstring_mon_t ports_in_all**
- **MHAParser::vstring_mon_t ports_out_all**
- **MHAParser::parser_t ports_parser**
- **MHAParser::float_mon_t state_cpuload**
- **MHAParser::int_mon_t state_xruns**
- **MHAParser::int_mon_t state_priority**
- **MHAParser::string_mon_t state_scheduler**
- **MHAParser::parser_t state_parser**
- **MHASignal::waveform_t** * **pwinner_out**
- **MHAEvents::patchbay_t**< **io_jack_t** > **patchbay**

Additional Inherited Members

5.210.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

5.210.2 Constructor & Destructor Documentation

5.210.2.1 io_jack_t::io_jack_t (

unsigned int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle*)

5.210.3 Member Function Documentation

5.210.3.1 void io_jack_t::prepare (

int *nch_in,*
int *nch_out*)

Allocate buffers, activate JACK client and install internal ports.

5.210.3.2 void io_jack_t::release (

void)

5.210.3.3 bool MHAIOJackdb::io_jack_t::fail_on_async_jackerror ( ) const [inline]

5.210.3.4 int io_jack_t::IOProcessEvent_inner (

void * *handle,*
mha_wave_t * *sIn,*
mha_wave_t ** *sOut*) [static], [private]

5.210.3.5 int io_jack_t::IOProcessEvent_inner (

mha_wave_t * *sIn,*
mha_wave_t ** *sOut*) [private]

5.210.3.6 void io_jack_t::reconnect_inports ( ) [private]

Connect the input ports when connection variable is accessed.

5.210.3.7 void io_jack_t::reconnect_outports ( ) [private]

Connect the output ports when connection variable is accessed.

5.210.3.8 void io_jack_t::get_physical_input_ports ( ) [private]

5.210.3.9 void io_jack_t::get_physical_output_ports ( ) [private]

5.210.3.10 void io_jack_t::get_all_input_ports ( ) [private]

5.210.3.11 void io_jack_t::get_all_output_ports ( ) [private]

5.210.3.12 void io_jack_t::read_get_cpu_load ( ) [private]

5.210.3.13 void io_jack_t::read_get_xruns ( ) [private]

5.210.3.14 void io_jack_t::read_get_scheduler ( ) [private]

5.210.3.15 void io_jack_t::set_use_jack_transport ( ) [private]

5.210.3.16 void io_jack_t::set_locate ( ) [private]

5.210.4 Member Data Documentation

5.210.4.1 IOProcessEvent_t MHAIOJackdb::io_jack_t::proc_event [private]

5.210.4.2 void* MHAIOJackdb::io_jack_t::proc_handle [private]

5.210.4.3 unsigned int MHAIOJackdb::io_jack_t::mha_fragsize [private]

5.210.4.4 float MHAIOJackdb::io_jack_t::mha_samplerate [private]

5.210.4.5 unsigned int MHAIOJackdb::io_jack_t::fragsize_ratio [private]

5.210.4.6 MHAParser::string_t MHAIOJackdb::io_jack_t::servername [private]

5.210.4.7 MHAParser::string_t MHAIOJackdb::io_jack_t::clientname [private]

5.210.4.8 MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_in [private]

5.210.4.9 MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_out [private]

5.210.4.10 MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_in [private]

5.210.4.11 MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_out [private]

5.210.4.12 MHAParser::bool_t MHAIOJackdb::io_jack_t::fail_on_async_jackerr [private]

5.210.4.13 MHAParser::bool_t MHAIOJackdb::io_jack_t::use_jack_transport [private]

5.210.4.14 MHAParser::float_t MHAIOJackdb::io_jack_t::locate [private]

5.210.4.15 MHAParser::float_mon_t MHAIOJackdb::io_jack_t::server_srate [private]

5.210.4.16 MHAParser::int_mon_t MHAIOJackdb::io_jack_t::server_fragsize [private]

5.210.4.17 MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_in_physical [private]

5.210.4.18 MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_out_physical [private]

5.210.4.19 MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_in_all [private]

5.210.4.20 MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_out_all [private]

5.210.4.21 MHAParser::parser_t MHAIOJackdb::io_jack_t::ports_parser [private]

5.210.4.22 MHAParser::float_mon_t MHAIOJackdb::io_jack_t::state_cpuload [private]

5.210.4.23 MHAParser::int_mon_t MHAIOJackdb::io_jack_t::state_xruns [private]

5.210.4.24 MHAParser::int_mon_t MHAIOJackdb::io_jack_t::state_priority [private]

5.210.4.25 MHAParser::string_mon_t MHAIOJackdb::io_jack_t::state_scheduler [private]

5.210.4.26 MHAParser::parser_t MHAIOJackdb::io_jack_t::state_parser [private]

5.210.4.27 MHASignal::waveform_t* MHAIOJackdb::io_jack_t::pwinner_out [private]

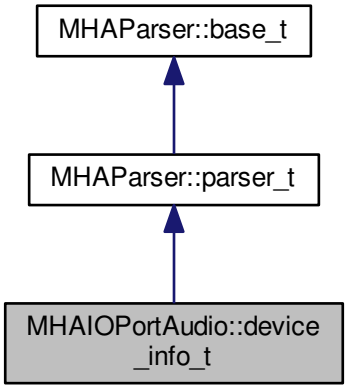
5.210.4.28 MHAEvents::patchbay_t<io_jack_t> MHAIOJackdb::io_jack_t::patchbay [private]

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

- **MHAIOJackdb.cpp**

5.211 MHAIOPortAudio::device_info_t Class Reference

Inheritance diagram for MHAIOPortAudio::device_info_t:



Public Member Functions

- **device_info_t** ()
- void **fill_info** ()

Public Attributes

- **MHAParser::int_mon_t numDevices**
- **MHAParser::vint_mon_t structVersion**
- **MHAParser::vstring_mon_t name**
- **MHAParser::vint_mon_t hostApi**
- **MHAParser::vint_mon_t maxInputChannels**
- **MHAParser::vint_mon_t maxOutputChannels**
- **MHAParser::vfloat_mon_t defaultLowInputLatency**
- **MHAParser::vfloat_mon_t defaultLowOutputLatency**
- **MHAParser::vfloat_mon_t defaultHighInputLatency**
- **MHAParser::vfloat_mon_t defaultHighOutputLatency**
- **MHAParser::vfloat_mon_t defaultSampleRate**

Additional Inherited Members

5.211.1 Constructor & Destructor Documentation

5.211.1.1 MHAIOPortAudio::device_info_t::device_info_t ( ) [inline]

5.211.2 Member Function Documentation

5.211.2.1 void MHAIOPortAudio::device_info_t::fill_info ( ) [inline]

5.211.3 Member Data Documentation

5.211.3.1 MHAParser::int_mon_t MHAIOPortAudio::device_info_t::numDevices

5.211.3.2 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::structVersion

5.211.3.3 MHAParser::vstring_mon_t MHAIOPortAudio::device_info_t::name

5.211.3.4 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::hostApi

5.211.3.5 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxInputChannels

5.211.3.6 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxOutputChannels

5.211.3.7 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowInputLatency

5.211.3.8 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowOutputLatency

5.211.3.9 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighInputLatency

5.211.3.10 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighOutputLatency

5.211.3.11 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultSampleRate

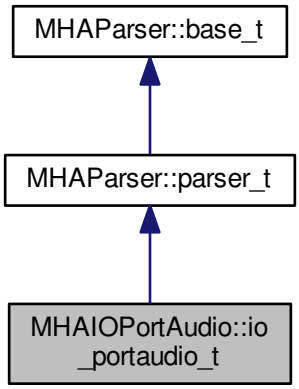
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.212 MHAIOPortAudio::io_portaudio_t Class Reference

Main class for Portaudio sound IO.

Inheritance diagram for MHAIOPortAudio::io_portaudio_t:



Public Member Functions

- **io_portaudio_t** (unsigned int **fragsize**, float **samplerate**, **IOProcessEvent_t proc_↩ event**, void ∗**proc_handle**, **IOStartedEvent_t start_event**, void ∗**start_handle**, **IO↩ StoppedEvent_t stop_event**, void ∗**stop_handle**)
- void **device_name_updated** ()
- void **device_index_updated** ()
- ∼**io_portaudio_t** ()
- void **cmd_prepare** (int, int)
- void **cmd_start** ()
- void **cmd_stop** ()
- void **cmd_release** ()
- int **portaudio_callback** (const void ∗input, void ∗output, unsigned long frame_count, const PaStreamCallbackTimeInfo ∗time_info, PaStreamCallbackFlags status_flags)

Private Attributes

- **device_info_t device_info**
- **MHASignal::waveform_t** ∗ **s_in**
- **mha_wave_t** ∗ **s_out**
- float **samplerate**
- unsigned int **nchannels_out**
- unsigned int **nchannels_in**

- unsigned int **fragsize**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- PaStream * **portaudio_stream**
- **MHAParser::string_t device_name**
- **MHAParser::int_t device_index**
- **MHAEvents::patchbay_t< io_portaudio_t > patchbay**

Additional Inherited Members

5.212.1 Detailed Description

Main class for Portaudio sound IO.

5.212.2 Constructor & Destructor Documentation

5.212.2.1 MHAIOPortAudio::io_portaudio_t::io_portaudio_t (unsigned int *fragsize,* float *samplerate,* **IOProcessEvent_t** *proc_event,* void * *proc_handle,* **IOStartedEvent_t** *start_event,* void * *start_handle,* **IOStoppedEvent_t** *stop_event,* void * *stop_handle*) [inline]

5.212.2.2 MHAIOPortAudio::io_portaudio_t::~io_portaudio_t ( ) [inline]

5.212.3 Member Function Documentation

5.212.3.1 void MHAIOPortAudio::io_portaudio_t::device_name_updated ( ) [inline]

5.212.3.2 void MHAIOPortAudio::io_portaudio_t::device_index_updated ( ) [inline]

5.212.3.3 void MHAIOPortAudio::io_portaudio_t::cmd_prepare (int *nchannels_in,* int *nchannels_out*)

5.212.3.4 void MHAIOPortAudio::io_portaudio_t::cmd_start ( )

5.212.3.5 void MHAIOPortAudio::io_portaudio_t::cmd_stop ( )

5.212.3.6 void MHAIOPortAudio::io_portaudio_t::cmd_release ( )

5.212.3.7 int MHAIOPortAudio::io_portaudio_t::portaudio_callback (

const void * *input,*
void * *output,*
unsigned long *frame_count,*
const PaStreamCallbackTimeInfo * *time_info,*
PaStreamCallbackFlags *status_flags*)

5.212.4 Member Data Documentation

5.212.4.1 device_info_t MHAIOPortAudio::io_portaudio_t::device_info [private]

5.212.4.2 MHASignal::waveform_t* MHAIOPortAudio::io_portaudio_t::s_in [private]

5.212.4.3 mha_wave_t* MHAIOPortAudio::io_portaudio_t::s_out [private]

5.212.4.4 float MHAIOPortAudio::io_portaudio_t::samplerate [private]

5.212.4.5 unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_out [private]

5.212.4.6 unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_in [private]

5.212.4.7 unsigned int MHAIOPortAudio::io_portaudio_t::fragsize [private]

5.212.4.8 IOProcessEvent_t MHAIOPortAudio::io_portaudio_t::proc_event [private]

5.212.4.9 void* MHAIOPortAudio::io_portaudio_t::proc_handle [private]

5.212.4.10 IOStartedEvent_t MHAIOPortAudio::io_portaudio_t::start_event [private]

5.212.4.11 void* MHAIOPortAudio::io_portaudio_t::start_handle [private]

5.212.4.12 IOStoppedEvent_t MHAIOPortAudio::io_portaudio_t::stop_event [private]

5.212.4.13 void* MHAIOPortAudio::io_portaudio_t::stop_handle [private]

5.212.4.14 PaStream* MHAIOPortAudio::io_portaudio_t::portaudio_stream [private]

5.212.4.15 MHAParser::string_t MHAIOPortAudio::io_portaudio_t::device_name [private]

5.212.4.16 MHAParser::int_t MHAIOPortAudio::io_portaudio_t::device_index [private]

5.212.4.17 MHAEvents::patchbay_t<io_portaudio_t> MHAIOPortAudio::io_portaudio_t↩ ::patchbay [private]

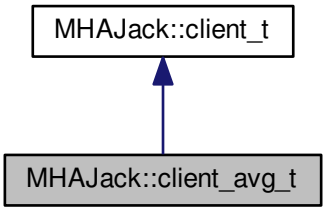
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.213 MHAJack::client_avg_t Class Reference

Generic JACK client for averaging a system response across time.

Inheritance diagram for MHAJack::client_avg_t:



Public Member Functions

- **client_avg_t** (const std::string &**name**, const unsigned int &nrep_)
  Constructor for averaging client.
- void **io** (**mha_wave_t** ∗**s_out**, **mha_wave_t** ∗**s_in**, const std::vector< std::string > &p_out, const std::vector< std::string > &p_in, float ∗srate=NULL, unsigned int ∗**fragsize**=NULL)
  Recording function.

Private Member Functions

- void **proc** (**mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- void **IOStoppedEvent** ()

Static Private Member Functions

- static int **proc** (void ∗handle, **mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- static void **IOStoppedEvent** (void ∗handle, int proc_err, int io_err)

Private Attributes

- bool **b_stopped**
- unsigned int **pos**
- **mha_wave_t** ∗ **sn_in**
- **mha_wave_t** ∗ **sn_out**
- std::string **name**
- **MHASignal::waveform_t** ∗ **frag_out**
- const unsigned int **nrep**
- unsigned int **n**
- bool **b_ready**

Additional Inherited Members

5.213.1 Detailed Description

Generic JACK client for averaging a system response across time.

5.213.2 Constructor & Destructor Documentation

5.213.2.1 MHAJack::client_avg_t::client_avg_t (const std::string & *name_*, const unsigned int & *nrep_*)

Constructor for averaging client.

Parameters

*name_*	Name of JACK client
*nrep_*	Number of repetitions

5.213.3 Member Function Documentation

5.213.3.1 void MHAJack::client_avg_t::io (mha_wave_t * *is_out*, mha_wave_t * *is_in*, const std::vector< std::string > & *p_out*, const std::vector< std::string > & *p_in*, float * *srate* = NULL, unsigned int * *fragsize* = NULL)

Recording function.

long-description

Parameters

is_out	Input (test) signal, which will be repeated
is_in	System response (averaged, same length as input required)
p_out	Ports to play back the test signal
p_in	Ports to record from the system response
srate	Pointer to sampling rate variable, will be filled with server sampling rate
fragsize	Pointer to fragment size variable, will be filled with server fragment size

5.213.3.2 int MHAJack::client_avg_t::proc (void * *handle,* mha_wave_t * *sIn,* mha_wave_t ** *sOut*) [static], [private]

5.213.3.3 void MHAJack::client_avg_t::IOStoppedEvent (void * *handle,* int *proc_err,* int *io_err*) [static], [private]

5.213.3.4 void MHAJack::client_avg_t::proc (mha_wave_t * *sIn,* mha_wave_t ** *sOut*) [private]

5.213.3.5 void MHAJack::client_avg_t::IOStoppedEvent ( ) [private]

5.213.4 Member Data Documentation

5.213.4.1 bool MHAJack::client_avg_t::b_stopped [private]

5.213.4.2 unsigned int MHAJack::client_avg_t::pos [private]

5.213.4.3 mha_wave_t* MHAJack::client_avg_t::sn_in [private]

5.213.4.4 mha_wave_t* MHAJack::client_avg_t::sn_out [private]

5.213.4.5 std::string MHAJack::client_avg_t::name [private]

5.213.4.6 MHASignal::waveform_t* MHAJack::client_avg_t::frag_out [private]

5.213.4.7 const unsigned int MHAJack::client_avg_t::nrep [private]

5.213.4.8 unsigned int MHAJack::client_avg_t::n [private]

5.213.4.9 bool MHAJack::client_avg_t::b_ready [private]

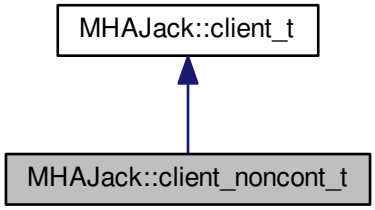
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

5.214 MHAJack::client_noncont_t Class Reference

Generic client for synchronous playback and recording of waveform fragments.

Inheritance diagram for MHAJack::client_noncont_t:



Public Member Functions

- **client_noncont_t** (const std::string &**name**, bool **use_jack_transport**=false)
- void **io** (**mha_wave_t** ∗**s_out**, **mha_wave_t** ∗**s_in**, const std::vector< std::string > &p_out, const std::vector< std::string > &p_in, float ∗srate=NULL, unsigned int ∗**fragsize**=NULL)

Private Member Functions

- void **proc** (**mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- void **IOStoppedEvent** ()

Static Private Member Functions

- static int **proc** (void ∗handle, **mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- static void **IOStoppedEvent** (void ∗handle, int proc_err, int io_err)

Private Attributes

- bool **b_stopped**
- unsigned int **pos**
- **mha_wave_t** ∗ **sn_in**
- **mha_wave_t** ∗ **sn_out**
- std::string **name**
- **MHASignal::waveform_t** ∗ **frag_out**

Additional Inherited Members

5.214.1 Detailed Description

Generic client for synchronous playback and recording of waveform fragments.

5.214.2 Constructor & Destructor Documentation

5.214.2.1 MHAJack::client_noncont_t::client_noncont_t (const std::string & *name,* bool *use_jack_transport* = `false`)

5.214.3 Member Function Documentation

5.214.3.1 void MHAJack::client_noncont_t::io (mha_wave_t * *s_out,* mha_wave_t * *s_in,* const std::vector< std::string > & *p_out,* const std::vector< std::string > & *p_in,* float * *srate* = `NULL`, unsigned int * *fragsize* = `NULL`)

5.214.3.2 int MHAJack::client_noncont_t::proc (void * *handle,* mha_wave_t * *sIn,* mha_wave_t ** *sOut*) `[static]`, `[private]`

5.214.3.3 void MHAJack::client_noncont_t::IOStoppedEvent (void * *handle,* int *proc_err,* int *io_err*) `[static]`, `[private]`

5.214.3.4 void MHAJack::client_noncont_t::proc (mha_wave_t * *sIn,* mha_wave_t ** *sOut*) `[private]`

5.214.3.5 void MHAJack::client_noncont_t::IOStoppedEvent ( ) `[private]`

5.214.4 Member Data Documentation

5.214.4.1 bool MHAJack::client_noncont_t::b_stopped `[private]`

5.214.4.2 unsigned int MHAJack::client_noncont_t::pos `[private]`

5.214.4.3 mha_wave_t* MHAJack::client_noncont_t::sn_in [private]

5.214.4.4 mha_wave_t* MHAJack::client_noncont_t::sn_out [private]

5.214.4.5 std::string MHAJack::client_noncont_t::name [private]

5.214.4.6 MHASignal::waveform_t* MHAJack::client_noncont_t::frag_out [private]

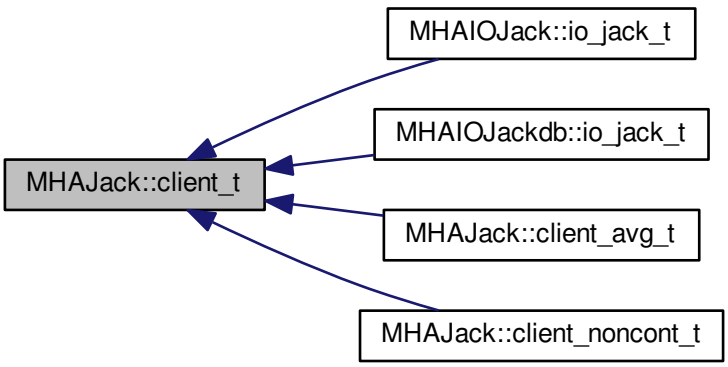
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

5.215 MHAJack::client_t Class Reference

Generic asynchronous JACK client.

Inheritance diagram for MHAJack::client_t:



Public Member Functions

- **client_t** (**IOProcessEvent_t proc_event**, void *__proc_handle__=NULL, **IOStartedEvent_t start_event**=NULL, void *__start_handle__=NULL, **IOStoppedEvent_t stop_event**=NULL, void *__stop_handle__=NULL, bool **use_jack_transport**=false)
- void **prepare** (const std::string &client_name, const unsigned int &**nchannels_in**, const unsigned int &**nchannels_out**)

  Allocate buffers, activate JACK client and install internal ports.

- void **prepare** (const std::string &server_name, const std::string &client_name, const unsigned int &**nchannels_in**, const unsigned int &**nchannels_out**)

  Allocate buffers, ports, and activates JACK client.
- void **release** ()

  Remove JACK client and deallocate internal ports and buffers.
- void **start** (bool fail_on_async_jack_error=true)
- void **stop** ()
- void **connect_input** (const std::vector< std::string > &)

  Connect the input ports when connection variable is accessed.
- void **connect_output** (const std::vector< std::string > &)

  Connect the output ports when connection variable is accessed.
- unsigned int **get_fragsize** () const
- float **get_srate** () const
- unsigned long **get_xruns** ()
- unsigned long **get_xruns_reset** ()
- std::string **str_error** (int err)
- void **get_ports** (std::vector< std::string > &, unsigned long jack_flags)

  Get a list of Jack ports.
- std::vector< std::string > **get_my_input_ports** ()
- std::vector< std::string > **get_my_output_ports** ()
- void **set_input_portnames** (const std::vector< std::string > &)
- void **set_output_portnames** (const std::vector< std::string > &)
- float **get_cpu_load** ()
- void **set_use_jack_transport** (bool ut)

Protected Attributes

- jack_client_t * **jc**

Private Member Functions

- void **prepare_impl** (const char *server_name, const char *client_name, const unsigned int &**nchannels_in**, const unsigned int &**nchannels_out**)

  Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.
- void **internal_start** ()
- void **internal_stop** ()
- void **stopped** (int, int)
- int **jack_proc_cb** (jack_nframes_t)

  This is the main processing callback.
- int **jack_xrun_cb** ()

Static Private Member Functions

- static int **jack_proc_cb** (jack_nframes_t, void *)
- static int **jack_xrun_cb** (void *)

Private Attributes

- unsigned long **num_xruns**
- unsigned int **fragsize**
- float **samplerate**
- unsigned int **nchannels_in**
- unsigned int **nchannels_out**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- **MHASignal::waveform_t * s_in**
- **mha_wave_t * s_out**
- **MHAJack::port_t ** inch**
- **MHAJack::port_t ** outch**
- unsigned int **flags**
- bool **b_prepared**
- bool **use_jack_transport**
- jack_transport_state_t **jstate_prev**
- std::vector< std::string > **input_portnames**
- std::vector< std::string > **output_portnames**
- bool **fail_on_async_jackerror**

5.215.1 Detailed Description

Generic asynchronous JACK client.

5.215.2 Constructor & Destructor Documentation

5.215.2.1 MHAJack::client_t::client_t (

IOProcessEvent_t ***proc_event,***
void * ***proc_handle =*** `NULL`**,**
IOStartedEvent_t ***start_event =*** `NULL`**,**
void * ***start_handle =*** `NULL`**,**
IOStoppedEvent_t ***stop_event =*** `NULL`**,**
void * ***stop_handle =*** `NULL`**,**
bool ***use_jack_transport =*** `false` **)**

5.215.3 Member Function Documentation

5.215.3.1 void MHAJack::client_t::prepare (

const std::string & ***client_name,***
const unsigned int & ***nch_in,***
const unsigned int & ***nch_out*** **)**

Allocate buffers, activate JACK client and install internal ports.

Registers the jack client with the default jack server and activates it.

Parameters

client_name	Name of this jack client
nch_in	Input ports to register
nch_out	Output ports to register

5.215.3.2 void MHAJack::client_t::prepare (const std::string & *server_name,* const std::string & *client_name,* const unsigned int & *nch_in,* const unsigned int & *nch_out*)

Allocate buffers, ports, and activates JACK client.

Registers the jack client with specified jack server and activates it.

Parameters

server_name	Name of the jack server to register with
client_name	Name of this jack client
nch_in	Input ports to register
nch_out	Output ports to register

5.215.3.3 void MHAJack::client_t::release (void)

Remove JACK client and deallocate internal ports and buffers.

5.215.3.4 void MHAJack::client_t::start (bool *fail_on_async_jack_error* = `true`)

5.215.3.5 void MHAJack::client_t::stop ( )

5.215.3.6 void MHAJack::client_t::connect_input (const std::vector< std::string > & *con*)

Connect the input ports when connection variable is accessed.

5.215.3.7 void MHAJack::client_t::connect_output (const std::vector< std::string > & *con*)

Connect the output ports when connection variable is accessed.

5.215.3.8 unsigned int MHAJack::client_t::get_fragsize ( ) const [inline]

5.215.3.9 float MHAJack::client_t::get_srate ( ) const [inline]

5.215.3.10 unsigned long MHAJack::client_t::get_xruns ( ) [inline]

5.215.3.11 unsigned long MHAJack::client_t::get_xruns_reset ( )

5.215.3.12 std::string MHAJack::client_t::str_error (int *err*)

5.215.3.13 void MHAJack::client_t::get_ports (std::vector< std::string > & *res,* unsigned long *jack_flags*)

Get a list of Jack ports.

Parameters

res	Result string vector
jack_flags	Jack port flags (JackPortInput etc.)

5.215.3.14 std::vector< std::string > MHAJack::client_t::get_my_input_ports ( )

5.215.3.15 std::vector< std::string > MHAJack::client_t::get_my_output_ports ( )

5.215.3.16 void MHAJack::client_t::set_input_portnames (const std::vector< std::string > & *names*)

5.215.3.17 void MHAJack::client_t::set_output_portnames (const std::vector< std::string > & *names*)

5.215.3.18 float MHAJack::client_t::get_cpu_load ( )

5.215.3.19 void MHAJack::client_t::set_use_jack_transport (bool *ut*) [inline]

5.215.3.20 void MHAJack::client_t::prepare_impl (const char * *server_name,* const char * *client_name,* const unsigned int & *nch_in,* const unsigned int & *nch_out*) [private]

Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.

Parameters

server_name	Name of the jack server to register with
client_name	Name of this jack client
nch_in	Input ports to register
nch_out	Output ports to register

5.215.3.21 void MHAJack::client_t::internal_start ( ) [private]

5.215.3.22 void MHAJack::client_t::internal_stop ( ) [private]

5.215.3.23 void MHAJack::client_t::stopped (int *proc_err,* int *io_err*) [private]

5.215.3.24 int MHAJack::client_t::jack_proc_cb (jack_nframes_t *n,* void * *h*) [static], [private]

5.215.3.25 int MHAJack::client_t::jack_proc_cb (jack_nframes_t *n*) [private]

This is the main processing callback.

Here happens double buffering and downsampling.

5.215.3.26 int MHAJack::client_t::jack_xrun_cb (void * *h*) [static], [private]

5.215.3.27 int MHAJack::client_t::jack_xrun_cb ( ) [inline], [private]

5.215.4 Member Data Documentation

5.215.4.1 unsigned long MHAJack::client_t::num_xruns [private]

5.215.4.2 unsigned int MHAJack::client_t::fragsize [private]

5.215.4.3 float MHAJack::client_t::samplerate [private]

5.215.4.4 unsigned int MHAJack::client_t::nchannels_in [private]

5.215.4.5 unsigned int MHAJack::client_t::nchannels_out [private]

5.215.4.6 IOProcessEvent_t MHAJack::client_t::proc_event [private]

5.215.4.7 void* MHAJack::client_t::proc_handle [private]

5.215.4.8 IOStartedEvent_t MHAJack::client_t::start_event [private]

5.215.4.9 void* MHAJack::client_t::start_handle [private]

5.215.4.10 IOStoppedEvent_t MHAJack::client_t::stop_event [private]

5.215.4.11 void* MHAJack::client_t::stop_handle [private]

5.215.4.12 MHASignal::waveform_t* MHAJack::client_t::s_in [private]

5.215.4.13 mha_wave_t* MHAJack::client_t::s_out [private]

5.215.4.14 MHAJack::port_t** MHAJack::client_t::inch [private]

5.215.4.15 MHAJack::port_t** MHAJack::client_t::outch [private]

5.215.4.16 jack_client_t* MHAJack::client_t::jc [protected]

5.215.4.17 unsigned int MHAJack::client_t::flags [private]

5.215.4.18 bool MHAJack::client_t::b_prepared [private]

5.215.4.19 bool MHAJack::client_t::use_jack_transport [private]

5.215.4.20 jack_transport_state_t MHAJack::client_t::jstate_prev [private]

5.215.4.21 std::vector<std::string> MHAJack::client_t::input_portnames [private]

5.215.4.22 std::vector<std::string> MHAJack::client_t::output_portnames [private]

5.215.4.23 bool MHAJack::client_t::fail_on_async_jackerror [private]

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

- **mhajack.h**
- **mhajack.cpp**

5.216 MHAJack::port_t Class Reference

Class for one channel/port.

Public Types

Public Member Functions

- **port_t** (jack_client_t ∗**jc**, **dir_t** dir, int id)
- **port_t** (jack_client_t ∗**jc**, **dir_t** dir, const std::string &id)
  Constructor to create port with specific name.
- ∼**port_t** ()
- void **read** (**mha_wave_t** ∗s, unsigned int ch)
- void **write** (**mha_wave_t** ∗s, unsigned int ch)
- void **mute** (unsigned int n)
- void **connect_to** (const char ∗pn)
- const char ∗ **get_short_name** ()
  Return the port name.

Private Attributes

- **dir_t dir_type**
- jack_port_t ∗ **port**
- jack_default_audio_sample_t ∗ **iob**
- jack_client_t ∗ **jc**

5.216.1 Detailed Description

Class for one channel/port.

This class represents one JACK port. Double buffering for asynchronous process callbacks is managed by this class.

5.216.2 Member Enumeration Documentation

5.216.2.1 enum MHAJack::port_t::dir_t

Enumerator

input

output

5.216.3 Constructor & Destructor Documentation

5.216.3.1 MHAJack::port_t::port_t (jack_client_t ∗ *jc,* dir_t *dir,* int *id*)

Parameters

jc	JACK client.
dir	Direction (input/output).
id	Number in port name (starting with 1).

5.216.3.2 MHAJack::port_t::port_t (jack_client_t * *jc*, dir_t *dir*, const std::string & *id*)

Constructor to create port with specific name.

Parameters

jc	JACK client.
dir	Direction (input/output).
id	Port name.

5.216.3.3 MHAJack::port_t::~port_t ( )

5.216.4 Member Function Documentation

5.216.4.1 void MHAJack::port_t::read (mha_wave_t * *s*, unsigned int *ch*)

Parameters

s	Signal structure to store the audio data.
ch	Channel number in audio data structure to be used.

5.216.4.2 void MHAJack::port_t::write (mha_wave_t * *s*, unsigned int *ch*)

Parameters

s	Signal structure from which the audio data is read.
ch	Channel number in audio data structure to be used.

5.216.4.3 void MHAJack::port_t::mute (unsigned int *n*)

Parameters

n	Number of samples to be muted (must be the same as reported by Jack processing callback).

5.216.4.4 void MHAJack::port_t::connect_to (const char * *pn*)

Parameters

pn	Port name to connect to

5.216.4.5 const char * MHAJack::port_t::get_short_name ( )

Return the port name.

5.216.5 Member Data Documentation

5.216.5.1 dir_t MHAJack::port_t::dir_type [private]

5.216.5.2 jack_port_t* MHAJack::port_t::port [private]

5.216.5.3 jack_default_audio_sample_t* MHAJack::port_t::iob [private]

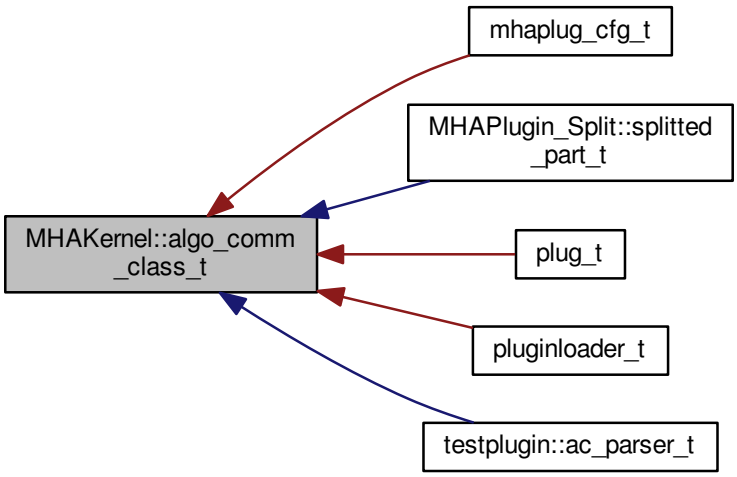
5.216.5.4 jack_client_t* MHAJack::port_t::jc [private]

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

- **mhajack.h**
- **mhajack.cpp**

5.217 MHAKernel::algo_comm_class_t Class Reference

Inheritance diagram for MHAKernel::algo_comm_class_t:



Public Member Functions

- **algo_comm_class_t** ()
- virtual ~**algo_comm_class_t** ()
- **algo_comm_t get_c_handle** ()
- virtual void **local_insert_var** (const char *, **comm_var_t**)
- virtual void **local_remove_var** (const char *)
- virtual void **local_remove_ref** (void *)
- virtual bool **local_is_var** (const char *)
- virtual void **local_get_var** (const char *, **comm_var_t** *)
- virtual std::string **local_get_entries** ()
- virtual comm_var_map_t::size_type **size** () const

Static Public Member Functions

- static int **insert_var** (void *, const char *, **comm_var_t**)
- static int **insert_var_int** (void *, const char *, int *)
- static int **insert_var_float** (void *, const char *, float *)
- static int **remove_var** (void *, const char *)
- static int **remove_ref** (void *, void *)
- static int **is_var** (void *, const char *)
- static int **get_var** (void *, const char *, **comm_var_t** *)
- static int **get_var_int** (void *, const char *, int *)
- static int **get_var_float** (void *, const char *, float *)
- static int **get_entries** (void *, char *, unsigned int)
- static const char * **get_error** (int)

Public Attributes

- char * **algo_comm_id_string**

Private Attributes

- **algo_comm_t ac**
- int **algo_comm_id_string_len**
- **comm_var_map_t vars**

5.217.1 Constructor & Destructor Documentation

5.217.1.1 MHAKernel::algo_comm_class_t::algo_comm_class_t ( )

5.217.1.2 MHAKernel::algo_comm_class_t::~algo_comm_class_t ( ) [virtual]

5.217.2 Member Function Documentation

5.217.2.1 algo_comm_t MHAKernel::algo_comm_class_t::get_c_handle ( )

5.217.2.2 int MHAKernel::algo_comm_class_t::insert_var (void * *handle*, const char * *name*, comm_var_t *var*) [static]

5.217.2.3 int MHAKernel::algo_comm_class_t::insert_var_int (void * *handle*, const char * *name*, int * *ivar*) [static]

5.217.2.4 int MHAKernel::algo_comm_class_t::insert_var_float (void * *handle*, const char * *name*, float * *ivar*) [static]

5.217.2.5 int MHAKernel::algo_comm_class_t::remove_var (void * *handle*, const char * *name*) [static]

5.217.2.6 int MHAKernel::algo_comm_class_t::remove_ref (void * *handle*, void * *ref*) [static]

5.217.2.7 int MHAKernel::algo_comm_class_t::is_var (void * *handle,* const char * *name*) [static]

5.217.2.8 int MHAKernel::algo_comm_class_t::get_var (void * *handle,* const char * *name,* comm_var_t * *var*) [static]

5.217.2.9 int MHAKernel::algo_comm_class_t::get_var_int (void * *handle,* const char * *name,* int * *ivar*) [static]

5.217.2.10 int MHAKernel::algo_comm_class_t::get_var_float (void * *handle,* const char * *name,* float * *ivar*) [static]

5.217.2.11 int MHAKernel::algo_comm_class_t::get_entries (void * *handle,* char * *ret,* unsigned int *len*) [static]

5.217.2.12 const char * MHAKernel::algo_comm_class_t::get_error (int *e*) [static]

5.217.2.13 void MHAKernel::algo_comm_class_t::local_insert_var (const char * *name,* comm_var_t *var*) [virtual]

5.217.2.14 void MHAKernel::algo_comm_class_t::local_remove_var (const char * *name*) [virtual]

5.217.2.15 void MHAKernel::algo_comm_class_t::local_remove_ref (void * *addr*) [virtual]

5.217.2.16 bool MHAKernel::algo_comm_class_t::local_is_var (const char * *name*) [virtual]

5.217.2.17 void MHAKernel::algo_comm_class_t::local_get_var (const char * *name,* comm_var_t * *var*) [virtual]

5.217.2.18 std::string MHAKernel::algo_comm_class_t::local_get_entries ( ) [virtual]

5.217.2.19 MHAKernel::comm_var_map_t::size_type MHAKernel::algo_comm_class_t::size ( ) const [virtual]

5.217.3 Member Data Documentation

5.217.3.1 char* MHAKernel::algo_comm_class_t::algo_comm_id_string

5.217.3.2 algo_comm_t MHAKernel::algo_comm_class_t::ac [private]

5.217.3.3 int MHAKernel::algo_comm_class_t::algo_comm_id_string_len [private]

5.217.3.4 comm_var_map_t MHAKernel::algo_comm_class_t::vars [private]

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

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.218 MHAKernel::comm_var_map_t Class Reference

Inherits map< std::string, comm_var_t >.

Public Member Functions

- bool **has_key** (const std::string &name)

5.218.1 Member Function Documentation

5.218.1.1 bool MHAKernel::comm_var_map_t::has_key (const std::string & *name*) [inline]

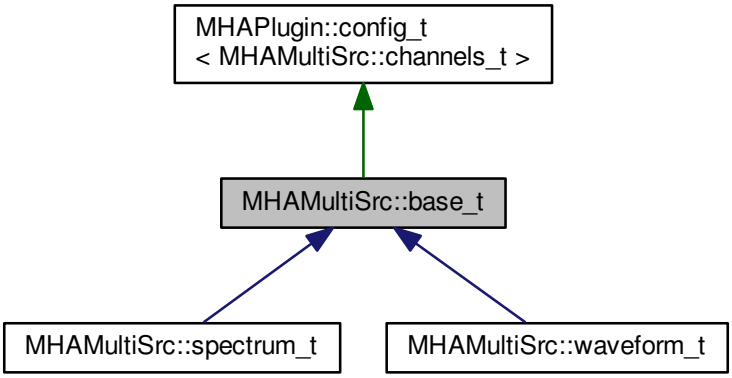
The documentation for this class was generated from the following file:

- **mha_algo_comm.hh**

5.219 MHAMultiSrc::base_t Class Reference

Base class for source selection.

Inheritance diagram for MHAMultiSrc::base_t:



Public Member Functions

- **base_t** (**algo_comm_t** iac)
- void **select_source** (const std::vector< std::string > &src, int in_channels)
  Change the selection of input sources.

Protected Attributes

- **algo_comm_t ac**

Additional Inherited Members

5.219.1 Detailed Description

Base class for source selection.

See also

MHAMultiSrc::channel_t (p. 608)
MHAMultiSrc::channels_t (p. 609)

5.219.2 Constructor & Destructor Documentation

5.219.2.1 MHAMultiSrc::base_t::base_t (algo_comm_t *iac*)

5.219.3 Member Function Documentation

5.219.3.1 void MHAMultiSrc::base_t::select_source (const std::vector< std::string > & *src,* int *in_channels*)

Change the selection of input sources.

This function is real-time and thread safe.

Parameters

src	List of input sources
in_channels	Number of input channels in direct input (the processed signal)

5.219.4 Member Data Documentation

5.219.4.1 algo_comm_t MHAMultiSrc::base_t::ac [protected]

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.220 MHAMultiSrc::channel_t Class Reference

Public Attributes

- std::string **name**
- int **channel**

5.220.1 Member Data Documentation

5.220.1.1 std::string MHAMultiSrc::channel_t::name

5.220.1.2 int MHAMultiSrc::channel_t::channel

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

- **mha_multisrc.h**

5.221 MHAMultiSrc::channels_t Class Reference

Inherits vector< MHAMultiSrc::channel_t >.

Public Member Functions

- **channels_t** (const std::vector< std::string > &src, int in_channels)
  Separate a list of input sources into a parsable channel list.

5.221.1 Constructor & Destructor Documentation

5.221.1.1 MHAMultiSrc::channels_t::channels_t (const std::vector< std::string > & *route,* int *in_channels*)

Separate a list of input sources into a parsable channel list.

The number of input channels if verified, a list of **MHAMultiSrc::channel_t** (p. 608) is filled.

Parameters

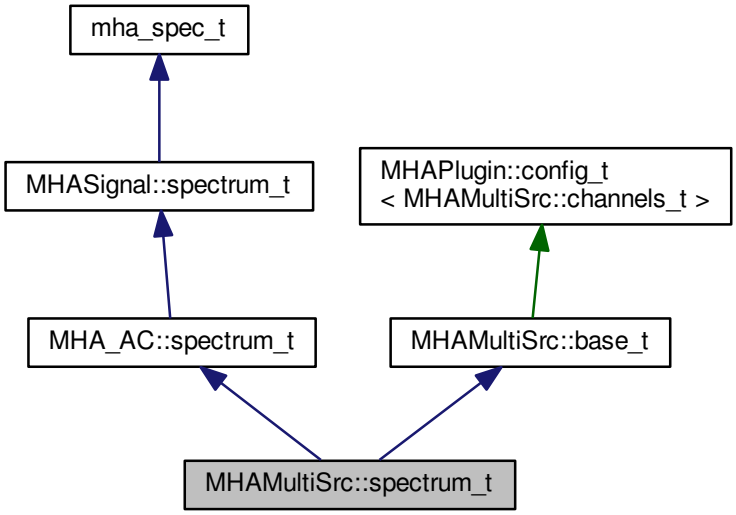
route	vector of source channel ids
in_channels	number of channels in the processed input signal

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.222 MHAMultiSrc::spectrum_t Class Reference

Inheritance diagram for MHAMultiSrc::spectrum_t:



Public Member Functions

- **spectrum_t** (**algo_comm_t** iac, std::string **name**, unsigned int frames, unsigned int **channels**)
- **mha_spec_t** ∗ **update** (**mha_spec_t** ∗s)

  Update data of spectrum to hold actual input data.

Additional Inherited Members

5.222.1 Constructor & Destructor Documentation

5.222.1.1 MHAMultiSrc::spectrum_t::spectrum_t (algo_comm_t *iac,* std::string *name,* unsigned int *frames,* unsigned int *channels*)

5.222.2 Member Function Documentation

5.222.2.1 mha_spec_t ∗ MHAMultiSrc::spectrum_t::update (mha_spec_t ∗ *s*)

Update data of spectrum to hold actual input data.

Parameters

s	Input signal chunk

Returns

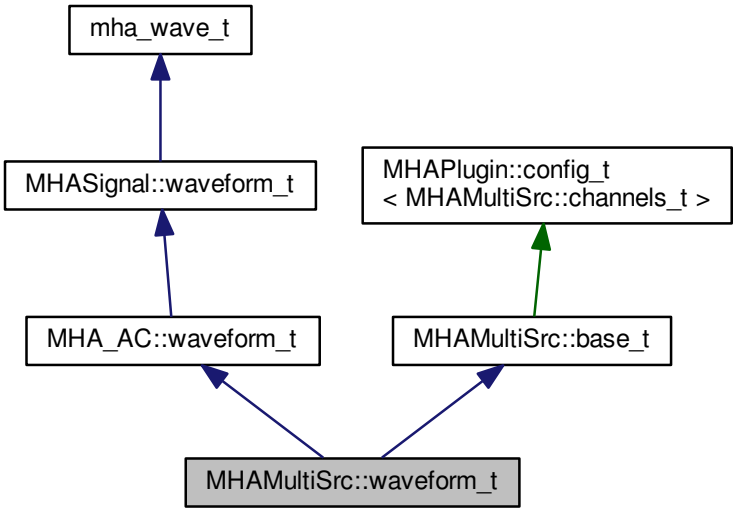
Return pointer to spectrum structure

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.223 MHAMultiSrc::waveform_t Class Reference

Inheritance diagram for MHAMultiSrc::waveform_t:



Public Member Functions

- **waveform_t** (**algo_comm_t** iac, std::string **name**, unsigned int frames, unsigned int **channels**)
- **mha_wave_t** ∗ **update** (**mha_wave_t** ∗s)

  Update data of waveform to hold actual input data.

Additional Inherited Members

5.223.1 Constructor & Destructor Documentation

5.223.1.1 MHAMultiSrc::waveform_t::waveform_t (algo_comm_t *iac,* std::string *name,* unsigned int *frames,* unsigned int *channels*)

5.223.2 Member Function Documentation

5.223.2.1 mha_wave_t * MHAMultiSrc::waveform_t::update (mha_wave_t * *s*)

Update data of waveform to hold actual input data.

Parameters

s	Input signal chunk

Returns

Return pointer to waveform structure

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.224 MHAOvlFilter::band_descriptor_t Class Reference

Public Attributes

- **mha_real_t cf_l**
- **mha_real_t ef_l**
- **mha_real_t cf**
- **mha_real_t ef_h**
- **mha_real_t cf_h**
- bool **low_side_flat**
- bool **high_side_flat**

5.224.1 Member Data Documentation

5.224.1.1 mha_real_t MHAOvlFilter::band_descriptor_t::cf_l

5.224.1.2 mha_real_t MHAOvlFilter::band_descriptor_t::ef_l

5.224.1.3 mha_real_t MHAOvlFilter::band_descriptor_t::cf

5.224.1.4 mha_real_t MHAOvlFilter::band_descriptor_t::ef_h

5.224.1.5 mha_real_t MHAOvlFilter::band_descriptor_t::cf_h

5.224.1.6 bool MHAOvlFilter::band_descriptor_t::low_side_flat

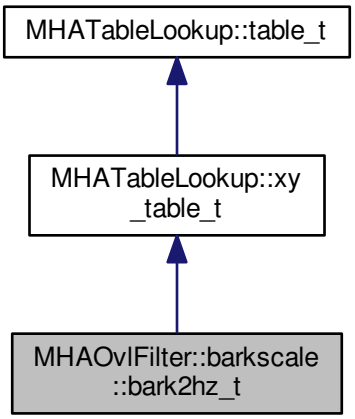
5.224.1.7 bool MHAOvlFilter::band_descriptor_t::high_side_flat

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

- **mha_fftfb.hh**

5.225 MHAOvlFilter::barkscale::bark2hz_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::bark2hz_t:



Public Member Functions

- **bark2hz_t** ()
- ∼**bark2hz_t** ()

Additional Inherited Members

5.225.1 Constructor & Destructor Documentation

5.225.1.1 MHAOvlFilter::barkscale::bark2hz_t::bark2hz_t ( )

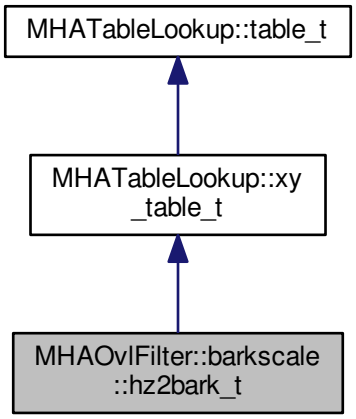
5.225.1.2 MHAOvlFilter::barkscale::bark2hz_t::∼bark2hz_t ( )

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

- **mha_fftfb.cpp**

5.226 MHAOvlFilter::barkscale::hz2bark_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::hz2bark_t:



Public Member Functions

- **hz2bark_t** ()
- ∼**hz2bark_t** ()

Additional Inherited Members

5.226.1 Constructor & Destructor Documentation

5.226.1.1 MHAOvlFilter::barkscale::hz2bark_t::hz2bark_t ( )

5.226.1.2 MHAOvlFilter::barkscale::hz2bark_t::~hz2bark_t ( )

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

- **mha_fftfb.cpp**

5.227 MHAOvlFilter::fftfb_ac_info_t Class Reference

Public Member Functions

- **fftfb_ac_info_t** (const **MHAOvlFilter::fftfb_t** &fb, **algo_comm_t** ac, const std::string &prefix)
- void **insert** ()

Private Attributes

- **MHA_AC::waveform_t cfv**
  vector of nominal center frequencies / Hz
- **MHA_AC::waveform_t efv**
  vector of edge frequencies / Hz
- **MHA_AC::waveform_t bwv**
  vector of band-weigths (sum of squared fft-bin-weigths)/num_frames
- **MHA_AC::waveform_t cLTASS**
  vector of LTASS correction

5.227.1 Constructor & Destructor Documentation

5.227.1.1 MHAOvlFilter::fftfb_ac_info_t::fftfb_ac_info_t (const MHAOvlFilter::fftfb_t & *fb,* algo_comm_t *ac,* const std::string & *prefix*)

5.227.2 Member Function Documentation

5.227.2.1 void MHAOvlFilter::fftfb_ac_info_t::insert ( )

5.227.3 Member Data Documentation

5.227.3.1 MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cfv [private]

vector of nominal center frequencies / Hz

5.227.3.2 MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::efv [private]

vector of edge frequencies / Hz

5.227.3.3 MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::bwv [private]

vector of band-weigths (sum of squared fft-bin-weigths)/num_frames

5.227.3.4 MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cLTASS [private]

vector of LTASS correction

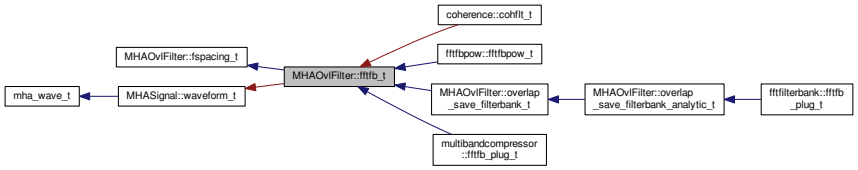
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.228 MHAOvlFilter::fftfb_t Class Reference

FFT based overlapping filter bank.

Inheritance diagram for MHAOvlFilter::fftfb_t:



Public Member Functions

- **fftfb_t** (**MHAOvlFilter::fftfb_vars_t** &par, unsigned int nfft, **mha_real_t** fs)
  Constructor for a FFT-based overlapping filter bank.
- ∼**fftfb_t** ()
- void **apply_gains** (**mha_spec_t** ∗s_out, const **mha_spec_t** ∗s_in, const **mha_wave_t** ∗gains)
- void **get_fbpower** (**mha_wave_t** ∗fbpow, const **mha_spec_t** ∗s_in)
- void **get_fbpower_db** (**mha_wave_t** ∗fbpow, const **mha_spec_t** ∗s_in)
- std::vector< **mha_real_t** > **get_ltass_gain_db** () const
- unsigned int **bin1** (unsigned int band) const
  Return index of first non-zero filter shape window.
- unsigned int **bin2** (unsigned int band) const
  Return index of first zero filter shape window above center frequency.
- unsigned int **get_fftlen** () const
  Return fft length.
- **mha_real_t w** (unsigned int k, unsigned int b) const
  Return filter shape window at index k in band b.

Private Attributes

- unsigned int ∗ **vbin1**
- unsigned int ∗ **vbin2**
- **mha_real_t**(∗ **shape**)(**mha_real_t**)
- unsigned int **fftlen**
- **mha_real_t samplingrate**

Additional Inherited Members

5.228.1 Detailed Description

FFT based overlapping filter bank.

5.228.2 Constructor & Destructor Documentation

5.228.2.1 MHAOvlFilter::fftfb_t::fftfb_t (

MHAOvlFilter::fftfb_vars_t & *par,*
unsigned int *nfft,*
mha_real_t *fs*)

Constructor for a FFT-based overlapping filter bank.

Parameters

par	Parameters for the FFT filterbank that can not be deduced from the signal dimensions are taken from this set of configuration variables.
nfft	FFT length
fs	Sampling rate / Hz

5.228.2.2 MHAOvlFilter::fftfb_t::∼fftfb_t ( )

5.228.3 Member Function Documentation

5.228.3.1 void MHAOvlFilter::fftfb_t::apply_gains (

mha_spec_t ∗ *s_out,*
const mha_spec_t ∗ *s_in,*
const mha_wave_t ∗ *gains*)

5.228.3.2 void MHAOvlFilter::fftfb_t::get_fbpower (

mha_wave_t ∗ *fbpow,*
const mha_spec_t ∗ *s_in*)

5.228.3.3 void MHAOvlFilter::fftfb_t::get_fbpower_db (mha_wave_t * *fbpow,* const mha_spec_t * *s_in*)

5.228.3.4 std::vector< float > MHAOvlFilter::fftfb_t::get_ltass_gain_db ( ) const

5.228.3.5 unsigned int MHAOvlFilter::fftfb_t::bin1 (unsigned int *band*) const [inline]

Return index of first non-zero filter shape window.

5.228.3.6 unsigned int MHAOvlFilter::fftfb_t::bin2 (unsigned int *band*) const [inline]

Return index of first zero filter shape window above center frequency.

5.228.3.7 unsigned int MHAOvlFilter::fftfb_t::get_fftlen ( ) const [inline]

Return fft length.

5.228.3.8 mha_real_t MHAOvlFilter::fftfb_t::w (unsigned int *k,* unsigned int *b*) const [inline]

Return filter shape window at index k in band b.

Parameters

k	Frequency index
b	Band index

5.228.4 Member Data Documentation

5.228.4.1 unsigned int* MHAOvlFilter::fftfb_t::vbin1 [private]

5.228.4.2 unsigned int* MHAOvlFilter::fftfb_t::vbin2 [private]

5.228.4.3 mha_real_t(* MHAOvlFilter::fftfb_t::shape) (mha_real_t) [private]

5.228.4.4 unsigned int MHAOvlFilter::fftfb_t::fftlen [private]

5.228.4.5 mha_real_t MHAOvlFilter::fftfb_t::samplingrate [private]

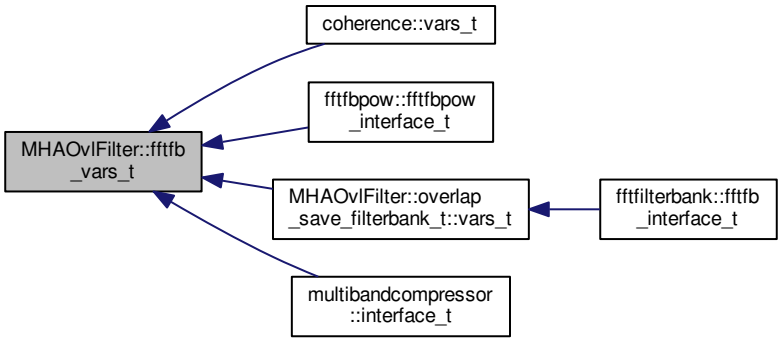
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.229 MHAOvlFilter::fftfb_vars_t Class Reference

Set of configuration variables for FFT-based overlapping filters.

Inheritance diagram for MHAOvlFilter::fftfb_vars_t:



Public Member Functions

- **fftfb_vars_t** (**MHAParser::parser_t** &p)

  construct a set of openMHA configuration language variables suitable for configuring the FFT-based overlapping filterbank.

Public Attributes

- **scale_var_t fscale**

  Frequency scale type (lin/bark/log/erb).
- **scale_var_t ovltype**

  Filter shape (rect/lin/hann).
- **MHAParser::float_t plateau**

  relative plateau width.
- **MHAParser::kw_t ftype**

  Flag to decide wether edge or center frequencies are used.
- **fscale_t f**

  Frequency.
- **MHAParser::bool_t normalize**

  Normalize sum of channels.
- **MHAParser::bool_t fail_on_nonmonotonic**

  Fail if frequency entries are non-monotonic (otherwise sort)
- **MHAParser::bool_t fail_on_unique_bins**

  Fail if center frequencies share the same FFT bin.

- **MHAParser::vfloat_mon_t cf**
  Final center frequencies in Hz.
- **MHAParser::vfloat_mon_t ef**
  Final edge frequencies in Hz.
- **MHAParser::vfloat_mon_t cLTASS**
  Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)
- **MHAParser::mfloat_mon_t shapes**

5.229.1 Detailed Description

Set of configuration variables for FFT-based overlapping filters.

This class enables easy configuration of the FFT-based overlapping filterbank. An instance of **fftfb_vars_t** (p. 619) creates openMHA configuration language variables needed for configuring the filterbank, and inserts these variables in the openMHA configuration tree.

This way, the variables are visible to the user and can be configured using the openMHA configuration language.

5.229.2 Constructor & Destructor Documentation

5.229.2.1 MHAOvlFilter::fftfb_vars_t::fftfb_vars_t (MHAParser::parser_t & *p*)

construct a set of openMHA configuration language variables suitable for configuring the FFT-based overlapping filterbank.

Parameters

p	The node of the configuration tree where the variables created by this instance are inserted.

5.229.3 Member Data Documentation

5.229.3.1 scale_var_t MHAOvlFilter::fftfb_vars_t::fscale

Frequency scale type (lin/bark/log/erb).

5.229.3.2 scale_var_t MHAOvlFilter::fftfb_vars_t::ovltype

Filter shape (rect/lin/hann).

5.229.3.3 MHAParser::float_t MHAOvlFilter::fftfb_vars_t::plateau

relative plateau width.

5.229.3.4 MHAParser::kw_t MHAOvlFilter::fftfb_vars_t::ftype

Flag to decide wether edge or center frequencies are used.

5.229.3.5 fscale_t MHAOvlFilter::fftfb_vars_t::f

Frequency.

5.229.3.6 MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::normalize

Normalize sum of channels.

5.229.3.7 MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_nonmonotonic

Fail if frequency entries are non-monotonic (otherwise sort)

5.229.3.8 MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_unique_bins

Fail if center frequencies share the same FFT bin.

5.229.3.9 MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cf

Final center frequencies in Hz.

5.229.3.10 MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::ef

Final edge frequencies in Hz.

5.229.3.11 MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cLTASS

Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)

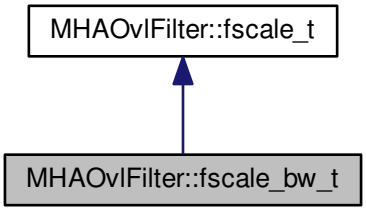
5.229.3.12 MHAParser::mfloat_mon_t MHAOvlFilter::fftfb_vars_t::shapes

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.230 MHAOvlFilter::fscale_bw_t Class Reference

Inheritance diagram for MHAOvlFilter::fscale_bw_t:



Public Member Functions

- **fscale_bw_t** (**MHAParser::parser_t** &parent)
- std::vector< **mha_real_t** > **get_bw_hz** () const

Protected Attributes

- **MHAParser::vfloat_t bw**
- **MHAParser::vfloat_mon_t bw_hz**

Private Member Functions

- void **update_hz** ()

Private Attributes

- **MHAEvents::connector_t**< **fscale_bw_t** > **updater**

Additional Inherited Members

5.230.1 Constructor & Destructor Documentation

5.230.1.1 MHAOvlFilter::fscale_bw_t::fscale_bw_t (MHAParser::parser_t & *parent*)

5.230.2 Member Function Documentation

5.230.2.1 std::vector< mha_real_t > MHAOvlFilter::fscale_bw_t::get_bw_hz ( ) const

5.230.2.2 void MHAOvlFilter::fscale_bw_t::update_hz ( ) [private]

5.230.3 Member Data Documentation

5.230.3.1 MHAParser::vfloat_t MHAOvlFilter::fscale_bw_t::bw [protected]

5.230.3.2 MHAParser::vfloat_mon_t MHAOvlFilter::fscale_bw_t::bw_hz [protected]

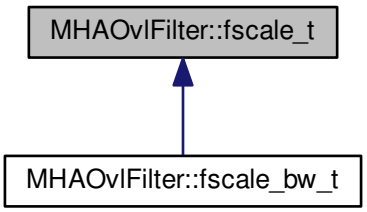
5.230.3.3 MHAEvents::connector_t<fscale_bw_t> MHAOvlFilter::fscale_bw_t::updater [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.231 MHAOvlFilter::fscale_t Class Reference

Inheritance diagram for MHAOvlFilter::fscale_t:



Public Member Functions

- **fscale_t** (**MHAParser::parser_t** &parent)
- std::vector< **mha_real_t** > **get_f_hz** () const

Public Attributes

- **scale_var_t unit**
- **MHAParser::vfloat_t f**
- **MHAParser::vfloat_mon_t f_hz**

Private Member Functions

- void **update_hz** ()

Private Attributes

- **MHAEvents::connector_t< fscale_t > updater**

5.231.1 Constructor & Destructor Documentation

5.231.1.1 MHAOvlFilter::fscale_t::fscale_t (MHAParser::parser_t & *parent*)

5.231.2 Member Function Documentation

5.231.2.1 std::vector< mha_real_t > MHAOvlFilter::fscale_t::get_f_hz ( ) const

5.231.2.2 void MHAOvlFilter::fscale_t::update_hz ( ) [private]

5.231.3 Member Data Documentation

5.231.3.1 scale_var_t MHAOvlFilter::fscale_t::unit

5.231.3.2 MHAParser::vfloat_t MHAOvlFilter::fscale_t::f

5.231.3.3 MHAParser::vfloat_mon_t MHAOvlFilter::fscale_t::f_hz

5.231.3.4 MHAEvents::connector_t<fscale_t> MHAOvlFilter::fscale_t::updater [private]

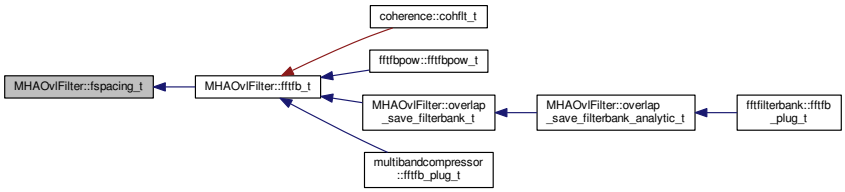
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.232 MHAOvlFilter::fspacing_t Class Reference

Class for frequency spacing, used by filterbank shape generator class.

Inheritance diagram for MHAOvlFilter::fspacing_t:



Public Member Functions

- **fspacing_t** (const **MHAOvlFilter::fftfb_vars_t** &par, unsigned int nfft, **mha_real_t** fs)
- std::vector< unsigned int > **get_cf_fftbin** () const
- std::vector< **mha_real_t** > **get_cf_hz** () const
- std::vector< **mha_real_t** > **get_ef_hz** () const
- unsigned int **nbands** () const

  Return number of bands in filter bank.

Protected Member Functions

- void **fail_on_nonmonotonic_cf** ()
- void **fail_on_unique_fftbins** ()

Protected Attributes

- std::vector< **MHAOvlFilter::band_descriptor_t** > **bands**
- **mha_real_t**(∗ **symmetry_scale**)(**mha_real_t**)

Private Member Functions

- void **ef2bands** (std::vector< **mha_real_t** > vef)
- void **cf2bands** (std::vector< **mha_real_t** > vcf)
- void **equidist2bands** (std::vector< **mha_real_t** > vcf)

Private Attributes

- unsigned int **nfft_**
- **mha_real_t fs_**

5.232.1 Detailed Description

Class for frequency spacing, used by filterbank shape generator class.

5.232.2 Constructor & Destructor Documentation

5.232.2.1 MHAOvlFilter::fspacing_t::fspacing_t (const MHAOvlFilter::fftfb_vars_t & *par,* unsigned int *nfft,* mha_real_t *fs*)

5.232.3 Member Function Documentation

5.232.3.1 std::vector< unsigned int > MHAOvlFilter::fspacing_t::get_cf_fftbin ( ) const

5.232.3.2 std::vector< mha_real_t > MHAOvlFilter::fspacing_t::get_cf_hz ( ) const

5.232.3.3 std::vector< mha_real_t > MHAOvlFilter::fspacing_t::get_ef_hz ( ) const

5.232.3.4 unsigned int MHAOvlFilter::fspacing_t::nbands ( ) const [inline]

Return number of bands in filter bank.

5.232.3.5 void MHAOvlFilter::fspacing_t::fail_on_nonmonotonic_cf ( ) [protected]

5.232.3.6 void MHAOvlFilter::fspacing_t::fail_on_unique_fftbins ( ) [protected]

5.232.3.7 void MHAOvlFilter::fspacing_t::ef2bands (std::vector< mha_real_t > *vef*) [private]

5.232.3.8 void MHAOvlFilter::fspacing_t::cf2bands (std::vector< mha_real_t > *vcf*) [private]

5.232.3.9 void MHAOvlFilter::fspacing_t::equidist2bands (std::vector< mha_real_t > *vcf*) [private]

5.232.4 Member Data Documentation

5.232.4.1 std::vector<MHAOvlFilter::band_descriptor_t> MHAOvlFilter::fspacing_t::bands [protected]

5.232.4.2 mha_real_t(* MHAOvlFilter::fspacing_t::symmetry_scale) (mha_real_t) [protected]

5.232.4.3 unsigned int MHAOvlFilter::fspacing_t::nfft_ [private]

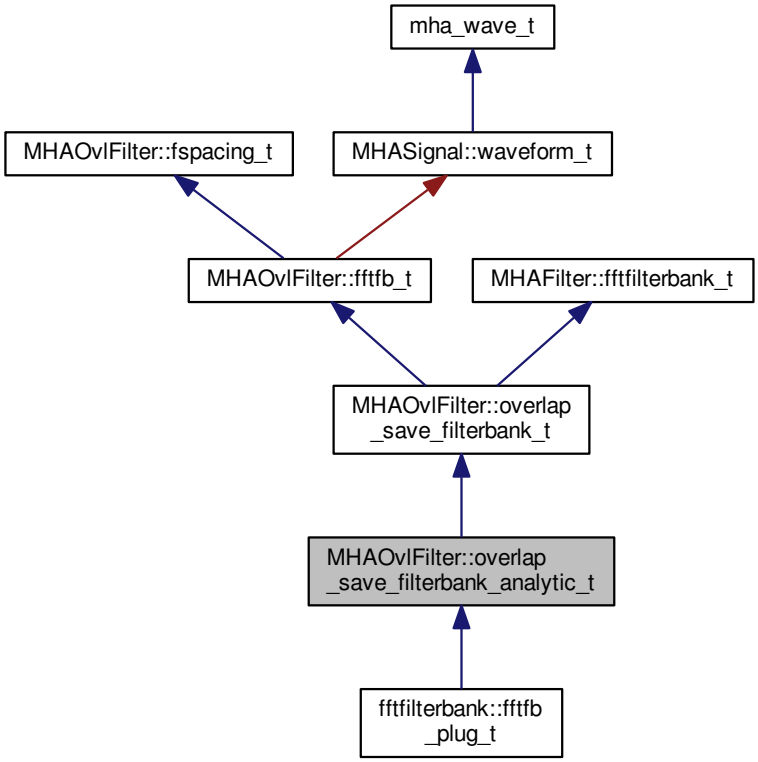
5.232.4.4 mha_real_t MHAOvlFilter::fspacing_t::fs_ [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.233 MHAOvlFilter::overlap_save_filterbank_analytic_t Class Reference

Inheritance diagram for MHAOvlFilter::overlap_save_filterbank_analytic_t:



Public Member Functions

- **overlap_save_filterbank_analytic_t** (**MHAOvlFilter::overlap_save_filterbank_t↩ ::vars_t** &fbpar, **mhaconfig_t** channelconfig_in)
- void **filter_analytic** (const **mha_wave_t** ∗sIn, **mha_wave_t** ∗∗fltRe, **mha_wave_t** ∗∗flt↩ Im)

Private Attributes

- **MHAFilter::fftfilterbank_t imagfb**

Additional Inherited Members

5.233.1 Constructor & Destructor Documentation

5.233.1.1 MHAOvlFilter::overlap_save_filterbank_analytic_t::overlap_save_filterbank_analytic_t (MHAOvlFilter::overlap_save_filterbank_t::vars_t & *fbpar,* mhaconfig_t *channelconfig_in*)

5.233.2 Member Function Documentation

5.233.2.1 void MHAOvlFilter::overlap_save_filterbank_analytic_t::filter_analytic (const mha_wave_t * *sIn,* mha_wave_t ** *fltRe,* mha_wave_t ** *fltIm*)

5.233.3 Member Data Documentation

5.233.3.1 MHAFilter::fftfilterbank_t MHAOvlFilter::overlap_save_filterbank_analytic_t::imagfb [private]

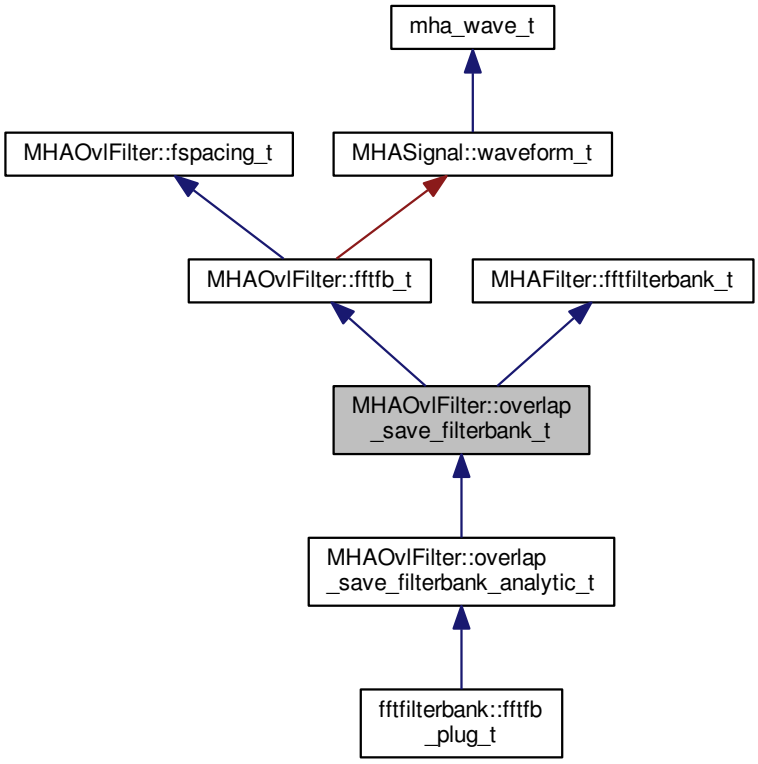
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.234 MHAOvlFilter::overlap_save_filterbank_t Class Reference

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 616).

Inheritance diagram for MHAOvlFilter::overlap_save_filterbank_t:



Classes

- class **vars_t**

Public Member Functions

- **overlap_save_filterbank_t** (**MHAOvlFilter::overlap_save_filterbank_t::vars_t** &fbpar, **mhaconfig_t** channelconfig_in)
- **mhaconfig_t get_channelconfig** () const

Private Attributes

- **mhaconfig_t channelconfig_out_**

Additional Inherited Members

5.234.1 Detailed Description

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 616).

5.234.2 Constructor & Destructor Documentation

5.234.2.1 MHAOvlFilter::overlap_save_filterbank_t::overlap_save_filterbank_t (**MHAOvlFilter::overlap_save_filterbank_t::vars_t** & *fbpar*, **mhaconfig_t** *channelconfig_in*)

5.234.3 Member Function Documentation

5.234.3.1 **mhaconfig_t** MHAOvlFilter::overlap_save_filterbank_t::get_channelconfig ( ) const [inline]

5.234.4 Member Data Documentation

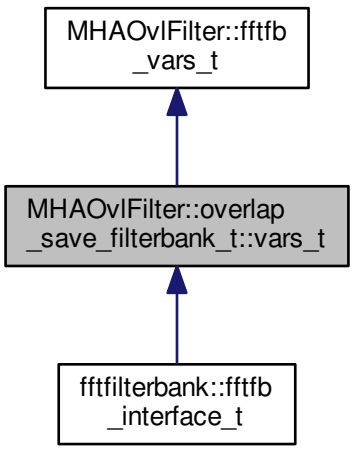
5.234.4.1 **mhaconfig_t** MHAOvlFilter::overlap_save_filterbank_t::channelconfig_out_ [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.235 MHAOvlFilter::overlap_save_filterbank_t::vars_t Class Reference

Inheritance diagram for MHAOvlFilter::overlap_save_filterbank_t::vars_t:



Public Member Functions

- **vars_t** (**MHAParser::parser_t** &p)

Public Attributes

- **MHAParser::int_t fftlen**
- **MHAParser::kw_t phasemodel**
- **MHAParser::window_t irswnd**

5.235.1 Constructor & Destructor Documentation

5.235.1.1 MHAOvlFilter::overlap_save_filterbank_t::vars_t::vars_t (MHAParser::parser_t & *p*)

5.235.2 Member Data Documentation

5.235.2.1 MHAParser::int_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::fftlen

5.235.2.2 MHAParser::kw_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::phasemodel

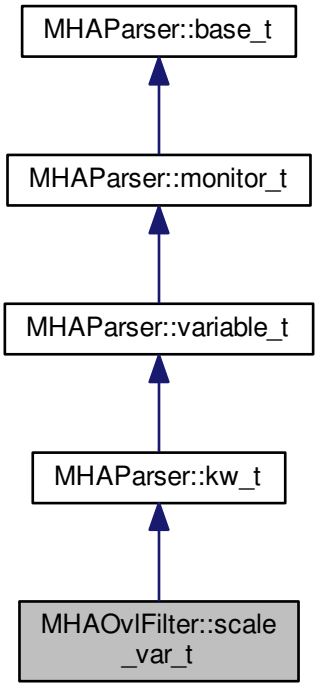
5.235.2.3 MHAParser::window_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::irswnd

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.236 MHAOvlFilter::scale_var_t Class Reference

Inheritance diagram for MHAOvlFilter::scale_var_t:



Public Member Functions

- **scale_var_t** (const std::string &**help**)
- void **add_fun** (const std::string &name, **scale_fun_t** ∗fun)
- std::string **get_name** () const
- **scale_fun_t** ∗ **get_fun** () const
- **mha_real_t hz2unit** (**mha_real_t** x) const
- **mha_real_t unit2hz** (**mha_real_t** x) const

Private Attributes

- std::vector< std::string > **names**
- std::vector< **scale_fun_t** ∗ > **funs**

Additional Inherited Members

5.236.1 Constructor & Destructor Documentation

5.236.1.1 MHAOvlFilter::scale_var_t::scale_var_t (const std::string & *help*)

5.236.2 Member Function Documentation

5.236.2.1 void MHAOvlFilter::scale_var_t::add_fun (const std::string & *name,* scale_fun_t * *fun*)

5.236.2.2 std::string MHAOvlFilter::scale_var_t::get_name ( ) const [inline]

5.236.2.3 scale_fun_t* MHAOvlFilter::scale_var_t::get_fun ( ) const [inline]

5.236.2.4 mha_real_t MHAOvlFilter::scale_var_t::hz2unit (mha_real_t *x*) const

5.236.2.5 mha_real_t MHAOvlFilter::scale_var_t::unit2hz (mha_real_t *x*) const

5.236.3 Member Data Documentation

5.236.3.1 std::vector<std::string> MHAOvlFilter::scale_var_t::names [private]

5.236.3.2 std::vector<scale_fun_t*> MHAOvlFilter::scale_var_t::funs [private]

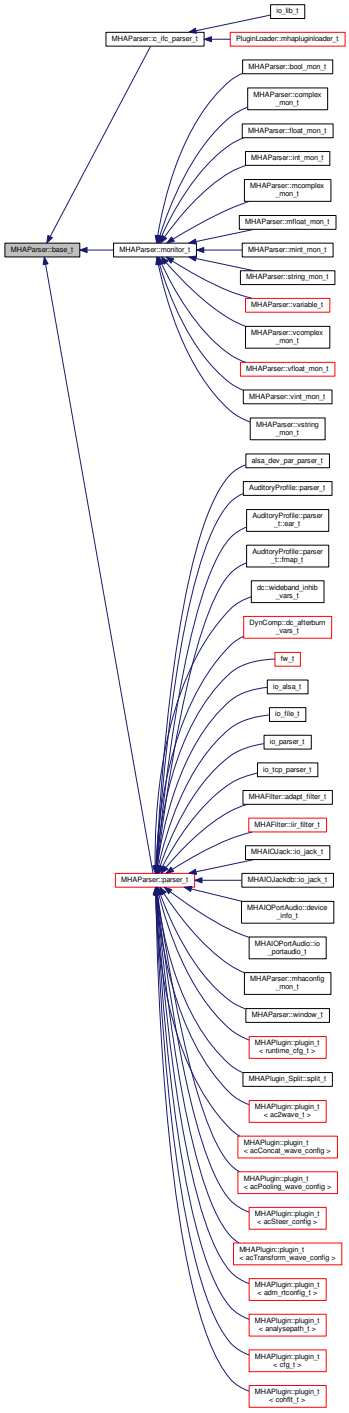
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.237 MHAParser::base_t Class Reference

Base class for all parser items.

Inheritance diagram for MHAParser::base_t:



Classes

- class **replace_t**

Public Member Functions

- **base_t** (const std::string &)
  Constructor for base class of all parser nodes.
- **base_t** (const **base_t** &)
- virtual ~**base_t** ()
- virtual std::string **parse** (const std::string &)
  Causes this node to process a command in the openMHA configuration language.
- virtual void **parse** (const char *, char *, unsigned int)
  This function parses a command and writes the parsing result into a C character array.
- virtual void **parse** (const std::vector< std::string > &, std::vector< std::string > &)
- virtual std::string **op_subparse** (**expression_t** &)
- virtual std::string **op_setval** (**expression_t** &)
- virtual std::string **op_query** (**expression_t** &)
- virtual std::string **query_dump** (const std::string &)
- virtual std::string **query_entries** (const std::string &)
- virtual std::string **query_perm** (const std::string &)
- virtual std::string **query_range** (const std::string &)
- virtual std::string **query_type** (const std::string &)
- virtual std::string **query_val** (const std::string &)
- virtual std::string **query_readfile** (const std::string &)
- virtual std::string **query_savefile** (const std::string &)
- virtual std::string **query_savefile_compact** (const std::string &)
- virtual std::string **query_savemons** (const std::string &)
- virtual std::string **query_listids** (const std::string &)
- std::string **query_version** (const std::string &)
- std::string **query_id** (const std::string &)
- std::string **query_subst** (const std::string &)
- std::string **query_addsubst** (const std::string &)
- std::string **query_help** (const std::string &)
- std::string **query_cmds** (const std::string &)
- void **set_node_id** (const std::string &)
  Set the identification string of this parser node.
- void **set_help** (const std::string &)
  Set the help comment of a variable or parser.
- void **add_parent_on_insert** (**parser_t** *, std::string)
- void **rm_parent_on_remove** (**parser_t** *)
- const std::string & **fullname** () const
  Return the full dot-separated path name of this parser node in the openMHA configuration tree.

Public Attributes

- **MHAEvents::emitter_t writeaccess**
  Event emitted on write access.
- **MHAEvents::emitter_t valuechanged**
  Event emitted if the value has changed.

- **MHAEvents::emitter_t readaccess**

  Event emitted on read access.
- **MHAEvents::emitter_t prereadaccess**

  Event emitted on read access, before the data field is accessed.

Protected Member Functions

- void **activate_query** (const std::string &, **query_t**)
- void **notify** ()

Protected Attributes

- **query_map_t queries**
- bool **data_is_initialized**

Private Types

- typedef std::vector< **replace_t** > **repl_list_t**

Private Member Functions

- void **add_replace_pair** (const std::string &, const std::string &)
- std::string **oplist** ()

Private Attributes

- std::string **help**
- std::string **id_str**
- **opact_map_t operators**
- **repl_list_t repl_list**
- bool **nested_lock**
- **parser_t** * **parent**
- std::string **thefullname**

5.237.1 Detailed Description

Base class for all parser items.

The key method of the parser base class is the std::string **parse(const std::string&)** (p. 637) method. Parser proxy derivatives which overwrite any of the other **parse()** (p. 637) methods to be the key method must make sure that the original **parse()** (p. 637) method utilizes the new key method.

5.237.2 Member Typedef Documentation

5.237.2.1 typedef std::vector<replace_t> MHAParser::base_t::repl_list_t [private]

5.237.3 Constructor & Destructor Documentation

5.237.3.1 MHAParser::base_t::base_t (const std::string & *h*)

Constructor for base class of all parser nodes.

Parameters

h	Help text describing this parser node. This help text is accessible to the configuration language through the "?help" query command.

5.237.3.2 MHAParser::base_t::base_t (const base_t & *src*)

5.237.3.3 MHAParser::base_t::~base_t ( ) [virtual]

5.237.4 Member Function Documentation

5.237.4.1 std::string MHAParser::base_t::parse (const std::string & *cs*) [virtual]

Causes this node to process a command in the openMHA configuration language.

Parameters

cs	The command to parse

Returns

The response to the command, if successful

Exceptions

MHA_Error *(p. 445)*	If the command cannot be executed successfully. The reason for failure is given in the message string of the exception.

Reimplemented in **PluginLoader::mhapluginloader_t** (p. 883), and **altplugs_t** (p. 215).

5.237.4.2 void MHAParser::base_t::parse (const char * *cmd,* char * *retv,* unsigned int *len*) [virtual]

This function parses a command and writes the parsing result into a C character array.

This base class implementation delegates to **parse(const std::string &)** (p. 637).

Parameters

cmd	Command to be parsed
retv	Buffer for the result
len	Length of buffer

Reimplemented in **altplugs_t** (p. 215).

5.237.4.3 void MHAParser::base_t::parse (const std::vector< std::string > & *cs,* std::vector< std::string > & *retv*) [virtual]

5.237.4.4 std::string MHAParser::base_t::op_subparse (expression_t &) [virtual]

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 649), and **MHAParser::parser_t** (p. 691).

5.237.4.5 std::string MHAParser::base_t::op_setval (expression_t &) [virtual]

Reimplemented in **MHAParser::mcomplex_t** (p. 675), **MHAParser::mfloat_t** (p. 679), **MHAParser::vcomplex_t** (p. 706), **MHAParser::vfloat_t** (p. 710), **MHAParser::vint_t** (p. 714), **MHAParser::complex_t** (p. 655), **MHAParser::float_t** (p. 661), **MHAParser::int_t** (p. 666), **MHAParser::bool_t** (p. 647), **MHAParser::vstring_t** (p. 718), **MHAParser::string_t** (p. 699), **MHAParser::kw_t** (p. 671), **MHAParser::variable_t** (p. 701), **MHAParser::c_ifc_parser_t** (p. 649), and **MHAParser::parser_t** (p. 691).

5.237.4.6 std::string MHAParser::base_t::op_query (expression_t &) [virtual]

Reimplemented in **MHAParser::monitor_t** (p. 687), **MHAParser::c_ifc_parser_t** (p. 649), and **MHAParser::parser_t** (p. 691).

5.237.4.7 std::string MHAParser::base_t::query_dump (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::monitor_t** (p. 687), and **MHAParser::parser_t** (p. 691).

5.237.4.8 std::string MHAParser::base_t::query_entries (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 691).

5.237.4.9 std::string MHAParser::base_t::query_perm (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::variable_t** (p. 701), and **MHAParser::monitor_t** (p. 687).

5.237.4.10 std::string MHAParser::base_t::query_range (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::kw_t** (p. 671), and **MHAParser::range_var_t** (p. 694).

5.237.4.11 std::string MHAParser::base_t::query_type (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 673), **MHAParser::vcomplex_mon_t** (p. 704), **MHAParser::complex_mon_t** (p. 653), **MHAParser::float_mon_t** (p. 659), **MHAParser::mfloat_mon_t** (p. 677), **MHAParser::vfloat_mon_t** (p. 708), **MHAParser::mint_mon_t** (p. 686), **MHAParser::vint_mon_t** (p. 712), **MHAParser::vstring_mon_t** (p. 716), **MHAParser::string_mon_t** (p. 697), **MHAParser::bool_mon_t** (p. 645), **MHAParser::int_mon_t** (p. 663), **MHAParser::mcomplex_t** (p. 675), **MHAParser::mfloat_t** (p. 679), **MHAParser::vcomplex_t** (p. 706), **MHAParser::vfloat_t** (p. 710), **MHAParser::vint_t** (p. 714), **MHAParser::complex_t** (p. 655), **MHAParser::float_t** (p. 661), **MHAParser::int_t** (p. 666), **MHAParser::bool_t** (p. 647), **MHAParser::vstring_t** (p. 718), **MHAParser::string_t** (p. 699), **MHAParser::kw_t** (p. 671), and **MHAParser::parser_t** (p. 691).

5.237.4.12 std::string MHAParser::base_t::query_val (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 673), **MHAParser::vcomplex_mon_t** (p. 704), **MHAParser::complex_mon_t** (p. 653), **MHAParser::float_mon_t** (p. 659), **MHAParser::mfloat_mon_t** (p. 677), **MHAParser::vfloat_mon_t** (p. 708), **MHAParser::mint_mon_t** (p. 686), **MHAParser::vint_mon_t** (p. 712), **MHAParser::vstring_mon_t** (p. 716), **MHAParser::string_mon_t** (p. 697), **MHAParser::bool_mon_t** (p. 645), **MHAParser::int_mon_t** (p. 663), **MHAParser::mcomplex_t** (p. 675), **MHAParser::mfloat_t** (p. 679), **MHAParser::vcomplex_t** (p. 706), **MHAParser::vfloat_t** (p. 710), **MHAParser::vint_t** (p. 714), **MHAParser::complex_t** (p. 655), **MHAParser::float_t** (p. 661), **MHAParser::int_t** (p. 666), **MHAParser::bool_t** (p. 647), **MHAParser::vstring_t** (p. 718), **MHAParser::string_t** (p. 699), **MHAParser::kw_t** (p. 671), and **MHAParser::parser_t** (p. 692).

5.237.4.13 std::string MHAParser::base_t::query_readfile (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 691).

5.237.4.14 std::string MHAParser::base_t::query_savefile (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 692).

5.237.4.15 std::string MHAParser::base_t::query_savefile_compact (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 692).

5.237.4.16 std::string MHAParser::base_t::query_savemons (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 692).

5.237.4.17 std::string MHAParser::base_t::query_listids (const std::string & *s*) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 692).

5.237.4.18 std::string MHAParser::base_t::query_version (const std::string &)

5.237.4.19 std::string MHAParser::base_t::query_id (const std::string &)

5.237.4.20 std::string MHAParser::base_t::query_subst (const std::string & *s*)

5.237.4.21 std::string MHAParser::base_t::query_addsubst (const std::string & *s*)

5.237.4.22 std::string MHAParser::base_t::query_help (const std::string & *s*)

5.237.4.23 std::string MHAParser::base_t::query_cmds (const std::string & *s*)

5.237.4.24 void MHAParser::base_t::set_node_id (const std::string & *s*)

Set the identification string of this parser node.

The id can be queried from the configuration language using the ?id query command. Nodes can be found by id using the ?listid query command on a containing parser node.

Parameters

s	The new identification string.

5.237.4.25 void MHAParser::base_t::set_help (const std::string & *s*)

Set the help comment of a variable or parser.

Parameters

s	New help comment.

5.237.4.26 void MHAParser::base_t::add_parent_on_insert (parser_t * *p*, std::string *n*)

5.237.4.27 void MHAParser::base_t::rm_parent_on_remove (parser_t *)

5.237.4.28 const std::string & MHAParser::base_t::fullname ( ) const

Return the full dot-separated path name of this parser node in the openMHA configuration tree.

5.237.4.29 void MHAParser::base_t::activate_query (const std::string & *n*, query_t *a*) [protected]

5.237.4.30 void MHAParser::base_t::notify ( ) [protected]

5.237.4.31 void MHAParser::base_t::add_replace_pair (const std::string & *a*, const std::string & *b*) [private]

5.237.4.32 std::string MHAParser::base_t::oplist ( ) [private]

5.237.5 Member Data Documentation

5.237.5.1 MHAEvents::emitter_t MHAParser::base_t::writeaccess

Event emitted on write access.

To connect a callback that is invoked on write access to this parser variable, use MHAEvents↩ ::patchbay_t<receiver_t> method connect(&writeaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

5.237.5.2 MHAEvents::emitter_t MHAParser::base_t::valuechanged

Event emitted if the value has changed.

To connect a callback that is invoked when write access to this parser variable actually changes its value, use MHAEvents::patchbay_t<receiver_t> method connect(&valuechanged,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

5.237.5.3 MHAEvents::emitter_t MHAParser::base_t::readaccess

Event emitted on read access.

To connect a callback that is invoked after the value of this variable has been read through the configuration interface, use MHAEvents::patchbay_t<receiver_t> method connect(&readaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

5.237.5.4 MHAEvents::emitter_t MHAParser::base_t::prereadaccess

Event emitted on read access, before the data field is accessed.

To connect a callback that is invoked when the value of this variable is about to be read through the configuration interface, so that the callback can influence the value that is reported, use MHAEvents::patchbay_t<receiver_t> method connect(&prereadaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

5.237.5.5 query_map_t MHAParser::base_t::queries [protected]

5.237.5.6 bool MHAParser::base_t::data_is_initialized [protected]

5.237.5.7 std::string MHAParser::base_t::help [private]

5.237.5.8 std::string MHAParser::base_t::id_str [private]

5.237.5.9 opact_map_t MHAParser::base_t::operators [private]

5.237.5.10 repl_list_t MHAParser::base_t::repl_list [private]

5.237.5.11 bool MHAParser::base_t::nested_lock [private]

5.237.5.12 parser_t* MHAParser::base_t::parent [private]

5.237.5.13 std::string MHAParser::base_t::thefullname [private]

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.238 MHAParser::base_t::replace_t Class Reference

Public Member Functions

- **replace_t** (const std::string &, const std::string &)
- void **replace** (std::string &)
- const std::string & **get_a** () const
- const std::string & **get_b** () const

Private Attributes

- std::string **a**
- std::string **b**

5.238.1 Constructor & Destructor Documentation

5.238.1.1 MHAParser::base_t::replace_t::replace_t (const std::string & *ia,* const std::string & *ib*)

5.238.2 Member Function Documentation

5.238.2.1 void MHAParser::base_t::replace_t::replace (std::string & *s*)

5.238.2.2 const std::string& MHAParser::base_t::replace_t::get_a ( ) const [inline]

5.238.2.3 const std::string& MHAParser::base_t::replace_t::get_b ( ) const [inline]

5.238.3 Member Data Documentation

5.238.3.1 std::string MHAParser::base_t::replace_t::a [private]

5.238.3.2 std::string MHAParser::base_t::replace_t::b [private]

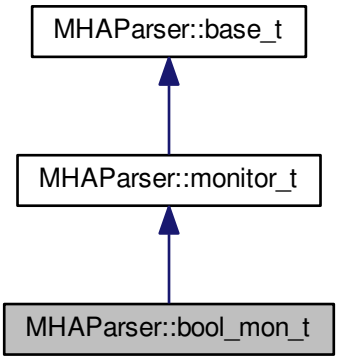
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.239 MHAParser::bool_mon_t Class Reference

Monitor with string value.

Inheritance diagram for MHAParser::bool_mon_t:



Public Member Functions

- **bool_mon_t** (const std::string &hlp)
  Create a monitor variable for string values.

Public Attributes

- bool **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.239.1 Detailed Description

Monitor with string value.

5.239.2 Constructor & Destructor Documentation

5.239.2.1 MHAParser::bool_mon_t::bool_mon_t (const std::string & *hlp*)

Create a monitor variable for string values.

Parameters

hlp	A help text describing this monitor variable.

5.239.3 Member Function Documentation

5.239.3.1 std::string MHAParser::bool_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.239.3.2 std::string MHAParser::bool_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.239.4 Member Data Documentation

5.239.4.1 bool MHAParser::bool_mon_t::data

Data field.

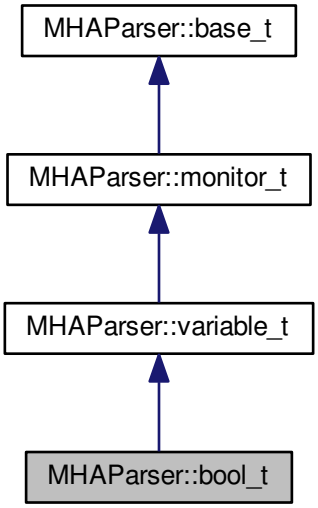
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.240 MHAParser::bool_t Class Reference

Variable with a boolean value ("yes"/"no")

Inheritance diagram for MHAParser::bool_t:



Public Member Functions

- **bool_t** (const std::string &help_text, const std::string &initial_value)
  Constructor for a configuration language variable for boolean values.

Public Attributes

- bool **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.240.1 Detailed Description

Variable with a boolean value ("yes"/"no")

5.240.2 Constructor & Destructor Documentation

5.240.2.1 MHAParser::bool_t::bool_t (const std::string & *help_text,* const std::string & *initial_value*)

Constructor for a configuration language variable for boolean values.

Parameters

help_text	A human-readable text describing the purpose of this configuration variable.
initial_value	The initial value for this variable as a string. The string representation of 'true' is either "yes" or "1". The string representation of 'false' is either "no" or "0".

5.240.3 Member Function Documentation

5.240.3.1 std::string MHAParser::bool_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.240.3.2 std::string MHAParser::bool_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.240.3.3 std::string MHAParser::bool_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.240.4 Member Data Documentation

5.240.4.1 bool MHAParser::bool_t::data

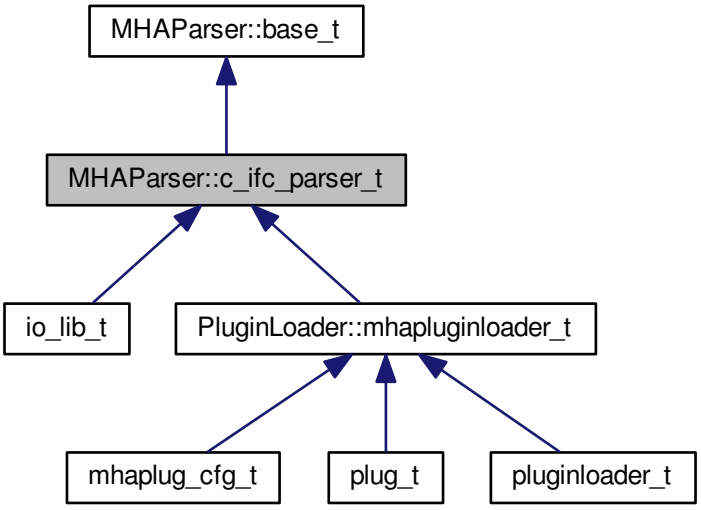
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.241 MHAParser::c_ifc_parser_t Class Reference

Inheritance diagram for MHAParser::c_ifc_parser_t:



Public Member Functions

- **c_ifc_parser_t** (const std::string &modulename_)
- ~**c_ifc_parser_t** ()
- void **set_parse_cb** (**c_parse_cmd_t**, **c_parse_err_t**, void *)

Protected Member Functions

- std::string **op_subparse** (**MHAParser::expression_t** &)
- std::string **op_setval** (**MHAParser::expression_t** &)
- std::string **op_query** (**MHAParser::expression_t** &)

Private Member Functions

- void **test_error** ()

Private Attributes

- std::string **modulename**
- **c_parse_cmd_t c_parse_cmd**
- **c_parse_err_t c_parse_err**
- int **liberr**
- void * **libdata**
- unsigned int **ret_size**
- char * **retv**

Additional Inherited Members

5.241.1 Constructor & Destructor Documentation

5.241.1.1 MHAParser::c_ifc_parser_t::c_ifc_parser_t (const std::string & *modulename_*)

5.241.1.2 MHAParser::c_ifc_parser_t::~c_ifc_parser_t ( )

5.241.2 Member Function Documentation

5.241.2.1 void MHAParser::c_ifc_parser_t::set_parse_cb (MHAParser::c_parse_cmd_t *cb,* MHAParser::c_parse_err_t *strerr,* void * *d*)

5.241.2.2 std::string MHAParser::c_ifc_parser_t::op_subparse (MHAParser::expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.241.2.3 std::string MHAParser::c_ifc_parser_t::op_setval (MHAParser::expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.241.2.4 std::string MHAParser::c_ifc_parser_t::op_query (MHAParser::expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.241.2.5 void MHAParser::c_ifc_parser_t::test_error ( ) [private]

5.241.3 Member Data Documentation

5.241.3.1 std::string MHAParser::c_ifc_parser_t::modulename [private]

5.241.3.2 c_parse_cmd_t MHAParser::c_ifc_parser_t::c_parse_cmd [private]

5.241.3.3 c_parse_err_t MHAParser::c_ifc_parser_t::c_parse_err [private]

5.241.3.4 int MHAParser::c_ifc_parser_t::liberr [private]

5.241.3.5 void* MHAParser::c_ifc_parser_t::libdata [private]

5.241.3.6 unsigned int MHAParser::c_ifc_parser_t::ret_size [private]

5.241.3.7 char* MHAParser::c_ifc_parser_t::retv [private]

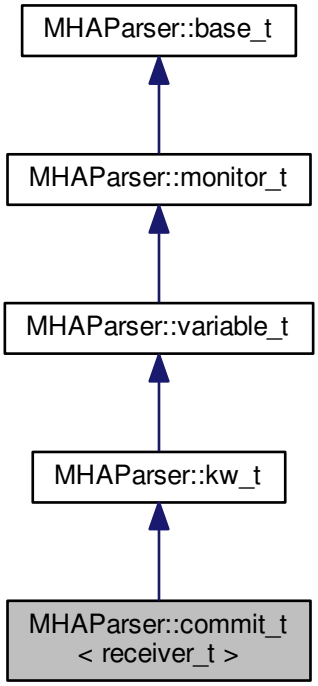
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.242 MHAParser::commit_t< receiver_t > Class Template Reference

Parser variable with event-emission functionality.

Inheritance diagram for MHAParser::commit_t< receiver_t >:



Public Member Functions

- **commit_t** (receiver_t ∗, void(receiver_t::∗)(), const std::string &**help**="Variable changes action")

Private Attributes

- **MHAEvents::connector_t**< receiver_t > **extern_connector**

Additional Inherited Members

5.242.1 Detailed Description

template<**class receiver_t**>
class MHAParser::commit_t< **receiver_t** >

Parser variable with event-emission functionality.

The **commit_t** (p. 650) variable can register an event receiver in its constructor, which is called whenever the variable is set to "commit".

5.242.2 Constructor & Destructor Documentation

5.242.2.1 template<class receiver_t > MHAParser::commit_t< receiver_t >::commit_t (receiver_t * *r*, void(receiver_t::*)() *rfun*, const std::string & *help* = "Variable changes action")

5.242.3 Member Data Documentation

5.242.3.1 template<class receiver_t > MHAEvents::connector_t<receiver_t> MHAParser::commit_t< receiver_t >::extern_connector [private]

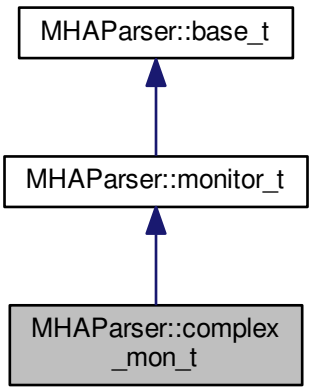
The documentation for this class was generated from the following file:

- **mha_parser.hh**

5.243 MHAParser::complex_mon_t Class Reference

Monitor with complex value.

Inheritance diagram for MHAParser::complex_mon_t:



Public Member Functions

- **complex_mon_t** (const std::string &hlp)
  Create a complex monitor variable.

Public Attributes

- **mha_complex_t data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.243.1 Detailed Description

Monitor with complex value.

5.243.2 Constructor & Destructor Documentation

5.243.2.1 MHAParser::complex_mon_t::complex_mon_t (const std::string & *hlp*)

Create a complex monitor variable.

Parameters

hlp	A help text describing this monitor variable.

5.243.3 Member Function Documentation

5.243.3.1 std::string MHAParser::complex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.243.3.2 std::string MHAParser::complex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.243.4 Member Data Documentation

5.243.4.1 mha_complex_t MHAParser::complex_mon_t::data

Data field.

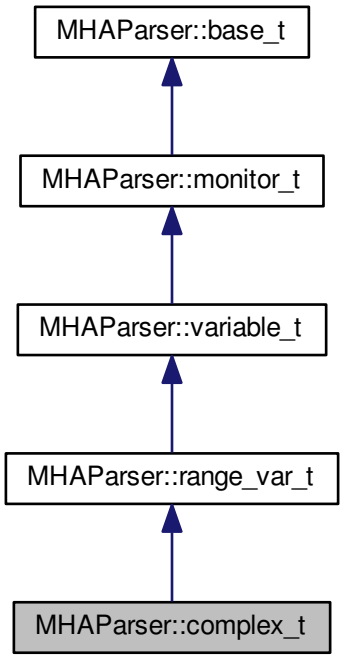
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.244 MHAParser::complex_t Class Reference

Variable with complex value.

Inheritance diagram for MHAParser::complex_t:



Public Member Functions

- **complex_t** (const std::string &, const std::string &, const std::string &="")

Public Attributes

- **mha_complex_t data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.244.1 Detailed Description

Variable with complex value.

5.244.2 Constructor & Destructor Documentation

5.244.2.1 MHAParser::complex_t::complex_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

5.244.3 Member Function Documentation

5.244.3.1 std::string MHAParser::complex_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.244.3.2 std::string MHAParser::complex_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.244.3.3 std::string MHAParser::complex_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.244.4 Member Data Documentation

5.244.4.1 mha_complex_t MHAParser::complex_t::data

Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.245 MHAParser::entry_t Class Reference

Public Member Functions

- **entry_t** (const std::string &, **base_t** ∗)

Public Attributes

- std::string **name**
- **base_t** ∗ **entry**

5.245.1 Constructor & Destructor Documentation

5.245.1.1 MHAParser::entry_t::entry_t (const std::string & *n,* base_t ∗ *e*)

5.245.2 Member Data Documentation

5.245.2.1 std::string MHAParser::entry_t::name

5.245.2.2 base_t∗ MHAParser::entry_t::entry

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.246 MHAParser::expression_t Class Reference

Public Member Functions

- **expression_t** (const std::string &, const std::string &)
  Constructor.
- **expression_t** ()

Public Attributes

- std::string **lval**
- std::string **rval**
- std::string **op**

5.246.1 Constructor & Destructor Documentation

5.246.1.1 expression_t::expression_t (const std::string & *s*, const std::string & *o*)

Constructor.

Parameters

s	String to be splitted
o	List of valid operators (single character only)

5.246.1.2 expression_t::expression_t ( )

5.246.2 Member Data Documentation

5.246.2.1 std::string MHAParser::expression_t::lval

5.246.2.2 std::string MHAParser::expression_t::rval

5.246.2.3 std::string MHAParser::expression_t::op

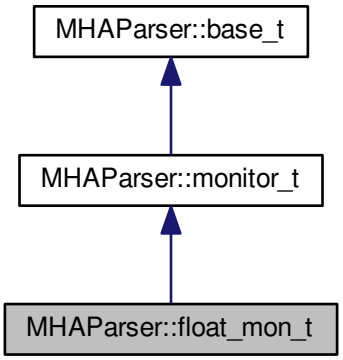
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.247 MHAParser::float_mon_t Class Reference

Monitor with float value.

Inheritance diagram for MHAParser::float_mon_t:



Public Member Functions

- **float_mon_t** (const std::string &hlp)
  Initialize a floating point (32 bits) monitor variable.

Public Attributes

- float **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.247.1 Detailed Description

Monitor with float value.

5.247.2 Constructor & Destructor Documentation

5.247.2.1 MHAParser::float_mon_t::float_mon_t (const std::string & *hlp*)

Initialize a floating point (32 bits) monitor variable.

Parameters

hlp	A help text describing this monitor variable.

5.247.3 Member Function Documentation

5.247.3.1 std::string MHAParser::float_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.247.3.2 std::string MHAParser::float_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.247.4 Member Data Documentation

5.247.4.1 float MHAParser::float_mon_t::data

Data field.

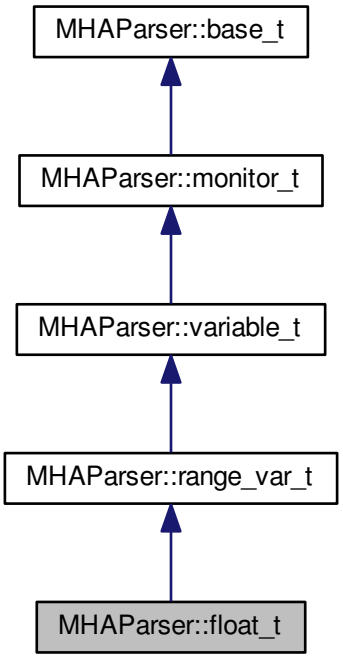
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.248 MHAParser::float_t Class Reference

Variable with float value.

Inheritance diagram for MHAParser::float_t:



Public Member Functions

- **float_t** (const std::string &help_text, const std::string &initial_value, const std::string &**range**="")

  Constructor for a configuration language variable for 32bit ieee floating-point values.

Public Attributes

- float **data**

  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.248.1 Detailed Description

Variable with float value.

5.248.2 Constructor & Destructor Documentation

5.248.2.1 MHAParser::float_t::float_t (const std::string & *help_text,* const std::string & *initial_value,* const std::string & *range* = "")

Constructor for a configuration language variable for 32bit ieee floating-point values.

Parameters

help_text	A human-readable text describing the purpose of this configuration variable.
initial_value	The initial value for this variable as a string (decimal representation of the floating-point variable). If a range is given in the third parameter, then the initial value has to be within the range. A human-readable text describing the purpose of this configuration variable.
range	The range of values that this variable can hold can be restricted. A range is a string of the form "[a,b]", where a and b are decimal representations of the inclusive boundaries of the range. a<=b. In a range of the form "]a,b[", both boundaries are excluded. Mixed forms are permitted. a or b can also be omitted if there is no lower or upper limit. The range of values is always restricted by the representable range of the underlying C data type.

5.248.3 Member Function Documentation

5.248.3.1 std::string MHAParser::float_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.248.3.2 std::string MHAParser::float_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.248.3.3 std::string MHAParser::float_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.248.4 Member Data Documentation

5.248.4.1 float MHAParser::float_t::data

Data field.

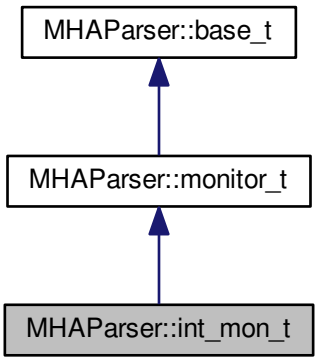
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.249 MHAParser::int_mon_t Class Reference

Monitor variable with int value.

Inheritance diagram for MHAParser::int_mon_t:



Public Member Functions

- **int_mon_t** (const std::string &hlp)
  Create a monitor variable for integral values.

Public Attributes

- int **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.249.1 Detailed Description

Monitor variable with int value.

Monitor variables can be of many types. These variables can be queried through the parser. The public data element contains the monitored state. Write access is only possible from the C++ code by direct access to the data field.

5.249.2 Constructor & Destructor Documentation

5.249.2.1 MHAParser::int_mon_t::int_mon_t (const std::string & *hlp*)

Create a monitor variable for integral values.

Parameters

hlp	A help text describing this monitor variable.

5.249.3 Member Function Documentation

5.249.3.1 std::string MHAParser::int_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.249.3.2 std::string MHAParser::int_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.249.4 Member Data Documentation

5.249.4.1 int MHAParser::int_mon_t::data

Data field.

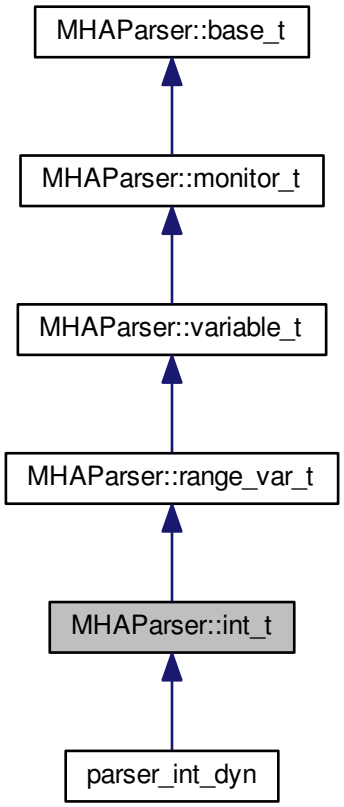
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.250 MHAParser::int_t Class Reference

Variable with integer value.

Inheritance diagram for MHAParser::int_t:



Public Member Functions

- **int_t** (const std::string &help_text, const std::string &initial_value, const std::string &**range**="")

  Constructor for a configuration language variable for integral values.

Public Attributes

- int **data**

  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.250.1 Detailed Description

Variable with integer value.

5.250.2 Constructor & Destructor Documentation

5.250.2.1 MHAParser::int_t::int_t (const std::string & *help_text,* const std::string & *initial_value,* const std::string & *range =* " ")

Constructor for a configuration language variable for integral values.

Parameters

help_text	A human-readable text describing the purpose of this configuration variable.
initial_value	The initial value for this variable as a string (decimal representation of the integer variable). If a range is given in the third parameter, then the initial value has to be within the range.
range	The range of values that this variable can hold can be restricted. A range is a string of the form "[a,b]", where a and b are decimal representations of the integral inclusive boundaries of the range. a<=b. In a range of the form "]a,b[", both boundaries are excluded. Mixed forms are permitted. a or b can also be omitted if there is no lower or upper limit. The range of values is always restricted by the representable range of the underlying C data type (usually 32 bits, [-2147483648,2147483647]).

5.250.3 Member Function Documentation

5.250.3.1 std::string MHAParser::int_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.250.3.2 std::string MHAParser::int_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.250.3.3 std::string MHAParser::int_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.250.4 Member Data Documentation

5.250.4.1 int MHAParser::int_t::data

Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.251 MHAParser::keyword_list_t Class Reference

Keyword list class.

Public Types

- typedef std::vector< std::string >::size_type **size_t**

Public Member Functions

- void **set_value** (const std::string &)
  Select a value from keyword list.
- void **set_entries** (const std::string &)
  Set keyword list entries.
- const std::string & **get_value** () const
  Return selected value.
- const std::vector< std::string > & **get_entries** () const
  Return keyword list.
- const **size_t** & **get_index** () const
  Return index of selected value.
- void **set_index** (unsigned int)
- void **validate** () const
  Check if index of selected value is valid.
- void **add_entry** (const std::string &en)
- **keyword_list_t** ()
  Constructor.

Private Attributes

- **size_t index**
  Index into list.
- std::vector< std::string > **entries**
  List of valid entries.
- std::string **empty_string**

5.251.1 Detailed Description

Keyword list class.

The stucture **keyword_list_t** (p. 666) defines a keyword list (vector of strings) with an index into the list. Used as **MHAParser::kw_t** (p. 669), it can be used to access a set of valid keywords through the parser (i.e. one of "pear apple banana").

5.251.2 Member Typedef Documentation

5.251.2.1 typedef std::vector<std::string>::size_type MHAParser::keyword_list_t::size_t

5.251.3 Constructor & Destructor Documentation

5.251.3.1 MHAParser::keyword_list_t::keyword_list_t ( )

Constructor.

5.251.4 Member Function Documentation

5.251.4.1 void MHAParser::keyword_list_t::set_value (const std::string & *s*)

Select a value from keyword list.

This function selects a value from the keyword list. The index is set to the last matching entry.

Parameters

s	Value to be selected.

5.251.4.2 void MHAParser::keyword_list_t::set_entries (const std::string & *s*)

Set keyword list entries.

With this function, the keyword list can be set from a space separated string list.

Parameters

s	Space separated entry list.

5.251.4.3 const std::string & MHAParser::keyword_list_t::get_value ( ) const

Return selected value.

5.251.4.4 const std::vector< std::string > & MHAParser::keyword_list_t::get_entries ( ) const

Return keyword list.

5.251.4.5 const MHAParser::keyword_list_t::size_t & MHAParser::keyword_list_t::get_index ( ) const

Return index of selected value.

5.251.4.6 void MHAParser::keyword_list_t::set_index (unsigned int *idx*)

5.251.4.7 void MHAParser::keyword_list_t::validate ( ) const

Check if index of selected value is valid.

5.251.4.8 void MHAParser::keyword_list_t::add_entry (const std::string & *en*) [inline]

5.251.5 Member Data Documentation

5.251.5.1 size_t MHAParser::keyword_list_t::index [private]

Index into list.

5.251.5.2 std::vector<std::string> MHAParser::keyword_list_t::entries [private]

List of valid entries.

5.251.5.3 std::string MHAParser::keyword_list_t::empty_string [private]

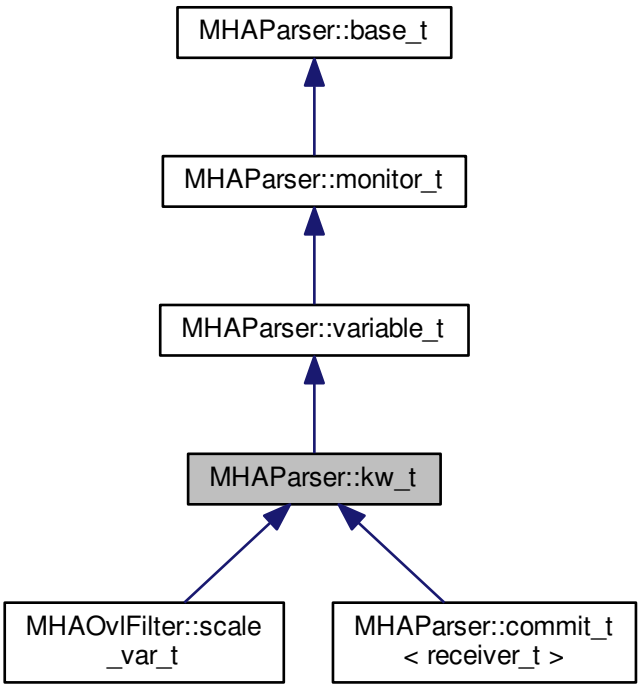
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.252 MHAParser::kw_t Class Reference

Variable with keyword list value.

Inheritance diagram for MHAParser::kw_t:



Public Member Functions

- **kw_t** (const std::string &, const std::string &, const std::string &)
  Constructor of a keyword list openMHA configuration variable.
- **kw_t** (const **kw_t** &)
  Copy constructor.
- void **set_range** (const std::string &)
  Set/change the list of valid entries.
- bool **isval** (const std::string &) const
  Test if the given value is selected.

Public Attributes

- **keyword_list_t data**
  Variable data in its native type.

Protected Member Functions

- void **validate** (const **keyword_list_t** &)
- std::string **op_setval** (**expression_t** &)
- std::string **query_range** (const std::string &)
- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.252.1 Detailed Description

Variable with keyword list value.

5.252.2 Constructor & Destructor Documentation

5.252.2.1 MHAParser::kw_t::kw_t (const std::string & *h,* const std::string & *v,* const std::string & *rg*)

Constructor of a keyword list openMHA configuration variable.

Parameters

h	A help string describing the purpose of this variable.
v	The initial value, has to be a value from the list of possible values given in the last parameter.
rg	A string containing the list of valid entries. The entries have to be separated by spaces. The list of entries has to be delimited by brackets "[", "]".

5.252.2.2 MHAParser::kw_t::kw_t (const kw_t & *src*)

Copy constructor.

5.252.3 Member Function Documentation

5.252.3.1 void MHAParser::kw_t::set_range (const std::string & *r*)

Set/change the list of valid entries.

Parameters

r	A string containing the list of valid entries. The entries have to be separated by spaces. The list of entries has to be delimited by brackets "[", "]".

5.252.3.2 bool MHAParser::kw_t::isval (const std::string & *testval*) const

Test if the given value is selected.

5.252.3.3 void MHAParser::kw_t::validate (const keyword_list_t & *s*) [protected]

5.252.3.4 std::string MHAParser::kw_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.252.3.5 std::string MHAParser::kw_t::query_range (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.252.3.6 std::string MHAParser::kw_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.252.3.7 std::string MHAParser::kw_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.252.4 Member Data Documentation

5.252.4.1 keyword_list_t MHAParser::kw_t::data

Variable data in its native type.

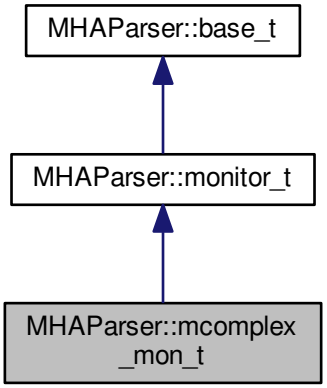
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.253 MHAParser::mcomplex_mon_t Class Reference

Matrix of complex numbers monitor.

Inheritance diagram for MHAParser::mcomplex_mon_t:



Public Member Functions

- **mcomplex_mon_t** (const std::string &hlp)
  Create a matrix of complex floating point monitor values.

Public Attributes

- std::vector< std::vector< **mha_complex_t** > > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.253.1 Detailed Description

Matrix of complex numbers monitor.

5.253.2 Constructor & Destructor Documentation

5.253.2.1 MHAParser::mcomplex_mon_t::mcomplex_mon_t (const std::string & *hlp*)

Create a matrix of complex floating point monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.253.3 Member Function Documentation

5.253.3.1 std::string MHAParser::mcomplex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.253.3.2 std::string MHAParser::mcomplex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.253.4 Member Data Documentation

5.253.4.1 std::vector< std::vector<mha_complex_t> > MHAParser::mcomplex_mon_t::data

Data field.

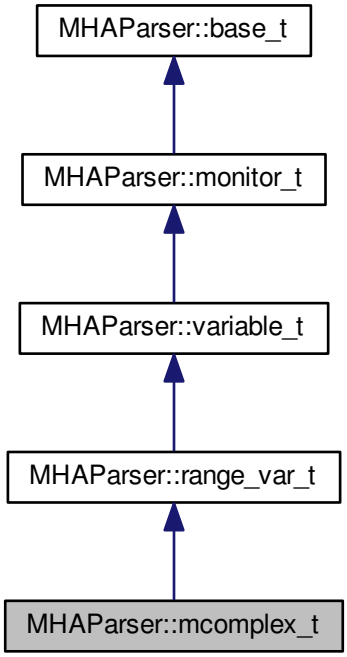
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.254 MHAParser::mcomplex_t Class Reference

Matrix variable with complex value.

Inheritance diagram for MHAParser::mcomplex_t:



Public Member Functions

- **mcomplex_t** (const std::string &, const std::string &, const std::string &="")

Public Attributes

- std::vector< std::vector< **mha_complex_t** > > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.254.1 Detailed Description

Matrix variable with complex value.

5.254.2 Constructor & Destructor Documentation

5.254.2.1 MHAParser::mcomplex_t::mcomplex_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

5.254.3 Member Function Documentation

5.254.3.1 std::string MHAParser::mcomplex_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.254.3.2 std::string MHAParser::mcomplex_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.254.3.3 std::string MHAParser::mcomplex_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.254.4 Member Data Documentation

5.254.4.1 std::vector<std::vector<**mha_complex_t**> > MHAParser::mcomplex_t::data

Data field.

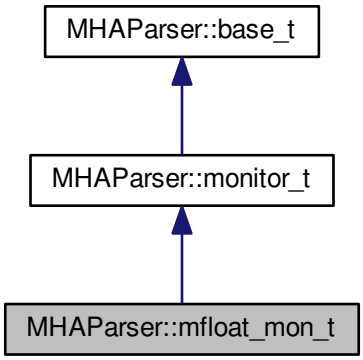
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.255 MHAParser::mfloat_mon_t Class Reference

Matrix of floats monitor.

Inheritance diagram for MHAParser::mfloat_mon_t:



Public Member Functions

- **mfloat_mon_t** (const std::string &hlp)
  Create a matrix of floating point monitor values.

Public Attributes

- std::vector< std::vector< float > > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.255.1 Detailed Description

Matrix of floats monitor.

5.255.2 Constructor & Destructor Documentation

5.255.2.1 MHAParser::mfloat_mon_t::mfloat_mon_t (const std::string & *hlp*)

Create a matrix of floating point monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.255.3 Member Function Documentation

5.255.3.1 std::string MHAParser::mfloat_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.255.3.2 std::string MHAParser::mfloat_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.255.4 Member Data Documentation

5.255.4.1 std::vector< std::vector<float> > MHAParser::mfloat_mon_t::data

Data field.

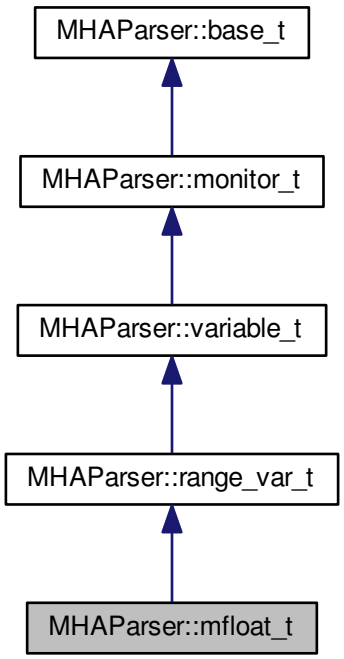
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.256 MHAParser::mfloat_t Class Reference

Matrix variable with float value.

Inheritance diagram for MHAParser::mfloat_t:



Public Member Functions

- **mfloat_t** (const std::string &, const std::string &, const std::string &="")
  Create a float matrix parser variable.

Public Attributes

- std::vector< std::vector< float > > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.256.1 Detailed Description

Matrix variable with float value.

5.256.2 Constructor & Destructor Documentation

5.256.2.1 MHAParser::mfloat_t::mfloat_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

Create a float matrix parser variable.

Parameters

h	A human-readable text describing the purpose of this configuration variable.
v	The initial value of the variable, as a string, in openMHA configuration language: (e.g. "[[0 1]; [2 3]]" for a matrix), described in the "Multidimensional Variables" s2.1.3 section of the openMHA User Manual.
rg	The numeric range to enforce on all members of the matrix.

5.256.3 Member Function Documentation

5.256.3.1 std::string MHAParser::mfloat_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.256.3.2 std::string MHAParser::mfloat_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.256.3.3 std::string MHAParser::mfloat_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.256.4 Member Data Documentation

5.256.4.1 std::vector<std::vector<float> > MHAParser::mfloat_t::data

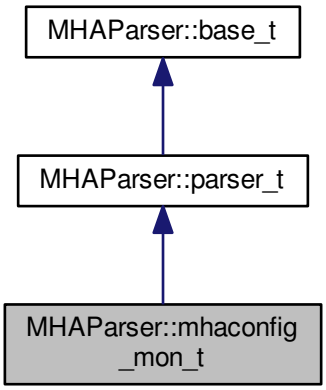
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.257 MHAParser::mhaconfig_mon_t Class Reference

Inheritance diagram for MHAParser::mhaconfig_mon_t:



Public Member Functions

- **mhaconfig_mon_t** (const std::string &**help**="")
- void **update** (const **mhaconfig_t** &cf)

Private Attributes

- **MHAParser::int_mon_t channels**
  Number of audio channels.
- **MHAParser::string_mon_t domain**
  Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)
- **MHAParser::int_mon_t fragsize**
  Fragment size of waveform data.
- **MHAParser::int_mon_t wndlen**
  Window length of spectral data.
- **MHAParser::int_mon_t fftlen**
  FFT length of spectral data.
- **MHAParser::float_mon_t srate**
  Sampling rate in Hz.

Additional Inherited Members

5.257.1 Constructor & Destructor Documentation

5.257.1.1 MHAParser::mhaconfig_mon_t::mhaconfig_mon_t (const std::string & *help* = " ")

5.257.2 Member Function Documentation

5.257.2.1 void MHAParser::mhaconfig_mon_t::update (const mhaconfig_t & *cf*)

5.257.3 Member Data Documentation

5.257.3.1 MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::channels [private]

Number of audio channels.

5.257.3.2 MHAParser::string_mon_t MHAParser::mhaconfig_mon_t::domain [private]

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.257.3.3 MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::fragsize [private]

Fragment size of waveform data.

5.257.3.4 MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::wndlen [private]

Window length of spectral data.

5.257.3.5 MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::fftlen [private]

FFT length of spectral data.

5.257.3.6 MHAParser::float_mon_t MHAParser::mhaconfig_mon_t::srate [private]

Sampling rate in Hz.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.258 MHAParser::mhapluginloader_t Class Reference

Class to create a plugin loader in a parser, including the load logic.

Public Member Functions

- **mhapluginloader_t** (**MHAParser::parser_t** &parent, **algo_comm_t** ac, const std::string &plugname_name="plugin_name", const std::string &prefix="")
- ∼**mhapluginloader_t** ()
- void **prepare** (**mhaconfig_t** &cf)
- void **release** ()
- void **process** (**mha_wave_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- void **process** (**mha_spec_t** ∗sIn, **mha_spec_t** ∗∗sOut)
- void **process** (**mha_wave_t** ∗sIn, **mha_spec_t** ∗∗sOut)
- void **process** (**mha_spec_t** ∗sIn, **mha_wave_t** ∗∗sOut)
- **mhaconfig_t get_cfin** () const
- **mhaconfig_t get_cfout** () const
- const std::string & **get_last_name** () const

Protected Attributes

- **PluginLoader::mhapluginloader_t** ∗ **plug**

Private Member Functions

- void **load_plug** ()

Private Attributes

- **MHAParser::parser_t** & **parent_**
- **MHAParser::string_t plugname**
- std::string **prefix_**
- **MHAEvents::connector_t**< **mhapluginloader_t** > **connector**
- **algo_comm_t ac_**
- std::string **last_name**
- std::string **plugname_name_**
- **mhaconfig_t cf_in_**
- **mhaconfig_t cf_out_**

Static Private Attributes

- static double **bookkeeping**

5.258.1 Detailed Description

Class to create a plugin loader in a parser, including the load logic.

5.258.2 Constructor & Destructor Documentation

5.258.2.1 MHAParser::mhapluginloader_t::mhapluginloader_t (MHAParser::parser_t & *parent,* algo_comm_t *ac,* const std::string & *plugname_name =* `"plugin_name"`, const std::string & *prefix =* `""`)

5.258.2.2 MHAParser::mhapluginloader_t::~mhapluginloader_t ( )

5.258.3 Member Function Documentation

5.258.3.1 void MHAParser::mhapluginloader_t::prepare (mhaconfig_t & *cf*)

5.258.3.2 void MHAParser::mhapluginloader_t::release ( )

5.258.3.3 void MHAParser::mhapluginloader_t::process (mha_wave_t * *sIn,* mha_wave_t ** *sOut*) `[inline]`

5.258.3.4 void MHAParser::mhapluginloader_t::process (mha_spec_t * *sIn,* mha_spec_t ** *sOut*) `[inline]`

5.258.3.5 void MHAParser::mhapluginloader_t::process (mha_wave_t * *sIn*, mha_spec_t ** *sOut*) [inline]

5.258.3.6 void MHAParser::mhapluginloader_t::process (mha_spec_t * *sIn*, mha_wave_t ** *sOut*) [inline]

5.258.3.7 mhaconfig_t MHAParser::mhapluginloader_t::get_cfin ( ) const [inline]

5.258.3.8 mhaconfig_t MHAParser::mhapluginloader_t::get_cfout ( ) const [inline]

5.258.3.9 const std::string& MHAParser::mhapluginloader_t::get_last_name ( ) const [inline]

5.258.3.10 void MHAParser::mhapluginloader_t::load_plug ( ) [private]

5.258.4 Member Data Documentation

5.258.4.1 PluginLoader::mhapluginloader_t* MHAParser::mhapluginloader_t::plug [protected]

5.258.4.2 MHAParser::parser_t& MHAParser::mhapluginloader_t::parent_ [private]

5.258.4.3 MHAParser::string_t MHAParser::mhapluginloader_t::plugname [private]

5.258.4.4 std::string MHAParser::mhapluginloader_t::prefix_ [private]

5.258.4.5 MHAEvents::connector_t<mhapluginloader_t> MHAParser::mhapluginloader_t::connector [private]

5.258.4.6 algo_comm_t MHAParser::mhapluginloader_t::ac_ [private]

5.258.4.7 std::string MHAParser::mhapluginloader_t::last_name [private]

5.258.4.8 std::string MHAParser::mhapluginloader_t::plugname_name_ [private]

5.258.4.9 mhaconfig_t MHAParser::mhapluginloader_t::cf_in_ [private]

5.258.4.10 mhaconfig_t MHAParser::mhapluginloader_t::cf_out_ [private]

5.258.4.11 double MHAParser::mhapluginloader_t::bookkeeping [static], [private]

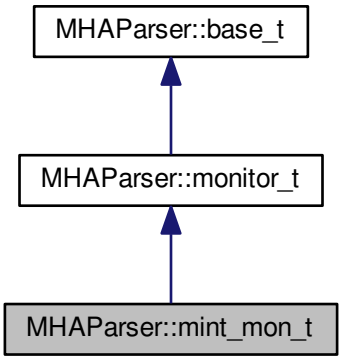
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.259 MHAParser::mint_mon_t Class Reference

Matrix of ints monitor.

Inheritance diagram for MHAParser::mint_mon_t:



Public Member Functions

- **mint_mon_t** (const std::string &hlp)
  Create a matrix of integer monitor values.

Public Attributes

- std::vector< std::vector< int > > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.259.1 Detailed Description

Matrix of ints monitor.

5.259.2 Constructor & Destructor Documentation

5.259.2.1 MHAParser::mint_mon_t::mint_mon_t (const std::string & *hlp*)

Create a matrix of integer monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.259.3 Member Function Documentation

5.259.3.1 std::string MHAParser::mint_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.259.3.2 std::string MHAParser::mint_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.259.4 Member Data Documentation

5.259.4.1 std::vector< std::vector<int> > MHAParser::mint_mon_t::data

Data field.

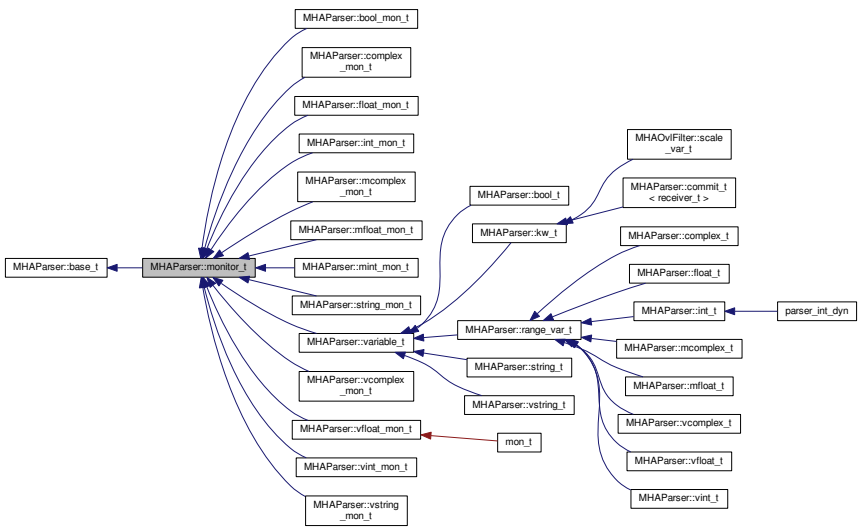
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.260 MHAParser::monitor_t Class Reference

Base class for monitors and variable nodes.

Inheritance diagram for MHAParser::monitor_t:



Public Member Functions

- **monitor_t** (const std::string &)
- **monitor_t** (const **monitor_t** &)
- std::string **op_query** (**expression_t** &)
- std::string **query_dump** (const std::string &)
- std::string **query_perm** (const std::string &)

Additional Inherited Members

5.260.1 Detailed Description

Base class for monitors and variable nodes.

5.260.2 Constructor & Destructor Documentation

5.260.2.1 MHAParser::monitor_t::monitor_t (const std::string & *h*)

5.260.2.2 MHAParser::monitor_t::monitor_t (const **monitor_t** & *src*)

5.260.3 Member Function Documentation

5.260.3.1 std::string MHAParser::monitor_t::op_query (**expression_t** & *x*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.260.3.2 std::string MHAParser::monitor_t::query_dump (const std::string & *s*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.260.3.3 std::string MHAParser::monitor_t::query_perm (const std::string & *s*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

Reimplemented in **MHAParser::variable_t** (p. 701).

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.261 MHAParser::parser_t Class Reference

Parser node class.

Inherits **MHAParser::base_t**.

Inherited by **alsa_dev_par_parser_t**, **AuditoryProfile::parser_t**, **AuditoryProfile::parser_t::ear_t**, **AuditoryProfile::parser_t::fmap_t**, **dc::wideband_inhib_vars_t**, **DynComp::dc_afterburn_vars_t**, **fw_t**, **io_alsa_t**, **io_file_t**, **io_parser_t**, **io_tcp_parser_t**, **MHAFilter::adapt_filter_t**, **MHAFilter::iir_filter_t**, **MHAIOJack::io_jack_t**, **MHAIOJackdb::io_jack_t**, **MHAIOPortAudio::device_info_t**, **MHAIOPortAudio::io_portaudio_t**, **MHAParser::mhaconfig_mon_t**, **MHAParser::window_t**, **MHAPlugin::plugin_t< runtime_cfg_t >**, **MHAPlugin_Split::split_t**, **MHAPlugin::plugin_t< ac2wave_t >**, **MHAPlugin::plugin_t< acConcat_wave_config >**, **MHAPlugin::plugin_t< acPooling_wave_config >**, **MHAPlugin::plugin_t< acSteer_config >**, **MHAPlugin::plugin_t< acTransform_wave_config >**, **MHAPlugin::plugin_t< adm_rtconfig_t >**, **MHAPlugin::plugin_t< analysepath_t >**, **MHAPlugin::plugin_t< cfg_t >**, **MHAPlugin::plugin_t< cohflt_t >**, **MHAPlugin::plugin_t< combc_t >**, **MHAPlugin::plugin_t< db_t >**, **MHAPlugin::plugin_t< dc_t >**, **MHAPlugin::plugin_t< delaysum_t >**, **MHAPlugin::plugin_t< doasvm_classification_config >**, **MHAPlugin::plugin_t< doasvm_feature_extraction_config >**, **MHAPlugin::plugin_t< example5_t >**, **MHAPlugin::plugin_t< fftfb_plug_t >**, **MHAPlugin::plugin_t< fftfbpow_t >**, **MHAPlugin::plugin_t< float >**, **MHAPlugin::plugin_t< fshift_config_t >**, **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >**, **MHAPlugin::plugin_t< hilbert_shifter_t >**, **MHAPlugin::plugin_t< int >**, **MHAPlugin::plugin_t< lpc_bl_predictor_config >**, **MHAPlugin::plugin_t< lpc_burglattice_config >**, **MHAPlugin::plugin_t< lpc_config >**, **MHAPlugin::plugin_t< MHA_AC::spectrum_t >**, **MHAPlugin::plugin_t< MHA_AC::waveform_t >**, **MHAPlugin::plugin_t< mhachain::plugs_t >**, **MHAPlugin::plugin_t< MHASignal::delay_t >**, **MHAPlugin::plugin_t< MHASignal::waveform_t >**, **MHAPlugin::plugin_t< MHAWindow::fun_t >**, **MHAPlugin::plugin_t< noisePowProposed >**, **MHAPlugin::plugin_t< overlapadd_t >**, **MHAPlugin::plugin_t< plingploing_t >**, **MHAPlugin::plugin_t< prediction_error_config >**, **MHAPlugin::plugin_t< resampling_t >**, **MHAPlugin::plugin_t< rmslevel_t >**, **MHAPlugin::plugin_t< route::process_t >**, **MHAPlugin::plugin_t< rt_nlms_t >**, **MHAPlugin::plugin_t< scaler_t >**, **MHAPlugin::plugin_t< sine_cfg_t >**, **MHAPlugin::plugin_t< smoothspec_wrap_t >**, **MHAPlugin::plugin_t< spec2wave_t >**, **MHAPlugin::plugin_t< spec_fader_t >**, **MHAPlugin::plugin_t< steerbf_config >**, **MHAPlugin::plugin_t< timoConfig >**, **MHAPlugin::plugin_t< wave2spec_t >**, **MHAPlugin::plugin_t< wavwriter_t >**, **softclipper_variables_t**, **testplugin::ac_parser_t**, **testplugin::config_parser_t**, and **testplugin::signal_parser_t**.

Public Member Functions

- **parser_t** (const std::string &help_text="")
  Construct detached node to be used in the configuration tree.
- ~**parser_t** ()
- void **insert_item** (const std::string &, **base_t** *)
  Register a parser item into this sub-parser.
- void **remove_item** (const std::string &)
  Remove an item by name.
- void **force_remove_item** (const std::string &)
  Remove an item by name.
- void **remove_item** (const **base_t** *)
  Remove an item by address.

Protected Member Functions

- std::string **op_subparse** (**expression_t** &)
- std::string **op_setval** (**expression_t** &)
- std::string **op_query** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_dump** (const std::string &)
- std::string **query_entries** (const std::string &)
- std::string **query_readfile** (const std::string &)
- std::string **query_savefile** (const std::string &)
- std::string **query_savefile_compact** (const std::string &)
- std::string **query_savemons** (const std::string &)
- std::string **query_val** (const std::string &)
- std::string **query_listids** (const std::string &)
- void **set_id_string** (const std::string &)

Private Attributes

- **entry_map_t entries**
- std::string **id_string**
  identification string
- std::string **srcfile**
- unsigned int **srcline**
- std::string **last_errormsg**

Additional Inherited Members

5.261.1 Detailed Description

Parser node class.

A **parser_t** (p. 688) instance is a node in the configuration tree. A parser node can contain any number of other **parser_t** (p. 688) instances or configuration language variables. These items are inserted into a parser node using the **parser_t::insert_item** (p. 690) method.

5.261.2 Constructor & Destructor Documentation

5.261.2.1 MHAParser::parser_t::parser_t (const std::string & *help_text* = " ")

Construct detached node to be used in the configuration tree.

Parameters

help_text	A text describing this node. E.g. if this node lives at the root of some openMHA plugin, then the help text should describe the functionality of the plugin.

5.261.2.2 MHAParser::parser_t::~parser_t ( )

5.261.3 Member Function Documentation

5.261.3.1 void MHAParser::parser_t::insert_item (const std::string & *n*, MHAParser::base_t * *e*)

Register a parser item into this sub-parser.

This function registers an item under a given name into this sub-parser and makes it accessible to the parser interface.

Parameters

n	Name of the item in the configuration tree
e	C++ pointer to the item instance. e can either point to a variable, to a monitor, or to another sub-parser.

5.261.3.2 void MHAParser::parser_t::remove_item (const std::string & *n*)

Remove an item by name.

If the item does not exist, an error is being reported.

Parameters

n	Name of parser item to be removed from list.

5.261.3.3 void MHAParser::parser_t::force_remove_item (const std::string & *n*)

Remove an item by name.

Non-existing items are ignored.

Parameters

n	Name of parser item to be removed from list.

5.261.3.4 void MHAParser::parser_t::remove_item (const base_t * *addr*)

Remove an item by address.

The item belonging to an address is being removed from the list of items.

Parameters

addr	Address of parser item to be removed.

5.261.3.5 std::string MHAParser::parser_t::op_subparse (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.261.3.6 std::string MHAParser::parser_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.261.3.7 std::string MHAParser::parser_t::op_query (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.261.3.8 std::string MHAParser::parser_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.261.3.9 std::string MHAParser::parser_t::query_dump (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

5.261.3.10 std::string MHAParser::parser_t::query_entries (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.261.3.11 std::string MHAParser::parser_t::query_readfile (const std::string & *fname*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.261.3.12 std::string MHAParser::parser_t::query_savefile (const std::string & *fname*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 640).

5.261.3.13 std::string MHAParser::parser_t::query_savefile_compact (const std::string & *fname*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 640).

5.261.3.14 std::string MHAParser::parser_t::query_savemons (const std::string & *fname*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 640).

5.261.3.15 std::string MHAParser::parser_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.261.3.16 std::string MHAParser::parser_t::query_listids (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 640).

5.261.3.17 void MHAParser::parser_t::set_id_string (const std::string & *s*) [protected]

5.261.4 Member Data Documentation

5.261.4.1 entry_map_t MHAParser::parser_t::entries [private]

5.261.4.2 std::string MHAParser::parser_t::id_string [private]

identification string

5.261.4.3 std::string MHAParser::parser_t::srcfile [private]

5.261.4.4 unsigned int MHAParser::parser_t::srcline [private]

5.261.4.5 std::string MHAParser::parser_t::last_errormsg [private]

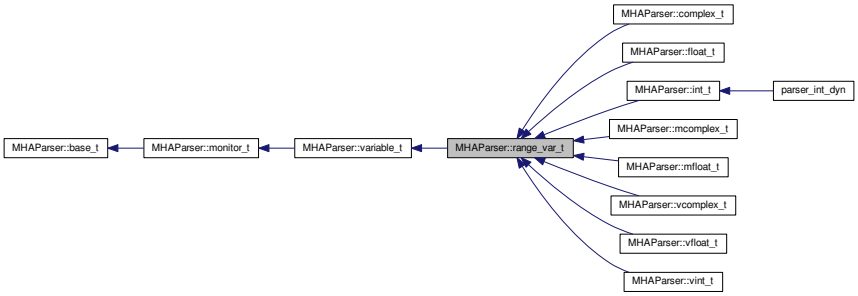
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.262 MHAParser::range_var_t Class Reference

Base class for all variables with a numeric value range.

Inheritance diagram for MHAParser::range_var_t:



Public Member Functions

- **range_var_t** (const std::string &, const std::string &="")
- **range_var_t** (const **range_var_t** &)
- std::string **query_range** (const std::string &)
- void **set_range** (const std::string &r)
  Change the valid range of a variable.
- void **validate** (const int &)
- void **validate** (const float &)
- void **validate** (const **mha_complex_t** &)
- void **validate** (const std::vector< int > &)
- void **validate** (const std::vector< float > &)
- void **validate** (const std::vector< **mha_complex_t** > &)
- void **validate** (const std::vector< std::vector< float > > &)
- void **validate** (const std::vector< std::vector< **mha_complex_t** > > &)

Protected Attributes

- float **low_limit**
  Lower limit of range.
- float **up_limit**
  Upper limit of range.
- bool **low_incl**
  Lower limit is included (or excluded) in range.
- bool **up_incl**
  Upper limit is included (or excluded) in range.
- bool **check_low**
  Check lower limit.
- bool **check_up**
  Check upper limit.
- bool **check_range**
  Range checking is active.

Additional Inherited Members

5.262.1 Detailed Description

Base class for all variables with a numeric value range.

5.262.2 Constructor & Destructor Documentation

5.262.2.1 MHAParser::range_var_t::range_var_t (const std::string & *h,* const std::string & *r =* " ")

5.262.2.2 MHAParser::range_var_t::range_var_t (const **range_var_t** & *src*)

5.262.3 Member Function Documentation

5.262.3.1 std::string MHAParser::range_var_t::query_range (const std::string & *s*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.262.3.2 void MHAParser::range_var_t::set_range (const std::string & *r*)

Change the valid range of a variable.

Parameters

r	New range of the variable (string representation)

5.262.3.3 void MHAParser::range_var_t::validate (const int & *v*)

5.262.3.4 void MHAParser::range_var_t::validate (const float & *v*)

5.262.3.5 void MHAParser::range_var_t::validate (const **mha_complex_t** & *v*)

5.262.3.6 void MHAParser::range_var_t::validate (const std::vector< int > & *v*)

5.262.3.7 void MHAParser::range_var_t::validate (const std::vector< float > & *v*)

5.262.3.8 void MHAParser::range_var_t::validate (const std::vector< **mha_complex_t** > & *v*)

5.262.3.9 void MHAParser::range_var_t::validate (const std::vector< std::vector< float > > & *v*)

5.262.3.10 void MHAParser::range_var_t::validate (const std::vector< std::vector< **mha_complex_t** > > & *v*)

5.262.4 Member Data Documentation

5.262.4.1 float MHAParser::range_var_t::low_limit [protected]

Lower limit of range.

5.262.4.2 float MHAParser::range_var_t::up_limit [protected]

Upper limit of range.

5.262.4.3 bool MHAParser::range_var_t::low_incl [protected]

Lower limit is included (or excluded) in range.

5.262.4.4 bool MHAParser::range_var_t::up_incl [protected]

Upper limit is included (or excluded) in range.

5.262.4.5 bool MHAParser::range_var_t::check_low [protected]

Check lower limit.

5.262.4.6 bool MHAParser::range_var_t::check_up [protected]

Check upper limit.

5.262.4.7 bool MHAParser::range_var_t::check_range [protected]

Range checking is active.

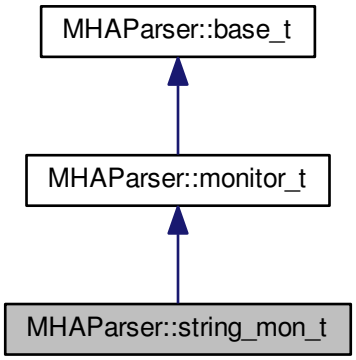
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.263 MHAParser::string_mon_t Class Reference

Monitor with string value.

Inheritance diagram for MHAParser::string_mon_t:



Public Member Functions

- **string_mon_t** (const std::string &hlp)
  Create a monitor variable for string values.

Public Attributes

- std::string **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.263.1 Detailed Description

Monitor with string value.

5.263.2 Constructor & Destructor Documentation

5.263.2.1 MHAParser::string_mon_t::string_mon_t (const std::string & *hlp*)

Create a monitor variable for string values.

Parameters

hlp	A help text describing this monitor variable.

5.263.3 Member Function Documentation

5.263.3.1 std::string MHAParser::string_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.263.3.2 std::string MHAParser::string_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.263.4 Member Data Documentation

5.263.4.1 std::string MHAParser::string_mon_t::data

Data field.

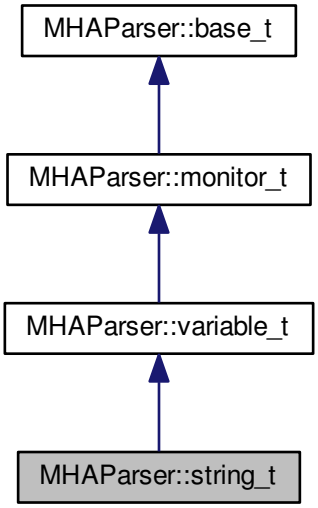
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.264 MHAParser::string_t Class Reference

Variable with a string value.

Inheritance diagram for MHAParser::string_t:



Public Member Functions

- **string_t** (const std::string &, const std::string &)
  Constructor of a openMHA configuration variable for string values.

Public Attributes

- std::string **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.264.1 Detailed Description

Variable with a string value.

5.264.2 Constructor & Destructor Documentation

5.264.2.1 MHAParser::string_t::string_t (const std::string & *h,* const std::string & *v*)

Constructor of a openMHA configuration variable for string values.

Parameters

h	A help string describing the purpose of this variable.
v	The initial string value

5.264.3 Member Function Documentation

5.264.3.1 std::string MHAParser::string_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.264.3.2 std::string MHAParser::string_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.264.3.3 std::string MHAParser::string_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.264.4 Member Data Documentation

5.264.4.1 std::string MHAParser::string_t::data

Data field.

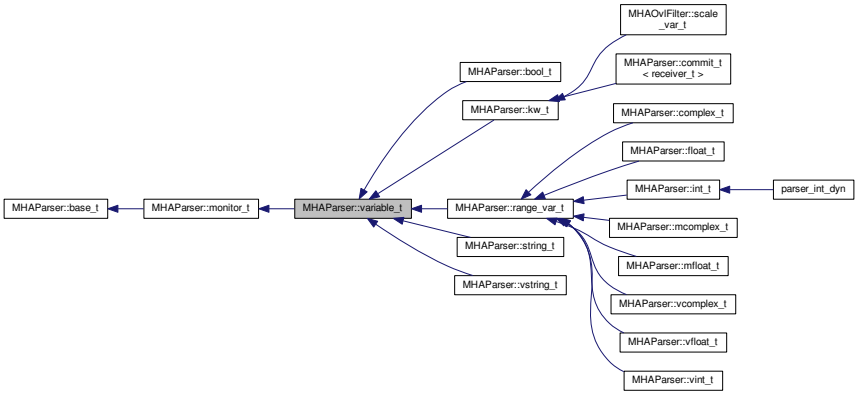
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.265 MHAParser::variable_t Class Reference

Base class for variable nodes.

Inheritance diagram for MHAParser::variable_t:



Public Member Functions

- **variable_t** (const std::string &)
- std::string **op_setval** (**expression_t** &)
- std::string **query_perm** (const std::string &)
- void **setlock** (const bool &)

  Lock a variable against write access.

Private Attributes

- bool **locked**

Additional Inherited Members

5.265.1 Detailed Description

Base class for variable nodes.

5.265.2 Constructor & Destructor Documentation

5.265.2.1 MHAParser::variable_t::variable_t (const std::string & *h*)

5.265.3 Member Function Documentation

5.265.3.1 std::string MHAParser::variable_t::op_setval (expression_t & *x*) [virtual]

Reimplemented from **MHAParser::base_t** (p. 638).

Reimplemented in **MHAParser::mcomplex_t** (p. 675), **MHAParser::mfloat_t** (p. 679), **MHAParser::vcomplex_t** (p. 706), **MHAParser::vfloat_t** (p. 710), **MHAParser::vint_t** (p. 714), **MHAParser::complex_t** (p. 655), **MHAParser::float_t** (p. 661), **MHAParser::int_t** (p. 666), **MHAParser::bool_t** (p. 647), **MHAParser::vstring_t** (p. 718), **MHAParser::string_t** (p. 699), and **MHAParser::kw_t** (p. 671).

5.265.3.2 std::string MHAParser::variable_t::query_perm (const std::string & *s*) [virtual]

Reimplemented from **MHAParser::monitor_t** (p. 687).

5.265.3.3 void MHAParser::variable_t::setlock (const bool & *b*)

Lock a variable against write access.

Parameters

b	Lock state

5.265.4 Member Data Documentation

5.265.4.1 bool MHAParser::variable_t::locked [private]

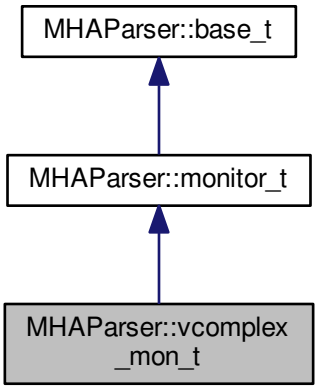
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.266 MHAParser::vcomplex_mon_t Class Reference

Monitor with vector of complex values.

Inheritance diagram for MHAParser::vcomplex_mon_t:



Public Member Functions

- **vcomplex_mon_t** (const std::string &hlp)
  Create a vector of complex monitor values.

Public Attributes

- std::vector< **mha_complex_t** > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.266.1 Detailed Description

Monitor with vector of complex values.

5.266.2 Constructor & Destructor Documentation

5.266.2.1 MHAParser::vcomplex_mon_t::vcomplex_mon_t (const std::string & *hlp*)

Create a vector of complex monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.266.3 Member Function Documentation

5.266.3.1 std::string MHAParser::vcomplex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.266.3.2 std::string MHAParser::vcomplex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.266.4 Member Data Documentation

5.266.4.1 std::vector<mha_complex_t> MHAParser::vcomplex_mon_t::data

Data field.

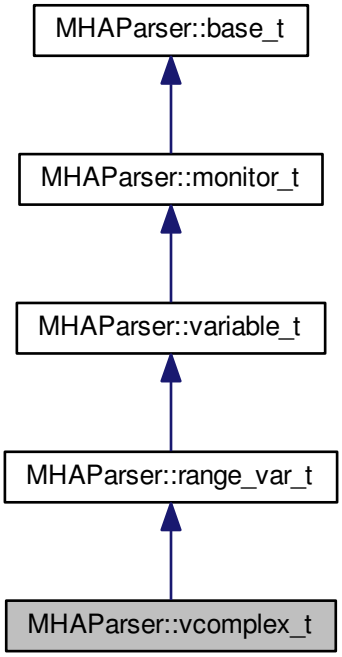
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.267 MHAParser::vcomplex_t Class Reference

Vector variable with complex value.

Inheritance diagram for MHAParser::vcomplex_t:



Public Member Functions

- **vcomplex_t** (const std::string &, const std::string &, const std::string &="")

Public Attributes

- std::vector< **mha_complex_t** > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.267.1 Detailed Description

Vector variable with complex value.

5.267.2 Constructor & Destructor Documentation

5.267.2.1 MHAParser::vcomplex_t::vcomplex_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

5.267.3 Member Function Documentation

5.267.3.1 std::string MHAParser::vcomplex_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.267.3.2 std::string MHAParser::vcomplex_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.267.3.3 std::string MHAParser::vcomplex_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.267.4 Member Data Documentation

5.267.4.1 std::vector<**mha_complex_t**> MHAParser::vcomplex_t::data

Data field.

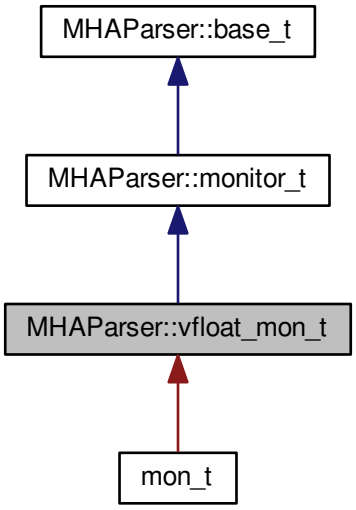
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.268 MHAParser::vfloat_mon_t Class Reference

Vector of floats monitor.

Inheritance diagram for MHAParser::vfloat_mon_t:



Public Member Functions

- **vfloat_mon_t** (const std::string &hlp)
  Create a vector of floating point monitor values.

Public Attributes

- std::vector< float > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.268.1 Detailed Description

Vector of floats monitor.

5.268.2 Constructor & Destructor Documentation

5.268.2.1 MHAParser::vfloat_mon_t::vfloat_mon_t (const std::string & *hlp*)

Create a vector of floating point monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.268.3 Member Function Documentation

5.268.3.1 std::string MHAParser::vfloat_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.268.3.2 std::string MHAParser::vfloat_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.268.4 Member Data Documentation

5.268.4.1 std::vector<float> MHAParser::vfloat_mon_t::data

Data field.

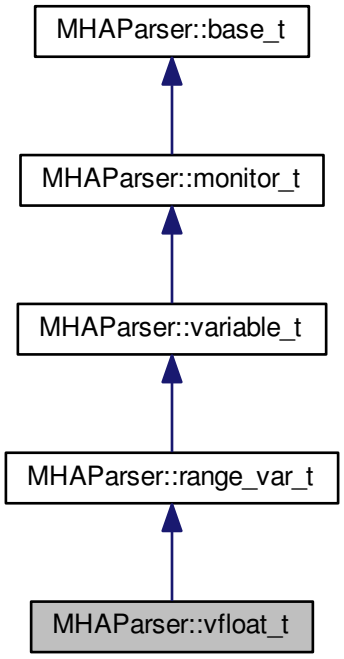
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.269 MHAParser::vfloat_t Class Reference

Vector variable with float value.

Inheritance diagram for MHAParser::vfloat_t:



Public Member Functions

- **vfloat_t** (const std::string &, const std::string &, const std::string &="")
  Create a float vector parser variable.

Public Attributes

- std::vector< float > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.269.1 Detailed Description

Vector variable with float value.

5.269.2 Constructor & Destructor Documentation

5.269.2.1 MHAParser::vfloat_t::vfloat_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

Create a float vector parser variable.

Parameters

h	A human-readable text describing the purpose of this configuration variable.
v	The initial value of the variable, as a string, in openMHA configuration language: (e.g. "[0 1 2.1 3]" for a vector), described in the "Multidimensional Variables" s2.1.3 section of the openMHA User Manual.
rg	The numeric range to enforce on all members of the vector.

-

5.269.3 Member Function Documentation

5.269.3.1 std::string MHAParser::vfloat_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.269.3.2 std::string MHAParser::vfloat_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.269.3.3 std::string MHAParser::vfloat_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.269.4 Member Data Documentation

5.269.4.1 std::vector<float> MHAParser::vfloat_t::data

Data field.

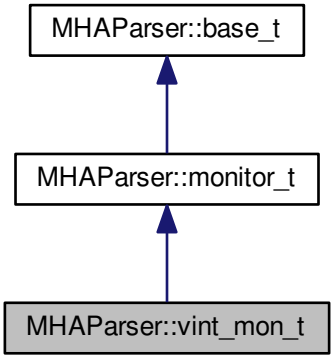
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.270 MHAParser::vint_mon_t Class Reference

Vector of ints monitor.

Inheritance diagram for MHAParser::vint_mon_t:



Public Member Functions

- **vint_mon_t** (const std::string &hlp)
  Create a vector of integer monitor values.

Public Attributes

- std::vector< int > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.270.1 Detailed Description

Vector of ints monitor.

5.270.2 Constructor & Destructor Documentation

5.270.2.1 MHAParser::vint_mon_t::vint_mon_t (const std::string & *hlp*)

Create a vector of integer monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.270.3 Member Function Documentation

5.270.3.1 std::string MHAParser::vint_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.270.3.2 std::string MHAParser::vint_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.270.4 Member Data Documentation

5.270.4.1 std::vector<int> MHAParser::vint_mon_t::data

Data field.

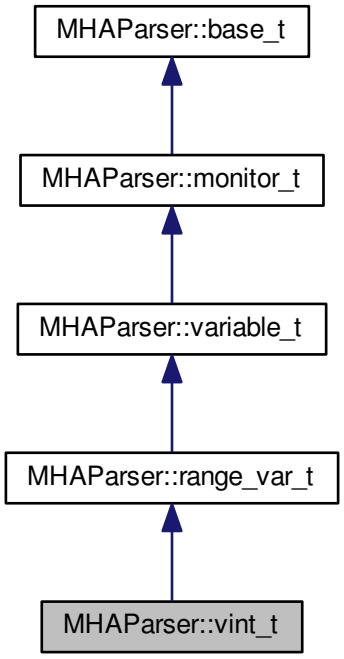
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.271 MHAParser::vint_t Class Reference

Variable with vector<int> value.

Inheritance diagram for MHAParser::vint_t:



Public Member Functions

- **vint_t** (const std::string &, const std::string &, const std::string &="")
  Constructor.

Public Attributes

- std::vector< int > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.271.1 Detailed Description

Variable with vector<int> value.

5.271.2 Constructor & Destructor Documentation

5.271.2.1 MHAParser::vint_t::vint_t (const std::string & *h,* const std::string & *v,* const std::string & *rg* = " ")

Constructor.

Parameters

h	help string
v	initial value
rg	optional: range constraint for all elements

5.271.3 Member Function Documentation

5.271.3.1 std::string MHAParser::vint_t::op_setval (expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.271.3.2 std::string MHAParser::vint_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.271.3.3 std::string MHAParser::vint_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.271.4 Member Data Documentation

5.271.4.1 std::vector<int> MHAParser::vint_t::data

Data field.

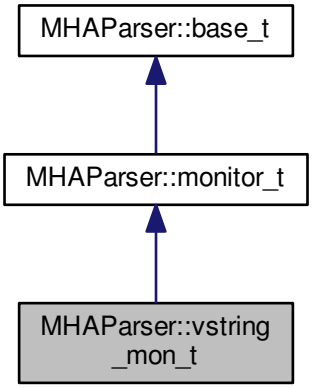
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.272 MHAParser::vstring_mon_t Class Reference

Vector of monitors with string value.

Inheritance diagram for MHAParser::vstring_mon_t:



Public Member Functions

- **vstring_mon_t** (const std::string &hlp)
  Create a vector of string monitor values.

Public Attributes

- std::vector< std::string > **data**
  Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

5.272.1 Detailed Description

Vector of monitors with string value.

5.272.2 Constructor & Destructor Documentation

5.272.2.1 MHAParser::vstring_mon_t::vstring_mon_t (const std::string & *hlp*)

Create a vector of string monitor values.

Parameters

hlp	A help text describing this monitor variable.

5.272.3 Member Function Documentation

5.272.3.1 std::string MHAParser::vstring_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.272.3.2 std::string MHAParser::vstring_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.272.4 Member Data Documentation

5.272.4.1 std::vector<std::string> MHAParser::vstring_mon_t::data

Data field.

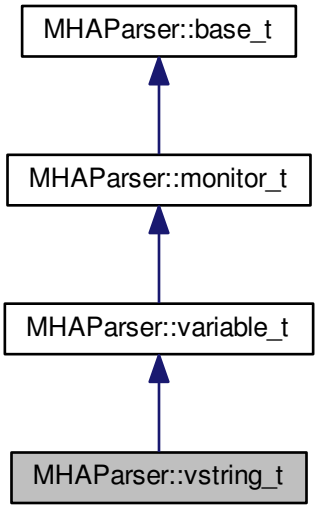
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.273 MHAParser::vstring_t Class Reference

Vector variable with string values.

Inheritance diagram for MHAParser::vstring_t:



Public Member Functions

- **vstring_t** (const std::string &, const std::string &)

Public Attributes

- std::vector< std::string > **data**
  Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

5.273.1 Detailed Description

Vector variable with string values.

5.273.2 Constructor & Destructor Documentation

5.273.2.1 MHAParser::vstring_t::vstring_t (const std::string & *h,* const std::string & *v*)

5.273.3 Member Function Documentation

5.273.3.1 std::string MHAParser::vstring_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 701).

5.273.3.2 std::string MHAParser::vstring_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.273.3.3 std::string MHAParser::vstring_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 639).

5.273.4 Member Data Documentation

5.273.4.1 std::vector<std::string> MHAParser::vstring_t::data

Data field.

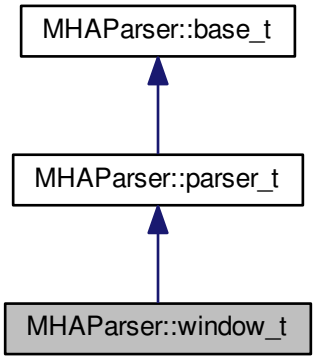
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.274 MHAParser::window_t Class Reference

MHA configuration interface for a window function generator.

Inheritance diagram for MHAParser::window_t:



Public Types

Public Member Functions

- **window_t** (const std::string &**help**="Window type configuration.")
  Constructor to create parser class.
- **MHAWindow::base_t get_window** (unsigned int len) const
  Create a window instance, use default parameters.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin) const
  Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax) const
  Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool minincluded) const
  Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool minincluded, bool maxincluded) const
  Create a window instance.
- **MHAParser::window_t::wtype_t get_type** () const
  Return currently selected window type.

Private Attributes

- **MHAParser::kw_t wtype**
- **MHAParser::vfloat_t user**

Additional Inherited Members

5.274.1 Detailed Description

MHA configuration interface for a window function generator.

This class implements a configuration interface (sub-parser) for window type selection and user-defined window type. It provides member functions to generate an instance of **MHA↩ Window::base_t** (p. 835) based on the values provided by the configuration interface.

The configuration interface is derived from **MHAParser::parser_t** (p. 688) and can thus be inserted into the configuration tree using the **insert_item()** (p. 690) method of the parent parser.

If one of the pre-defined window types is used, then the window is generated using the **MHA↩ Window::fun_t** (p. 838) class constructor; for the user-defined type the values from the "user" variable are copied.

5.274.2 Member Enumeration Documentation

5.274.2.1 enum MHAParser::window_t::wtype_t

Enumerator

wnd_rect

wnd_hann

wnd_hamming

wnd_blackman

wnd_bartlett

wnd_user

5.274.3 Constructor & Destructor Documentation

5.274.3.1 MHAParser::window_t::window_t (const std::string & *help* = "Window type configuration.")

Constructor to create parser class.

5.274.4 Member Function Documentation

5.274.4.1 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len*) const

Create a window instance, use default parameters.

5.274.4.2 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len,* float *xmin*) const

Create a window instance.

5.274.4.3 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len,* float *xmin,* float *xmax*) const

Create a window instance.

5.274.4.4 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len,* float *xmin,* float *xmax,* bool *minincluded*) const

Create a window instance.

5.274.4.5 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len,* float *xmin,* float *xmax,* bool *minincluded,* bool *maxincluded*) const

Create a window instance.

5.274.4.6 MHAParser::window_t::wtype_t MHAParser::window_t::get_type ( ) const

Return currently selected window type.

5.274.5 Member Data Documentation

5.274.5.1 MHAParser::kw_t MHAParser::window_t::wtype [private]

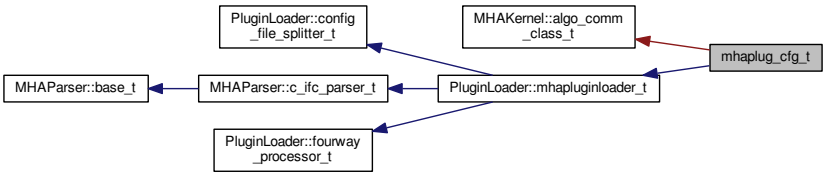
5.274.5.2 MHAParser::vfloat_t MHAParser::window_t::user [private]

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.275 mhaplug_cfg_t Class Reference

Inheritance diagram for mhaplug_cfg_t:



Public Member Functions

- **mhaplug_cfg_t** (**algo_comm_t** iac, const std::string &**libname**, bool use_own_ac)
- ∼**mhaplug_cfg_t** () throw ()

Additional Inherited Members

5.275.1 Constructor & Destructor Documentation

5.275.1.1 mhaplug_cfg_t::mhaplug_cfg_t (algo_comm_t *iac,* const std::string & *libname,* bool *use_own_ac*)

5.275.1.2 mhaplug_cfg_t::∼mhaplug_cfg_t ( ) throw) [inline]

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

- **altplugs.cpp**

5.276 MHAPlugin::cfg_chain_t< runtime_cfg_t > Class Template Reference

Public Member Functions

- **cfg_chain_t** (runtime_cfg_t ∗id)
- ∼**cfg_chain_t** ()

Public Attributes

- **cfg_chain_t**< runtime_cfg_t > * **next**
- bool **not_in_use**
- runtime_cfg_t * **data**

5.276.1 Constructor & Destructor Documentation

5.276.1.1 template<class runtime_cfg_t> MHAPlugin::cfg_chain_t< runtime_cfg_t >::cfg_chain_t (runtime_cfg_t * *id*)

5.276.1.2 template<class runtime_cfg_t > MHAPlugin::cfg_chain_t< runtime_cfg_t >::~cfg_chain_t ( )

5.276.2 Member Data Documentation

5.276.2.1 template<class runtime_cfg_t> cfg_chain_t< runtime_cfg_t >* MHAPlugin::cfg_chain_t< runtime_cfg_t >::next

5.276.2.2 template<class runtime_cfg_t> bool MHAPlugin::cfg_chain_t< runtime_cfg_t >::not_in_use

5.276.2.3 template<class runtime_cfg_t> runtime_cfg_t* MHAPlugin::cfg_chain_t< runtime_cfg_t >::data

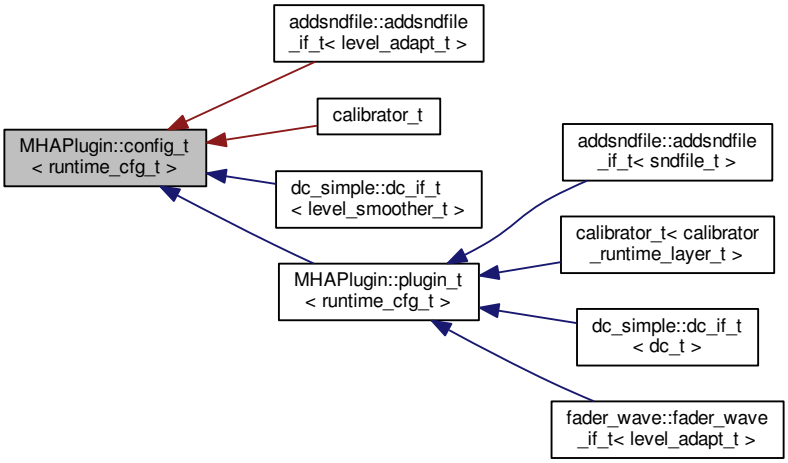
The documentation for this class was generated from the following file:

- **mha_plugin.hh**

5.277 MHAPlugin::config_t< runtime_cfg_t > Class Template Reference

Template class for thread safe configuration.

Inheritance diagram for MHAPlugin::config_t< runtime_cfg_t >:



Public Member Functions

- **config_t** ()
- ∼**config_t** ()

Protected Member Functions

- runtime_cfg_t ∗ **poll_config** ()
  Receive the latest run time configuration.
- runtime_cfg_t ∗ **last_config** ()
  Receive the latest run time configuration.
- void **push_config** (runtime_cfg_t ∗ncfg)
  Push a new run time configuration into the configuration fifo.
- void **cleanup_unused_cfg** ()

Protected Attributes

- runtime_cfg_t ∗ **cfg**

Private Member Functions

- void **remove_all_cfg** ()

Private Attributes

- **MHAPlugin::cfg_chain_t**< runtime_cfg_t > ∗ **cfg_chain**
- **MHAPlugin::cfg_chain_t**< runtime_cfg_t > ∗ **cfg_chain_current**

5.277.1 Detailed Description

template<**class runtime_cfg_t**>
class MHAPlugin::config_t< **runtime_cfg_t** >

Template class for thread safe configuration.

This template class provides a mechanism for the handling of thread safe configuration which is required for run time configuration changes of the openMHA plugins.

The template parameter runtime_cfg_t is the run time configuration class of the openMHA plugin. The constructor of that class should transform the **MHAParser** (p. 107) variables into derived runtime configuration. The constructor should fail if the configuration is invalid by any reason.

A new runtime configuration is provided by the function **push_config()** (p. 726). In the processing thread, the actual configuration can be received by a call of **poll_config()** (p. 726).

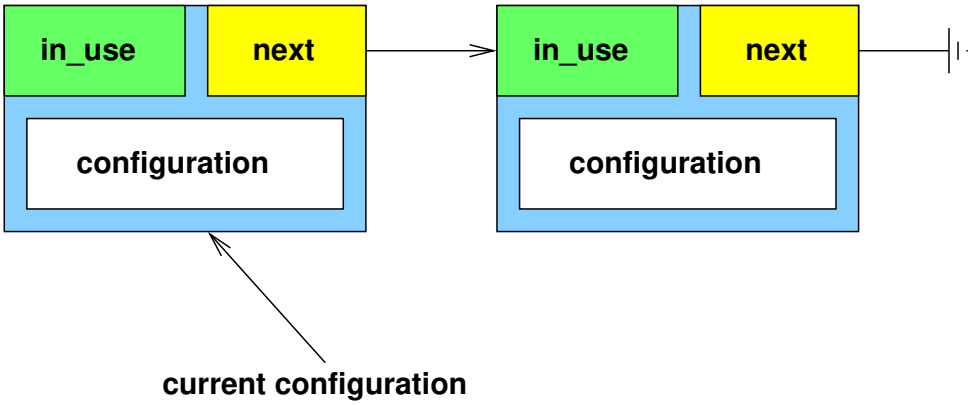


Figure 5 Schematic drawing of runtime configuration update: configuration updated, but not used yet.

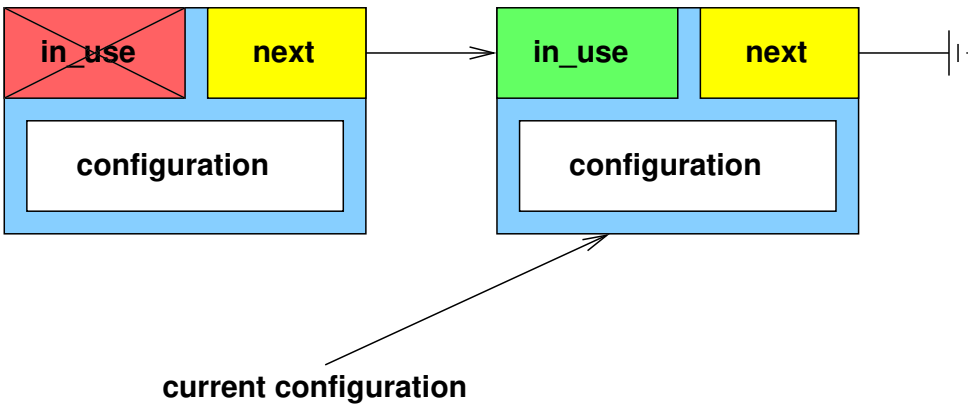


Figure 6 Schematic drawing of runtime configuration update: configuration in use.

5.277.2 Constructor & Destructor Documentation

5.277.2.1 template<class runtime_cfg_t > MHAPlugin::config_t< runtime_cfg_t >::config_t ( )

5.277.2.2 template<class runtime_cfg_t > MHAPlugin::config_t< runtime_cfg_t >::~config_t ( )

5.277.3 Member Function Documentation

5.277.3.1 template<class runtime_cfg_t > runtime_cfg_t * MHAPlugin::config_t< runtime_cfg_t >::poll_config ( ) [protected]

Receive the latest run time configuration.

This function stores the latest run time configuration into the protected class member variable 'cfg'. If no configuration exists, then an exception will be thrown. If no changes occured, then the value of 'cfg' will be untouched. This function should be called before any access to the 'cfg' variable, typically once in each signal processing call.

This function should be only called from the *processing* thread.

Exceptions

MHA_Error *(p. 445)*	if the resulting runtime configuration is NULL. This usually means that no push_config has occured.

5.277.3.2 template<class runtime_cfg_t > runtime_cfg_t * MHAPlugin::config_t< runtime_cfg_t >::last_config ( ) [protected]

Receive the latest run time configuration.

This function stores the latest run time configuration into the protected class member variable 'cfg'. If no configuration exists, then an exception will be thrown. If no changes occured, then the value of 'cfg' will be untouched. This function may be called instead of poll_config.

The difference between poll_config and last_config is that poll_config marks previous configurations as ready for deletion, while this function does not. Therefore, memory usage of all runtime configurations will accumulate if only this function is called, but it enables safe access to previous runtime configurations.

Also, last_config does not raise an Exception when the latest run time configuration is NULL.

5.277.3.3 template<class runtime_cfg_t> void MHAPlugin::config_t< runtime_cfg_t >::push_config (runtime_cfg_t * *ncfg*) [protected]

Push a new run time configuration into the configuration fifo.

This function adds a new run time configuration. The next time **poll_config** (p. 726) is called, this configuration will be available. Configurations which are not in use or are outdated will be removed.

This function should be only called from the *configuration* thread.

Parameters

ncfg	pointer on a new configuration

Warning

The runtime configuration passed to this function will be removed by the internal garbage collector. Do not free manually.

5.277.3.4 template<class runtime_cfg_t > void MHAPlugin::config_t< runtime_cfg_t >::cleanup_unused_cfg ( ) [protected]

5.277.3.5 template<class runtime_cfg_t > void MHAPlugin::config_t< runtime_cfg_t >::remove_all_cfg ( ) [private]

5.277.4 Member Data Documentation

5.277.4.1 template<class runtime_cfg_t> runtime_cfg_t* MHAPlugin::config_t< runtime_cfg_t >::cfg [protected]

5.277.4.2 template<class runtime_cfg_t> MHAPlugin::cfg_chain_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_chain [private]

5.277.4.3 template<class runtime_cfg_t> MHAPlugin::cfg_chain_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_chain_current [private]

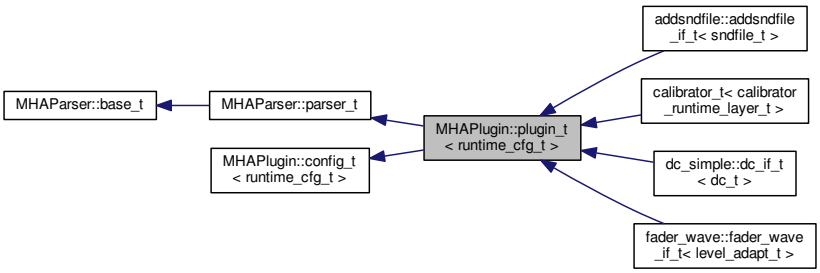
The documentation for this class was generated from the following file:

- **mha_plugin.hh**

5.278 MHAPlugin::plugin_t< runtime_cfg_t > Class Template Reference

The template class for C++ openMHA plugins.

Inheritance diagram for MHAPlugin::plugin_t< runtime_cfg_t >:



Public Member Functions

- **plugin_t** (const std::string &, const **algo_comm_t** &)
  Constructor of plugin template.
- virtual ~**plugin_t** ()
- virtual void **prepare** (**mhaconfig_t** &)=0
- virtual void **release** ()
- void **prepare_** (**mhaconfig_t** &)
- void **release_** ()
- bool **is_prepared** () const
  Flag, if the prepare method is successfully called (or currently evaluated)
- **mhaconfig_t input_cfg** () const
  Current input channel configuration.
- **mhaconfig_t output_cfg** () const
  Current output channel configuration.

Protected Attributes

- **mhaconfig_t tftype**
  Member for storage of plugin interface configuration.
- **algo_comm_t ac**
  AC handle of the chain.

Private Attributes

- bool **is_prepared_**
- **mhaconfig_t input_cfg_**
- **mhaconfig_t output_cfg_**
- **MHAParser::mhaconfig_mon_t mhaconfig_in**
- **MHAParser::mhaconfig_mon_t mhaconfig_out**

Additional Inherited Members

5.278.1 Detailed Description

template<class runtime_cfg_t>
class MHAPlugin::plugin_t< runtime_cfg_t >

The template class for C++ openMHA plugins.

Template Parameters

runtime_cfg_t	run-time configuration.

Todo Describe all services provided by this class, so that the reason why it is recommended that all plugins use this class as their base is evident. Document all relevant methods and fields.

This template class provides thread safe configuration handling and standard methods to be compatible to the C++ openMHA plugin wrapper macro **MHAPLUGIN_CALLBACKS** (p. 10).

The template parameter runtime_cfg_t should be the runtime configuration of the plugin.

See **MHAPlugin::config_t** (p. 723) for details on the thread safe communication update mechanism.

5.278.2 Constructor & Destructor Documentation

5.278.2.1 template<class runtime_cfg_t > MHAPlugin::plugin_t< runtime_cfg_t >::plugin_t (const std::string & *help,* const algo_comm_t & *iac*)

Constructor of plugin template.

Parameters

help	Help comment to provide some general information about the plugin.
iac	AC space handle (will be stored into the member variable ac).

5.278.2.2 template<class runtime_cfg_t > MHAPlugin::plugin_t< runtime_cfg_t >::~plugin_t ( ) [virtual]

5.278.3 Member Function Documentation

5.278.3.1 template<class runtime_cfg_t> virtual void MHAPlugin::plugin_t< runtime_cfg_t >::prepare (mhaconfig_t &) [pure virtual]

Implemented in **bbcalib_interface_t** (p. 229), **calibrator_t** (p. 232), **addsndfile::addsndfile_if_t** (p. 181), **testplugin::if_t** (p. 935), **analysispath_if_t** (p. 220), **adm_if_t** (p. 196), **ac2lsl::ac2lsl_t** (p. 132), **overlapadd::overlapadd_if_t** (p. 860), **noiseProposedScale::interface_t** (p. 856), **dc_simple::dc_if_t** (p. 264), **dc::dc_if_t** (p. 254),

multibandcompressor::interface_t (p. 848), **plingploing::if_t** (p. 866), **combc_if_t** (p. 244), **gtfb_analyzer::gtfb_analyzer_t** (p. 357), **coherence::cohflt_if_t** (p. 238), **plugin_interface_t** (p. 873), **example6_t** (p. 317), **smoothgains_bridge::overlapadd_if_t** (p. 913), **MHAPlugin_Resampling::resampling_if_t** (p. 734), **shadowfilter_end::shadowfilter_end_t** (p. 909), **ac2wave_if_t** (p. 149), **noise_t** (p. 854), **nlms_t** (p. 852), **prediction_error** (p. 888), **fshift_hilbert::frequency_translator_t** (p. 338), **acPooling_wave** (p. 161), **spec2wave_if_t** (p. 921), **mhachain::chain_base_t** (p. 499), **acsave::acsave_t** (p. 166), **fader_wave::fader_wave_if_t** (p. 321), **doasvm_feature_extraction** (p. 284), **fshift::fshift_t** (p. 335), **rmslevel_if_t** (p. 893), **shadowfilter_begin::shadowfilter_begin_t** (p. 906), **example3_t** (p. 310), **example4_t** (p. 314), **lpc_bl_predictor** (p. 399), **lpc_burglattice** (p. 404), **steerbf** (p. 927), **delaysum::delaysum_if_t** (p. 276), **lpc** (p. 396), **db_if_t** (p. 250), **fader_if_t** (p. 319), **acConcat_wave** (p. 153), **acSteer** (p. 173), **acTransform_wave** (p. 177), **wave2spec_if_t** (p. 950), **fftfbpow::fftfbpow_interface_t** (p. 325), **gain::gain_if_t** (p. 351), **droptect_t** (p. 288), **example1_t** (p. 304), **sine_t** (p. 911), **example2_t** (p. 307), **doasvm_classification** (p. 280), **fftfilterbank::fftfb_interface_t** (p. 329), **wavrec_t** (p. 954), **matrixmixer::matmix_t** (p. 411), **route::interface_t** (p. 896), **altplugs_t** (p. 214), **softclip_t** (p. 917), **ac2osc_t** (p. 146), **save_spec_t** (p. 902), **save_wave_t** (p. 903), **acmon::acmon_t** (p. 158), **timoSmooth** (p. 946), **identity_t** (p. 360), **cpuload_t** (p. 248), **delay::interface_t** (p. 274), **ds_t** (p. 291), and **us_t** (p. 948).

5.278.3.2 template<class runtime_cfg_t > void MHAPlugin::plugin_t< runtime_cfg_t >::release ( ) [virtual]

Reimplemented in **bbcalib_interface_t** (p. 229), **calibrator_t** (p. 232), **addsndfile::addsndfile_if_t** (p. 181), **analysispath_if_t** (p. 220), **ac2lsl::ac2lsl_t** (p. 132), **adm_if_t** (p. 196), **overlapadd::overlapadd_if_t** (p. 861), **dc_simple::dc_if_t** (p. 264), **multibandcompressor::interface_t** (p. 848), **coherence::cohflt_if_t** (p. 238), **smoothgains_bridge::overlapadd_if_t** (p. 913), **MHAPlugin_Resampling::resampling_if_t** (p. 734), **ac2wave_if_t** (p. 149), **nlms_t** (p. 852), **prediction_error** (p. 888), **fshift_hilbert::frequency_translator_t** (p. 338), **acPooling_wave** (p. 162), **mhachain::chain_base_t** (p. 499), **acsave::acsave_t** (p. 166), **fader_wave::fader_wave_if_t** (p. 321), **doasvm_feature_extraction** (p. 285), **fshift::fshift_t** (p. 336), **example3_t** (p. 311), **example4_t** (p. 314), **lpc_bl_predictor** (p. 400), **lpc_burglattice** (p. 404), **steerbf** (p. 928), **delaysum::delaysum_if_t** (p. 276), **lpc** (p. 397), **db_if_t** (p. 250), **acConcat_wave** (p. 153), **acSteer** (p. 174), **acTransform_wave** (p. 178), **droptect_t** (p. 288), **gain::gain_if_t** (p. 351), **example2_t** (p. 307), **doasvm_classification** (p. 281), **fftfilterbank::fftfb_interface_t** (p. 329), **wavrec_t** (p. 954), **route::interface_t** (p. 896), **ac2osc_t** (p. 146), **example1_t** (p. 304), **altplugs_t** (p. 215), **acmon::acmon_t** (p. 159), **timoSmooth** (p. 946), **identity_t** (p. 360), **ds_t** (p. 291), and **us_t** (p. 948).

5.278.3.3 template<class runtime_cfg_t > void MHAPlugin::plugin_t< runtime_cfg_t >::prepare_ (mhaconfig_t & *cf*)

5.278.3.4 template<class runtime_cfg_t > void MHAPlugin::plugin_t< runtime_cfg_t >::release_ ( )

5.278.3.5 template<class runtime_cfg_t> bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared ( ) const [inline]

Flag, if the prepare method is successfully called (or currently evaluated)

5.278.3.6 template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg ( ) const [inline]

Current input channel configuration.

5.278.3.7 template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg ( ) const [inline]

Current output channel configuration.

5.278.4 Member Data Documentation

5.278.4.1 template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::tftype [protected]

Member for storage of plugin interface configuration.

This member is defined for convenience of the developer. Typically, the actual contents of **mhaconfig_t** (p. 504) are stored in this member in the **prepare()** (p. 730) method.

Note

This member is likely to be removed in later versions, use **input_cfg()** (p. 732) and **output_cfg()** (p. 732) instead.

5.278.4.2 template<class runtime_cfg_t> algo_comm_t MHAPlugin::plugin_t< runtime_cfg_t >::ac [protected]

AC handle of the chain.

This variable is initialized in the constructor and can be used by derived plugins to access the AC space. Its contents should not be modified.

5.278.4.3 template<class runtime_cfg_t> bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared_ [private]

5.278.4.4 template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg_ [private]

5.278.4.5 template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg_ [private]

5.278.4.6 template<class runtime_cfg_t> MHAParser::mhaconfig_mon_t MHAPlugin::plugin_t< runtime_cfg_t >::mhaconfig_in [private]

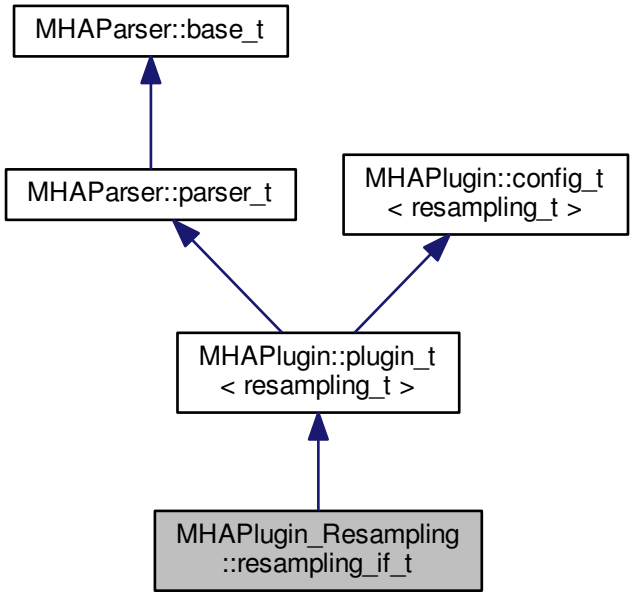
5.278.4.7 template<class runtime_cfg_t> MHAParser::mhaconfig_mon_t MHAPlugin::plugin_t< runtime_cfg_t >::mhaconfig_out [private]

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

- **mha_plugin.hh**

5.279 MHAPlugin_Resampling::resampling_if_t Class Reference

Inheritance diagram for MHAPlugin_Resampling::resampling_if_t:



Public Member Functions

- **resampling_if_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Attributes

- **MHAParser::float_t srate**
- **MHAParser::int_t fragsize**
- **MHAParser::float_t nyquist_ratio**
- **MHAParser::float_t irslen_outer2inner**
- **MHAParser::float_t irslen_inner2outer**
- **MHAParser::mhapluginloader_t plugloader**
- std::string **chain**
- std::string **algo**

Additional Inherited Members

5.279.1 Constructor & Destructor Documentation

5.279.1.1 MHAPlugin_Resampling::resampling_if_t::resampling_if_t (algo_comm_t *iac,* std::string *th,* std::string *al*)

5.279.2 Member Function Documentation

5.279.2.1 mha_wave_t * MHAPlugin_Resampling::resampling_if_t::process (mha_wave_t * *s*)

5.279.2.2 void MHAPlugin_Resampling::resampling_if_t::prepare (mhaconfig_t & *conf*) [virtual]

Implements **MHAPlugin::plugin_t< resampling_t >** (p. 730).

5.279.2.3 void MHAPlugin_Resampling::resampling_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< resampling_t >** (p. 731).

5.279.3 Member Data Documentation

5.279.3.1 MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::srate [private]

5.279.3.2 MHAParser::int_t MHAPlugin_Resampling::resampling_if_t::fragsize [private]

5.279.3.3 MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::nyquist_ratio [private]

5.279.3.4 MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::irslen_outer2inner [private]

5.279.3.5 MHAParser::float_t MHAPlugin_Resampling::resampling_if_t::irslen_inner2outer [private]

5.279.3.6 MHAParser::mhapluginloader_t MHAPlugin_Resampling::resampling_if_t::plugloader [private]

5.279.3.7 std::string MHAPlugin_Resampling::resampling_if_t::chain [private]

5.279.3.8 std::string MHAPlugin_Resampling::resampling_if_t::algo [private]

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

- **resampling.cpp**

5.280 MHAPlugin_Resampling::resampling_t Class Reference

Public Member Functions

- **resampling_t** (unsigned int **outer_fragsize**, float **outer_srate**, unsigned int **inner_↩ fragsize**, float inner_scrate, unsigned int nch_in, float filter_length_in, unsigned int nch↩ _out, float filter_length_out, float nyquist_ratio, **MHAParser::mhapluginloader_t** &plug)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- unsigned **outer_fragsize**
- unsigned **inner_fragsize**
- float **outer_srate**
- float **inner_srate**
- unsigned **nchannels_in**
- unsigned **nchannels_out**
- **MHAFilter::blockprocessing_polyphase_resampling_t outer2inner_resampling**
- **MHAFilter::blockprocessing_polyphase_resampling_t inner2outer_resampling**
- **MHAParser::mhapluginloader_t** & **plugloader**
- **MHASignal::waveform_t inner_signal**
- **MHASignal::waveform_t output_signal**

5.280.1 Constructor & Destructor Documentation

5.280.1.1 MHAPlugin_Resampling::resampling_t::resampling_t (unsigned int *outer_fragsize,* float *outer_srate,* unsigned int *inner_fragsize,* float *inner_scrate,* unsigned int *nch_in,* float *filter_length_in,* unsigned int *nch_out,* float *filter_length_out,* float *nyquist_ratio,* **MHAParser::mhapluginloader_t** & *plug*)

5.280.2 Member Function Documentation

5.280.2.1 mha_wave_t ∗ MHAPlugin_Resampling::resampling_t::process (mha_wave_t ∗ *s*)

5.280.3 Member Data Documentation

5.280.3.1 unsigned MHAPlugin_Resampling::resampling_t::outer_fragsize [private]

5.280.3.2 unsigned MHAPlugin_Resampling::resampling_t::inner_fragsize [private]

5.280.3.3 float MHAPlugin_Resampling::resampling_t::outer_srate [private]

5.280.3.4 float MHAPlugin_Resampling::resampling_t::inner_srate [private]

5.280.3.5 unsigned MHAPlugin_Resampling::resampling_t::nchannels_in [private]

5.280.3.6 unsigned MHAPlugin_Resampling::resampling_t::nchannels_out [private]

5.280.3.7 MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugin_Resampling::resampling_t::outer2inner_resampling [private]

5.280.3.8 MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugin_Resampling::resampling_t::inner2outer_resampling [private]

5.280.3.9 MHAParser::mhapluginloader_t& MHAPlugin_Resampling::resampling_t::plugloader [private]

5.280.3.10 MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::inner_signal [private]

5.280.3.11 MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::output_signal [private]

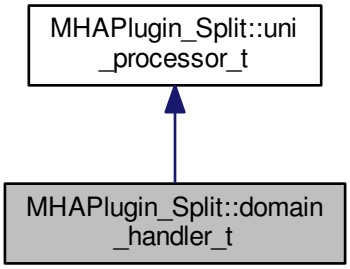
The documentation for this class was generated from the following file:

- **resampling.cpp**

5.281 MHAPlugin_Split::domain_handler_t Class Reference

Handles domain-specific partial input and output signal.

Inheritance diagram for MHAPlugin_Split::domain_handler_t:



Public Member Functions

- void **set_input_domain** (const **mhaconfig_t** &settings_in)
  Set parameters of input signal.
- void **set_output_domain** (const **mhaconfig_t** &settings_out)
  Set output signal parameters.
- void **deallocate_domains** ()
  Deallocate domain indicators and signal holders.
- **domain_handler_t** (const **mhaconfig_t** &settings_in, const **mhaconfig_t** &settings_out, **PluginLoader::fourway_processor_t** ∗**processor**)
  Construct a new domain handler once the domains and dimensions of input and output signal of one of the child plugins of split are known.
- virtual ∼**domain_handler_t** ()
  Deallocation of signal holders.
- unsigned **put_signal** (**mha_wave_t** ∗s_in, unsigned start_channel)
  Store the relevant channels from the input signal for processing.
- unsigned **put_signal** (**mha_spec_t** ∗s_in, unsigned start_channel)
  Store the relevant channels from the input signal for processing.
- unsigned **get_signal** (**MHASignal::waveform_t** ∗s_out, unsigned start_channel)
  Store all partial signal output channels in the combined waveform signal with the given channel offset.
- unsigned **get_signal** (**MHASignal::spectrum_t** ∗s_out, unsigned start_channel)
  Store all partial signal output channels in the combined spectrum signal with the given channel offset.
- void **process** ()
  Call the processing method of the processor with configured input/output signal domains.

Public Attributes

- **MHASignal::waveform_t** ∗ **wave_in**
  Partial wave input signal.
- **mha_wave_t** ∗∗ **wave_out**
  Partial wave output signal.
- **MHASignal::spectrum_t** ∗ **spec_in**
  Partial spec input signal.
- **mha_spec_t** ∗∗ **spec_out**
  Partial spec input signal.
- **PluginLoader::fourway_processor_t** ∗ **processor**
  The domain-specific signal processing methods are implemented here.

Private Member Functions

- **domain_handler_t** (const **domain_handler_t** &)
  Disallow copy constructor.
- **domain_handler_t** & **operator=** (const **domain_handler_t** &)
  Disallow assignment operator.

5.281.1 Detailed Description

Handles domain-specific partial input and output signal.

5.281.2 Constructor & Destructor Documentation

5.281.2.1 MHAPlugin_Split::domain_handler_t::domain_handler_t (const domain_handler_t &) [private]

Disallow copy constructor.

5.281.2.2 MHAPlugin_Split::domain_handler_t::domain_handler_t (const mhaconfig_t & *settings_in,* const mhaconfig_t & *settings_out,* PluginLoader::fourway_processor_t * *processor*) [inline]

Construct a new domain handler once the domains and dimensions of input and output signal of one of the child plugins of split are known.

5.281.2.3 virtual MHAPlugin_Split::domain_handler_t::~domain_handler_t ( ) [inline], [virtual]

Deallocation of signal holders.

5.281.3 Member Function Documentation

5.281.3.1 domain_handler_t& MHAPlugin_Split::domain_handler_t::operator= (const domain_handler_t &) [private]

Disallow assignment operator.

5.281.3.2 void MHAPlugin_Split::domain_handler_t::set_input_domain (const mhaconfig_t & *settings_in*) [inline]

Set parameters of input signal.

Parameters

settings_in	domain and dimensions of partial input signal

5.281.3.3 void MHAPlugin_Split::domain_handler_t::set_output_domain (const mhaconfig_t & *settings_out*) [inline]

Set output signal parameters.

Parameters

settings_out	domain and dimensions of partial output signal

5.281.3.4 void MHAPlugin_Split::domain_handler_t::deallocate_domains ( ) [inline]

Deallocate domain indicators and signal holders.

5.281.3.5 unsigned MHAPlugin_Split::domain_handler_t::put_signal (mha_wave_t * *s_in*, unsigned *start_channel*) [inline]

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **wave_in** (p. 741).

Parameters

s_in	The combined waveform input signal.
start_channel	The index (0-based) of the first channel in s_in to be copied to the partial input signal.

Returns

The number of channels that were copied from the input signal

5.281.3.6 unsigned MHAPlugin_Split::domain_handler_t::put_signal (mha_spec_t * *s_in*, unsigned *start_channel*) [inline]

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **spec_in** (p. 741).

Parameters

s_in	The combined spectrum input signal.
start_channel	The index (0-based) of the first channel in s_in to be copied to the partial input signal.

Returns

The number of channels that were copied from the input signal

5.281.3.7 unsigned MHAPlugin_Split::domain_handler_t::get_signal (MHASignal::waveform_t * *s_out,* unsigned *start_channel*) [inline]

Store all partial signal output channels in the combined waveform signal with the given channel offset.

All channels present in **wave_out** (p. 741) will be copied. Caller may use (*wave_out)->num↩ _channels to check the number of channels in advance.

Parameters

s_out	The combined waveform output signal.
start_channel	The channel offset (0-based) in s_out.

Returns

The number of channels that were copied to the output signal

5.281.3.8 unsigned MHAPlugin_Split::domain_handler_t::get_signal (MHASignal::spectrum_t * *s_out,* unsigned *start_channel*) [inline]

Store all partial signal output channels in the combined spectrum signal with the given channel offset.

All channels present in **spec_out** (p. 741) will be copied. Caller may use (*spec_out)->num↩ _channels to check the number of channels in advance.

Parameters

s_out	The combined spectrum output signal.
start_channel	The channel offset (0-based) in s_out.

Returns

The number of channels that were copied to the output signal

5.281.3.9 void MHAPlugin_Split::domain_handler_t::process (void) [inline], [virtual]

Call the processing method of the processor with configured input/output signal domains.

The input signal has to be stored using **put_signal** (p. 739) before this method may be called.

Implements **MHAPlugin_Split::uni_processor_t** (p. 760).

5.281.4 Member Data Documentation

5.281.4.1 MHASignal::waveform_t* MHAPlugin_Split::domain_handler_t::wave_in

Partial wave input signal.

5.281.4.2 mha_wave_t** MHAPlugin_Split::domain_handler_t::wave_out

Partial wave output signal.

5.281.4.3 MHASignal::spectrum_t* MHAPlugin_Split::domain_handler_t::spec_in

Partial spec input signal.

5.281.4.4 mha_spec_t** MHAPlugin_Split::domain_handler_t::spec_out

Partial spec input signal.

5.281.4.5 PluginLoader::fourway_processor_t* MHAPlugin_Split::domain_handler_t::processor

The domain-specific signal processing methods are implemented here.

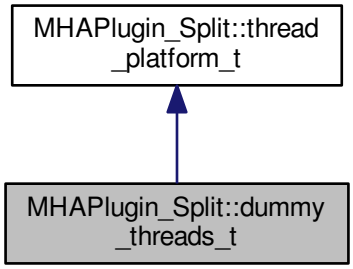
The documentation for this class was generated from the following file:

- **split.cpp**

5.282 MHAPlugin_Split::dummy_threads_t Class Reference

Dummy specification of a thread platform: This class implements everything in a single thread.

Inheritance diagram for MHAPlugin_Split::dummy_threads_t:



Public Member Functions

- void **kick_thread** ()

  perform signal processing immediately (no multiple threads in this dummy class)
- void **catch_thread** ()

  No implementation needed: Processing has been completed during ummy_threads_t::kick_↩ thread.
- **dummy_threads_t** (**uni_processor_t** ∗proc, const std::string &thread_scheduler, int thread_priority)

  Constructor.

Additional Inherited Members

5.282.1 Detailed Description

Dummy specification of a thread platform: This class implements everything in a single thread.

5.282.2 Constructor & Destructor Documentation

5.282.2.1 MHAPlugin_Split::dummy_threads_t::dummy_threads_t (uni_processor_t ∗ *proc,* const std::string & *thread_scheduler,* int *thread_priority*) [inline]

Constructor.

Parameters

proc	Pointer to the associated plugin loader
thread_scheduler	Unused in dummy thread platform
thread_priority	Unused in dummy thread platform

5.282.3 Member Function Documentation

5.282.3.1 void MHAPlugin_Split::dummy_threads_t::kick_thread ( ) [inline], [virtual]

perform signal processing immediately (no multiple threads in this dummy class)

Implements **MHAPlugin_Split::thread_platform_t** (p. 758).

5.282.3.2 void MHAPlugin_Split::dummy_threads_t::catch_thread ( ) [inline],[virtual]

No implementation needed: Processing has been completed during ummy_threads_t::kick_↩ thread.

Implements **MHAPlugin_Split::thread_platform_t** (p. 758).

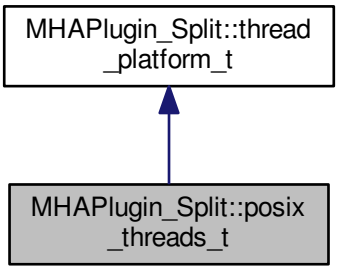
The documentation for this class was generated from the following file:

- **split.cpp**

5.283 MHAPlugin_Split::posix_threads_t Class Reference

Posix threads specification of thread platform.

Inheritance diagram for MHAPlugin_Split::posix_threads_t:



Public Member Functions

- void **kick_thread** ()
  Start signal processing in separate thread.
- void **catch_thread** ()
  Wait for signal processing to finish.
- **posix_threads_t** (**uni_processor_t** ∗proc, const std::string &thread_scheduler, int thread_priority)
  Constructor.
- ∼**posix_threads_t** ()
  Terminate thread.
- void **main** ()
  Thread main loop. Wait for process/termination trigger, then act.

Static Public Member Functions

- static void * **thread_start** (void *thr)

  Thread start function.
- static std::string **current_thread_scheduler** ()
- static int **current_thread_priority** ()

Private Attributes

- pthread_mutex_t **mutex**

  The mutex.
- pthread_cond_t **kick_condition**

  The condition for signalling the kicking and termination.
- pthread_cond_t **catch_condition**

  The condition for signalling the processing is finished.
- pthread_attr_t **attr**

  Thread attributes.
- struct sched_param **priority**

  Thread scheduling priority.
- int **scheduler**
- pthread_t **thread**

  The thread object.
- bool **kicked**

  A flag that is set to true by kick_thread and to false by the thread after it has woken up from the kicking.
- bool **processing_done**

  A flag that is set to true by the thread when it returns from processing and to false by catch_↩ thread after it has waited for that return.
- bool **termination_request**

  Set to true by the destructor.

Additional Inherited Members

5.283.1 Detailed Description

Posix threads specification of thread platform.

5.283.2 Constructor & Destructor Documentation

5.283.2.1 MHAPlugin_Split::posix_threads_t::posix_threads_t (uni_processor_t * *proc*, const std::string & *thread_scheduler*, int *thread_priority*) [inline]

Constructor.

Parameters

proc	Pointer to the associated signal processor instance
thread_scheduler	A string describing the posix thread scheduler. Possible values: "SCHED_OTHER", "SCHED_RR", "SCHED_FIFO".
thread_priority	The scheduling priority of the new thread.

5.283.2.2 MHAPlugin_Split::posix_threads_t::~posix_threads_t ( ) [inline]

Terminate thread.

5.283.3 Member Function Documentation

5.283.3.1 void MHAPlugin_Split::posix_threads_t::kick_thread ( ) [inline], [virtual]

Start signal processing in separate thread.

Implements **MHAPlugin_Split::thread_platform_t** (p. 758).

5.283.3.2 void MHAPlugin_Split::posix_threads_t::catch_thread ( ) [inline], [virtual]

Wait for signal processing to finish.

Implements **MHAPlugin_Split::thread_platform_t** (p. 758).

5.283.3.3 static void* MHAPlugin_Split::posix_threads_t::thread_start (void * *thr*) [inline], [static]

Thread start function.

5.283.3.4 void MHAPlugin_Split::posix_threads_t::main ( ) [inline]

Thread main loop. Wait for process/termination trigger, then act.

5.283.3.5 static std::string MHAPlugin_Split::posix_threads_t::current_thread_scheduler ( ) [inline], [static]

5.283.3.6 static int MHAPlugin_Split::posix_threads_t::current_thread_priority ( ) [inline], [static]

5.283.4 Member Data Documentation

5.283.4.1 pthread_mutex_t MHAPlugin_Split::posix_threads_t::mutex [private]

The mutex.

5.283.4.2 pthread_cond_t MHAPlugin_Split::posix_threads_t::kick_condition [private]

The condition for signalling the kicking and termination.

5.283.4.3 pthread_cond_t MHAPlugin_Split::posix_threads_t::catch_condition [private]

The condition for signalling the processing is finished.

5.283.4.4 pthread_attr_t MHAPlugin_Split::posix_threads_t::attr [private]

Thread attributes.

5.283.4.5 struct sched_param MHAPlugin_Split::posix_threads_t::priority [private]

Thread scheduling priority.

5.283.4.6 int MHAPlugin_Split::posix_threads_t::scheduler [private]

5.283.4.7 pthread_t MHAPlugin_Split::posix_threads_t::thread [private]

The thread object.

5.283.4.8 bool MHAPlugin_Split::posix_threads_t::kicked [private]

A flag that is set to true by kick_thread and to false by the thread after it has woken up from the kicking.

5.283.4.9 bool MHAPlugin_Split::posix_threads_t::processing_done [private]

A flag that is set to true by the thread when it returns from processing and to false by catch_↩ thread after it has waited for that return.

5.283.4.10 bool MHAPlugin_Split::posix_threads_t::termination_request [private]

Set to true by the destructor.

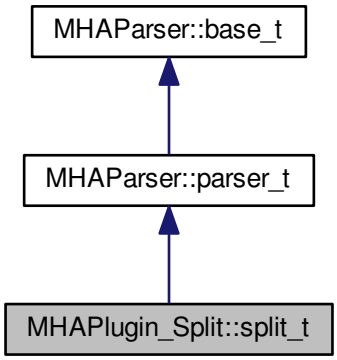
The documentation for this class was generated from the following file:

- **split.cpp**

5.284 MHAPlugin_Split::split_t Class Reference

Implements split plugin.

Inheritance diagram for MHAPlugin_Split::split_t:



Public Member Functions

- **split_t** (**algo_comm_t** iac, const std::string &chain_name, const std::string &algo_name)
  Plugin constructor.
- ∼**split_t** ()
  Plugin destructor. Unloads nested plugins.
- void **prepare_** (**mhaconfig_t** &)
  Check signal parameters, prepare chains, and allocate output signal holders.
- void **release_** ()
  Delete output signal holder and release chains.
- template<class SigTypeIn , class SigTypeOut >
  void **process** (SigTypeIn ∗, SigTypeOut ∗∗)
  Let the parallel plugins process channel groups of the input signal.

Private Member Functions

- void **update** ()
  Load plugins in response to a value change in the algos variable.
- void **clear_chains** ()
  Unload the plugins.
- **mha_wave_t** ∗ **copy_output_wave** ()
- **mha_spec_t** ∗ **copy_output_spec** ()

- template<class SigType >
  void **trigger_processing** (SigType *s_in)
  Split the argument input signal to groups of channels for the plugins and initiate signal processing.
- template<class SigType >
  void **collect_result** (SigType *s_out)
  Combine the output signal from the plugins.
- **MHASignal::waveform_t * signal_out** (**mha_wave_t** **)
  Waveform domain output signal structure accessor.
- **MHASignal::spectrum_t * signal_out** (**mha_spec_t** **)
  Spectrum domain output signal structure. Parameter is ignored.

Private Attributes

- **MHAEvents::patchbay_t< split_t > patchbay**
  Reload plugins when the algos variable changes.
- **MHAParser::vstring_t algos**
  Vector of plugins to load in parallel.
- **MHAParser::vint_t channels**
  Number of channels to route through each plugin.
- **MHAParser::kw_t thread_platform**
  Thread platform chooser.
- **MHAParser::kw_t worker_thread_scheduler**
  Scheduler used for worker threads.
- **MHAParser::int_t worker_thread_priority**
  Priority of worker threads.
- **MHAParser::string_mon_t framework_thread_scheduler**
  Scheduler of the signal processing thread.
- **MHAParser::int_mon_t framework_thread_priority**
  Priority of signal processing thread.
- **MHAParser::bool_t delay**
  Switch to activate parallel processing of plugins at the cost of one block of additional delay.
- std::vector< **splitted_part_t** * > **chains**
  Interfaces to parallel plugins.
- **MHASignal::waveform_t * wave_out**
  Combined output waveforms structure.
- **MHASignal::spectrum_t * spec_out**
  Combined output spectra structure.

Additional Inherited Members

5.284.1 Detailed Description

Implements split plugin.

An instance of class **split_t** (p. 747) implements the split plugin functionality: The audio channels are splitted and groups of audio channels are processed by different plugins in parallel.

5.284.2 Constructor & Destructor Documentation

5.284.2.1 MHAPlugin_Split::split_t::split_t (algo_comm_t *iac,* const std::string & *chain_name,* const std::string & *algo_name*)

Plugin constructor.

5.284.2.2 MHAPlugin_Split::split_t::~split_t ( )

Plugin destructor. Unloads nested plugins.

5.284.3 Member Function Documentation

5.284.3.1 void MHAPlugin_Split::split_t::prepare_ (mhaconfig_t & *signal_parameters*)

Check signal parameters, prepare chains, and allocate output signal holders.

5.284.3.2 void MHAPlugin_Split::split_t::release_ ( )

Delete output signal holder and release chains.

5.284.3.3 template<class SigTypeIn , class SigTypeOut > void MHAPlugin_Split::split_t::process (SigTypeIn * *s_in,* SigTypeOut ** *s_out*)

Let the parallel plugins process channel groups of the input signal.

5.284.3.4 void MHAPlugin_Split::split_t::update ( ) [private]

Load plugins in response to a value change in the algos variable.

5.284.3.5 void MHAPlugin_Split::split_t::clear_chains ( ) [private]

Unload the plugins.

5.284.3.6 mha_wave_t* MHAPlugin_Split::split_t::copy_output_wave ( ) [private]

5.284.3.7 mha_spec_t* MHAPlugin_Split::split_t::copy_output_spec ( ) [private]

5.284.3.8 template<class SigType > void MHAPlugin_Split::split_t::trigger_processing (SigType * *s_in*) [private]

Split the argument input signal to groups of channels for the plugins and initiate signal processing.

5.284.3.9 template<class SigType > void MHAPlugin_Split::split_t::collect_result (SigType * *s_out*) [private]

Combine the output signal from the plugins.

5.284.3.10 MHASignal::waveform_t* MHAPlugin_Split::split_t::signal_out (mha_wave_t **) [inline], [private]

Waveform domain output signal structure accessor.

Parameter is only for domain disambiguation and is ignored.

5.284.3.11 MHASignal::spectrum_t* MHAPlugin_Split::split_t::signal_out (mha_spec_t **) [inline], [private]

Spectrum domain output signal structure. Parameter is ignored.

5.284.4 Member Data Documentation

5.284.4.1 MHAEvents::patchbay_t<split_t> MHAPlugin_Split::split_t::patchbay [private]

Reload plugins when the algos variable changes.

5.284.4.2 MHAParser::vstring_t MHAPlugin_Split::split_t::algos [private]

Vector of plugins to load in parallel.

5.284.4.3 MHAParser::vint_t MHAPlugin_Split::split_t::channels [private]

Number of channels to route through each plugin.

5.284.4.4 MHAParser::kw_t MHAPlugin_Split::split_t::thread_platform [private]

Thread platform chooser.

5.284.4.5 MHAParser::kw_t MHAPlugin_Split::split_t::worker_thread_scheduler [private]

Scheduler used for worker threads.

5.284.4.6 MHAParser::int_t MHAPlugin_Split::split_t::worker_thread_priority [private]

Priority of worker threads.

5.284.4.7 MHAParser::string_mon_t MHAPlugin_Split::split_t::framework_thread_scheduler [private]

Scheduler of the signal processing thread.

5.284.4.8 MHAParser::int_mon_t MHAPlugin_Split::split_t::framework_thread_priority [private]

Priority of signal processing thread.

5.284.4.9 MHAParser::bool_t MHAPlugin_Split::split_t::delay [private]

Switch to activate parallel processing of plugins at the cost of one block of additional delay.

5.284.4.10 std::vector<splitted_part_t*> MHAPlugin_Split::split_t::chains [private]

Interfaces to parallel plugins.

5.284.4.11 MHASignal::waveform_t* MHAPlugin_Split::split_t::wave_out [private]

Combined output waveforms structure.

5.284.4.12 MHASignal::spectrum_t* MHAPlugin_Split::split_t::spec_out [private]

Combined output spectra structure.

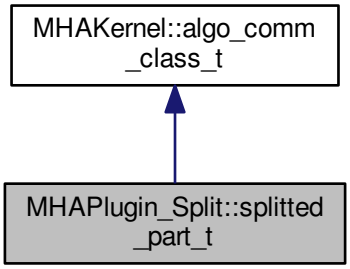
The documentation for this class was generated from the following file:

- **split.cpp**

5.285 MHAPlugin_Split::splitted_part_t Class Reference

The **splitted_part_t** (p. 751) instance manages the plugin that performs processing on the reduced set of channels.

Inheritance diagram for MHAPlugin_Split::splitted_part_t:



Public Member Functions

- **splitted_part_t** (const std::string &plugname, **MHAParser::parser_t** *parent)

  Load the plugin for this partial signal path.
- **splitted_part_t** (**PluginLoader::fourway_processor_t** *plugin)

  Create the handler for the partial signal.
- ~**splitted_part_t** () throw ()

  *Destructor. Deletes the plugin **plug** (p. 755).*
- void **prepare** (**mhaconfig_t** &signal_parameters, const std::string &thread_platform, const std::string &thread_scheduler, int thread_priority)

  *Delegates the prepare method to the plugin and allocates a suitable **MHAPlugin_Split::domain_handler_t** (p. 736) instance.*
- void **release** ()

  *Delegates the release method to the plugin and deletes the **MHAPlugin_Split::domain_handler_t** (p. 736) instance.*
- std::string **parse** (const std::string &str)

  Delegates parser incovation to plugin.
- template<class SigType >
  unsigned **trigger_processing** (SigType *s_in, unsigned start_channel)

  The domain handler copies the input signal channels.
- template<class SigType >
  unsigned **collect_result** (SigType *s_out, unsigned start_channel)

  Wait until processing is finished, then copy the output data.

Private Member Functions

- **splitted_part_t** (const **splitted_part_t** &)

  Disallow copy constructor.
- **splitted_part_t** & **operator=** (const **splitted_part_t** &)

  Disallow assignment operator.

Private Attributes

- **PluginLoader::fourway_processor_t** * **plug**

  The plugin that performs the signal processing on the prepared channels.
- **domain_handler_t** * **domain**

  The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.
- **thread_platform_t** * **thread**

  The platform-dependent thread synchronization implementation.

Additional Inherited Members

5.285.1 Detailed Description

The **splitted_part_t** (p. 751) instance manages the plugin that performs processing on the reduced set of channels.

The signal is split by channels by this instance, but the signal is combined again by the calling class.

5.285.2 Constructor & Destructor Documentation

5.285.2.1 MHAPlugin_Split::splitted_part_t::splitted_part_t (const splitted_part_t &) [private]

Disallow copy constructor.

5.285.2.2 MHAPlugin_Split::splitted_part_t::splitted_part_t (const std::string & *plugname,* MHAParser::parser_t * *parent*)

Load the plugin for this partial signal path.

Loads the MHA plugin for a signal path of these audio channels.

Parameters

plugname	The name of the MHA plugin, optionally followed by a colon and the algorithm name.
parent	The parser node where the configuration of the new plugin is inserted. The plugin's parser name is the configured name (colon syntax).

5.285.2.3 MHAPlugin_Split::splitted_part_t::splitted_part_t (PluginLoader::fourway_processor_t * *plugin*)

Create the handler for the partial signal.

The plugin is loaded by the caller, but it will be deleted by the destructor of this class. This constructor exists solely for testing purposes.

Parameters

plugin	The plugin used for processing the signal. The new **splitted_part_t** (p. 753) instance will take ownership of this instance and release it in the destructor.

5.285.2.4 MHAPlugin_Split::splitted_part_t::~splitted_part_t ( ) throw)

Destructor. Deletes the plugin **plug** (p. 755).

5.285.3 Member Function Documentation

5.285.3.1 splitted_part_t& MHAPlugin_Split::splitted_part_t::operator= (const splitted_part_t &) [private]

Disallow assignment operator.

5.285.3.2 void MHAPlugin_Split::splitted_part_t::prepare (mhaconfig_t & *signal_parameters,* const std::string & *thread_platform,* const std::string & *thread_scheduler,* int *thread_priority*)

Delegates the prepare method to the plugin and allocates a suitable **MHAPlugin_Split::domain_handler_t** (p. 736) instance.

Prepare the loaded plugin.

Plugin preparation.

Parameters

signal_parameters	The signal description parameters for this path.
thread_platform	The name of the thread platform to use. Possible values: "posix", "win32", "dummy".
thread_scheduler	The name of the scheduler to use. Posix threads support "SCHED_OTHER", "SCHED_RR", "SCHED_FIFO". The other thread platforms do not support different thread schedulers. This value is not used for platforms other than "posix".
thread_priority	The new thread priority. Interpretation and permitted range depend on the thread platform and possibly on the scheduler.

5.285.3.3 void MHAPlugin_Split::splitted_part_t::release (void)

Delegates the release method to the plugin and deletes the **MHAPlugin_Split::domain_handler_t** (p. 736) instance.

Release the loaded plugin.

Plugin release.

5.285.3.4 std::string MHAPlugin_Split::splitted_part_t::parse (const std::string & *str*) [inline]

Delegates parser incovation to plugin.

5.285.3.5 template<class SigType > unsigned MHAPlugin_Split::splitted_part_t::trigger_processing (SigType * *s_in*, unsigned *start_channel*) [inline]

The domain handler copies the input signal channels.

Then, processing is initiated.

Parameters

s_in	The combined input signal.
start_channel	The index (0-based) of the first channel in s_in to be copied to the partial input signal.

Returns

The number of channels that were copied from the input signal

5.285.3.6 template<class SigType > unsigned MHAPlugin_Split::splitted_part_t::collect_result (SigType * *s_out*, unsigned *start_channel*) [inline]

Wait until processing is finished, then copy the output data.

Parameters

s_out	The combined waveform output signal.
start_channel	The channel offset (0-based) in s_out.

Returns

The number of channels that were copied to the output signal

5.285.4 Member Data Documentation

5.285.4.1 PluginLoader::fourway_processor_t* MHAPlugin_Split::splitted_part_t::plug [private]

The plugin that performs the signal processing on the prepared channels.

5.285.4.2 domain_handler_t* MHAPlugin_Split::splitted_part_t::domain [private]

The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.

5.285.4.3 thread_platform_t* MHAPlugin_Split::splitted_part_t::thread [private]

The platform-dependent thread synchronization implementation.

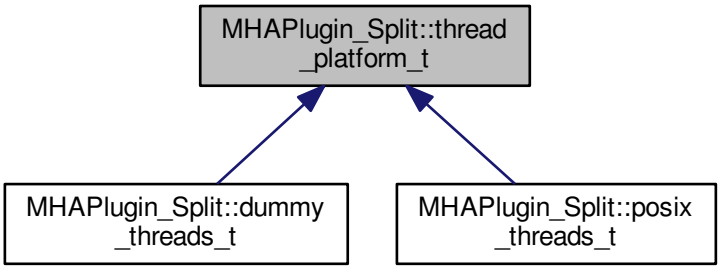
The documentation for this class was generated from the following file:

- **split.cpp**

5.286 MHAPlugin_Split::thread_platform_t Class Reference

Basic interface for encapsulating thread creation, thread priority setting, and synchronization on any threading platform (i.e., pthreads or win32threads).

Inheritance diagram for MHAPlugin_Split::thread_platform_t:



Public Member Functions

- **thread_platform_t** (**uni_processor_t** *proc)
  Constructor.
- virtual ~**thread_platform_t** ()
  Make derived classes destructable via pointer to this base class.
- virtual void **kick_thread** ()=0
  Derived classes notify their processing thread that it should call processor->process().
- virtual void **catch_thread** ()=0
  Derived classes wait for their signal processing thread to return from the call to part->process().

Protected Attributes

- **uni_processor_t** ∗ **processor**

  A pointer to the plugin loader that processes the sound data in the channels for which this thread was created.

Private Member Functions

- **thread_platform_t** (const **thread_platform_t** &)

  Disallow copy constructor.
- **thread_platform_t** & **operator=** (const **thread_platform_t** &)

  Disallow assignment operator.

5.286.1 Detailed Description

Basic interface for encapsulating thread creation, thread priority setting, and synchronization on any threading platform (i.e., pthreads or win32threads).

Derived classes specialize in the actual thread platform.

5.286.2 Constructor & Destructor Documentation

5.286.2.1 MHAPlugin_Split::thread_platform_t::thread_platform_t (const thread_platform_t &) [private]

Disallow copy constructor.

5.286.2.2 MHAPlugin_Split::thread_platform_t::thread_platform_t (uni_processor_t ∗ *proc*) [inline]

Constructor.

Derived classes create the thread in the constructor.

Parameters

proc	Pointer to the associated plugin loader. This plugin loader has to live at least as long as this instance. This instance does not take possession of the plugin loader. In production code, this thread platform and the plugin loader are both created and destroyed by the **MHAPlugin_Split::splitted_part_t** (p. 751) instance.

5.286.2.3 virtual MHAPlugin_Split::thread_platform_t::~thread_platform_t ( ) [inline], [virtual]

Make derived classes destructable via pointer to this base class.

Derived classes' destructors notify the thread that it should terminate itself, and wait for the termination to occur.

5.286.3 Member Function Documentation

5.286.3.1 thread_platform_t& MHAPlugin_Split::thread_platform_t::operator= (const thread_platform_t &) [private]

Disallow assignment operator.

5.286.3.2 virtual void MHAPlugin_Split::thread_platform_t::kick_thread ( ) [pure virtual]

Derived classes notify their processing thread that it should call processor->process().

Implemented in **MHAPlugin_Split::posix_threads_t** (p. 745), and **MHAPlugin_Split::dummy_threads_t** (p. 742).

5.286.3.3 virtual void MHAPlugin_Split::thread_platform_t::catch_thread ( ) [pure virtual]

Derived classes wait for their signal processing thread to return from the call to part->process().

Implemented in **MHAPlugin_Split::posix_threads_t** (p. 745), and **MHAPlugin_Split::dummy_threads_t** (p. 743).

5.286.4 Member Data Documentation

5.286.4.1 uni_processor_t* MHAPlugin_Split::thread_platform_t::processor [protected]

A pointer to the plugin loader that processes the sound data in the channels for which this thread was created.

Using the **MHAPlugin_Split::uni_processor_t** (p. 759) interface instead of the mhaplugin-loader class directly for testability (no need to load real plugins for testing the thread platform).

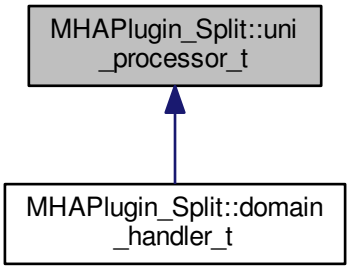
The documentation for this class was generated from the following file:

- **split.cpp**

5.287 MHAPlugin_Split::uni_processor_t Class Reference

An interface to a class that sports a process method with no parameters and no return value.

Inheritance diagram for MHAPlugin_Split::uni_processor_t:



Public Member Functions

- virtual void **process** ()=0

  This method uses some input signal, performs processing and stores the output signal somewhere.
- virtual ~**uni_processor_t** ()

  Classes containing virtual methods need virtual destructors.

5.287.1 Detailed Description

An interface to a class that sports a process method with no parameters and no return value.

No signal transfer occurs through this interface, because the signal transfer is performed in another thread than the processing.

5.287.2 Constructor & Destructor Documentation

5.287.2.1 virtual MHAPlugin_Split::uni_processor_t::~uni_processor_t ( ) [inline], [virtual]

Classes containing virtual methods need virtual destructors.

5.287.3 Member Function Documentation

5.287.3.1 virtual void MHAPlugin_Split::uni_processor_t::process ( ) [pure virtual]

This method uses some input signal, performs processing and stores the output signal somewhere.

This method also has to dispatch the process call based on the configured domains.

Signal transfer and domain configuration have to be done in derived class in different methods.

Implemented in **MHAPlugin_Split::domain_handler_t** (p. 740).

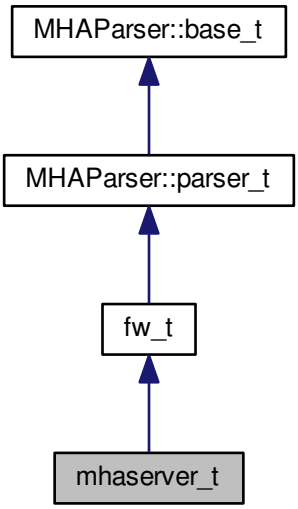
The documentation for this class was generated from the following file:

- **split.cpp**

5.288 mhaserver_t Class Reference

MHA Framework listening on TCP port for commands.

Inheritance diagram for mhaserver_t:



Public Member Functions

- **mhaserver_t** (const std::string &ao, const std::string &af, const std::string &lf)
- ~**mhaserver_t** ()
- virtual std::string **received_group** (const std::string &line)

  A line of text was received from network client.
- virtual void **acceptor_started** (int status)

  Notification: "TCP port is open".
- virtual void **set_announce_port** (unsigned short **announce_port**)

  If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.
- void **logstring** (const std::string &)

  Log a message to log file.
- int **run** (unsigned short **port**, const std::string &_interface)

  Accept network connections and act on commands.

Public Attributes

- **MHAParser::int_t port**

Private Attributes

- **MHA_TCP::Server** ∗ **tcpserver**
- std::string **ack_ok**
- std::string **ack_fail**
- std::string **logfile**
- unsigned short **announce_port**
- **MHAParser::int_mon_t pid_mon**

Additional Inherited Members

5.288.1 Detailed Description

MHA Framework listening on TCP port for commands.

5.288.2 Constructor & Destructor Documentation

5.288.2.1 mhaserver_t::mhaserver_t (const std::string & *ao,* const std::string & *af,* const std::string & *lf*)

Parameters

ao	Acknowledgement string at end of successful command responses
af	Achknoledgement string at end of failed command responses
lf	File system path of file to use as log file. MHA appends.

5.288.2.2 mhaserver_t::~mhaserver_t ( )

5.288.3 Member Function Documentation

5.288.3.1 std::string mhaserver_t::received_group (const std::string & *line*) [virtual]

A line of text was received from network client.

5.288.3.2 void mhaserver_t::acceptor_started (int *status*) [virtual]

Notification: "TCP port is open".

5.288.3.3 void mhaserver_t::set_announce_port (unsigned short *announce_port*) [virtual]

If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.

5.288.3.4 void mhaserver_t::logstring (const std::string & *s*) [inline]

Log a message to log file.

5.288.3.5 int mhaserver_t::run (unsigned short *port,* const std::string & *_interface*)

Accept network connections and act on commands.

Calls **acceptor_started()** (p. 762) when the TCP port is opened. Calls received_group for every line received.

Returns

exit code that can be used as process exit code

5.288.4 Member Data Documentation

5.288.4.1 MHA_TCP::Server* mhaserver_t::tcpserver [private]

5.288.4.2 std::string mhaserver_t::ack_ok [private]

5.288.4.3 std::string mhaserver_t::ack_fail [private]

5.288.4.4 std::string mhaserver_t::logfile [private]

5.288.4.5 unsigned short mhaserver_t::announce_port [private]

5.288.4.6 MHAParser::int_mon_t mhaserver_t::pid_mon [private]

5.288.4.7 MHAParser::int_t mhaserver_t::port

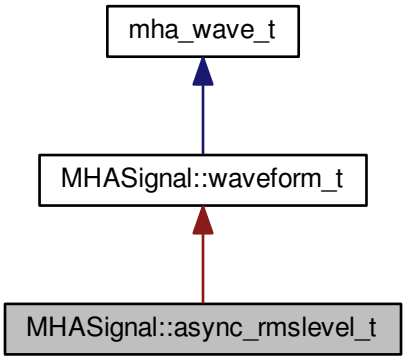
The documentation for this class was generated from the following file:

- **mhamain.cpp**

5.289 MHASignal::async_rmslevel_t Class Reference

Class for asynchronous level metering.

Inheritance diagram for MHASignal::async_rmslevel_t:



Public Member Functions

- **async_rmslevel_t** (unsigned int frames, unsigned int **channels**)
  Constructor for level metering class.
- std::vector< float > **rmslevel** () const
  Read-only function for querying the current RMS level.
- std::vector< float > **peaklevel** () const
  Read-only function for querying the current peak level.
- void **process** (**mha_wave_t** *s)
  Function to store a chunk of audio in the level meter.

Private Attributes

- unsigned int **pos**
- unsigned int **filled**

Additional Inherited Members

5.289.1 Detailed Description

Class for asynchronous level metering.

5.289.2 Constructor & Destructor Documentation

5.289.2.1 MHASignal::async_rmslevel_t::async_rmslevel_t (unsigned int *frames,* unsigned int *channels*)

Constructor for level metering class.

Allocate memory for metering. The RMS integration time corresponds to the number of frames in the buffer.

Parameters

frames	Number of frames to integrate.
channels	Number of channels used for level-metering.

5.289.3 Member Function Documentation

5.289.3.1 std::vector< float > MHASignal::async_rmslevel_t::rmslevel ( ) const

Read-only function for querying the current RMS level.

Returns

Vector of floats, one value for each channel, containing the RMS level in dB (SPL if calibrated properly).

5.289.3.2 std::vector< float > MHASignal::async_rmslevel_t::peaklevel ( ) const

Read-only function for querying the current peak level.

Returns

Vector of floats, one value for each channel, containing the peak level in dB (SPL if calibrated properly).

5.289.3.3 void MHASignal::async_rmslevel_t::process (mha_wave_t * *s*)

Function to store a chunk of audio in the level meter.

Parameters

s	Audio chunk (same number of channels required as given in the constructor).

5.289.4 Member Data Documentation

5.289.4.1 unsigned int MHASignal::async_rmslevel_t::pos [private]

5.289.4.2 unsigned int MHASignal::async_rmslevel_t::filled [private]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.290 MHASignal::delay_spec_t Class Reference

Public Member Functions

- **delay_spec_t** (unsigned int **delay**, unsigned int frames, unsigned int **channels**)
- ~**delay_spec_t** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- unsigned int **delay**
- **MHASignal::spectrum_t** ∗∗ **buffer**
- unsigned int **pos**

5.290.1 Constructor & Destructor Documentation

5.290.1.1 MHASignal::delay_spec_t::delay_spec_t (unsigned int *delay,* unsigned int *frames,* unsigned int *channels*)

5.290.1.2 MHASignal::delay_spec_t::∼delay_spec_t ( )

5.290.2 Member Function Documentation

5.290.2.1 mha_spec_t ∗ MHASignal::delay_spec_t::process (mha_spec_t ∗ *s*)

5.290.3 Member Data Documentation

5.290.3.1 unsigned int MHASignal::delay_spec_t::delay [private]

5.290.3.2 MHASignal::spectrum_t∗∗ MHASignal::delay_spec_t::buffer [private]

5.290.3.3 unsigned int MHASignal::delay_spec_t::pos [private]

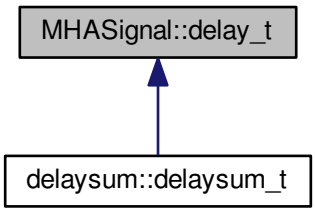
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.291 MHASignal::delay_t Class Reference

Class to realize a simple delay of waveform streams.

Inheritance diagram for MHASignal::delay_t:



Public Member Functions

- **delay_t** (std::vector< int > **delays**, unsigned int **channels**)
  Constructor.
- **mha_wave_t** * **process** (**mha_wave_t** *s)
  Processing method.
- ~**delay_t** ()
- std::string **inspect** () const

Private Attributes

- unsigned int **channels**
- unsigned int * **delays**
- unsigned int * **pos**
- **mha_real_t** ** **buffer**

5.291.1 Detailed Description

Class to realize a simple delay of waveform streams.

5.291.2 Constructor & Destructor Documentation

5.291.2.1 MHASignal::delay_t::delay_t (std::vector< int > *delays*, unsigned int *channels*)

Constructor.

Parameters

delays	Vector of delays, one entry for each channel.
channels	Number of channels expected.

5.291.2.2 MHASignal::delay_t::~delay_t ( )

5.291.3 Member Function Documentation

5.291.3.1 mha_wave_t * MHASignal::delay_t::process (mha_wave_t * *s*)

Processing method.

Parameters

s	Input waveform fragment, with number of channels provided in constructor.

Returns

Output waveform fragment.

5.291.3.2 std::string MHASignal::delay_t::inspect ( ) const [inline]

5.291.4 Member Data Documentation

5.291.4.1 unsigned int MHASignal::delay_t::channels [private]

5.291.4.2 unsigned int* MHASignal::delay_t::delays [private]

5.291.4.3 unsigned int* MHASignal::delay_t::pos [private]

5.291.4.4 mha_real_t** MHASignal::delay_t::buffer [private]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.292 MHASignal::delay_wave_t Class Reference

Delayline containing wave fragments.

Public Member Functions

- **delay_wave_t** (unsigned int **delay**, unsigned int frames, unsigned int **channels**)
- ~**delay_wave_t** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Attributes

- unsigned int **delay**
- **MHASignal::waveform_t** ** **buffer**
- unsigned int **pos**

5.292.1 Detailed Description

Delayline containing wave fragments.

The delayline contains waveform fragments. The delay can be configured in integer fragments (sample delay or sub-sample delay is not possible).

5.292.2 Constructor & Destructor Documentation

5.292.2.1 MHASignal::delay_wave_t::delay_wave_t (unsigned int *delay,* unsigned int *frames,* unsigned int *channels*)

5.292.2.2 MHASignal::delay_wave_t::~delay_wave_t ( )

5.292.3 Member Function Documentation

5.292.3.1 mha_wave_t * MHASignal::delay_wave_t::process (mha_wave_t * *s*)

5.292.4 Member Data Documentation

5.292.4.1 unsigned int MHASignal::delay_wave_t::delay [private]

5.292.4.2 MHASignal::waveform_t** MHASignal::delay_wave_t::buffer [private]

5.292.4.3 unsigned int MHASignal::delay_wave_t::pos [private]

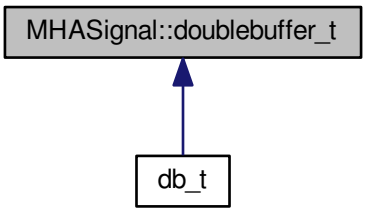
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.293 MHASignal::doublebuffer_t Class Reference

Double-buffering class.

Inheritance diagram for MHASignal::doublebuffer_t:



Public Member Functions

- **doublebuffer_t** (unsigned int nchannels_in, unsigned int nchannels_out, unsigned int outer_fragsize, unsigned int inner_fragsize)
  Constructor of double buffer.
- virtual ∼**doublebuffer_t** ()
- **mha_wave_t** ∗ **outer_process** (**mha_wave_t** ∗s)
  Method to pass audio fragments into the inner layer.

Protected Member Functions

- virtual **mha_wave_t** ∗ **inner_process** (**mha_wave_t** ∗s)=0
  Method to realize inner processing callback.

Private Member Functions

- unsigned int **min** (unsigned int a, unsigned int b)

Private Attributes

- **waveform_t outer_out**
- **mha_wave_t this_outer_out**
- **waveform_t inner_in**
- **waveform_t inner_out**
- unsigned int **k_inner**
- unsigned int **k_outer**
- unsigned int **ch**

5.293.1 Detailed Description

Double-buffering class.

This class has two layers: The outer layer, with an outer fragment size, and an inner layer, with its own fragment size. Data is passed into the inner layer through the doublebuffer_t↩ ::outr_process() callback. The pure virtual method **doublebuffer_t::inner_process()** (p. 771) is called whenever enough data is available.

5.293.2 Constructor & Destructor Documentation

5.293.2.1 MHASignal::doublebuffer_t::doublebuffer_t (unsigned int *nchannels_in,* unsigned int *nchannels_out,* unsigned int *outer_fragsize,* unsigned int *inner_fragsize*)

Constructor of double buffer.

Parameters

nchannels_in	Number of channels at the input (both layers).
nchannels_out	Number of channels at the output (both layers).
outer_fragsize	Fragment size of the outer layer (e.g., hardware fragment size)
inner_fragsize	Fragment size of the inner layer (e.g., software fragment size)

5.293.2.2 MHASignal::doublebuffer_t::~doublebuffer_t ( ) [virtual]

5.293.3 Member Function Documentation

5.293.3.1 mha_wave_t * MHASignal::doublebuffer_t::outer_process (mha_wave_t * *s*)

Method to pass audio fragments into the inner layer.

Parameters

s	Pointer to input waveform fragment.

Returns

Pointer to output waveform fragment.

5.293.3.2 virtual mha_wave_t* MHASignal::doublebuffer_t::inner_process (mha_wave_t * *s*) [protected], [pure virtual]

Method to realize inner processing callback.

To be overwritten by derived classes.

Parameters

s	Pointer to input waveform fragment.

Returns

Pointer to output waveform fragment.

Implemented in **db_t** (p. 252).

5.293.3.3 unsigned int MHASignal::doublebuffer_t::min (unsigned int *a,* unsigned int *b*) [inline], [private]

5.293.4 Member Data Documentation

5.293.4.1 waveform_t MHASignal::doublebuffer_t::outer_out [private]

5.293.4.2 mha_wave_t MHASignal::doublebuffer_t::this_outer_out [private]

5.293.4.3 waveform_t MHASignal::doublebuffer_t::inner_in [private]

5.293.4.4 waveform_t MHASignal::doublebuffer_t::inner_out [private]

5.293.4.5 unsigned int MHASignal::doublebuffer_t::k_inner [private]

5.293.4.6 unsigned int MHASignal::doublebuffer_t::k_outer [private]

5.293.4.7 unsigned int MHASignal::doublebuffer_t::ch [private]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.294 MHASignal::fft_t Class Reference

Public Member Functions

- **fft_t** (const unsigned int &)
- ~**fft_t** ()
- void **wave2spec** (const **mha_wave_t** ∗, **mha_spec_t** ∗, bool swap)
  fast fourier transform.
- void **spec2wave** (const **mha_spec_t** ∗, **mha_wave_t** ∗)
- void **spec2wave** (const **mha_spec_t** ∗, **mha_wave_t** ∗, unsigned int offset)
  wave may have fewer number of frames than needed for a complete iFFT.
- void **forward** (**mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)
- void **backward** (**mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)
- void **wave2spec_scale** (const **mha_wave_t** ∗, **mha_spec_t** ∗, bool swap)
- void **spec2wave_scale** (const **mha_spec_t** ∗, **mha_wave_t** ∗)
- void **forward_scale** (**mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)
- void **backward_scale** (**mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)

Private Member Functions

- void **sort_fftw2spec** (fftw_real ∗s_fftw, **mha_spec_t** ∗s_spec, unsigned int ch)
  Arrange the order of an fftw spectrum to the internal order.
- void **sort_spec2fftw** (fftw_real ∗s_fftw, const **mha_spec_t** ∗s_spec, unsigned int ch)
  Arrange the order of an internal spectrum to the fftw order.

Private Attributes

- unsigned int **nfft**
- unsigned int **n_re**
- unsigned int **n_im**
- **mha_real_t scale**
- **mha_real_t** ∗ **buf_in**
- **mha_real_t** ∗ **buf_out**
- rfftw_plan **fftw_plan_wave2spec**
- rfftw_plan **fftw_plan_spec2wave**
- fftw_plan **fftw_plan_fft**
- fftw_plan **fftw_plan_ifft**

5.294.1 Constructor & Destructor Documentation

5.294.1.1 MHASignal::fft_t::fft_t (const unsigned int & *n*)

5.294.1.2 MHASignal::fft_t::∼fft_t ( )

5.294.2 Member Function Documentation

5.294.2.1 void MHASignal::fft_t::wave2spec (const **mha_wave_t** ∗ *wave,* **mha_spec_t** ∗ *spec,* bool *swap*)

fast fourier transform.

if swap is set, the buffer halfes of the wave signal are exchanged before computing the fft.

5.294.2.2 void MHASignal::fft_t::spec2wave (const **mha_spec_t** ∗ *spec,* **mha_wave_t** ∗ *wave*)

5.294.2.3 void MHASignal::fft_t::spec2wave (const **mha_spec_t** ∗ *spec,* **mha_wave_t** ∗ *wave,* unsigned int *offset*)

wave may have fewer number of frames than needed for a complete iFFT.

Only as many frames are written into wave as fit, starting with offset offset of the complete iFFT.

5.294.2.4 void MHASignal::fft_t::forward (
mha_spec_t * *sIn,*
mha_spec_t * *sOut*)

5.294.2.5 void MHASignal::fft_t::backward (
mha_spec_t * *sIn,*
mha_spec_t * *sOut*)

5.294.2.6 void MHASignal::fft_t::wave2spec_scale (
const mha_wave_t * *wave,*
mha_spec_t * *spec,*
bool *swap*)

5.294.2.7 void MHASignal::fft_t::spec2wave_scale (
const mha_spec_t * *spec,*
mha_wave_t * *wave*)

5.294.2.8 void MHASignal::fft_t::forward_scale (
mha_spec_t * *sIn,*
mha_spec_t * *sOut*)

5.294.2.9 void MHASignal::fft_t::backward_scale (
mha_spec_t * *sIn,*
mha_spec_t * *sOut*)

5.294.2.10 void MHASignal::fft_t::sort_fftw2spec (
fftw_real * *s_fftw,*
mha_spec_t * *s_spec,*
unsigned int *ch*) [private]

Arrange the order of an fftw spectrum to the internal order.

The fftw spectrum is arranged [r0 r1 r2 ... rn-1 in in-1 ... i1], while the interal order is [r0 – r1 i1 r2 i2 ... rn-1 in-1 rn –].

5.294.2.11 void MHASignal::fft_t::sort_spec2fftw (
fftw_real * *s_fftw,*
const mha_spec_t * *s_spec,*
unsigned int *ch*) [private]

Arrange the order of an internal spectrum to the fftw order.

5.294.3 Member Data Documentation

5.294.3.1 unsigned int MHASignal::fft_t::nfft [private]

5.294.3.2 unsigned int MHASignal::fft_t::n_re [private]

5.294.3.3 unsigned int MHASignal::fft_t::n_im [private]

5.294.3.4 mha_real_t MHASignal::fft_t::scale [private]

5.294.3.5 mha_real_t* MHASignal::fft_t::buf_in [private]

5.294.3.6 mha_real_t* MHASignal::fft_t::buf_out [private]

5.294.3.7 rfftw_plan MHASignal::fft_t::fftw_plan_wave2spec [private]

5.294.3.8 rfftw_plan MHASignal::fft_t::fftw_plan_spec2wave [private]

5.294.3.9 fftw_plan MHASignal::fft_t::fftw_plan_fft [private]

5.294.3.10 fftw_plan MHASignal::fft_t::fftw_plan_ifft [private]

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

- **mha_signal_fft.h**
- **mha_signal.cpp**

5.295 MHASignal::hilbert_fftw_t Class Reference

Public Member Functions

- **hilbert_fftw_t** (unsigned int len)
- void **hilbert** (const **mha_wave_t** *, **mha_wave_t** *)

Private Attributes

- unsigned int **n**
- rfftw_plan **p1**
- fftw_plan **p2**
- fftw_real * **buf_r_in**
- fftw_real * **buf_r_out**
- fftw_complex * **buf_c_in**
- fftw_complex * **buf_c_out**
- **mha_real_t sc**

5.295.1 Constructor & Destructor Documentation

5.295.1.1 MHASignal::hilbert_fftw_t::hilbert_fftw_t (unsigned int *len*)

5.295.2 Member Function Documentation

5.295.2.1 void MHASignal::hilbert_fftw_t::hilbert (const **mha_wave_t** ∗ *s_in*, **mha_wave_t** ∗ *s_out*)

5.295.3 Member Data Documentation

5.295.3.1 unsigned int MHASignal::hilbert_fftw_t::n [private]

5.295.3.2 rfftw_plan MHASignal::hilbert_fftw_t::p1 [private]

5.295.3.3 fftw_plan MHASignal::hilbert_fftw_t::p2 [private]

5.295.3.4 fftw_real∗ MHASignal::hilbert_fftw_t::buf_r_in [private]

5.295.3.5 fftw_real∗ MHASignal::hilbert_fftw_t::buf_r_out [private]

5.295.3.6 fftw_complex∗ MHASignal::hilbert_fftw_t::buf_c_in [private]

5.295.3.7 fftw_complex∗ MHASignal::hilbert_fftw_t::buf_c_out [private]

5.295.3.8 mha_real_t MHASignal::hilbert_fftw_t::sc [private]

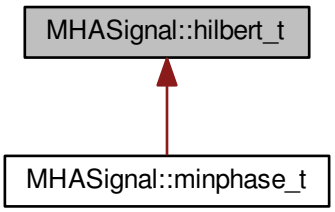
The documentation for this class was generated from the following file:

- **mha_signal.cpp**

5.296 MHASignal::hilbert_t Class Reference

Hilbert transformation of a waveform segment.

Inheritance diagram for MHASignal::hilbert_t:



Public Member Functions

- **hilbert_t** (unsigned int len)
- ∼**hilbert_t** ()
- void **operator()** (const **mha_wave_t** ∗, **mha_wave_t** ∗)
  Apply Hilbert transformation on a waveform segment.

Private Attributes

- void ∗ **h**

5.296.1 Detailed Description

Hilbert transformation of a waveform segment.

Returns the imaginary part of the inverse Fourier transformation of the Fourier transformed input signal with negative frequencies set to zero.

5.296.2 Constructor & Destructor Documentation

5.296.2.1 MHASignal::hilbert_t::hilbert_t (unsigned int *len*)

Parameters

len	Length of waveform segment

5.296.2.2 MHASignal::hilbert_t::∼hilbert_t ( )

5.296.3 Member Function Documentation

5.296.3.1 void MHASignal::hilbert_t::operator() (const **mha_wave_t** ∗ ***s_in***, **mha_wave_t** ∗ ***s_out***)

Apply Hilbert transformation on a waveform segment.

5.296.4 Member Data Documentation

5.296.4.1 void∗ MHASignal::hilbert_t::h [private]

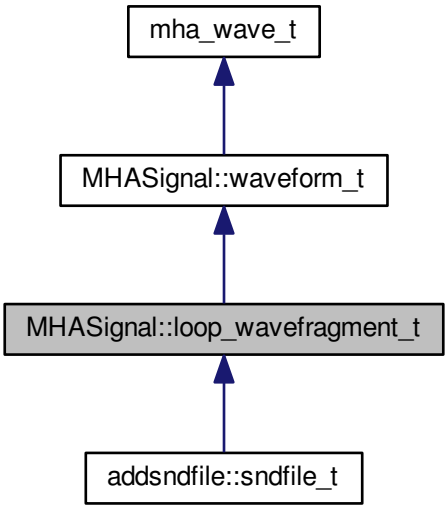
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.297 MHASignal::loop_wavefragment_t Class Reference

Copy a fixed waveform fragment to a series of waveform fragments of other size.

Inheritance diagram for MHASignal::loop_wavefragment_t:



Public Types

Public Member Functions

- **loop_wavefragment_t** (const **mha_wave_t** &src, bool loop, **level_mode_t** level_mode, std::vector< int > **channels**, unsigned int startpos=0)

  *Constructor to create an instance of **loop_wavefragment_t** (p. 778) based on an existing waveform block.*
- std::vector< int > **get_mapping** (unsigned int **channels**)
- void **playback** (**mha_wave_t** *s, **playback_mode_t** pmode, **mha_wave_t** *level_pa, const std::vector< int > &**channels**)

  Add source waveform block to an output block.
- void **playback** (**mha_wave_t** *s, **playback_mode_t** pmode, **mha_wave_t** *level_pa)

  Add source waveform block to an output block.
- void **playback** (**mha_wave_t** *s, **playback_mode_t** pmode)

  Add source waveform block to an output block.
- void **set_level_lin** (**mha_real_t** l)
- void **set_level_db** (**mha_real_t** l)
- void **rewind** ()
- void **locate_end** ()
- bool **is_playback_active** () const

Private Attributes

- std::vector< int > **playback_channels**
- bool **b_loop**
- unsigned int **pos**
- **MHASignal::waveform_t intern_level**

Additional Inherited Members

5.297.1 Detailed Description

Copy a fixed waveform fragment to a series of waveform fragments of other size.

This class is designed to continously play back a waveform to an output stream, with variable output block size.

5.297.2 Member Enumeration Documentation

5.297.2.1 enum MHASignal::loop_wavefragment_t::level_mode_t

Switch for playback level mode.

Enumerator

relative The nominal level is applied as a gain to the source signal.

peak The nominal level is the peak level of source signal in Pascal.

rms The nominal level is the RMS level of the source signal in Pascal.

rms_limit40

5.297.2.2 enum MHASignal::loop_wavefragment_t::playback_mode_t

Switch for playback mode.

Enumerator

add Add source signal to output stream.

replace Replace output stream by source signal.

input Do nothing, keep output stream (source position is unchanged).

mute Mute output stream (source position is unchanged).

5.297.3 Constructor & Destructor Documentation

5.297.3.1 MHASignal::loop_wavefragment_t::loop_wavefragment_t (

const **mha_wave_t** & *src,*
bool *loop,*
level_mode_t *level_mode,*
std::vector< int > *channels,*
unsigned int *startpos* = 0)

Constructor to create an instance of **loop_wavefragment_t** (p. 778) based on an existing waveform block.

Parameters

src	Waveform block to copy data from.
loop	Flag whether the block should be looped or played once.
level_mode	Configuration of playback level (see **MHASignal::loop_wavefragment_t::level_mode_t** (p. 779) for details)
channels	Mapping of input to output channels.
startpos	Starting position

5.297.4 Member Function Documentation

5.297.4.1 std::vector< int > MHASignal::loop_wavefragment_t::get_mapping (unsigned int *channels*)

5.297.4.2 void MHASignal::loop_wavefragment_t::playback (mha_wave_t * *s*, playback_mode_t *pmode*, mha_wave_t * *level_pa*, const std::vector< int > & *channels*)

Add source waveform block to an output block.

Parameters

s	Output block (streamed signal).
pmode	Playback mode (add, replace, input, mute).
level_pa	Linear output level/gain (depending on level_mode parameter in constructor); one value for each sample in output block.
channels	Output channels

5.297.4.3 void MHASignal::loop_wavefragment_t::playback (mha_wave_t * *s*, playback_mode_t *pmode*, mha_wave_t * *level_pa*)

Add source waveform block to an output block.

Parameters

s	Output block (streamed signal).
pmode	Playback mode (add, replace, input, mute).
level_pa	Linear output level/gain (depending on level_mode parameter in constructor); one value for each sample in output block.

5.297.4.4 void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode)

Add source waveform block to an output block.

Parameters

s	Output block (streamed signal).
pmode	Playback mode (add, replace, input, mute).

5.297.4.5 void MHASignal::loop_wavefragment_t::set_level_lin (mha_real_t l)

5.297.4.6 void MHASignal::loop_wavefragment_t::set_level_db (mha_real_t l)

5.297.4.7 void MHASignal::loop_wavefragment_t::rewind ( ) [inline]

5.297.4.8 void MHASignal::loop_wavefragment_t::locate_end ( ) [inline]

5.297.4.9 bool MHASignal::loop_wavefragment_t::is_playback_active ( ) const [inline]

5.297.5 Member Data Documentation

5.297.5.1 std::vector<int> MHASignal::loop_wavefragment_t::playback_channels [private]

5.297.5.2 bool MHASignal::loop_wavefragment_t::b_loop [private]

5.297.5.3 unsigned int MHASignal::loop_wavefragment_t::pos [private]

5.297.5.4 MHASignal::waveform_t MHASignal::loop_wavefragment_t::intern_level [private]

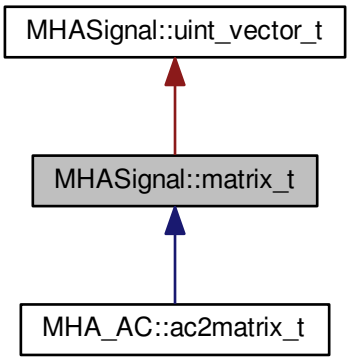
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.298 MHASignal::matrix_t Class Reference

n-dimensional matrix with real or complex floating point values.

Inheritance diagram for MHASignal::matrix_t:



Public Member Functions

- **matrix_t** (unsigned int nrows, unsigned int ncols, bool b_is_complex=true)
  Create a two-dimensional matrix.
- **matrix_t** (const **mha_spec_t** &spec)
  Create a two-dimensional matrix from a spectrum, copy values.
- **matrix_t** (const **MHASignal::uint_vector_t** &**size**, bool b_is_complex=true)
  Create n-dimensional matrix, descriped by size argument.
- **matrix_t** (const **MHASignal::matrix_t** &)
- **matrix_t** (const uint8_t ∗buf, unsigned int len)
  Construct from memory area.
- ∼**matrix_t** ()
- **MHASignal::matrix_t** & **operator=** (const **MHASignal::matrix_t** &)
- **MHASignal::matrix_t** & **operator=** (const **comm_var_t** &v)
  Fill matrix with data of an AC variable object.
- **comm_var_t get_comm_var** ()
  Return a AC communication variable pointing to the data of the current matrix.
- unsigned int **dimension** () const
  Return the dimension of the matrix.
- unsigned int **size** (unsigned int k) const
  Return the size of the matrix.
- unsigned int **get_nelements** () const
  Return total number of elements.

- bool **is_same_size** (const **MHASignal::matrix_t** &)

  Test if matrix has same size as other.
- bool **iscomplex** () const

  Return information about complexity.
- **mha_real_t** & **real** (const **MHASignal::uint_vector_t** &index)

  Access real part of an element in a n-dimensional matrix.
- **mha_real_t** & **imag** (const **MHASignal::uint_vector_t** &index)

  Access imaginary part of an element in a n-dimensional matrix.
- **mha_complex_t** & **operator()** (const **MHASignal::uint_vector_t** &index)

  Access complex value of an element in a n-dimensional matrix.
- const **mha_real_t** & **real** (const **MHASignal::uint_vector_t** &index) const

  Access real part of an element in a n-dimensional matrix.
- const **mha_real_t** & **imag** (const **MHASignal::uint_vector_t** &index) const

  Access imaginary part of an element in a n-dimensional matrix.
- const **mha_complex_t** & **operator()** (const **MHASignal::uint_vector_t** &index) const

  Access complex value of an element in a n-dimensional matrix.
- **mha_real_t** & **real** (unsigned int row, unsigned int col)

  Access real part of an element in a two-dimensional matrix.
- **mha_real_t** & **imag** (unsigned int row, unsigned int col)

  Access imaginary part of an element in a two-dimensional matrix.
- **mha_complex_t** & **operator()** (unsigned int row, unsigned int col)

  Access complex value of an element in a two-dimensional matrix.
- const **mha_real_t** & **real** (unsigned int row, unsigned int col) const

  Access real part of an element in a two-dimensional matrix.
- const **mha_real_t** & **imag** (unsigned int row, unsigned int col) const

  Access imaginary part of an element in a two-dimensional matrix.
- const **mha_complex_t** & **operator()** (unsigned int row, unsigned int col) const

  Access complex value of an element in a two-dimensional matrix.
- unsigned int **get_nreals** () const
- unsigned int **get_index** (unsigned int row, unsigned int col) const
- unsigned int **get_index** (const **MHASignal::uint_vector_t** &index) const
- unsigned int **numbytes** () const

  Return number of bytes needed to store into memory.
- unsigned int **write** (uint8_t ∗buf, unsigned int len) const

  Copy to memory area.
- const **mha_real_t** ∗ **get_rdata** () const

  Return pointer of real data.
- const **mha_complex_t** ∗ **get_cdata** () const

  Return pointer of complex data.

Private Attributes

- uint32_t **complex_ofs**
- uint32_t **nelements**
- union {
  mha_real_t ∗ **rdata**
  mha_complex_t ∗ **cdata**
  };

Additional Inherited Members

5.298.1 Detailed Description

n-dimensional matrix with real or complex floating point values.

Warning

The member functions **imag()** (p. 787) and operator() should only be called if the matrix is defined to hold complex values.

5.298.2 Constructor & Destructor Documentation

5.298.2.1 MHASignal::matrix_t::matrix_t (unsigned int *nrows,* unsigned int *ncols,* bool *b_is_complex* = `true`)

Create a two-dimensional matrix.

Parameters

nrows	Number of rows
ncols	Number of columns
b_is_complex	Add space for complex values

5.298.2.2 MHASignal::matrix_t::matrix_t (const **mha_spec_t** & *spec*)

Create a two-dimensional matrix from a spectrum, copy values.

Parameters

spec	Source spectrum structure

5.298.2.3 MHASignal::matrix_t::matrix_t (const MHASignal::uint_vector_t & *size*, bool *b_is_complex* = `true`)

Create n-dimensional matrix, descriped by size argument.

Parameters

size	Size vector
b_is_complex	Add space for complex values

5.298.2.4 MHASignal::matrix_t::matrix_t (const MHASignal::matrix_t & *src*)

5.298.2.5 MHASignal::matrix_t::matrix_t (const uint8_t * *buf*, unsigned int *len*)

Construct from memory area.

Warning

This constructor is not real time safe

5.298.2.6 MHASignal::matrix_t::~matrix_t ( )

5.298.3 Member Function Documentation

5.298.3.1 matrix_t & MHASignal::matrix_t::operator= (const MHASignal::matrix_t & *src*)

5.298.3.2 MHASignal::matrix_t & MHASignal::matrix_t::operator= (const comm_var_t & *v*)

Fill matrix with data of an AC variable object.

Parameters

v	Source AC variable (**comm_var_t** (p. 246))

Note

The type and dimension of the AC variable must match the type and dimension of the matrix.

5.298.3.3 comm_var_t MHASignal::matrix_t::get_comm_var ( )

Return a AC communication variable pointing to the data of the current matrix.

Returns

AC variable object (**comm_var_t** (p. 246)), valid for the life time of the matrix.

5.298.3.4 unsigned int MHASignal::matrix_t::dimension ( ) const [inline]

Return the dimension of the matrix.

Returns

Dimension of the matrix

5.298.3.5 unsigned int MHASignal::matrix_t::size (unsigned int *k*) const [inline]

Return the size of the matrix.

Parameters

k	Dimension

Returns

Size of the matrix in dimension k

5.298.3.6 unsigned int MHASignal::matrix_t::get_nelements ( ) const

Return total number of elements.

5.298.3.7 bool MHASignal::matrix_t::is_same_size (const MHASignal::matrix_t & *src*)

Test if matrix has same size as other.

5.298.3.8 bool MHASignal::matrix_t::iscomplex ( ) const [inline]

Return information about complexity.

5.298.3.9 mha_real_t& MHASignal::matrix_t::real (const MHASignal::uint_vector_t & *index*) [inline]

Access real part of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.10 mha_real_t& MHASignal::matrix_t::imag (const MHASignal::uint_vector_t & *index*) [inline]

Access imaginary part of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.11 mha_complex_t& MHASignal::matrix_t::operator() (const MHASignal::uint_vector_t & *index*) [inline]

Access complex value of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.12 const mha_real_t& MHASignal::matrix_t::real (const MHASignal::uint_vector_t & *index*) const [inline]

Access real part of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.13 const mha_real_t& MHASignal::matrix_t::imag (const MHASignal::uint_vector_t & *index*) const [inline]

Access imaginary part of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.14 const mha_complex_t& MHASignal::matrix_t::operator() (const MHASignal::uint_vector_t & *index*) const [inline]

Access complex value of an element in a n-dimensional matrix.

Parameters

index	Index vector

5.298.3.15 mha_real_t& MHASignal::matrix_t::real (unsigned int *row,* unsigned int *col*) [inline]

Access real part of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.16 mha_real_t& MHASignal::matrix_t::imag (unsigned int *row,* unsigned int *col*) [inline]

Access imaginary part of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.17 mha_complex_t& MHASignal::matrix_t::operator() (unsigned int *row,* unsigned int *col*) [inline]

Access complex value of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.18 const mha_real_t& MHASignal::matrix_t::real (unsigned int *row,* unsigned int *col*) const [inline]

Access real part of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.19 const mha_real_t& MHASignal::matrix_t::imag (unsigned int *row,* unsigned int *col*) const [inline]

Access imaginary part of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.20 const mha_complex_t& MHASignal::matrix_t::operator() (unsigned int *row,* unsigned int *col*) const [inline]

Access complex value of an element in a two-dimensional matrix.

Parameters

row	Row number of element
col	Column number of element

5.298.3.21 unsigned int MHASignal::matrix_t::get_nreals ( ) const [inline]

5.298.3.22 unsigned int MHASignal::matrix_t::get_index (unsigned int *row,* unsigned int *col*) const

5.298.3.23 unsigned int MHASignal::matrix_t::get_index (const MHASignal::uint_vector_t & *index*) const

5.298.3.24 unsigned int MHASignal::matrix_t::numbytes ( ) const

Return number of bytes needed to store into memory.

5.298.3.25 unsigned int MHASignal::matrix_t::write (uint8_t * *buf*, unsigned int *len*) const

Copy to memory area.

5.298.3.26 const mha_real_t* MHASignal::matrix_t::get_rdata ( ) const [inline]

Return pointer of real data.

5.298.3.27 const mha_complex_t* MHASignal::matrix_t::get_cdata ( ) const [inline]

Return pointer of complex data.

5.298.4 Member Data Documentation

5.298.4.1 uint32_t MHASignal::matrix_t::complex_ofs [private]

5.298.4.2 uint32_t MHASignal::matrix_t::nelements [private]

5.298.4.3 mha_real_t* MHASignal::matrix_t::rdata

5.298.4.4 mha_complex_t* MHASignal::matrix_t::cdata

5.298.4.5 union { ... } [private]

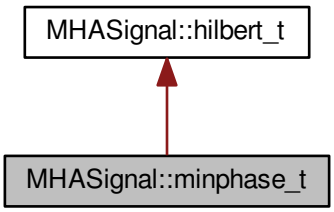
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.299 MHASignal::minphase_t Class Reference

Minimal phase function.

Inheritance diagram for MHASignal::minphase_t:



Public Member Functions

- **minphase_t** (unsigned int fftlen, unsigned int ch)
  Constructor.
- void **operator()** (**mha_spec_t** $*$s)
  Transform input spectrum to a minimal-phase spectrum, discarding the original phase.

Private Attributes

- **MHASignal::waveform_t phase**

Additional Inherited Members

5.299.1 Detailed Description

Minimal phase function.

The output spectrum $Y(f)$ is

$$Y(f) = |X(f)|e^{i\mathcal{H}\{\log|X(f)|\}},$$

with the input spectrum $X(f)$ and the Hilbert transformation $\mathcal{H}\{\cdots\}$.

5.299.2 Constructor & Destructor Documentation

5.299.2.1 MHASignal::minphase_t::minphase_t (unsigned int *fftlen,* unsigned int *ch*)

Constructor.

Parameters

fftlen	FFT length
ch	Number of channels

5.299.3 Member Function Documentation

5.299.3.1 void MHASignal::minphase_t::operator() (mha_spec_t $*$ *s*)

Transform input spectrum to a minimal-phase spectrum, discarding the original phase.

Parameters

s	Spectrum to operate on.

5.299.4 Member Data Documentation

5.299.4.1 MHASignal::waveform_t MHASignal::minphase_t::phase [private]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.300 MHASignal::quantizer_t Class Reference

Simple simulation of fixpoint quantization.

Public Member Functions

- **quantizer_t** (unsigned int num_bits)
  Constructor.
- void **operator()** (**mha_wave_t** &s)
  Quantization of a waveform fragment.

Private Attributes

- bool **limit**
- **mha_real_t upscale**
- **mha_real_t downscale**
- **mha_real_t up_limit**

5.300.1 Detailed Description

Simple simulation of fixpoint quantization.

5.300.2 Constructor & Destructor Documentation

5.300.2.1 MHASignal::quantizer_t::quantizer_t (unsigned int *num_bits*)

Constructor.

Parameters

num_bits	Number of bits to simulate, or zero for limiting to [-1,1] only.

5.300.3 Member Function Documentation

5.300.3.1 void MHASignal::quantizer_t::operator() (mha_wave_t & *s*)

Quantization of a waveform fragment.

Parameters

s	Waveform fragment to be quantized.

5.300.4 Member Data Documentation

5.300.4.1 bool MHASignal::quantizer_t::limit [private]

5.300.4.2 mha_real_t MHASignal::quantizer_t::upscale [private]

5.300.4.3 mha_real_t MHASignal::quantizer_t::downscale [private]

5.300.4.4 mha_real_t MHASignal::quantizer_t::up_limit [private]

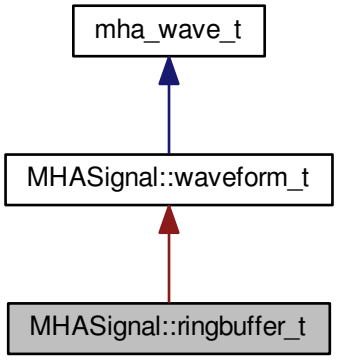
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.301 MHASignal::ringbuffer_t Class Reference

A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.

Inheritance diagram for MHASignal::ringbuffer_t:



Public Member Functions

- **ringbuffer_t** (unsigned frames, unsigned **channels**, unsigned prefilled_frames)
  Creates new ringbuffer for time domain signal.
- unsigned **contained_frames** () const
  number of currently contained frames
- **mha_real_t** & **value** (unsigned frame, unsigned channel)
  Access to value stored in ringbuffer.
- void **discard** (unsigned frames)
  Discards the oldest frames.
- void **write** (**mha_wave_t** &signal)
  Copies the contents of the signal into the ringbuffer if there is enough space.

Private Attributes

- unsigned **next_read_frame_index**
  Index of oldest frame in underlying storage for the ringbuffer.
- unsigned **next_write_frame_index**
  Index of first free frame in underlying storage.

Additional Inherited Members

5.301.1 Detailed Description

A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.

Blocks of audio signal can be placed into the ringbuffer using the **write** (p. 796) method. Individual audio samples can be accessed and altered using the **value** (p. 795) method. Blocks of audio data can be deleted from the ringbuffer using the **discard** (p. 796) method.

5.301.2 Constructor & Destructor Documentation

5.301.2.1 ringbuffer_t::ringbuffer_t (unsigned *frames,* unsigned *channels,* unsigned *prefilled_frames*)

Creates new ringbuffer for time domain signal.

Constructor allocates enough storage so that *frames* audio samples can be stored in the ringbuffer.

Parameters

frames	Size of ringbuffer in samples per channel. Maximum number of frames that can be stored in the ringbuffer at one time. This number cannot be changed after instance creation.
channels	Number of audio channels.
prefilled_frames	Number of frames to be prefilled with zero values. Many applications of a ringbuffer require the introduction of a delay. In practice, this delay is achieved by inserting silence audio samples (zeros) into the ringbuffer before the start of the actual signal is inserted for the first time.

Exceptions

MHA_Error *(p. 445)*	if prefilled_frames > frames

5.301.3 Member Function Documentation

5.301.3.1 unsigned MHASignal::ringbuffer_t::contained_frames ( ) const [inline]

number of currently contained frames

5.301.3.2 mha_real_t& MHASignal::ringbuffer_t::value (unsigned *frame,* unsigned *channel*) [inline]

Access to value stored in ringbuffer.

frame index is relative to the oldest frame stored in the ringbuffer, therefore, the meaning of the *frame* changes when the **discard** (p. 796) method is called.

Parameters

frame	frame index, 0 corresponds to oldest frame stored.
channel	audio channel

Returns

reference to contained sample value

Exceptions

MHA_Error *(p. 445)*	if channel or frame out of bounds.

5.301.3.3 void MHASignal::ringbuffer_t::discard (unsigned *frames*) [inline]

Discards the oldest frames.

Makes room for new **write** (p. 796), alters base frame index for **value** (p. 795)

Parameters

frames	how many frames to discard.

Exceptions

MHA_Error *(p. 445)*	if frames > **contained_frames** (p. 795)

5.301.3.4 void MHASignal::ringbuffer_t::write (mha_wave_t & *signal*) [inline]

Copies the contents of the signal into the ringbuffer if there is enough space.

Parameters

signal	New signal to be appended to the signal already present in the ringbuffer

Exceptions

MHA_Error *(p. 445)*	if there is not enough space or if the channel count mismatches. Nothing is copied if the space is insufficient.

5.301.4 Member Data Documentation

5.301.4.1 unsigned MHASignal::ringbuffer_t::next_read_frame_index [private]

Index of oldest frame in underlying storage for the ringbuffer.

This value is added to the frame parameter of the **value** (p. 795) method, and this value is altered when **discard** (p. 796) is called.

5.301.4.2 unsigned MHASignal::ringbuffer_t::next_write_frame_index [private]

Index of first free frame in underlying storage.

Next frame to be stored will be placed here.

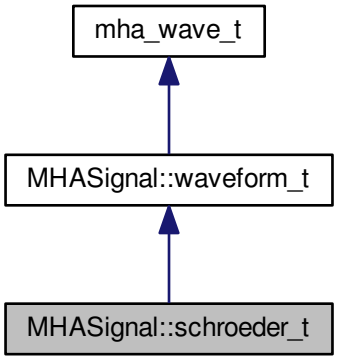
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.302 MHASignal::schroeder_t Class Reference

Schroeder tone complex class.

Inheritance diagram for MHASignal::schroeder_t:



Public Types

- typedef float(∗ **groupdelay_t**) (float f, float fmin, float fmax)

  Function type for group delay definition.

Public Member Functions

- **schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::sign_t** sign=**up**, **mha_real_t** speed=1)

  Constructor.
- **schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::groupdelay_t** freqfun=**MHASignal::schroeder_t::identity**, float fmin=0, float fmax=1, float eps=1e-10)

  Construct create Schroeder tone complex from a given frequency function.

Static Public Member Functions

- static float **identity** (float x, float, float)
- static float **log_up** (float x, float fmin, float fmax)
- static float **log_down** (float x, float fmin, float fmax)

Additional Inherited Members

5.302.1 Detailed Description

Schroeder tone complex class.

The Schroeder tone complex is a sweep defined in the sampled spectrum:

$$\Phi(f) = \sigma 2\pi\tau(2f/f_s)^{2\alpha}, \quad S(f) = e^{i\Phi(f)}$$

f is the sampled frequency in Hz, σ is the sign of the sweep (-1 for up sweep, +1 for down sweep), τ is the sweep duration in samples, f_s is the sampling rate in Hz and α is the relative sweep speed.

5.302.2 Member Typedef Documentation

5.302.2.1 typedef float(∗ MHASignal::schroeder_t::groupdelay_t) (float f, float fmin, float fmax)

Function type for group delay definition.

Parameters

f	Frequency relative to Nyquist frequency.
fmin	Minimum frequency relative to Nyquist frequency.
fmax	Maximum frequency relative to Nyquist frequency.

5.302.3 Member Enumeration Documentation

5.302.3.1 enum MHASignal::schroeder_t::sign_t

Enumerator for sign of Schroeder tone complex sweep direction.

Enumerator

up Sweep from zero to Nyquist frequency ($\sigma = -1$)

down Sweep from Nyquist frequency to zero ($\sigma = +1$)

5.302.4 Constructor & Destructor Documentation

5.302.4.1 MHASignal::schroeder_t::schroeder_t (unsigned int *len*, unsigned int *channels* = 1, schroeder_t::sign_t *sign* = up, mha_real_t *speed* = 1)

Constructor.

Parameters of the Schroeder tone complex are configured in the constructor.

Parameters

len	Length τ of the Schroeder tone complex in samples
channels	Number of channels
sign	Sign σ of Schroeder sweep
speed	Relative speed α (curvature of phase function)

5.302.4.2 MHASignal::schroeder_t::schroeder_t (unsigned int *len*, unsigned int *channels* = 1, schroeder_t::groupdelay_t *freqfun* = MHASignal::schroeder_t::identity, float *fmin* = 0, float *fmax* = 1, float *eps* = 1e-10)

Construct create Schroeder tone complex from a given frequency function.

The frequency function $g(f)$ defines the sweep speed and sign (based on the group delay). It must be defined in the interval [0,1) and should return values in the interval [0,1].

$$\Phi(f) = -4\pi\tau \int_0^{\tau} g(f)\,\mathrm{d}f, \quad S(f) = e^{i\Phi(f)}$$

Parameters

len	Length τ of the Schroeder tone complex in samples.
channels	Number of channels.
freqfun	Frequency function $g(f)$.
fmin	Start frequency (relative to Nyquist frequency).
fmax	End frequency (relative to Nyquist frequency).
eps	Stability constant for frequency ranges not covered by Schroeder tone complex.

5.302.5 Member Function Documentation

5.302.5.1 static float MHASignal::schroeder_t::identity (float *x,* float *,* float) [inline], [static]

5.302.5.2 static float MHASignal::schroeder_t::log_up (float *x,* float *fmin,* float *fmax*) [inline], [static]

5.302.5.3 static float MHASignal::schroeder_t::log_down (float *x,* float *fmin,* float *fmax*) [inline], [static]

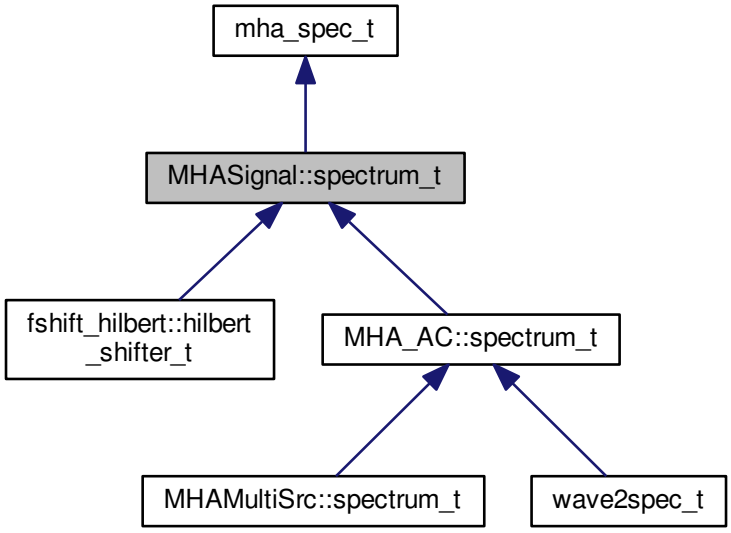
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.303 MHASignal::spectrum_t Class Reference

a signal processing class for spectral data (based on **mha_spec_t** (p. 465))

Inheritance diagram for MHASignal::spectrum_t:



Public Member Functions

- **spectrum_t** (const unsigned int &frames, const unsigned int &**channels**)
  constructor of spectrum class
- **spectrum_t** (const **mha_spec_t** &)
  Copy constructor.
- **spectrum_t** (const **MHASignal::spectrum_t** &)
  Copy constructor.
- **spectrum_t** (const std::vector< **mha_complex_t** > &)
- virtual ~**spectrum_t** (void)
- **mha_complex_t** & **operator()** (unsigned int f, unsigned int ch)
  Access to element.
- **mha_complex_t** & **operator[ ]** (unsigned int k)
  Access to a single element, direct index into data buffer.
- **mha_complex_t** & **value** (unsigned int f, unsigned int ch)
  Access to element.
- void **copy** (const **mha_spec_t** &)
  copy all elements from a spectrum
- void **copy_channel** (const **mha_spec_t** &s, unsigned sch, unsigned dch)
  Copy one channel of a given spectrum signal to a target channel.
- void **export_to** (**mha_spec_t** &)
  copy elements to spectrum structure

- void **scale** (const unsigned int &, const unsigned int &, const unsigned int &, const **mha_real_t** &)
  scale section [a,b) in channel "ch" by "val"
- void **scale_channel** (const unsigned int &, const **mha_real_t** &)
  scale all elements in one channel

Additional Inherited Members

5.303.1 Detailed Description

a signal processing class for spectral data (based on **mha_spec_t** (p. 465))

5.303.2 Constructor & Destructor Documentation

5.303.2.1 spectrum_t::spectrum_t (const unsigned int & *frames,* const unsigned int & *channels*)

constructor of spectrum class

Allocates buffers and initializes memory to zeros.

Parameters

frames	number of frames (fft bins) in one channel. Number of Frames is usually fftlen / 2 + 1
channels	number of channels

5.303.2.2 spectrum_t::spectrum_t (const **mha_spec_t** & *src*) [explicit]

Copy constructor.

5.303.2.3 spectrum_t::spectrum_t (const **MHASignal::spectrum_t** & *src*)

Copy constructor.

5.303.2.4 spectrum_t::spectrum_t (const std::vector< **mha_complex_t** > & *src*)

5.303.2.5 spectrum_t::~spectrum_t (void) [virtual]

Reimplemented in **MHA_AC::spectrum_t** (p. 423).

5.303.3 Member Function Documentation

5.303.3.1 mha_complex_t& MHASignal::spectrum_t::operator() (unsigned int *f,* unsigned int *ch*) [inline]

Access to element.

Parameters

f	Bin number
ch	Channel number

Returns

Reference to element

5.303.3.2 mha_complex_t& MHASignal::spectrum_t::operator[ ] (unsigned int *k*) [inline]

Access to a single element, direct index into data buffer.

Parameters

k	Buffer index

Returns

Reference to element

5.303.3.3 mha_complex_t& MHASignal::spectrum_t::value (unsigned int *f,* unsigned int *ch*) [inline]

Access to element.

Parameters

f	Bin number
ch	Channel number

Returns

Reference to element

5.303.3.4 void spectrum_t::copy (const mha_spec_t & *src*)

copy all elements from a spectrum

Parameters

src	input spectrum

5.303.3.5 void spectrum_t::copy_channel (const mha_spec_t & *s*, unsigned *sch*, unsigned *dch*)

Copy one channel of a given spectrum signal to a target channel.

Parameters

s	Input spectrum signal
sch	Channel index in source signal
dch	Channel index in destination (this) signal

5.303.3.6 void spectrum_t::export_to (mha_spec_t & *dest*)

copy elements to spectrum structure

Parameters

dest	destination spectrum structure

5.303.3.7 void spectrum_t::scale (const unsigned int & *a*, const unsigned int & *b*, const unsigned int & *ch*, const mha_real_t & *val*)

scale section [a,b) in channel "ch" by "val"

Parameters

a	starting frame
b	end frame (excluded)
ch	channel number
val	scale factor

5.303.3.8 void spectrum_t::scale_channel (const unsigned int & *ch,* const **mha_real_t** & *src*)

scale all elements in one channel

Parameters

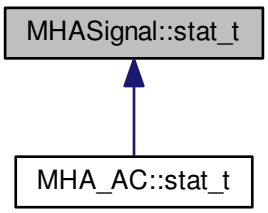
ch	channel number
src	scale factor

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.304 MHASignal::stat_t Class Reference

Inheritance diagram for MHASignal::stat_t:



Public Member Functions

- **stat_t** (const unsigned int &frames, const unsigned int &**channels**)
- void **mean** (**mha_wave_t** &m)
- void **mean_std** (**mha_wave_t** &m, **mha_wave_t** &s)
- void **push** (const **mha_wave_t** &)
- void **push** (const **mha_real_t** &x, const unsigned int &k, const unsigned int &ch)

Private Attributes

- **MHASignal::waveform_t n**
- **MHASignal::waveform_t sum**
- **MHASignal::waveform_t sum2**

5.304.1 Constructor & Destructor Documentation

5.304.1.1 MHASignal::stat_t::stat_t (const unsigned int & *frames,* const unsigned int & *channels*)

5.304.2 Member Function Documentation

5.304.2.1 void MHASignal::stat_t::mean (**mha_wave_t** & *m*)

5.304.2.2 void MHASignal::stat_t::mean_std (**mha_wave_t** & *m,* **mha_wave_t** & *s*)

5.304.2.3 void MHASignal::stat_t::push (const **mha_wave_t** & *x*)

5.304.2.4 void MHASignal::stat_t::push (const **mha_real_t** & *x,* const unsigned int & *k,* const unsigned int & *ch*)

5.304.3 Member Data Documentation

5.304.3.1 **MHASignal::waveform_t** MHASignal::stat_t::n [private]

5.304.3.2 **MHASignal::waveform_t** MHASignal::stat_t::sum [private]

5.304.3.3 **MHASignal::waveform_t** MHASignal::stat_t::sum2 [private]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.305 MHASignal::subsample_delay_t Class Reference

implements subsample delay in spectral domain.

Public Member Functions

- **subsample_delay_t** (const std::vector< float > &subsample_delay, unsigned fftlen)
  Constructor computes complex phase factors to apply to achieve subsample delay.
- void **process** (**mha_spec_t** *s)
  Apply the phase_gains to s to achieve the subsample delay.
- void **process** (**mha_spec_t** *s, unsigned idx)
  Apply the pase gains to channel idx in s to achieve the subsample delay in channel idx.

Public Attributes

- **spectrum_t phase_gains**
  The complex factors to apply to achieve the necessary phase shift.

Private Attributes

- unsigned **last_complex_bin**
  index of the last complex fft bin for the used fft length.

5.305.1 Detailed Description

implements subsample delay in spectral domain.

When transformed back to the time domain, the signal is delayed by the configured fraction of a sample. This operation must not be used in a smoothgains bracket.

5.305.2 Constructor & Destructor Documentation

5.305.2.1 MHASignal::subsample_delay_t::subsample_delay_t (const std::vector< float > & *subsample_delay*, unsigned *fftlen*)

Constructor computes complex phase factors to apply to achieve subsample delay.

Parameters

subsample_delay	The subsample delay to apply. -0.5 <= subsample_delay <= 0.5
fftlen	FFT length

Exceptions

MHA_Error *(p. 445)*	if the parameters are out of range

5.305.3 Member Function Documentation

5.305.3.1 void MHASignal::subsample_delay_t::process (mha_spec_t * s)

Apply the phase_gains to s to achieve the subsample delay.

5.305.3.2 void MHASignal::subsample_delay_t::process (mha_spec_t * s, unsigned idx)

Apply the pase gains to channel idx in s to achieve the subsample delay in channel idx.

Parameters

s	signal
idx	channel index, 0-based

Exceptions

MHA_Error *(p. 445)*	if idx >= s->num_channels

5.305.4 Member Data Documentation

5.305.4.1 spectrum_t MHASignal::subsample_delay_t::phase_gains

The complex factors to apply to achieve the necessary phase shift.

5.305.4.2 unsigned MHASignal::subsample_delay_t::last_complex_bin [private]

index of the last complex fft bin for the used fft length.

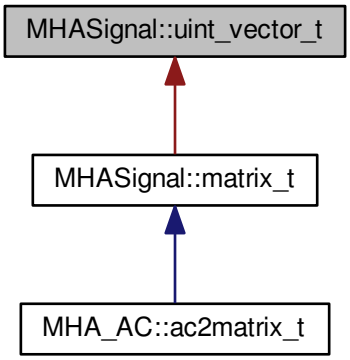
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.306 MHASignal::uint_vector_t Class Reference

Vector of unsigned values, used for size and index description of n-dimensional matrixes.

Inheritance diagram for MHASignal::uint_vector_t:



Public Member Functions

- **uint_vector_t** (unsigned int len)
  Constructor, initializes all elements to zero.
- **uint_vector_t** (const **uint_vector_t** &)
- **uint_vector_t** (const uint8_t *buf, unsigned int len)
  Construct from memory area.
- ~**uint_vector_t** ()
- bool **operator==** (const **uint_vector_t** &) const
  Check for equality.
- **uint_vector_t** & **operator=** (const **uint_vector_t** &)
  *Assign from other **uint_vector_t** (p. 809).*
- unsigned int **get_length** () const
  Return the length of the vector.
- const uint32_t & **operator[ ]** (unsigned int k) const
  Read-only access to elements.
- uint32_t & **operator[ ]** (unsigned int k)
  Access to elements.
- unsigned int **numbytes** () const
  Return number of bytes needed to store into memory.
- unsigned int **write** (uint8_t *buf, unsigned int len) const
  Copy to memory area.
- const uint32_t * **getdata** () const
  Return pointer to the data field.

Protected Attributes

- uint32_t **length**
- uint32_t * **data**

5.306.1 Detailed Description

Vector of unsigned values, used for size and index description of n-dimensional matrixes.

5.306.2 Constructor & Destructor Documentation

5.306.2.1 MHASignal::uint_vector_t::uint_vector_t (unsigned int *len*)

Constructor, initializes all elements to zero.

Parameters

len	Length of vector.

5.306.2.2 MHASignal::uint_vector_t::uint_vector_t (const uint_vector_t & *src*)

5.306.2.3 MHASignal::uint_vector_t::uint_vector_t (const uint8_t * *buf,* unsigned int *len*)

Construct from memory area.

Warning

This constructor is not real time safe

5.306.2.4 MHASignal::uint_vector_t::~uint_vector_t ( )

5.306.3 Member Function Documentation

5.306.3.1 bool MHASignal::uint_vector_t::operator== (const uint_vector_t & *src*) const

Check for equality.

5.306.3.2 uint_vector_t & MHASignal::uint_vector_t::operator= (const uint_vector_t & *src*)

Assign from other **uint_vector_t** (p. 809).

Warning

This assignment will fail if the lengths mismatch.

5.306.3.3 unsigned int MHASignal::uint_vector_t::get_length ( ) const [inline]

Return the length of the vector.

5.306.3.4 const uint32_t& MHASignal::uint_vector_t::operator[ ] (unsigned int *k*) const [inline]

Read-only access to elements.

5.306.3.5 uint32_t& MHASignal::uint_vector_t::operator[ ] (unsigned int *k*) [inline]

Access to elements.

5.306.3.6 unsigned int MHASignal::uint_vector_t::numbytes ( ) const

Return number of bytes needed to store into memory.

5.306.3.7 unsigned int MHASignal::uint_vector_t::write (uint8_t * *buf,* unsigned int *len*) const

Copy to memory area.

5.306.3.8 const uint32_t* MHASignal::uint_vector_t::getdata ( ) const [inline]

Return pointer to the data field.

5.306.4 Member Data Documentation

5.306.4.1 uint32_t MHASignal::uint_vector_t::length [protected]

5.306.4.2 uint32_t* MHASignal::uint_vector_t::data [protected]

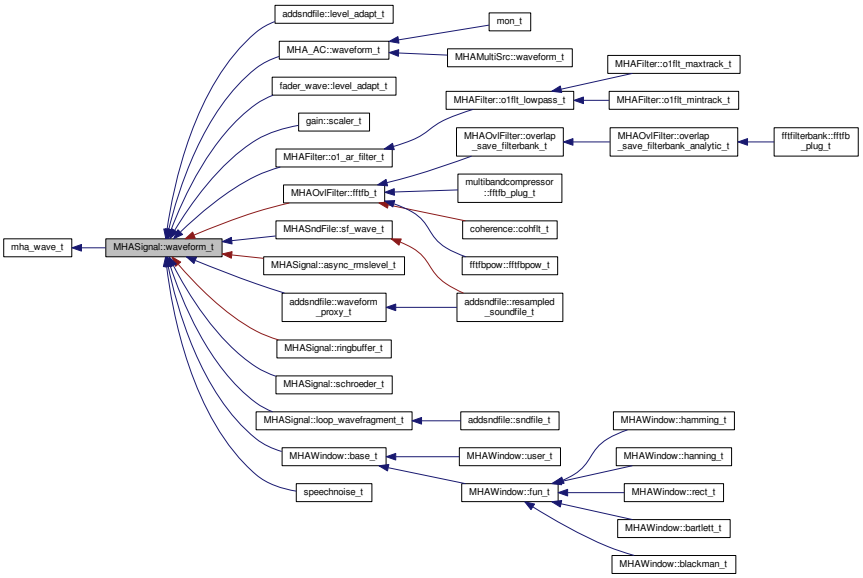
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.307 MHASignal::waveform_t Class Reference

signal processing class for waveform data (based on **mha_wave_t** (p. 496))

Inheritance diagram for MHASignal::waveform_t:



Public Member Functions

- **waveform_t** (const unsigned int &frames, const unsigned int &**channels**)
  *constructor of **waveform_t** (p. 812)*
- **waveform_t** (const **mhaconfig_t** &cf)
  Constructor to create a waveform from plugin configuration.
- **waveform_t** (const **mha_wave_t** &src)
  *Copy contructor for **mha_wave_t** (p. 496) source.*
- **waveform_t** (const **MHASignal::waveform_t** &src)
  Copy contructor.
- **waveform_t** (const std::vector< **mha_real_t** > &src)
  Copy contructor for std::vector<mha_real_t> source.
- virtual ~**waveform_t** (void)
- void **operator=** (const **mha_real_t** &v)
- **mha_real_t** & **operator[ ]** (unsigned int k)
- const **mha_real_t** & **operator[ ]** (unsigned int k) const
- **mha_real_t** & **value** (unsigned int t, unsigned int ch)
  Element accessor.
- **mha_real_t** & **operator()** (unsigned int t, unsigned int ch)
  Element accessor.
- const **mha_real_t** & **value** (unsigned int t, unsigned int ch) const

Constant element accessor.

- const **mha_real_t** & **operator()** (unsigned int t, unsigned int ch) const
  Constant element accessor.
- **mha_real_t sum** (const unsigned int &a, const unsigned int &b)
  sum of all elements between [a,b) in all channels
- **mha_real_t sum** (const unsigned int &a, const unsigned int &b, const unsigned int &ch)
  sum of all elements between [a,b) in channel ch
- **mha_real_t sum** ()
  sum of all elements
- **mha_real_t sumsqr** ()
  sum of square of all elements
- **mha_real_t sum_channel** (const unsigned int &)
  return sum of all elements in one channel
- void **assign** (const unsigned int &k, const unsigned int &ch, const **mha_real_t** &val)
  set frame "k" in channel "ch" to value "val"
- void **assign** (const **mha_real_t** &)
  set all elements to value
- void **assign_frame** (const unsigned int &k, const **mha_real_t** &val)
  assign value "val" to frame k in all channels
- void **assign_channel** (const unsigned int &c, const **mha_real_t** &val)
  assign value "val" to channel ch in all frames
- void **copy** (const std::vector< **mha_real_t** > &v)
- void **copy** (const **mha_wave_t** &)
  copy data from source into current waveform
- void **copy** (const **mha_wave_t** *)
- void **copy_channel** (const **mha_wave_t** &, unsigned int, unsigned int)
  Copy one channel of a given waveform signal to a target channel.
- void **copy_from_at** (unsigned int, unsigned int, const **mha_wave_t** &, unsigned int)
  Copy part of the source signal into part of this waveform object.
- void **export_to** (**mha_wave_t** &)
  *copy data into allocated **mha_wave_t** (p. 496) structure*
- void **limit** (const **mha_real_t** &**min**, const **mha_real_t** &**max**)
  limit target to range [min,max]
- void **power** (const **waveform_t** &)
  transform waveform signal (in Pa) to squared signal (in W/m^2)
- void **powspec** (const **mha_spec_t** &)
  get the power spectrum (in W/m^2) from a complex spectrum
- void **scale** (const unsigned int &a, const unsigned int &b, const unsigned int &ch, const **mha_real_t** &val)
  scale section [a,b) in channel "ch" by "val"
- void **scale** (const unsigned int &k, const unsigned int &ch, const **mha_real_t** &val)
  scale one element
- void **scale_channel** (const unsigned int &, const **mha_real_t** &)
  scale one channel of target with a scalar
- void **scale_frame** (const unsigned int &, const **mha_real_t** &)
- unsigned int **get_size** () const

Additional Inherited Members

5.307.1 Detailed Description

signal processing class for waveform data (based on **mha_wave_t** (p. 496))

5.307.2 Constructor & Destructor Documentation

5.307.2.1 waveform_t::waveform_t (const unsigned int & *frames,* const unsigned int & *channels*)

constructor of **waveform_t** (p. 812)

Allocates buffer memory and initializes values to zero.

Parameters

frames	number of frames in each channel
channels	number of channels

5.307.2.2 waveform_t::waveform_t (const **mhaconfig_t** & *cf*) [explicit]

Constructor to create a waveform from plugin configuration.

Parameters

cf	Plugin configuration

5.307.2.3 waveform_t::waveform_t (const **mha_wave_t** & *src*) [explicit]

Copy contructor for **mha_wave_t** (p. 496) source.

5.307.2.4 waveform_t::waveform_t (const **MHASignal::waveform_t** & *src*)

Copy contructor.

5.307.2.5 waveform_t::waveform_t (const std::vector< **mha_real_t** > & *src*)

Copy contructor for std::vector<mha_real_t> source.

A waveform structure with a single channel is created, the length is equal to the number of elements in the source vector.

5.307.2.6 waveform_t::~waveform_t (void) [virtual]

Reimplemented in **MHA_AC::waveform_t** (p. 427).

5.307.3 Member Function Documentation

5.307.3.1 void MHASignal::waveform_t::operator= (const mha_real_t & *v*) [inline]

5.307.3.2 mha_real_t& MHASignal::waveform_t::operator[ ] (unsigned int *k*) [inline]

5.307.3.3 const mha_real_t& MHASignal::waveform_t::operator[ ] (unsigned int *k*) const [inline]

5.307.3.4 mha_real_t& MHASignal::waveform_t::value (unsigned int *t,* unsigned int *ch*) [inline]

Element accessor.

Parameters

t	Frame number
ch	Channel number

Returns

Reference to element

5.307.3.5 mha_real_t& MHASignal::waveform_t::operator() (unsigned int *t,* unsigned int *ch*) [inline]

Element accessor.

Parameters

t	Frame number
ch	Channel number

Returns

Reference to element

5.307.3.6 const mha_real_t& MHASignal::waveform_t::value (unsigned int *t,* unsigned int *ch*) const [inline]

Constant element accessor.

Parameters

t	Frame number
ch	Channel number

Returns

Reference to element

5.307.3.7 const mha_real_t& MHASignal::waveform_t::operator() (unsigned int *t,* unsigned int *ch*) const [inline]

Constant element accessor.

Parameters

t	Frame number
ch	Channel number

Returns

Reference to element

5.307.3.8 mha_real_t waveform_t::sum (const unsigned int & *a,* const unsigned int & *b*)

sum of all elements between [a,b) in all channels

Parameters

a	starting frame
b	end frame (excluded)

Returns

sum

5.307.3.9 mha_real_t waveform_t::sum (const unsigned int & *a,* const unsigned int & *b,* const unsigned int & *ch*)

sum of all elements between [a,b) in channel ch

Parameters

a	starting frame
b	end frame (exluded)
ch	channel number

Returns

sum

5.307.3.10 mha_real_t waveform_t::sum ( )

sum of all elements

Returns

sum of all elements

5.307.3.11 mha_real_t waveform_t::sumsqr ( )

sum of square of all elements

Returns

sum of square of all elements

5.307.3.12 mha_real_t waveform_t::sum_channel (const unsigned int & *ch*)

return sum of all elements in one channel

Parameters

ch	channel number

Returns

sum

5.307.3.13 void waveform_t::assign (const unsigned int & *k,* const unsigned int & *ch,* const **mha_real_t** & *val*)

set frame "k" in channel "ch" to value "val"

Parameters

k	frame number
ch	channel number
val	new value

5.307.3.14 void waveform_t::assign (const **mha_real_t** & *val*)

set all elements to value

Parameters

val	new value

5.307.3.15 void waveform_t::assign_frame (const unsigned int & *k,* const **mha_real_t** & *val*)

assign value "val" to frame k in all channels

Parameters

k	frame number
val	new value

5.307.3.16 void waveform_t::assign_channel (const unsigned int & *ch,* const **mha_real_t** & *val*)

assign value "val" to channel ch in all frames

Parameters

ch	channel number
val	new value

5.307.3.17 void waveform_t::copy (const std::vector< **mha_real_t** > & *v*)

5.307.3.18 void waveform_t::copy (const **mha_wave_t** & *src*)

copy data from source into current waveform

Parameters

src	input data (need to be same size as target)

5.307.3.19 void waveform_t::copy (const **mha_wave_t** * *src*)

5.307.3.20 void waveform_t::copy_channel (const **mha_wave_t** & *src,* unsigned int *src_channel,* unsigned int *dest_channel*)

Copy one channel of a given waveform signal to a target channel.

Parameters

src	Input waveform signal
src_channel	Channel in source signal
dest_channel	Channel number in destination signal

5.307.3.21 void waveform_t::copy_from_at (unsigned int *to_pos*, unsigned int *len*, const **mha_wave_t** & *src*, unsigned int *from_pos*)

Copy part of the source signal into part of this waveform object.

Source and target have to have the same number of channels.

Parameters

to_pos	Offset in target
len	Number of frames copied
src	Source
from_pos	Offset in source

5.307.3.22 void waveform_t::export_to (**mha_wave_t** & *dest*)

copy data into allocated **mha_wave_t** (p. 496) structure

Parameters

dest	destination structure

5.307.3.23 void waveform_t::limit (const **mha_real_t** & *min*, const **mha_real_t** & *max*)

limit target to range [min,max]

Parameters

min	lower limit
max	upper limit

5.307.3.24 void waveform_t::power (const **waveform_t** & *src*)

transform waveform signal (in Pa) to squared signal (in W/m^2)

Parameters

src	linear waveform signal (in Pa)

5.307.3.25 void waveform_t::powspec (const mha_spec_t & *src*)

get the power spectrum (in W/m^2) from a complex spectrum

Parameters

src	complex spectrum (normalized to Pa)

5.307.3.26 void waveform_t::scale (const unsigned int & *a,* const unsigned int & *b,* const unsigned int & *ch,* const mha_real_t & *val*)

scale section [a,b) in channel "ch" by "val"

Parameters

a	starting frame
b	end frame (excluded)
ch	channel number
val	scale factor

5.307.3.27 void waveform_t::scale (const unsigned int & *k,* const unsigned int & *ch,* const mha_real_t & *val*)

scale one element

Parameters

k	frame number
ch	channel number
val	scale factor

5.307.3.28 void waveform_t::scale_channel (const unsigned int & *ch,* const mha_real_t & *src*)

scale one channel of target with a scalar

Parameters

ch	channel number
src	factor

5.307.3.29 void waveform_t::scale_frame (const unsigned int & *frame,* const **mha_real_t** & *val*)

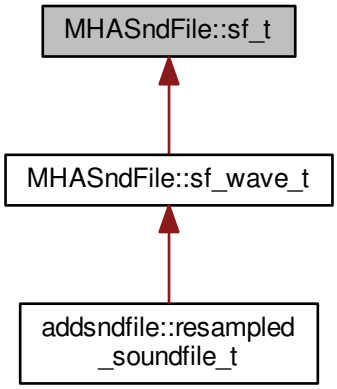
5.307.3.30 unsigned int MHASignal::waveform_t::get_size ( ) const [inline]

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.308 MHASndFile::sf_t Class Reference

Inheritance diagram for MHASndFile::sf_t:



Public Member Functions

- **sf_t** (const std::string &fname)
- ~**sf_t** ()

Public Attributes

- SNDFILE ∗ **sf**

5.308.1 Constructor & Destructor Documentation

5.308.1.1 MHASndFile::sf_t::sf_t (const std::string & *fname*)

5.308.1.2 MHASndFile::sf_t::∼sf_t ( )

5.308.2 Member Data Documentation

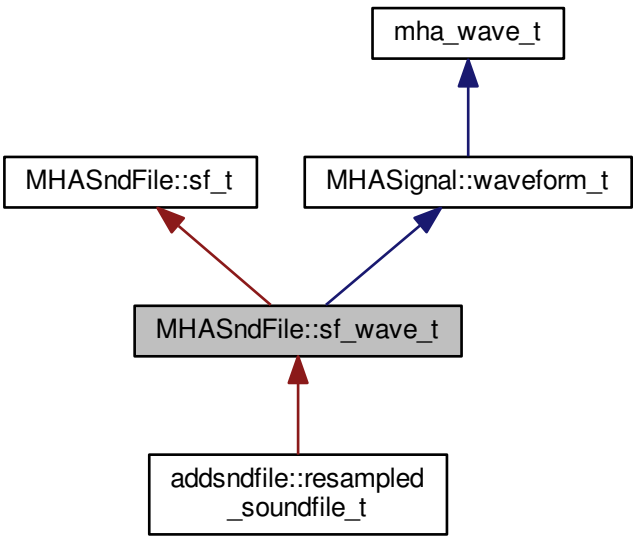
5.308.2.1 SNDFILE∗ MHASndFile::sf_t::sf

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

- **mhasndfile.h**
- **mhasndfile.cpp**

5.309 MHASndFile::sf_wave_t Class Reference

Inheritance diagram for MHASndFile::sf_wave_t:



Public Member Functions

- **sf_wave_t** (const std::string &fname, **mha_real_t** peaklevel_db, unsigned int maxlen=std::numeric_limits< unsigned int >::**max**(), unsigned int startpos=0, std::vector< int > channel_map=std::vector< int >())

Additional Inherited Members

5.309.1 Constructor & Destructor Documentation

5.309.1.1 MHASndFile::sf_wave_t::sf_wave_t (

const std::string & *fname,*
mha_real_t *peaklevel_db,*
unsigned int *maxlen =* `std::numeric_limits<unsigned int>::` **max** `()`*,*
unsigned int *startpos =* `0`*,*
std::vector< int > *channel_map =* `std::vector<int>()`)

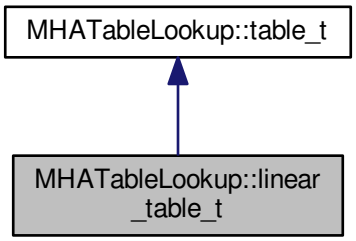
The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

5.310 MHATableLookup::linear_table_t Class Reference

Class for interpolation with equidistant x values.

Inheritance diagram for MHATableLookup::linear_table_t:



Public Member Functions

- **linear_table_t** (void)

  *contructor creates an empty **linear_table_t** (p. 824) object.*
- **mha_real_t lookup** (**mha_real_t** x) const

  look up the y value that is stored for the mesh point where x is lower than or equal to the x value given here.
- **mha_real_t interp** (**mha_real_t** x) const

  interpolate y value for the given x value.
- ∼**linear_table_t** (void)

  destructor
- void **set_xmin** (**mha_real_t xmin**)

  set the x value for the first mesh point.
- void **add_entry** (**mha_real_t** y)

  set the y value for the next mesh point.
- void **set_xmax** (**mha_real_t xmax**)

  this sets the x value for a past-the-end, not added mesh point.
- void **prepare** (void)

  prepare computes the x distance of the mesh points based on the values given to set_xmin, set_xmax, and the number of times that add_entry was called.
- void **clear** (void)

  clear resets the state of this object to the state directly after construction.

Protected Attributes

- **mha_real_t** ∗ **vy**
- unsigned int **len**

Private Attributes

- vector< **mha_real_t** > **vec_y**
- **mha_real_t xmin**
- **mha_real_t xmax**
- **mha_real_t scalefac**

Additional Inherited Members

5.310.1 Detailed Description

Class for interpolation with equidistant x values.

This class can be used for linear interpolation tasks where the mesh points are known for equidistant x values.

Before the class can be used for interpolation, it has to be filled with the y values for the mesh points, the x range has to be specified, and when all values are given, the prepare method has to be called so that the object can determine the distance between x values from the range and the number of mesh points given.

Only after prepare has returned, the object may be used for interpolation.

5.310.2 Constructor & Destructor Documentation

5.310.2.1 linear_table_t::linear_table_t (void)

contructor creates an empty **linear_table_t** (p. 824) object.

add_entry, set_xmin, set_xmax and prepare methods have to be called before the object can be used to lookup and interpolate values.

5.310.2.2 linear_table_t::~linear_table_t (void)

destructor

5.310.3 Member Function Documentation

5.310.3.1 mha_real_t linear_table_t::lookup (mha_real_t *x*) const [virtual]

look up the y value that is stored for the mesh point where x is lower than or equal to the x value given here.

This method does not extrapolate, so for x $<$ xmin, the y value for xmin is returned. For all x greater than the x of the last mesh point, the y value of the last mesh point is returned.

Precondition

prepare must have been called before lookup may be called.

Implements **MHATableLookup::table_t** (p. 829).

5.310.3.2 mha_real_t linear_table_t::interp (mha_real_t *x*) const [virtual]

interpolate y value for the given x value.

The y values for the neighbouring mesh points are looked up and linearly interpolated. For x values outside the range of mesh points, the y value is extrapolated from the nearest two mesh points.

Precondition

prepare must have been called before interp may be called.

Implements **MHATableLookup::table_t** (p. 829).

5.310.3.3 void linear_table_t::set_xmin (mha_real_t *xmin*)

set the x value for the first mesh point.

Must be called before prepare can be called.

5.310.3.4 void linear_table_t::add_entry (mha_real_t *y*)

set the y value for the next mesh point.

Must be called at least twice before prepare can be called.

5.310.3.5 void linear_table_t::set_xmax (mha_real_t *xmax*)

this sets the x value for a past-the-end, not added mesh point.

Example:

```
t.set_xmin(100);
t.add_entry(0); // mesh point {100,0}
t.add_entry(1); // mesh point {110,1}
// the next mesh point would be at x=120, but we do not add this
t.set_xmax(120); // the x where the next mesh point would be
t.prepare();
```

now, t.interp(100) == 0; t.interp(110) == 1; t.interp(105) == 0.5;

5.310.3.6 void linear_table_t::prepare (void)

prepare computes the x distance of the mesh points based on the values given to set_xmin, set_xmax, and the number of times that add_entry was called.

Precondition

set_xmin, set_xmax, add_entry functions must have been called before calling prepare, add_entry must have been called at least twice.

Only after this method has been called, interp or lookup may be called.

5.310.3.7 void linear_table_t::clear (void) [virtual]

clear resets the state of this object to the state directly after construction.

mesh entries and x range are deleted.

interp and lookup may not be called after this function has been called unless prepare and before that its precondition methods are called again.

Implements **MHATableLookup::table_t** (p. 829).

5.310.4 Member Data Documentation

5.310.4.1 mha_real_t* MHATableLookup::linear_table_t::vy [protected]

5.310.4.2 unsigned int MHATableLookup::linear_table_t::len [protected]

5.310.4.3 vector<mha_real_t> MHATableLookup::linear_table_t::vec_y [private]

5.310.4.4 mha_real_t MHATableLookup::linear_table_t::xmin [private]

5.310.4.5 mha_real_t MHATableLookup::linear_table_t::xmax [private]

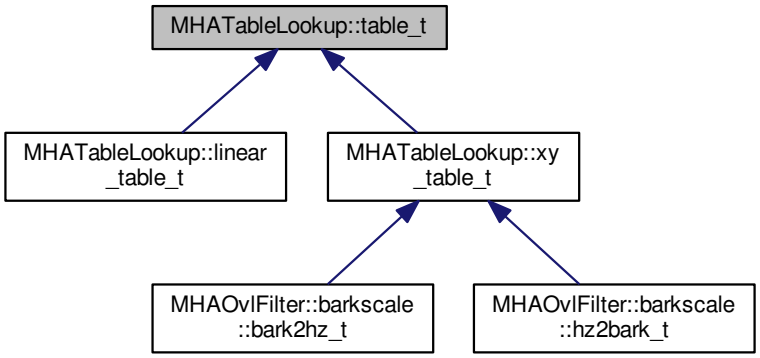
5.310.4.6 mha_real_t MHATableLookup::linear_table_t::scalefac [private]

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

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.311 MHATableLookup::table_t Class Reference

Inheritance diagram for MHATableLookup::table_t:



Public Member Functions

- **table_t** (void)
- virtual ~**table_t** (void)
- virtual **mha_real_t lookup** (**mha_real_t**) const =0
- virtual **mha_real_t interp** (**mha_real_t**) const =0

Protected Member Functions

- virtual void **clear** (void)=0

5.311.1 Constructor & Destructor Documentation

5.311.1.1 table_t::table_t (void)

5.311.1.2 table_t::~table_t (void) [virtual]

5.311.2 Member Function Documentation

5.311.2.1 virtual mha_real_t MHATableLookup::table_t::lookup (mha_real_t) const [pure virtual]

Implemented in **MHATableLookup::xy_table_t** (p. 831), and **MHATableLookup::linear_table_t** (p. 826).

5.311.2.2 virtual mha_real_t MHATableLookup::table_t::interp (mha_real_t) const [pure virtual]

Implemented in **MHATableLookup::xy_table_t** (p. 832), and **MHATableLookup::linear_table_t** (p. 826).

5.311.2.3 virtual void MHATableLookup::table_t::clear (void) [protected], [pure virtual]

Implemented in **MHATableLookup::xy_table_t** (p. 832), and **MHATableLookup::linear_table_t** (p. 828).

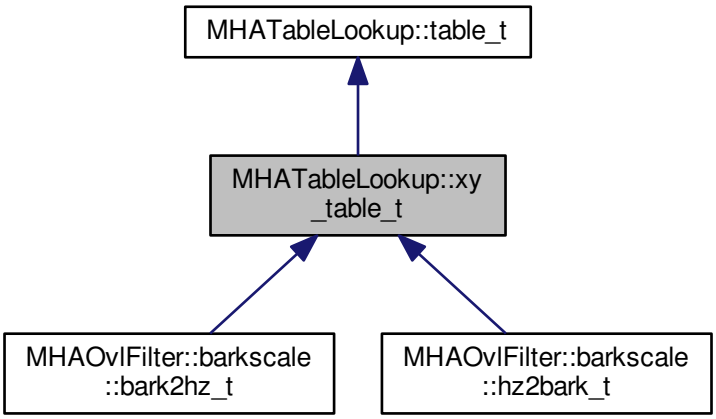
The documentation for this class was generated from the following files:

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.312 MHATableLookup::xy_table_t Class Reference

Class for interpolation with non-equidistant x values.

Inheritance diagram for MHATableLookup::xy_table_t:



Public Member Functions

- **xy_table_t** ()
- **mha_real_t lookup** (**mha_real_t** x) const
  Return the y-value at the position of the nearest x value below input.
- **mha_real_t interp** (**mha_real_t** x) const
  Linear interpolation function.
- void **add_entry** (**mha_real_t** x, **mha_real_t** y)
  Add a single x-y pair entry.
- void **add_entry** (**mha_real_t** ∗pVX, **mha_real_t** ∗pVY, unsigned int len)
  Add multiple entries at once.
- void **clear** ()
  Clear the table and transformation functions.
- void **set_xfun** (float(∗pXFun)(float))
  Set transformation function for x values.
- void **set_yfun** (float(∗pYFun)(float))
  Set transformation function for y values during insertion.
- void **set_xyfun** (float(∗pYFun)(float, float))
  Set transformation function for y values during insertion, based on x and y values.
- std::pair< **mha_real_t**, **mha_real_t** > **get_xlimits** () const
  returns the min and max x of all mesh points that are stored in the lookup table, i.e.

Private Attributes

- std::map< **mha_real_t**, **mha_real_t** > **mXY**
- float(* **xfun**)(float)
- float(* **yfun**)(float)
- float(* **xyfun**)(float, float)

Additional Inherited Members

5.312.1 Detailed Description

Class for interpolation with non-equidistant x values.

Linear interpolation of the x-y table is performed. A transformation of x and y-values is possible; if a transformation function is provided for the x-values, the same function is applied to the argument of **xy_table_t::interp()** (p. 832) and **xy_table_t::lookup()** (p. 831). The transformation of y values is applied only during insertion into the table. Two functions for y-transformation can be provided: a simple transformation which depends only on the y values, or a transformation which takes both (non-transformed) x and y value as an argument. The two-argument transformation is applied before the one-argument transformation.

5.312.2 Constructor & Destructor Documentation

5.312.2.1 xy_table_t::xy_table_t ( )

5.312.3 Member Function Documentation

5.312.3.1 mha_real_t xy_table_t::lookup (mha_real_t *x*) const [virtual]

Return the y-value at the position of the nearest x value below input.

Parameters

x	Input value

Returns

y value at nearest x value below input.

Implements **MHATableLookup::table_t** (p. 829).

5.312.3.2 mha_real_t xy_table_t::interp (mha_real_t *x*) const [virtual]

Linear interpolation function.

Parameters

x	x value

Returns

interpolated y value

Implements **MHATableLookup::table_t** (p. 829).

5.312.3.3 void xy_table_t::add_entry (mha_real_t *x*, mha_real_t *y*)

Add a single x-y pair entry.

Parameters

x	x value
y	corresponding y value

5.312.3.4 void xy_table_t::add_entry (mha_real_t * *pVX*, mha_real_t * *pVY*, unsigned int *uLength*)

Add multiple entries at once.

Parameters

pVX	array of x values
pVY	array of y values
uLength	Length of x and y arrays

5.312.3.5 void xy_table_t::clear (void) [virtual]

Clear the table and transformation functions.

Implements **MHATableLookup::table_t** (p. 829).

5.312.3.6 void xy_table_t::set_xfun (float(∗)(float) *fun*)

Set transformation function for x values.

Parameters

fun	Transformation function.

5.312.3.7 void xy_table_t::set_yfun (float(∗)(float) *fun*)

Set transformation function for y values during insertion.

Parameters

fun	Transformation function.

5.312.3.8 void xy_table_t::set_xyfun (float(∗)(float, float) *fun*)

Set transformation function for y values during insertion, based on x and y values.

Parameters

fun	Transformation function.

5.312.3.9 std::pair<mha_real_t,mha_real_t> MHATableLookup::xy_table_t::get_xlimits ( ) const [inline]

returns the min and max x of all mesh points that are stored in the lookup table, i.e. after transformation with xfun, if any. Not real-time safe

5.312.4 Member Data Documentation

5.312.4.1 std::map<mha_real_t,mha_real_t> MHATableLookup::xy_table_t::mXY [private]

5.312.4.2 float(∗ MHATableLookup::xy_table_t::xfun) (float) [private]

5.312.4.3 float(∗ MHATableLookup::xy_table_t::yfun) (float) [private]

5.312.4.4 float(∗ MHATableLookup::xy_table_t::xyfun) (float, float) [private]

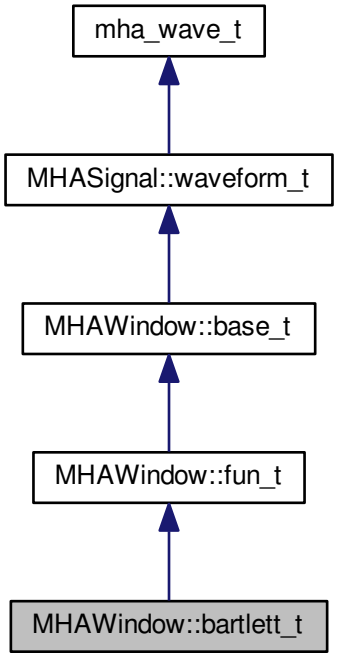
The documentation for this class was generated from the following files:

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.313 MHAWindow::bartlett_t Class Reference

Bartlett window.

Inheritance diagram for MHAWindow::bartlett_t:



Public Member Functions

- **bartlett_t** (unsigned int n)

Additional Inherited Members

5.313.1 Detailed Description

Bartlett window.

5.313.2 Constructor & Destructor Documentation

5.313.2.1 MHAWindow::bartlett_t::bartlett_t (unsigned int *n*) [inline]

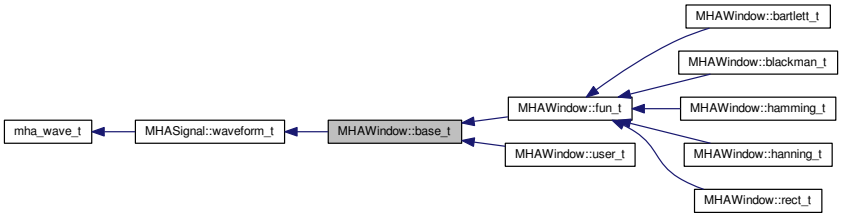
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.314 MHAWindow::base_t Class Reference

Common base for window types.

Inheritance diagram for MHAWindow::base_t:



Public Member Functions

- **base_t** (unsigned int len)
  Constructor.
- **base_t** (const **MHAWindow::base_t** &src)
  Copy constructor.
- void **operator()** (**mha_wave_t** &) const
  Apply window to waveform segment (reference)
- void **operator()** (**mha_wave_t** ∗) const
  Apply window to waveform segment (pointer)
- void **ramp_begin** (**mha_wave_t** &) const
  Apply a ramp at the begining.
- void **ramp_end** (**mha_wave_t** &) const
  Apply a ramp at the end.

Additional Inherited Members

5.314.1 Detailed Description

Common base for window types.

5.314.2 Constructor & Destructor Documentation

5.314.2.1 MHAWindow::base_t::base_t (unsigned int *len*)

Constructor.

Parameters

len	Window length in samples.

5.314.2.2 MHAWindow::base_t::base_t (const MHAWindow::base_t & *src*)

Copy constructor.

Parameters

src	Source to be copied

5.314.3 Member Function Documentation

5.314.3.1 void MHAWindow::base_t::operator() (mha_wave_t & *s*) const

Apply window to waveform segment (reference)

5.314.3.2 void MHAWindow::base_t::operator() (mha_wave_t * *s*) const

Apply window to waveform segment (pointer)

5.314.3.3 void MHAWindow::base_t::ramp_begin (mha_wave_t & *s*) const

Apply a ramp at the begining.

5.314.3.4 void MHAWindow::base_t::ramp_end (mha_wave_t & *s*) const

Apply a ramp at the end.

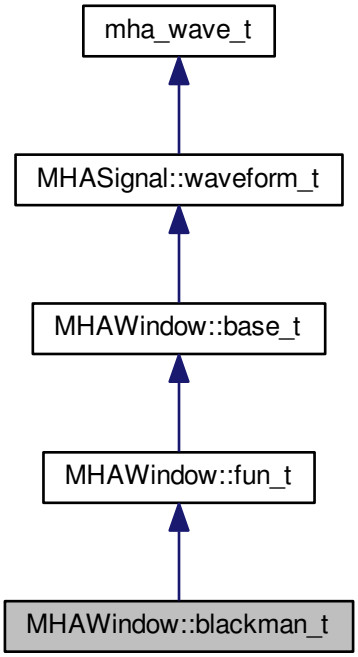
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.315 MHAWindow::blackman_t Class Reference

Blackman window.

Inheritance diagram for MHAWindow::blackman_t:



Public Member Functions

- **blackman_t** (unsigned int n)

Additional Inherited Members

5.315.1 Detailed Description

Blackman window.

5.315.2 Constructor & Destructor Documentation

5.315.2.1 MHAWindow::blackman_t::blackman_t (unsigned int *n*) [inline]

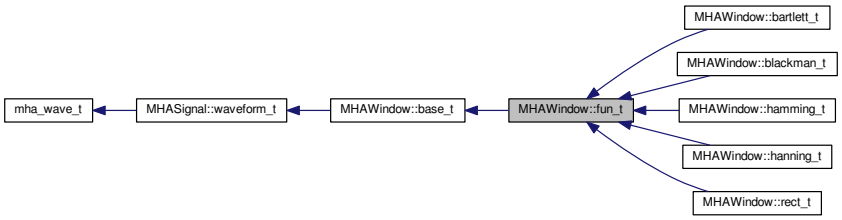
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.316 MHAWindow::fun_t Class Reference

Generic window based on a generator function.

Inheritance diagram for MHAWindow::fun_t:



Public Member Functions

- **fun_t** (unsigned int n, float(∗fun)(float), float xmin=-1, float xmax=1, bool min_↩ included=true, bool max_included=false)

  Constructor.

Additional Inherited Members

5.316.1 Detailed Description

Generic window based on a generator function.

The generator function should return a valid window function in the interval [-1,1[.

5.316.2 Constructor & Destructor Documentation

5.316.2.1 MHAWindow::fun_t::fun_t (

unsigned int ***n,***
float(∗)(float) ***fun,***
float ***xmin =*** `−1`**,**
float ***xmax =*** `1`**,**
bool ***min_included =*** `true`**,**
bool ***max_included =*** `false` **)**

Constructor.

Parameters

n	Window length
fun	Generator function, i.e. **MHAWindow::hanning()** (p. 127)
xmin	Start value of window, i.e. -1 for full window or 0 for fade-out ramp.
xmax	Last value of window, i.e. 1 for full window
min_included	Flag if minimum value is included
max_included	Flag if maximum value is included

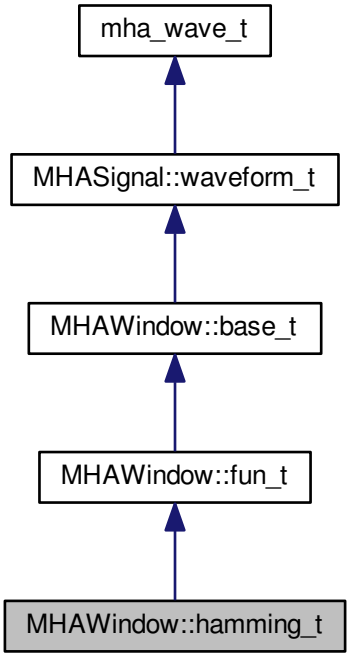
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.317 MHAWindow::hamming_t Class Reference

Hamming window.

Inheritance diagram for MHAWindow::hamming_t:



Public Member Functions

- **hamming_t** (unsigned int n)

Additional Inherited Members

5.317.1 Detailed Description

Hamming window.

5.317.2 Constructor & Destructor Documentation

5.317.2.1 MHAWindow::hamming_t::hamming_t (unsigned int *n*) [inline]

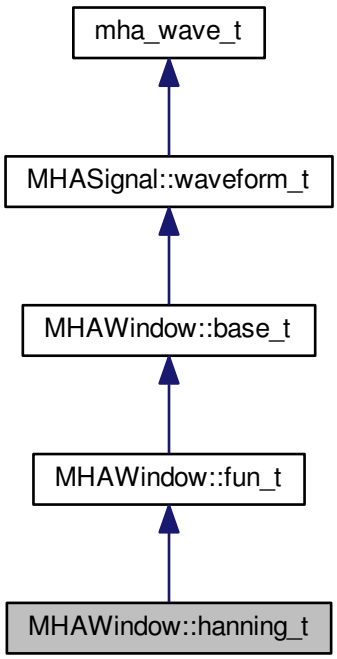
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.318 MHAWindow::hanning_t Class Reference

von-Hann window

Inheritance diagram for MHAWindow::hanning_t:



Public Member Functions

- **hanning_t** (unsigned int n)

Additional Inherited Members

5.318.1 Detailed Description

von-Hann window

5.318.2 Constructor & Destructor Documentation

5.318.2.1 MHAWindow::hanning_t::hanning_t (unsigned int *n*) [inline]

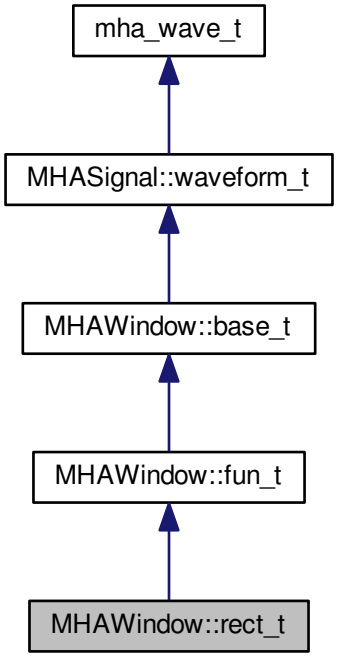
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.319 MHAWindow::rect_t Class Reference

Rectangular window.

Inheritance diagram for MHAWindow::rect_t:



Public Member Functions

- **rect_t** (unsigned int n)

Additional Inherited Members

5.319.1 Detailed Description

Rectangular window.

5.319.2 Constructor & Destructor Documentation

5.319.2.1 MHAWindow::rect_t::rect_t (unsigned int *n*) [inline]

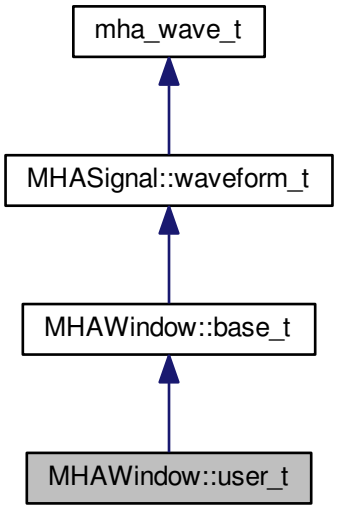
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.320 MHAWindow::user_t Class Reference

User defined window.

Inheritance diagram for MHAWindow::user_t:



Public Member Functions

- **user_t** (const std::vector< **mha_real_t** > &wnd)
  Constructor.

Additional Inherited Members

5.320.1 Detailed Description

User defined window.

5.320.2 Constructor & Destructor Documentation

5.320.2.1 MHAWindow::user_t::user_t (const std::vector< **mha_real_t** > & *wnd*)

Constructor.

Parameters

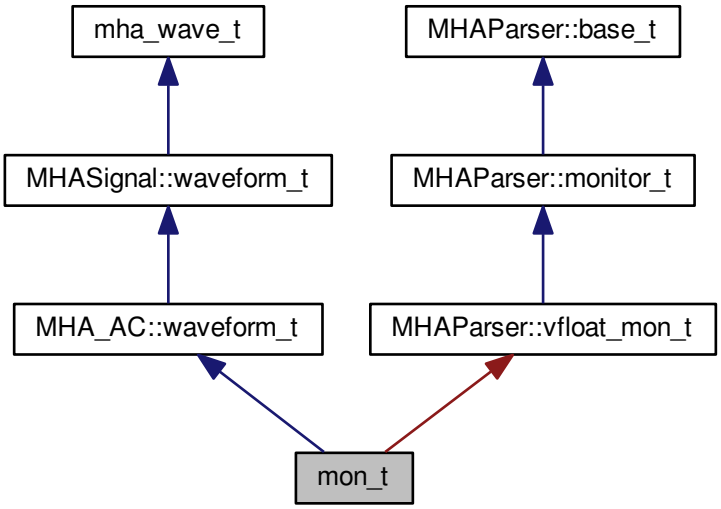
wnd	User defined window

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.321 mon_t Class Reference

Inheritance diagram for mon_t:



Public Member Functions

- **mon_t** (unsigned int nch, std::string **name**, **algo_comm_t ac**, std::string base, **MHA↩ Parser::parser_t** &p, std::string **help**)
- void **store** ()

Additional Inherited Members

5.321.1 Constructor & Destructor Documentation

5.321.1.1 mon_t::mon_t (

unsigned int *nch,*
std::string *name,*
algo_comm_t *ac,*
std::string *base,*
MHAParser::parser_t & *p,*
std::string *help*)

5.321.2 Member Function Documentation

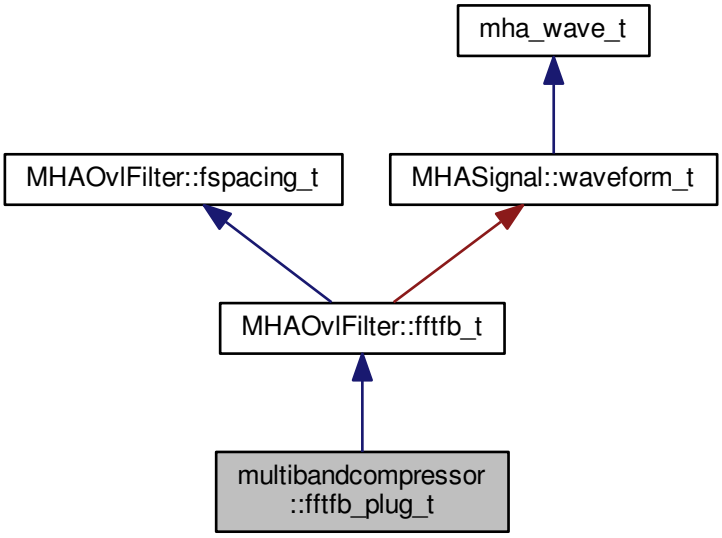
5.321.2.1 void mon_t::store ( )

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

- **rmslevel.cpp**

5.322 multibandcompressor::fftfb_plug_t Class Reference

Inheritance diagram for multibandcompressor::fftfb_plug_t:



Public Member Functions

- **fftfb_plug_t** (**MHAOvlFilter::fftfb_vars_t** &, const **mhaconfig_t** &cfg, **algo_comm_↩ t** ac, std::string alg)
- void **insert** ()

Private Attributes

- **MHA_AC::waveform_t cfv**

  vector of nominal center frequencies / Hz
- **MHA_AC::waveform_t efv**

  vector of edge frequencies / Hz
- **MHA_AC::waveform_t bwv**

  vector of band-weigths (sum of squared fft-bin-weigths)/num_frames

Additional Inherited Members

5.322.1 Constructor & Destructor Documentation

5.322.1.1 multibandcompressor::fftfb_plug_t::fftfb_plug_t (MHAOvlFilter::fftfb_vars_t & *vars,* const mhaconfig_t & *cfg,* algo_comm_t *ac,* std::string *alg*)

5.322.2 Member Function Documentation

5.322.2.1 void multibandcompressor::fftfb_plug_t::insert ( )

5.322.3 Member Data Documentation

5.322.3.1 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::cfv [private]

vector of nominal center frequencies / Hz

5.322.3.2 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::efv [private]

vector of edge frequencies / Hz

5.322.3.3 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::bwv [private]

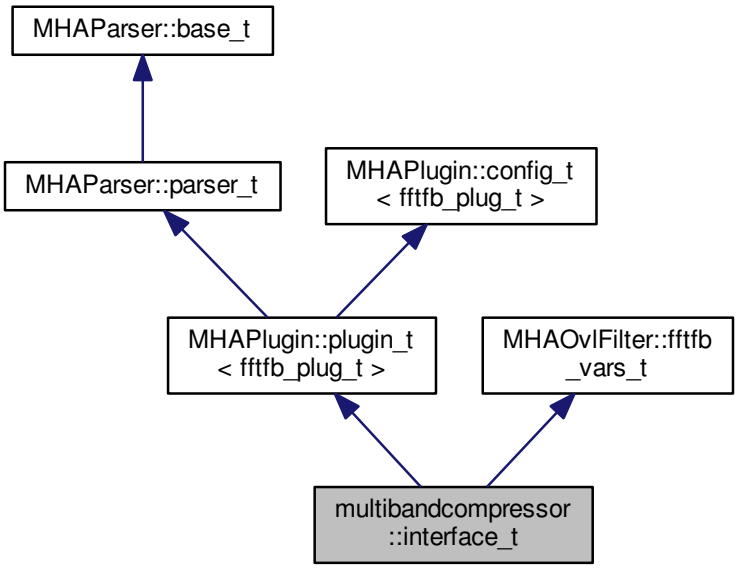
vector of band-weigths (sum of squared fft-bin-weigths)/num_frames

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

- **multibandcompressor.cpp**

5.323 multibandcompressor::interface_t Class Reference

Inheritance diagram for multibandcompressor::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- int **num_channels**
- **DynComp::dc_afterburn_t burn**
- **MHAEvents::patchbay_t**< **interface_t** > **patchbay**
- std::string **algo**
- **MHAParser::mhapluginloader_t plug**
- **plugin_signals_t** ∗ **plug_sigs**

Additional Inherited Members

5.323.1 Constructor & Destructor Documentation

5.323.1.1 multibandcompressor::interface_t::interface_t (const algo_comm_t & *ac_*, const std::string & *th*, const std::string & *al*)

Default values are set and MHA configuration variables registered into the parser.

Parameters

*ac_*	algorithm communication handle
th	chain name
al	algorithm name

5.323.2 Member Function Documentation

5.323.2.1 void multibandcompressor::interface_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 730).

5.323.2.2 void multibandcompressor::interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 731).

5.323.2.3 mha_spec_t * multibandcompressor::interface_t::process (mha_spec_t * *s*)

5.323.2.4 void multibandcompressor::interface_t::update_cfg (void) [private]

5.323.3 Member Data Documentation

5.323.3.1 int multibandcompressor::interface_t::num_channels [private]

5.323.3.2 DynComp::dc_afterburn_t multibandcompressor::interface_t::burn [private]

5.323.3.3 MHAEvents::patchbay_t<interface_t> multibandcompressor::interface_t::patchbay [private]

5.323.3.4 std::string multibandcompressor::interface_t::algo [private]

5.323.3.5 MHAParser::mhapluginloader_t multibandcompressor::interface_t::plug [private]

5.323.3.6 plugin_signals_t* multibandcompressor::interface_t::plug_sigs [private]

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

- **multibandcompressor.cpp**

5.324 multibandcompressor::plugin_signals_t Class Reference

Public Member Functions

- **plugin_signals_t** (unsigned int **channels**, unsigned int bands)
- void **update_levels** (**MHAOvlFilter::fftfb_t** *, **mha_spec_t** *s_in)
- void **apply_gains** (**MHAOvlFilter::fftfb_t** *, **DynComp::dc_afterburn_t** &burn, **mha_spec_t** *s_out)

Public Attributes

- **mha_wave_t** * **plug_output**

Private Attributes

- **MHASignal::waveform_t plug_level**
- **MHASignal::waveform_t gain**

5.324.1 Constructor & Destructor Documentation

5.324.1.1 multibandcompressor::plugin_signals_t::plugin_signals_t (unsigned int *channels,* unsigned int *bands*)

5.324.2 Member Function Documentation

5.324.2.1 void multibandcompressor::plugin_signals_t::update_levels (MHAOvlFilter::fftfb_t * *pFb,* mha_spec_t * *s_in*)

5.324.2.2 void multibandcompressor::plugin_signals_t::apply_gains (MHAOvlFilter::fftfb_t * *pFb,* DynComp::dc_afterburn_t & *burn,* mha_spec_t * *s_out*)

5.324.3 Member Data Documentation

5.324.3.1 MHASignal::waveform_t multibandcompressor::plugin_signals_t::plug_level [private]

5.324.3.2 MHASignal::waveform_t multibandcompressor::plugin_signals_t::gain [private]

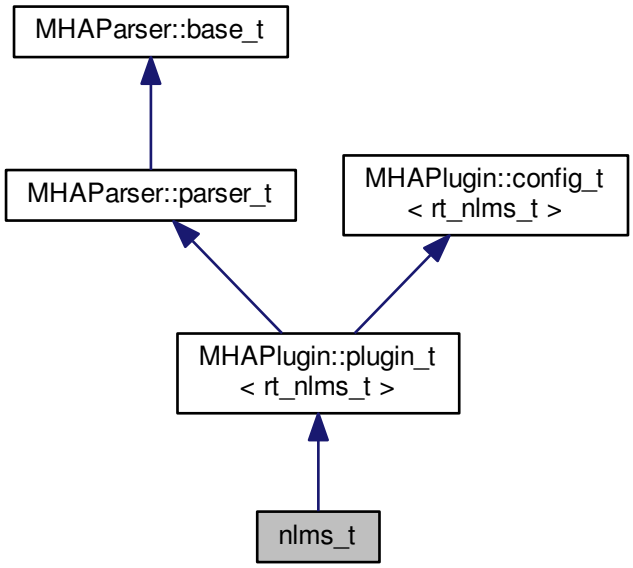
5.324.3.3 mha_wave_t* multibandcompressor::plugin_signals_t::plug_output

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

- **multibandcompressor.cpp**

5.325 nlms_t Class Reference

Inheritance diagram for nlms_t:



Public Member Functions

- **nlms_t** (**algo_comm_t**, const char ∗, const char ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::float_t rho**
- **MHAParser::float_t c**
- **MHAParser::int_t ntaps**
- **MHAParser::string_t name_u**
- **MHAParser::string_t name_d**
- **MHAParser::kw_t normtype**

- **MHAParser::kw_t estimtype**
- **MHAParser::float_t lambda_smoothing_power**
- **MHAParser::string_t name_e**
- **MHAParser::string_t name_f**
- **MHAParser::int_t n_no_update**
- std::string **algo**
- **MHAEvents::patchbay_t< nlms_t > patchbay**

Additional Inherited Members

5.325.1 Constructor & Destructor Documentation

5.325.1.1 nlms_t::nlms_t (

algo_comm_t *ac*,
const char * ,
const char * *ialg*)

5.325.2 Member Function Documentation

5.325.2.1 void nlms_t::prepare (

mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< rt_nlms_t >** (p. 730).

5.325.2.2 void nlms_t::release (

void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< rt_nlms_t >** (p. 731).

5.325.2.3 mha_wave_t * nlms_t::process (

mha_wave_t * *s*)

5.325.2.4 void nlms_t::update ( ) [private]

5.325.3 Member Data Documentation

5.325.3.1 MHAParser::float_t nlms_t::rho [private]

5.325.3.2 MHAParser::float_t nlms_t::c [private]

5.325.3.3 MHAParser::int_t nlms_t::ntaps [private]

5.325.3.4 MHAParser::string_t nlms_t::name_u [private]

5.325.3.5 MHAParser::string_t nlms_t::name_d [private]

5.325.3.6 MHAParser::kw_t nlms_t::normtype [private]

5.325.3.7 MHAParser::kw_t nlms_t::estimtype [private]

5.325.3.8 MHAParser::float_t nlms_t::lambda_smoothing_power [private]

5.325.3.9 MHAParser::string_t nlms_t::name_e [private]

5.325.3.10 MHAParser::string_t nlms_t::name_f [private]

5.325.3.11 MHAParser::int_t nlms_t::n_no_update [private]

5.325.3.12 std::string nlms_t::algo [private]

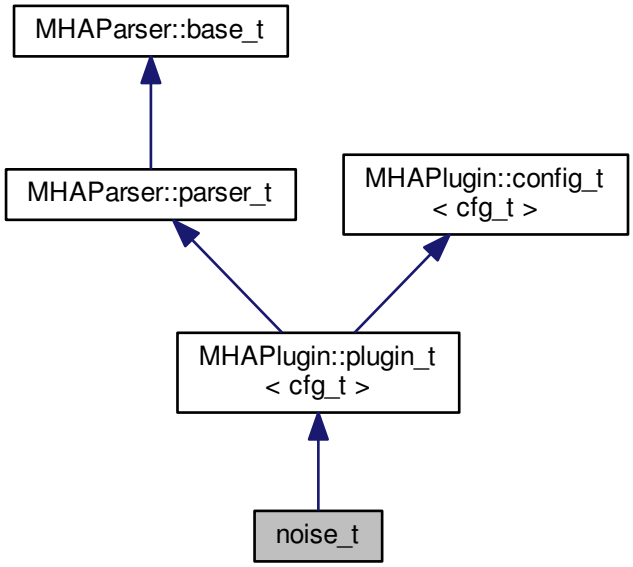
5.325.3.13 MHAEvents::patchbay_t<nlms_t> nlms_t::patchbay [private]

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

- **nlms_wave.cpp**

5.326 noise_t Class Reference

Inheritance diagram for noise_t:



Public Member Functions

- **noise_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t lev**
- **MHAParser::kw_t mode**
- **MHAParser::float_t frozennoise_length**
- **MHAEvents::patchbay_t**< **noise_t** > **patchbay**

Additional Inherited Members

5.326.1 Constructor & Destructor Documentation

5.326.1.1 noise_t::noise_t (
const **algo_comm_t** & *iac,*
const **std::string** & *,*
const **std::string** &)

5.326.2 Member Function Documentation

5.326.2.1 mha_wave_t ∗ noise_t::process (
mha_wave_t ∗ *s*)

5.326.2.2 mha_spec_t ∗ noise_t::process (
mha_spec_t ∗ *s*)

5.326.2.3 void noise_t::prepare (
mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 730).

5.326.2.4 void noise_t::update_cfg (void)

5.326.3 Member Data Documentation

5.326.3.1 MHAParser::float_t noise_t::lev [private]

5.326.3.2 MHAParser::kw_t noise_t::mode [private]

5.326.3.3 MHAParser::float_t noise_t::frozennoise_length [private]

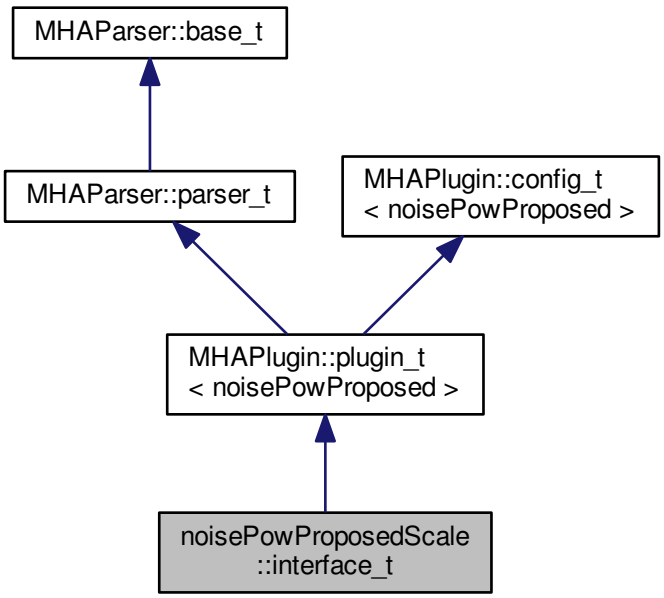
5.326.3.4 MHAEvents::patchbay_t<noise_t> noise_t::patchbay [private]

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

- **noise.cpp**

5.327 noisePowProposedScale::interface_t Class Reference

Inheritance diagram for noisePowProposedScale::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t alphaPH1mean**
- **MHAParser::float_t alphaPSD**
- **MHAParser::float_t q**
- **MHAParser::float_t xiOptDb**
- std::string **name**
- **MHAEvents::patchbay_t**< **interface_t** > **patchbay**

Additional Inherited Members

5.327.1 Constructor & Destructor Documentation

5.327.1.1 noisePowProposedScale::interface_t::interface_t (const algo_comm_t & *iac,* const std::string & *,* const std::string & *iname*)

5.327.2 Member Function Documentation

5.327.2.1 mha_spec_t ∗ noisePowProposedScale::interface_t::process (mha_spec_t ∗ *s*)

5.327.2.2 void noisePowProposedScale::interface_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t**< **noisePowProposed** > (p. 730).

5.327.2.3 void noisePowProposedScale::interface_t::update_cfg (void) [private]

5.327.3 Member Data Documentation

5.327.3.1 MHAParser::float_t noisePowProposedScale::interface_t::alphaPH1mean [private]

5.327.3.2 MHAParser::float_t noisePowProposedScale::interface_t::alphaPSD [private]

5.327.3.3 MHAParser::float_t noisePowProposedScale::interface_t::q [private]

5.327.3.4 MHAParser::float_t noisePowProposedScale::interface_t::xiOptDb [private]

5.327.3.5 std::string noisePowProposedScale::interface_t::name [private]

5.327.3.6 MHAEvents::patchbay_t<interface_t> noisePowProposedScale::interface_t::patchbay [private]

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

- **noisePowProposedScale.cpp**

5.328 noisePowProposedScale::noisePowProposed Class Reference

Public Member Functions

- **noisePowProposed** (const **mhaconfig_t** &cf, **algo_comm_t** ac, const std::string &name, float alphaPH1mean, float alphaPSD, float q, float xiOptDb)
- void **process** (**mha_spec_t** *noisyDftFrame)
- void **insert** ()

Private Attributes

- **MHASignal::waveform_t noisyPer**
- **MHASignal::waveform_t PH1mean**
- **MHA_AC::waveform_t noisePow**
- **MHA_AC::waveform_t inputPow**
- **MHA_AC::waveform_t snrPost1Debug**
- **MHA_AC::waveform_t GLRDebug**
- **MHA_AC::waveform_t PH1Debug**
- **MHA_AC::waveform_t estimateDebug**
- **MHA_AC::spectrum_t inputSpec**
- float **alphaPH1mean_**
- float **alphaPSD_**
- float **priorFact**
- float **xiOpt**
- float **logGLRFact**
- float **GLRexp**
- int **frameno**

5.328.1 Constructor & Destructor Documentation

5.328.1.1 noisePowProposedScale::noisePowProposed::noisePowProposed (
const **mhaconfig_t** & *cf,*
algo_comm_t *ac,*
const std::string & *name,*
float *alphaPH1mean,*
float *alphaPSD,*
float *q,*
float *xiOptDb*)

5.328.2 Member Function Documentation

5.328.2.1 void noisePowProposedScale::noisePowProposed::process (
mha_spec_t ∗ *noisyDftFrame*)

5.328.2.2 void noisePowProposedScale::noisePowProposed::insert ( ) [inline]

5.328.3 Member Data Documentation

5.328.3.1 MHASignal::waveform_t noisePowProposedScale::noisePowProposed::noisyPer [private]

5.328.3.2 MHASignal::waveform_t noisePowProposedScale::noisePowProposed::PH1mean [private]

5.328.3.3 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::noisePow [private]

5.328.3.4 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::inputPow [private]

5.328.3.5 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::snrPost1Debug [private]

5.328.3.6 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::GLRDebug [private]

5.328.3.7 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::PH1Debug [private]

5.328.3.8 MHA_AC::waveform_t noisePowProposedScale::noisePowProposed::estimateDebug [private]

5.328.3.9 MHA_AC::spectrum_t noisePowProposedScale::noisePowProposed::inputSpec [private]

5.328.3.10 float noisePowProposedScale::noisePowProposed::alphaPH1mean_ [private]

5.328.3.11 float noisePowProposedScale::noisePowProposed::alphaPSD_ [private]

5.328.3.12 float noisePowProposedScale::noisePowProposed::priorFact [private]

5.328.3.13 float noisePowProposedScale::noisePowProposed::xiOpt [private]

5.328.3.14 float noisePowProposedScale::noisePowProposed::logGLRFact [private]

5.328.3.15 float noisePowProposedScale::noisePowProposed::GLRexp [private]

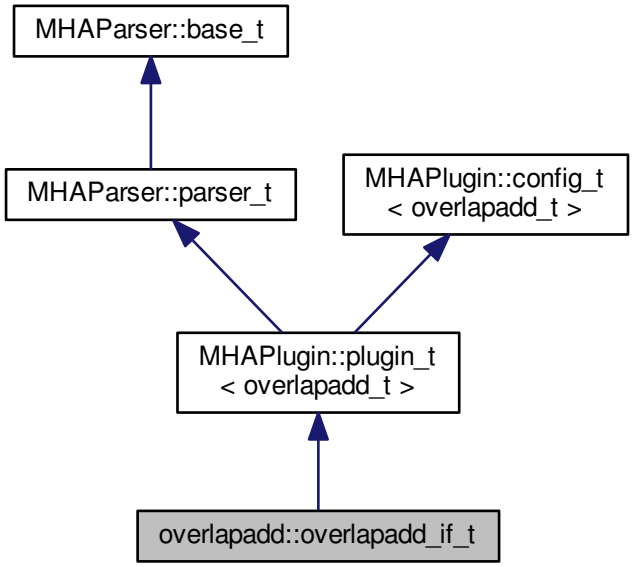
5.328.3.16 int noisePowProposedScale::noisePowProposed::frameno [private]

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

- **noisePowProposedScale.cpp**

5.329 overlapadd::overlapadd_if_t Class Reference

Inheritance diagram for overlapadd::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- ∼**overlapadd_if_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **overlapadd_if_t** > **patchbay**
- **MHAParser::int_t nfft**
  FFT length to be used, zero-padding is FFT length-wndlength.
- **MHAParser::int_t nwnd**
  Window length to be used (overlap is 1-fragsize/wndlength)
- **MHAParser::float_t wndpos**
  Relative position of zero padding (0 end, 0.5 center, 1 start)
- **MHAParser::window_t window**
- **MHAParser::float_t wndexp**
- **MHAParser::window_t zerowindow**
- **MHAParser::mhapluginloader_t plugloader**
- **MHAParser::float_mon_t prescale**
- **MHAParser::float_mon_t postscale**
- std::string **algo**
- **mhaconfig_t cf_in**
- **mhaconfig_t cf_out**

Additional Inherited Members

5.329.1 Constructor & Destructor Documentation

5.329.1.1 overlapadd::overlapadd_if_t::overlapadd_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string & *ialg*)

5.329.1.2 overlapadd::overlapadd_if_t::∼overlapadd_if_t ( )

5.329.2 Member Function Documentation

5.329.2.1 void overlapadd::overlapadd_if_t::prepare (mhaconfig_t & *t*) [virtual]

Implements **MHAPlugin::plugin_t**< **overlapadd_t** > (p. 730).

5.329.2.2 void overlapadd::overlapadd_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< overlapadd_t >** (p. 731).

5.329.2.3 mha_wave_t * overlapadd::overlapadd_if_t::process (mha_wave_t * *wave_in*)

5.329.2.4 void overlapadd::overlapadd_if_t::update ( ) [private]

5.329.3 Member Data Documentation

5.329.3.1 MHAEvents::patchbay_t<overlapadd_if_t> overlapadd::overlapadd_if_t::patchbay [private]

5.329.3.2 MHAParser::int_t overlapadd::overlapadd_if_t::nfft [private]

FFT length to be used, zero-padding is FFT length-wndlength.

5.329.3.3 MHAParser::int_t overlapadd::overlapadd_if_t::nwnd [private]

Window length to be used (overlap is 1-fragsize/wndlength)

5.329.3.4 MHAParser::float_t overlapadd::overlapadd_if_t::wndpos [private]

Relative position of zero padding (0 end, 0.5 center, 1 start)

5.329.3.5 MHAParser::window_t overlapadd::overlapadd_if_t::window [private]

5.329.3.6 MHAParser::float_t overlapadd::overlapadd_if_t::wndexp [private]

5.329.3.7 MHAParser::window_t overlapadd::overlapadd_if_t::zerowindow [private]

5.329.3.8 MHAParser::mhapluginloader_t overlapadd::overlapadd_if_t::plugloader [private]

5.329.3.9 MHAParser::float_mon_t overlapadd::overlapadd_if_t::prescale [private]

5.329.3.10 MHAParser::float_mon_t overlapadd::overlapadd_if_t::postscale [private]

5.329.3.11 std::string overlapadd::overlapadd_if_t::algo [private]

5.329.3.12 mhaconfig_t overlapadd::overlapadd_if_t::cf_in [private]

5.329.3.13 mhaconfig_t overlapadd::overlapadd_if_t::cf_out [private]

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

- **overlapadd.cpp**

5.330 overlapadd::overlapadd_t Class Reference

Public Member Functions

- **overlapadd_t** (**mhaconfig_t** spar_in, **mhaconfig_t** spar_out, float wexp, float wndpos, const **MHAParser::window_t** &window, const **MHAParser::window_t** &zerowindow, float &prescale_fac, float &postscale_fac)
- ∼**overlapadd_t** ()
- **mha_spec_t** ∗ **ola1** (**mha_wave_t** ∗)
- **mha_wave_t** ∗ **ola2** (**mha_spec_t** ∗)

Private Attributes

- **mha_fft_t fft**
- **MHAWindow::base_t prewnd**
- **MHAWindow::base_t postwnd**
- **MHASignal::waveform_t wave_in1**
- **MHASignal::waveform_t wave_out1**
- **MHASignal::spectrum_t spec_in**
- **MHASignal::waveform_t calc_out**
- **MHASignal::waveform_t out_buf**
- **MHASignal::waveform_t write_buf**
- unsigned int **n_zero**
- unsigned int **n_pad1**
- unsigned int **n_pad2**

5.330.1 Constructor & Destructor Documentation

5.330.1.1 overlapadd::overlapadd_t::overlapadd_t (

mhaconfig_t *spar_in,*
mhaconfig_t *spar_out,*
float *wexp,*
float *wndpos,*
const MHAParser::window_t & *window,*
const MHAParser::window_t & *zerowindow,*
float & *prescale_fac,*
float & *postscale_fac*)

5.330.1.2 overlapadd::overlapadd_t::∼overlapadd_t ( )

5.330.2 Member Function Documentation

5.330.2.1 mha_spec_t ∗ overlapadd::overlapadd_t::ola1 (

mha_wave_t ∗ ***s***)

5.330.2.2 mha_wave_t * overlapadd::overlapadd_t::ola2 (mha_spec_t * *s*)

5.330.3 Member Data Documentation

5.330.3.1 mha_fft_t overlapadd::overlapadd_t::fft [private]

5.330.3.2 MHAWindow::base_t overlapadd::overlapadd_t::prewnd [private]

5.330.3.3 MHAWindow::base_t overlapadd::overlapadd_t::postwnd [private]

5.330.3.4 MHASignal::waveform_t overlapadd::overlapadd_t::wave_in1 [private]

5.330.3.5 MHASignal::waveform_t overlapadd::overlapadd_t::wave_out1 [private]

5.330.3.6 MHASignal::spectrum_t overlapadd::overlapadd_t::spec_in [private]

5.330.3.7 MHASignal::waveform_t overlapadd::overlapadd_t::calc_out [private]

5.330.3.8 MHASignal::waveform_t overlapadd::overlapadd_t::out_buf [private]

5.330.3.9 MHASignal::waveform_t overlapadd::overlapadd_t::write_buf [private]

5.330.3.10 unsigned int overlapadd::overlapadd_t::n_zero [private]

5.330.3.11 unsigned int overlapadd::overlapadd_t::n_pad1 [private]

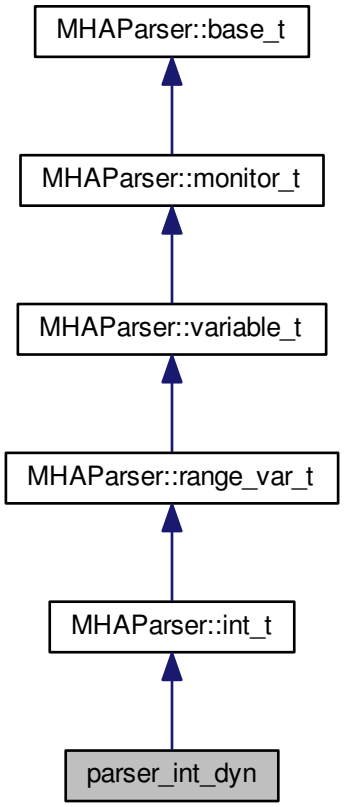
5.330.3.12 unsigned int overlapadd::overlapadd_t::n_pad2 [private]

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

- **overlapadd.cpp**

5.331 parser_int_dyn Class Reference

Inheritance diagram for parser_int_dyn:



Public Member Functions

- **parser_int_dyn** (const std::string &help_text, const std::string &initial_value, const std↩ ::string &**range**)
- void **set_max_angle_ind** (unsigned int max_ind)

Additional Inherited Members

5.331.1 Constructor & Destructor Documentation

5.331.1.1 parser_int_dyn::parser_int_dyn (const std::string & *help_text,* const std::string & *initial_value,* const std::string & *range*) [inline]

5.331.2 Member Function Documentation

5.331.2.1 void parser_int_dyn::set_max_angle_ind (unsigned int *max_ind*) [inline]

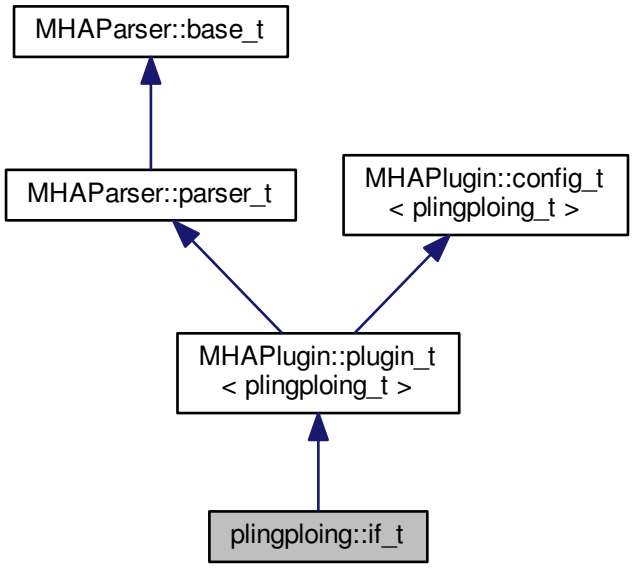
The documentation for this class was generated from the following file:

- **steerbf.h**

5.332 plingploing::if_t Class Reference

Plugin class of the plingploing music generator.

Inheritance diagram for plingploing::if_t:



Public Member Functions

- **if_t** (**algo_comm_t**, const char ∗, const char ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &cf)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t< if_t > patchbay**
- **MHAParser::float_t level**
  Output level in dB SPL.
- **MHAParser::float_t pitch**
  Bass pitch in Hz.
- **MHAParser::float_t fun1_key**
  Key1.
- **MHAParser::float_t fun1_range**
  Range1.
- **MHAParser::float_t fun2_key**
  Key 2.
- **MHAParser::float_t fun2_range**
  Range 2.
- **MHAParser::float_t bpm**
  Speed in beats per minute (bpm)
- **MHAParser::float_t minlen**
  Minimum note length in beats.
- **MHAParser::float_t maxlen**
  Maximum note length in beats.
- **MHAParser::float_t bassmod**
  Bass key modulation depth.
- **MHAParser::float_t bassperiod**
  Bass key modulation period.

Additional Inherited Members

5.332.1 Detailed Description

Plugin class of the plingploing music generator.

5.332.2 Constructor & Destructor Documentation

5.332.2.1 plingploing::if_t::if_t (

algo_comm_t *iac,*
const char ∗ **,**
const char ∗ **)**

5.332.3 Member Function Documentation

5.332.3.1 mha_wave_t ∗ plingploing::if_t::process (

mha_wave_t ∗ ***s*** **)**

5.332.3.2 void plingploing::if_t::prepare (

mhaconfig_t & ***cf*** **)** `[virtual]`

Implements **MHAPlugin::plugin_t< plingploing_t >** (p. 730).

5.332.3.3 void plingploing::if_t::update ( ) [private]

5.332.4 Member Data Documentation

5.332.4.1 MHAEvents::patchbay_t<if_t> plingploing::if_t::patchbay [private]

5.332.4.2 MHAParser::float_t plingploing::if_t::level [private]

Output level in dB SPL.

5.332.4.3 MHAParser::float_t plingploing::if_t::pitch [private]

Bass pitch in Hz.

5.332.4.4 MHAParser::float_t plingploing::if_t::fun1_key [private]

Key1.

5.332.4.5 MHAParser::float_t plingploing::if_t::fun1_range [private]

Range1.

5.332.4.6 MHAParser::float_t plingploing::if_t::fun2_key [private]

Key 2.

5.332.4.7 MHAParser::float_t plingploing::if_t::fun2_range [private]

Range 2.

5.332.4.8 MHAParser::float_t plingploing::if_t::bpm [private]

Speed in beats per minute (bpm)

5.332.4.9 MHAParser::float_t plingploing::if_t::minlen [private]

Minimum note length in beats.

5.332.4.10 MHAParser::float_t plingploing::if_t::maxlen [private]

Maximum note length in beats.

5.332.4.11 MHAParser::float_t plingploing::if_t::bassmod [private]

Bass key modulation depth.

5.332.4.12 MHAParser::float_t plingploing::if_t::bassperiod [private]

Bass key modulation period.

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

- **plingploing.cpp**

5.333 plingploing::plingploing_t Class Reference

Run-time configuration of the plingploing music generator.

Public Member Functions

- **plingploing_t** (**mhaconfig_t**, **mha_real_t level**, **mha_real_t** pitch, **mha_real_t** k1, **mha_real_t** k2, **mha_real_t** i1, **mha_real_t** i2, **mha_real_t** bpm, **mha_real_t** minlen, **mha_real_t** maxlen, **mha_real_t** bassmod, **mha_real_t** bassperiod)
- void **process** (**mha_wave_t** $*$)

Private Attributes

- **mhaconfig_t cf**
- **mha_real_t pitch_**
- unsigned int **bt**
- unsigned int **t**
- unsigned int **len**
- **mha_real_t dur_**
- **mha_real_t minlen_**
- **mha_real_t maxlen_**
- **mha_real_t bass**
- **mha_real_t freq**
- **mha_real_t fun1_key**
- **mha_real_t fun1_range**
- **mha_real_t fun1**
- **mha_real_t fun2**
- **mha_real_t fun2_key**
- **mha_real_t fun2_range**
- **mha_real_t dist**
- **mha_real_t dist1**
- **mha_real_t alph**
- **mha_real_t rms**
- **mha_real_t bassmod_**
- **mha_real_t bassperiod_**
- **MHAWindow::hanning_t hann1**
- **MHAWindow::hanning_t hann2**
- **mha_real_t level**

5.333.1 Detailed Description

Run-time configuration of the plingploing music generator.

5.333.2 Constructor & Destructor Documentation

5.333.2.1 plingploing::plingploing_t::plingploing_t (

mhaconfig_t *c,*
mha_real_t *level,*
mha_real_t *pitch,*
mha_real_t *k1,*
mha_real_t *k2,*
mha_real_t *i1,*
mha_real_t *i2,*
mha_real_t *bpm,*
mha_real_t *minlen,*
mha_real_t *maxlen,*
mha_real_t *bassmod,*
mha_real_t *bassperiod*)

5.333.3 Member Function Documentation

5.333.3.1 void plingploing::plingploing_t::process (

mha_wave_t $*$ *s*)

5.333.4 Member Data Documentation

5.333.4.1 mhaconfig_t plingploing::plingploing_t::cf [private]

5.333.4.2 mha_real_t plingploing::plingploing_t::pitch_ [private]

5.333.4.3 unsigned int plingploing::plingploing_t::bt [private]

5.333.4.4 unsigned int plingploing::plingploing_t::t [private]

5.333.4.5 unsigned int plingploing::plingploing_t::len [private]

5.333.4.6 mha_real_t plingploing::plingploing_t::dur_ [private]

5.333.4.7 mha_real_t plingploing::plingploing_t::minlen_ [private]

5.333.4.8 mha_real_t plingploing::plingploing_t::maxlen_ [private]

5.333.4.9 mha_real_t plingploing::plingploing_t::bass [private]

5.333.4.10 mha_real_t plingploing::plingploing_t::freq [private]

5.333.4.11 mha_real_t plingploing::plingploing_t::fun1_key [private]

5.333.4.12 mha_real_t plingploing::plingploing_t::fun1_range [private]

5.333.4.13 mha_real_t plingploing::plingploing_t::fun1 [private]

5.333.4.14 mha_real_t plingploing::plingploing_t::fun2 [private]

5.333.4.15 mha_real_t plingploing::plingploing_t::fun2_key [private]

5.333.4.16 mha_real_t plingploing::plingploing_t::fun2_range [private]

5.333.4.17 mha_real_t plingploing::plingploing_t::dist [private]

5.333.4.18 mha_real_t plingploing::plingploing_t::dist1 [private]

5.333.4.19 mha_real_t plingploing::plingploing_t::alph [private]

5.333.4.20 mha_real_t plingploing::plingploing_t::rms [private]

5.333.4.21 mha_real_t plingploing::plingploing_t::bassmod_ [private]

5.333.4.22 mha_real_t plingploing::plingploing_t::bassperiod_ [private]

5.333.4.23 MHAWindow::hanning_t plingploing::plingploing_t::hann1 [private]

5.333.4.24 MHAWindow::hanning_t plingploing::plingploing_t::hann2 [private]

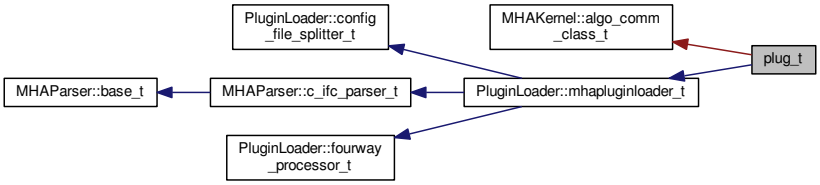
5.333.4.25 mha_real_t plingploing::plingploing_t::level [private]

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

- **plingploing.cpp**

5.334 plug_t Class Reference

Inheritance diagram for plug_t:



Public Member Functions

- **plug_t** (const std::string &**libname**, const std::string &chain, const std::string &algo)
- ∼**plug_t** () throw ()
- **MHAProc_wave2wave_t get_process_wave** ()
- **MHAProc_wave2spec_t get_process_spec** ()
- void ∗ **get_handle** ()
- **algo_comm_t get_ac** ()

Additional Inherited Members

5.334.1 Constructor & Destructor Documentation

5.334.1.1 plug_t::plug_t (const std::string & *libname,* const std::string & *chain,* const std::string & *algo*)

5.334.1.2 plug_t::∼plug_t ( ) throw) [inline]

5.334.2 Member Function Documentation

5.334.2.1 MHAProc_wave2wave_t plug_t::get_process_wave ( )

5.334.2.2 MHAProc_wave2spec_t plug_t::get_process_spec ( )

5.334.2.3 void ∗ plug_t::get_handle ( )

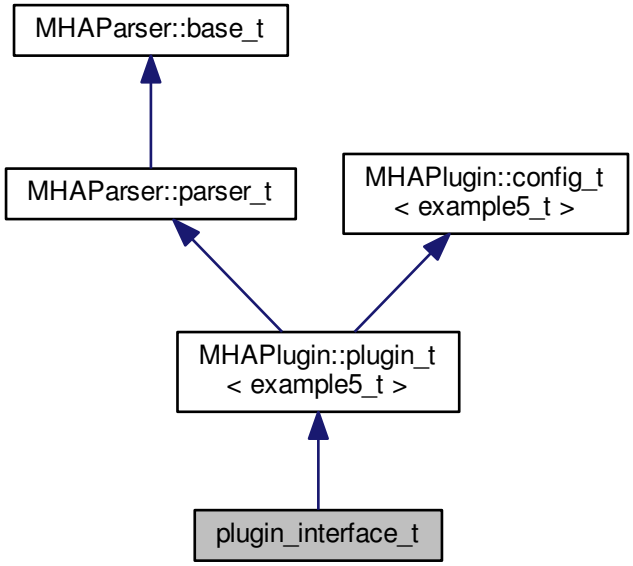
5.334.2.4 algo_comm_t plug_t::get_ac ( ) [inline]

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

- **analysispath.cpp**

5.335 plugin_interface_t Class Reference

Inheritance diagram for plugin_interface_t:



Public Member Functions

- **plugin_interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::int_t scale_ch**
- **MHAParser::float_t factor**
- **MHAEvents::patchbay_t**< **plugin_interface_t** > **patchbay**

Additional Inherited Members

5.335.1 Constructor & Destructor Documentation

5.335.1.1 plugin_interface_t::plugin_interface_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.335.2 Member Function Documentation

5.335.2.1 mha_spec_t * plugin_interface_t::process (mha_spec_t * *spec*)

5.335.2.2 void plugin_interface_t::prepare (mhaconfig_t & *tfcfg*) [virtual]

Implements **MHAPlugin::plugin_t< example5_t >** (p. 730).

5.335.2.3 void plugin_interface_t::update_cfg ( ) [private]

5.335.3 Member Data Documentation

5.335.3.1 MHAParser::int_t plugin_interface_t::scale_ch [private]

5.335.3.2 MHAParser::float_t plugin_interface_t::factor [private]

5.335.3.3 MHAEvents::patchbay_t<plugin_interface_t> plugin_interface_t::patchbay [private]

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

- **example5.cpp**

5.336 pluginbrowser_t Class Reference

Public Member Functions

- **pluginbrowser_t** ()
- void **get_paths** ()
- **plugindescription_t scan_plugin** (const std::string &name)
- void **add_plugins** ()
- void **clear_plugins** ()
- void **scan_plugins** ()
- void **add_plugin** (const std::string &name)
- std::list< **plugindescription_t** > **get_plugins** () const

Private Attributes

- std::string **plugin_extension**
- std::list< std::string > **library_paths**
- std::list< **plugindescription_t** > **plugins**
- std::map< std::string, **pluginloader_t** * > **p**

5.336.1 Constructor & Destructor Documentation

5.336.1.1 pluginbrowser_t::pluginbrowser_t ( )

5.336.2 Member Function Documentation

5.336.2.1 void pluginbrowser_t::get_paths ( )

5.336.2.2 plugindescription_t pluginbrowser_t::scan_plugin (const std::string & *name*)

5.336.2.3 void pluginbrowser_t::add_plugins ( )

5.336.2.4 void pluginbrowser_t::clear_plugins ( )

5.336.2.5 void pluginbrowser_t::scan_plugins ( )

5.336.2.6 void pluginbrowser_t::add_plugin (const std::string & *name*)

5.336.2.7 std::list<plugindescription_t> pluginbrowser_t::get_plugins ( ) const [inline]

5.336.3 Member Data Documentation

5.336.3.1 std::string pluginbrowser_t::plugin_extension [private]

5.336.3.2 std::list<std::string> pluginbrowser_t::library_paths [private]

5.336.3.3 std::list<plugindescription_t> pluginbrowser_t::plugins [private]

5.336.3.4 std::map<std::string,pluginloader_t*> pluginbrowser_t::p [private]

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

- **pluginbrowser.h**
- **pluginbrowser.cpp**

5.337 plugindescription_t Class Reference

Public Attributes

- std::string **name**
- std::string **fullname**
- std::string **documentation**
- std::vector< std::string > **categories**
- bool **wave2wave**
- bool **wave2spec**
- bool **spec2wave**
- bool **spec2spec**
- std::vector< std::string > **query_cmds**
- std::map< std::string, std::string > **queries**

5.337.1 Member Data Documentation

5.337.1.1 std::string plugindescription_t::name

5.337.1.2 std::string plugindescription_t::fullname

5.337.1.3 std::string plugindescription_t::documentation

5.337.1.4 std::vector<std::string> plugindescription_t::categories

5.337.1.5 bool plugindescription_t::wave2wave

5.337.1.6 bool plugindescription_t::wave2spec

5.337.1.7 bool plugindescription_t::spec2wave

5.337.1.8 bool plugindescription_t::spec2spec

5.337.1.9 std::vector<std::string> plugindescription_t::query_cmds

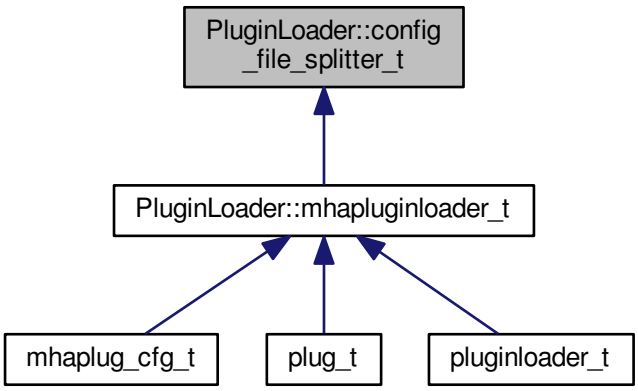
5.337.1.10 std::map<std::string,std::string> plugindescription_t::queries

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

- **pluginbrowser.h**

5.338 PluginLoader::config_file_splitter_t Class Reference

Inheritance diagram for PluginLoader::config_file_splitter_t:



Public Member Functions

- **config_file_splitter_t** (const std::string &name)
- const std::string & **get_configname** () const
- const std::string & **get_libname** () const
- const std::string & **get_origname** () const
- const std::string & **get_configfile** () const

Private Attributes

- std::string **libname**
- std::string **configname**
- std::string **origname**
- std::string **configfile**

5.338.1 Constructor & Destructor Documentation

5.338.1.1 PluginLoader::config_file_splitter_t::config_file_splitter_t (const std::string & *name*)

5.338.2 Member Function Documentation

5.338.2.1 const std::string& PluginLoader::config_file_splitter_t::get_configname ( ) const [inline]

5.338.2.2 const std::string& PluginLoader::config_file_splitter_t::get_libname ( ) const [inline]

5.338.2.3 const std::string& PluginLoader::config_file_splitter_t::get_origname ( ) const [inline]

5.338.2.4 const std::string& PluginLoader::config_file_splitter_t::get_configfile ( ) const [inline]

5.338.3 Member Data Documentation

5.338.3.1 std::string PluginLoader::config_file_splitter_t::libname [private]

5.338.3.2 std::string PluginLoader::config_file_splitter_t::configname [private]

5.338.3.3 std::string PluginLoader::config_file_splitter_t::origname [private]

5.338.3.4 std::string PluginLoader::config_file_splitter_t::configfile [private]

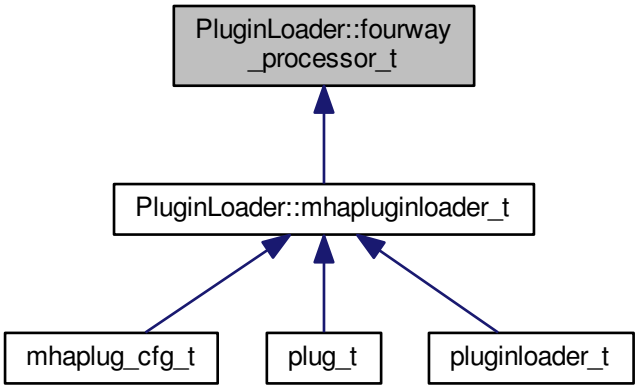
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.339 PluginLoader::fourway_processor_t Class Reference

This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.

Inheritance diagram for PluginLoader::fourway_processor_t:



Public Member Functions

- virtual void **process** (**mha_wave_t** ∗s_in, **mha_wave_t** ∗∗s_out)=0
  Pure waveform processing.
- virtual void **process** (**mha_spec_t** ∗s_in, **mha_spec_t** ∗∗s_out)=0
  Pure spectrum processing.
- virtual void **process** (**mha_wave_t** ∗s_in, **mha_spec_t** ∗∗s_out)=0
  Signal processing with domain transformation from waveform to spectrum.
- virtual void **process** (**mha_spec_t** ∗s_in, **mha_wave_t** ∗∗s_out)=0
  Signal processing with domain transformation from spectrum to waveform.
- virtual void **prepare** (**mhaconfig_t** &settings)=0
  Prepares the processor for signal processing.
- virtual void **release** ()=0
  *Resources allocated for signal processing in **fourway_processor_t::prepare*** *(p. 880)* *are released here in **fourway_processor_t::release*** *(p. 880)**.*
- virtual std::string **parse** (const std::string &query)=0
  Parser interface.
- virtual ∼**fourway_processor_t** ()
  Classes with virtual methods need virtual destructor.

5.339.1 Detailed Description

This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.

For supporting different output domains for the same input domain, the processing methods are overloaded with respect to input domain and output domain.

5.339.2 Constructor & Destructor Documentation

5.339.2.1 virtual PluginLoader::fourway_processor_t::∼fourway_processor_t ( ) [inline], [virtual]

Classes with virtual methods need virtual destructor.

This destructor is empty.

5.339.3 Member Function Documentation

5.339.3.1 virtual void PluginLoader::fourway_processor_t::process (mha_wave_t ∗ *s_in*, mha_wave_t ∗∗ *s_out*) [pure virtual]

Pure waveform processing.

Parameters

s_in	input waveform signal
s_out	output waveform signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.2 virtual void PluginLoader::fourway_processor_t::process (mha_spec_t * s_in, mha_spec_t ** s_out) [pure virtual]

Pure spectrum processing.

Parameters

s_in	input spectrum signal
s_out	output spectrum signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.3 virtual void PluginLoader::fourway_processor_t::process (mha_wave_t * s_in, mha_spec_t ** s_out) [pure virtual]

Signal processing with domain transformation from waveform to spectrum.

Parameters

s_in	input waveform signal
s_out	output spectrum signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.4 virtual void PluginLoader::fourway_processor_t::process (mha_spec_t * s_in, mha_wave_t ** s_out) [pure virtual]

Signal processing with domain transformation from spectrum to waveform.

Parameters

s_in	input spectrum signal
s_out	output waveform signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.5 virtual void PluginLoader::fourway_processor_t::prepare (mhaconfig_t & *settings*) [pure virtual]

Prepares the processor for signal processing.

Parameters

settings	domain and dimensions of the signal. The contents of settings may be modified by the prepare implementation. Upon calling **fourway_processor_t::prepare** (p. 880), settings reflects domain and dimensions of the input signal. When **fourway_processor_t::prepare** (p. 880) returns, settings reflects domain and dimensions of the output signal.

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.6 virtual void PluginLoader::fourway_processor_t::release ( ) [pure virtual]

Resources allocated for signal processing in **fourway_processor_t::prepare** (p. 880) are released here in **fourway_processor_t::release** (p. 880).

Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

5.339.3.7 virtual std::string PluginLoader::fourway_processor_t::parse (const std::string & *query*) [pure virtual]

Parser interface.

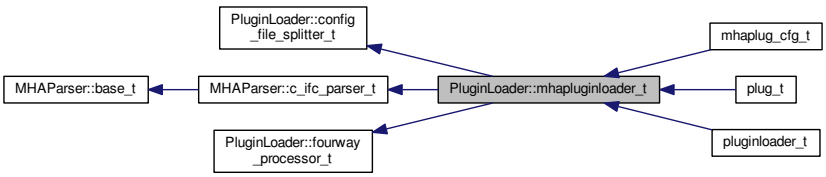
Implemented in **PluginLoader::mhapluginloader_t** (p. 883).

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

- **mhapluginloader.h**

5.340 PluginLoader::mhapluginloader_t Class Reference

Inheritance diagram for PluginLoader::mhapluginloader_t:



Public Member Functions

- std::string **parse** (const std::string &str)
- **mhapluginloader_t** (**algo_comm_t** iac, const std::string &**libname**, bool check_↩ version=true)

  Loads and initializes mha plugin and establishes interface.
- ∼**mhapluginloader_t** () throw ()
- bool **has_process** (**mha_domain_t** in, **mha_domain_t** out) const
- bool **has_parser** () const
- **mha_domain_t input_domain** () const
- **mha_domain_t output_domain** () const
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** (**mha_wave_t** ∗, **mha_wave_t** ∗∗)
- void **process** (**mha_spec_t** ∗, **mha_spec_t** ∗∗)
- void **process** (**mha_wave_t** ∗, **mha_spec_t** ∗∗)
- void **process** (**mha_spec_t** ∗, **mha_wave_t** ∗∗)
- std::string **getfullname** () const
- std::string **get_documentation** () const
- std::vector< std::string > **get_categories** () const
- bool **is_prepared** () const

Protected Member Functions

- void **test_error** ()
- void **test_version** ()
- void **mha_test_struct_size** (unsigned int s)
- void **resolve_and_init** ()

Protected Attributes

- int **lib_err**
- **algo_comm_t ac**
- **dynamiclib_t lib_handle**
- void ∗ **lib_data**
- **MHAGetVersion_t MHAGetVersion_cb**
- **MHAInit_t MHAInit_cb**
- **MHADestroy_t MHADestroy_cb**
- **MHAPrepare_t MHAPrepare_cb**
- **MHARelease_t MHARelease_cb**
- **MHAProc_wave2wave_t MHAProc_wave2wave_cb**
- **MHAProc_spec2spec_t MHAProc_spec2spec_cb**
- **MHAProc_wave2spec_t MHAProc_wave2spec_cb**
- **MHAProc_spec2wave_t MHAProc_spec2wave_cb**
- **MHASet_t MHASet_cb**
- **MHAStrError_t MHAStrError_cb**

- **mhaconfig_t cf_input**
- **mhaconfig_t cf_output**
- std::string **plugin_documentation**
- std::vector< std::string > **plugin_categories**
- bool **b_check_version**
- bool **b_is_prepared**

Additional Inherited Members

5.340.1 Constructor & Destructor Documentation

5.340.1.1 PluginLoader::mhapluginloader_t::mhapluginloader_t (algo_comm_t *iac,* const std::string & *libname,* bool *check_version =* `true`)

Loads and initializes mha plugin and establishes interface.

Parameters

iac	AC space (algorithm communication variables)
libname	Either file name of MHA plugin without platform-specific extension (i.e. "identity" for "identity.so" or "identity.dll") to be found on the MHA_LIBRARY_PATH (which is an environment variable). Or the same file name without extension followed by a colon ":" followed by the "configuration name" of the MHA plugin, which may be used to differentiate between multiple identical MHA plugins or to give the plugin a self-documenting name that fits its purpose. The library name - configuration name expression can be followed by a "<" followed by a configuration file name, which will be read after initialization of the plugin.

Example: "overlapadd:agc<compression.cfg" will load the plugin "overlapadd.so" or "overlapadd.dll", insert it as the configuration node "agc", and reads the configuration file "compression.cfg" into that node.

Parameters

check_version	Pluginloader will not check that the plugin was built using a known compatible MHA version if this flag is set to false. Disabling version check is discouraged.

5.340.1.2 PluginLoader::mhapluginloader_t::~mhapluginloader_t ( ) throw)

5.340.2 Member Function Documentation

5.340.2.1 std::string PluginLoader::mhapluginloader_t::parse (const std::string & *str*) [inline], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 880).

5.340.2.2 bool PluginLoader::mhapluginloader_t::has_process (mha_domain_t *in*, mha_domain_t *out*) const

5.340.2.3 bool PluginLoader::mhapluginloader_t::has_parser ( ) const

5.340.2.4 mha_domain_t PluginLoader::mhapluginloader_t::input_domain ( ) const

5.340.2.5 mha_domain_t PluginLoader::mhapluginloader_t::output_domain ( ) const

5.340.2.6 void PluginLoader::mhapluginloader_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 880).

5.340.2.7 void PluginLoader::mhapluginloader_t::release ( ) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 880).

5.340.2.8 void PluginLoader::mhapluginloader_t::process (mha_wave_t * *s_in*, mha_wave_t ** *s_out*) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 878).

5.340.2.9 void PluginLoader::mhapluginloader_t::process (mha_spec_t * *s_in*, mha_spec_t ** *s_out*) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 879).

5.340.2.10 void PluginLoader::mhapluginloader_t::process (mha_wave_t * *s_in*, mha_spec_t ** *s_out*) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 879).

5.340.2.11 void PluginLoader::mhapluginloader_t::process (mha_spec_t * *s_in*, mha_wave_t ** *s_out*) [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 879).

5.340.2.12 std::string PluginLoader::mhapluginloader_t::getfullname ( ) const [inline]

5.340.2.13 std::string PluginLoader::mhapluginloader_t::get_documentation ( ) const [inline]

5.340.2.14 std::vector<std::string> PluginLoader::mhapluginloader_t::get_categories ( ) const [inline]

5.340.2.15 bool PluginLoader::mhapluginloader_t::is_prepared ( ) const [inline]

5.340.2.16 void PluginLoader::mhapluginloader_t::test_error ( ) [protected]

5.340.2.17 void PluginLoader::mhapluginloader_t::test_version ( ) [protected]

5.340.2.18 void PluginLoader::mhapluginloader_t::mha_test_struct_size (unsigned int *s*) [protected]

5.340.2.19 void PluginLoader::mhapluginloader_t::resolve_and_init ( ) [protected]

5.340.3 Member Data Documentation

5.340.3.1 int PluginLoader::mhapluginloader_t::lib_err [protected]

5.340.3.2 algo_comm_t PluginLoader::mhapluginloader_t::ac [protected]

5.340.3.3 dynamiclib_t PluginLoader::mhapluginloader_t::lib_handle [protected]

5.340.3.4 void* PluginLoader::mhapluginloader_t::lib_data [protected]

5.340.3.5 MHAGetVersion_t PluginLoader::mhapluginloader_t::MHAGetVersion_cb [protected]

5.340.3.6 MHAInit_t PluginLoader::mhapluginloader_t::MHAInit_cb [protected]

5.340.3.7 MHADestroy_t PluginLoader::mhapluginloader_t::MHADestroy_cb [protected]

5.340.3.8 MHAPrepare_t PluginLoader::mhapluginloader_t::MHAPrepare_cb [protected]

5.340.3.9 MHARelease_t PluginLoader::mhapluginloader_t::MHARelease_cb [protected]

5.340.3.10 MHAProc_wave2wave_t PluginLoader::mhapluginloader_t::MHAProc_wave2wave_cb [protected]

5.340.3.11 MHAProc_spec2spec_t PluginLoader::mhapluginloader_t::MHAProc_spec2spec_cb [protected]

5.340.3.12 MHAProc_wave2spec_t PluginLoader::mhapluginloader_t::MHAProc_wave2spec_cb [protected]

5.340.3.13 MHAProc_spec2wave_t PluginLoader::mhapluginloader_t::MHAProc_spec2wave_cb [protected]

5.340.3.14 MHASet_t PluginLoader::mhapluginloader_t::MHASet_cb [protected]

5.340.3.15 MHAStrError_t PluginLoader::mhapluginloader_t::MHAStrError_cb [protected]

5.340.3.16 mhaconfig_t PluginLoader::mhapluginloader_t::cf_input [protected]

5.340.3.17 mhaconfig_t PluginLoader::mhapluginloader_t::cf_output [protected]

5.340.3.18 std::string PluginLoader::mhapluginloader_t::plugin_documentation [protected]

5.340.3.19 std::vector<std::string> PluginLoader::mhapluginloader_t::plugin_categories [protected]

5.340.3.20 bool PluginLoader::mhapluginloader_t::b_check_version [protected]

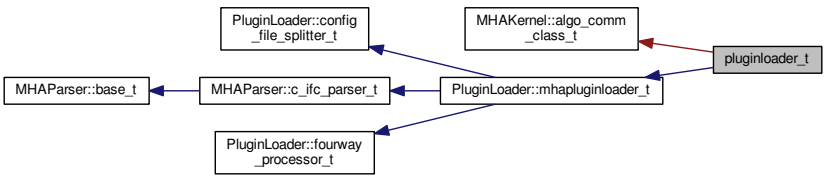
5.340.3.21 bool PluginLoader::mhapluginloader_t::b_is_prepared [protected]

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

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.341 pluginloader_t Class Reference

Inheritance diagram for pluginloader_t:



Public Member Functions

- **pluginloader_t** (const std::string &name)
- ~**pluginloader_t** () throw ()

Additional Inherited Members

5.341.1 Constructor & Destructor Documentation

5.341.1.1 pluginloader_t::pluginloader_t (const std::string & *name*)

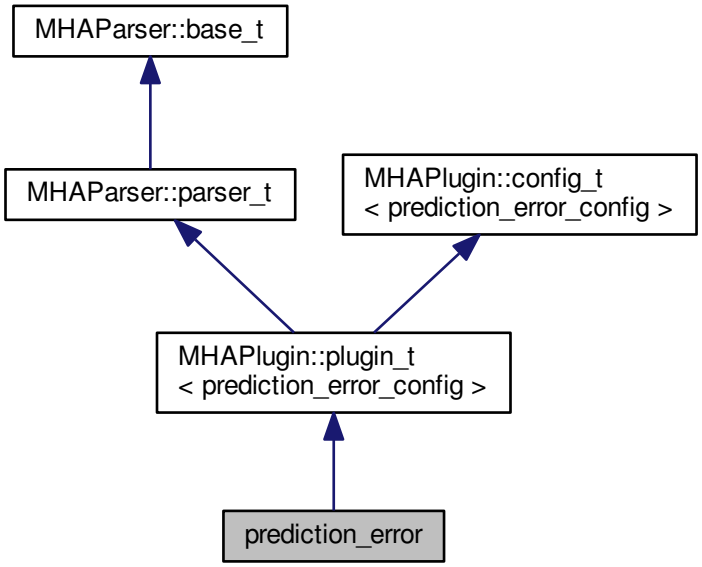
5.341.1.2 pluginloader_t::~pluginloader_t ( ) throw)

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

- **pluginbrowser.h**
- **pluginbrowser.cpp**

5.342 prediction_error Class Reference

Inheritance diagram for prediction_error:



Public Member Functions

- **prediction_error** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
  Constructs our plugin.
- ∼**prediction_error** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::float_t rho**
- **MHAParser::float_t c**
- **MHAParser::int_t ntaps**
- **MHAParser::vfloat_t gains**
- **MHAParser::string_t name_e**
- **MHAParser::string_t name_f**
- **MHAParser::string_t name_lpc**
- **MHAParser::int_t lpc_order**
- **MHAParser::vint_t pred_err_delay**
- **MHAParser::vint_t delay_w**
- **MHAParser::vint_t delay_d**
- **MHAParser::int_t n_no_update**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **prediction_error** > **patchbay**

Additional Inherited Members

5.342.1 Constructor & Destructor Documentation

5.342.1.1 prediction_error::prediction_error (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.342.1.2 prediction_error::~prediction_error ( )

5.342.2 Member Function Documentation

5.342.2.1 mha_wave_t * prediction_error::process (mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.342.2.2 void prediction_error::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< prediction_error_config >** (p. 730).

5.342.2.3 void prediction_error::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< prediction_error_config >** (p. 731).

5.342.2.4 void prediction_error::update_cfg (void) [private]

5.342.3 Member Data Documentation

5.342.3.1 MHAParser::float_t prediction_error::rho

5.342.3.2 MHAParser::float_t prediction_error::c

5.342.3.3 MHAParser::int_t prediction_error::ntaps

5.342.3.4 MHAParser::vfloat_t prediction_error::gains

5.342.3.5 MHAParser::string_t prediction_error::name_e

5.342.3.6 MHAParser::string_t prediction_error::name_f

5.342.3.7 MHAParser::string_t prediction_error::name_lpc

5.342.3.8 MHAParser::int_t prediction_error::lpc_order

5.342.3.9 MHAParser::vint_t prediction_error::pred_err_delay

5.342.3.10 MHAParser::vint_t prediction_error::delay_w

5.342.3.11 MHAParser::vint_t prediction_error::delay_d

5.342.3.12 MHAParser::int_t prediction_error::n_no_update

5.342.3.13 MHAEvents::patchbay_t<prediction_error> prediction_error::patchbay [private]

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

- **prediction_error.h**
- **prediction_error.cpp**

5.343 prediction_error_config Class Reference

Public Member Functions

- **prediction_error_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **prediction_error** *pred_err)
- ~**prediction_error_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *s_Y, **mha_real_t** rho, **mha_real_t** c)
- void **insert** ()

Private Attributes

- **algo_comm_t ac**
- unsigned int **ntaps**
- unsigned int **frames**
- unsigned int **channels**
- **MHA_AC::waveform_t s_E**
- **MHA_AC::waveform_t F**
- **MHASignal::waveform_t Pu**

  Power of input signal delayline.
- std::string **name_d_**
- std::string **name_lpc_**
- int **n_no_update_**
- int **no_iter**

- int **iter**
- **MHASignal::waveform_t v_G**
- **MHASignal::waveform_t s_U**
- **MHASignal::delay_t s_E_pred_err_delay**
- **MHASignal::delay_t s_W**
- **MHASignal::ringbuffer_t s_Wflt**
- **MHASignal::delay_t s_U_delay**
- **MHASignal::ringbuffer_t s_U_delayflt**
- **MHASignal::waveform_t F_Uflt**
- **MHASignal::delay_t s_Y_delay**
- **MHASignal::ringbuffer_t s_Y_delayflt**
- **MHASignal::ringbuffer_t UbufferPrew**
- **mha_wave_t s_LPC**
- **mha_wave_t UPrew**
- **mha_wave_t YPrew**
- **mha_wave_t EPrew**
- **mha_wave_t UPrewW**
- **mha_wave_t smpl**
- **mha_wave_t * s_Usmpl**

5.343.1 Constructor & Destructor Documentation

5.343.1.1 prediction_error_config::prediction_error_config (algo_comm_t & *ac,* const mhaconfig_t *in_cfg,* prediction_error * *pred_err*)

5.343.1.2 prediction_error_config::~prediction_error_config ( )

5.343.2 Member Function Documentation

5.343.2.1 mha_wave_t * prediction_error_config::process (mha_wave_t * *s_Y,* mha_real_t *rho,* mha_real_t *c*)

5.343.2.2 void prediction_error_config::insert ( )

5.343.3 Member Data Documentation

5.343.3.1 algo_comm_t prediction_error_config::ac [private]

5.343.3.2 unsigned int prediction_error_config::ntaps [private]

5.343.3.3 unsigned int prediction_error_config::frames [private]

5.343.3.4 unsigned int prediction_error_config::channels [private]

5.343.3.5 MHA_AC::waveform_t prediction_error_config::s_E [private]

5.343.3.6 MHA_AC::waveform_t prediction_error_config::F [private]

5.343.3.7 MHASignal::waveform_t prediction_error_config::Pu [private]

Power of input signal delayline.

5.343.3.8 std::string prediction_error_config::name_d_ [private]

5.343.3.9 std::string prediction_error_config::name_lpc_ [private]

5.343.3.10 int prediction_error_config::n_no_update_ [private]

5.343.3.11 int prediction_error_config::no_iter [private]

5.343.3.12 int prediction_error_config::iter [private]

5.343.3.13 MHASignal::waveform_t prediction_error_config::v_G [private]

5.343.3.14 MHASignal::waveform_t prediction_error_config::s_U [private]

5.343.3.15 MHASignal::delay_t prediction_error_config::s_E_pred_err_delay [private]

5.343.3.16 MHASignal::delay_t prediction_error_config::s_W [private]

5.343.3.17 MHASignal::ringbuffer_t prediction_error_config::s_Wflt [private]

5.343.3.18 MHASignal::delay_t prediction_error_config::s_U_delay [private]

5.343.3.19 MHASignal::ringbuffer_t prediction_error_config::s_U_delayflt [private]

5.343.3.20 MHASignal::waveform_t prediction_error_config::F_Uflt [private]

5.343.3.21 MHASignal::delay_t prediction_error_config::s_Y_delay [private]

5.343.3.22 MHASignal::ringbuffer_t prediction_error_config::s_Y_delayflt [private]

5.343.3.23 MHASignal::ringbuffer_t prediction_error_config::UbufferPrew [private]

5.343.3.24 mha_wave_t prediction_error_config::s_LPC [private]

5.343.3.25 mha_wave_t prediction_error_config::UPrew [private]

5.343.3.26 mha_wave_t prediction_error_config::YPrew [private]

5.343.3.27 mha_wave_t prediction_error_config::EPrew [private]

5.343.3.28 mha_wave_t prediction_error_config::UPrewW [private]

5.343.3.29 mha_wave_t prediction_error_config::smpl [private]

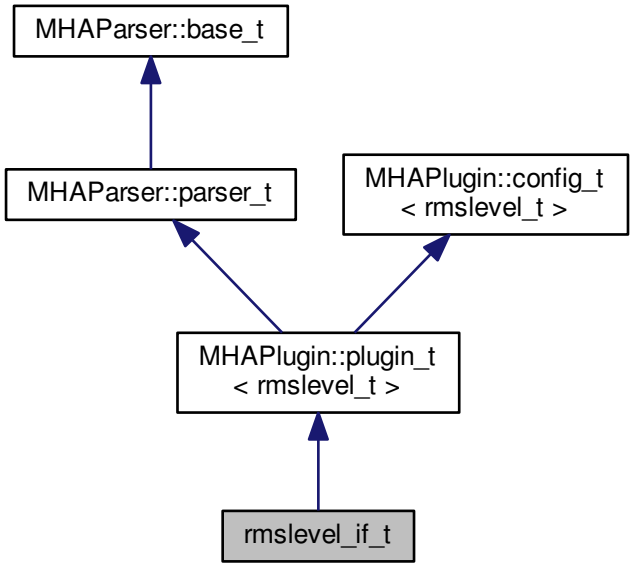
5.343.3.30 mha_wave_t* prediction_error_config::s_Usmpl [private]

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

- **prediction_error.h**
- **prediction_error.cpp**

5.344 rmslevel_if_t Class Reference

Inheritance diagram for rmslevel_if_t:



Public Member Functions

- **rmslevel_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- std::string **name**

Additional Inherited Members

5.344.1 Constructor & Destructor Documentation

5.344.1.1 rmslevel_if_t::rmslevel_if_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.344.2 Member Function Documentation

5.344.2.1 mha_spec_t ∗ rmslevel_if_t::process (mha_spec_t ∗ *s*)

5.344.2.2 mha_wave_t ∗ rmslevel_if_t::process (mha_wave_t ∗ *s*)

5.344.2.3 void rmslevel_if_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **rmslevel_t** > (p. 730).

5.344.3 Member Data Documentation

5.344.3.1 std::string rmslevel_if_t::name [private]

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

- **rmslevel.cpp**

5.345 rmslevel_t Class Reference

Public Member Functions

- **rmslevel_t** (unsigned int nch, **algo_comm_t** ac, std::string name, **MHAParser::parser_t** &p, unsigned int fftlen_)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **insert** ()

Private Attributes

- **mon_t level_db**
- **mon_t peak_db**
- **mon_t level**
- **mon_t peak**
- unsigned int **fftlen**

5.345.1 Constructor & Destructor Documentation

5.345.1.1 rmslevel_t::rmslevel_t (unsigned int *nch,* algo_comm_t *ac,* std::string *name,* MHAParser::parser_t & *p,* unsigned int *fftlen_*)

5.345.2 Member Function Documentation

5.345.2.1 mha_spec_t * rmslevel_t::process (mha_spec_t * *s*)

5.345.2.2 mha_wave_t * rmslevel_t::process (mha_wave_t * *s*)

5.345.2.3 void rmslevel_t::insert ( )

5.345.3 Member Data Documentation

5.345.3.1 mon_t rmslevel_t::level_db [private]

5.345.3.2 mon_t rmslevel_t::peak_db [private]

5.345.3.3 mon_t rmslevel_t::level [private]

5.345.3.4 mon_t rmslevel_t::peak [private]

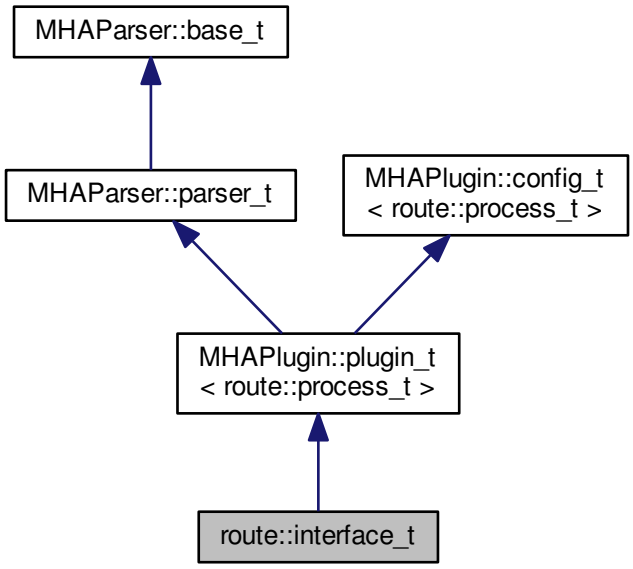
5.345.3.5 unsigned int rmslevel_t::fftlen [private]

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

- **rmslevel.cpp**

5.346 route::interface_t Class Reference

Inheritance diagram for route::interface_t:



Public Member Functions

- **interface_t** (**algo_comm_t** iac, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **route::interface_t** > **patchbay**
- **MHAParser::vstring_t route_out**
- **MHAParser::vstring_t route_ac**
- **mhaconfig_t cfin**
- **mhaconfig_t cfout**
- **mhaconfig_t cfac**
- bool **prepared**
- bool **stopped**
- std::string **algo**

Additional Inherited Members

5.346.1 Constructor & Destructor Documentation

5.346.1.1 route::interface_t::interface_t (algo_comm_t *iac,* const std::string & , const std::string & *ialg*)

5.346.2 Member Function Documentation

5.346.2.1 void route::interface_t::prepare (mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t< route::process_t >** (p. 730).

5.346.2.2 void route::interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< route::process_t >** (p. 731).

5.346.2.3 mha_wave_t * route::interface_t::process (mha_wave_t * *s*)

5.346.2.4 mha_spec_t * route::interface_t::process (mha_spec_t * *s*)

5.346.2.5 void route::interface_t::update ( ) [private]

5.346.3 Member Data Documentation

5.346.3.1 MHAEvents::patchbay_t<route::interface_t> route::interface_t::patchbay [private]

5.346.3.2 MHAParser::vstring_t route::interface_t::route_out [private]

5.346.3.3 MHAParser::vstring_t route::interface_t::route_ac [private]

5.346.3.4 mhaconfig_t route::interface_t::cfin [private]

5.346.3.5 mhaconfig_t route::interface_t::cfout [private]

5.346.3.6 mhaconfig_t route::interface_t::cfac [private]

5.346.3.7 bool route::interface_t::prepared [private]

5.346.3.8 bool route::interface_t::stopped [private]

5.346.3.9 std::string route::interface_t::algo [private]

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

- **route.cpp**

5.347 route::process_t Class Reference

Public Member Functions

- **process_t** (**algo_comm_t** iac, const std::string acname, const std::vector< std::string > &r_out, const std::vector< std::string > &r_ac, const **mhaconfig_t** &cf_in, const **mhaconfig_t** &cf_out, const **mhaconfig_t** &cf_ac, bool sync)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHAMultiSrc::waveform_t wout**
- **MHAMultiSrc::spectrum_t sout**
- **MHAMultiSrc::waveform_t wout_ac**
- **MHAMultiSrc::spectrum_t sout_ac**

5.347.1 Constructor & Destructor Documentation

5.347.1.1 route::process_t::process_t (

algo_comm_t *iac,*
const std::string *acname,*
const std::vector< std::string > & *r_out,*
const std::vector< std::string > & *r_ac,*
const mhaconfig_t & *cf_in,*
const mhaconfig_t & *cf_out,*
const mhaconfig_t & *cf_ac,*
bool *sync*)

5.347.2 Member Function Documentation

5.347.2.1 mha_wave_t * route::process_t::process (

mha_wave_t * *s*)

5.347.2.2 mha_spec_t * route::process_t::process (

mha_spec_t * *s*)

5.347.3 Member Data Documentation

5.347.3.1 MHAMultiSrc::waveform_t route::process_t::wout [private]

5.347.3.2 MHAMultiSrc::spectrum_t route::process_t::sout [private]

5.347.3.3 MHAMultiSrc::waveform_t route::process_t::wout_ac [private]

5.347.3.4 MHAMultiSrc::spectrum_t route::process_t::sout_ac [private]

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

- **route.cpp**

5.348 rt_nlms_t Class Reference

Public Member Functions

- **rt_nlms_t** (**algo_comm_t** iac, const std::string &name, const **mhaconfig_t** &cfg, unsigned int ntaps_, const std::string &name_u, const std::string &name_d, const std::string &name_e, const std::string &name_f, const int n_no_update)
- ~**rt_nlms_t** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗sUD, **mha_real_t** rho, **mha_real_t** c, unsigned int norm_type, unsigned int estim_type, **mha_real_t** lambda_smooth)
- void **insert** ()

Private Attributes

- **algo_comm_t ac**
- unsigned int **ntaps**
- unsigned int **frames**
- unsigned int **channels**
- **MHA_AC::waveform_t F**
- **MHASignal::waveform_t U**

  Input signal cache.
- **MHASignal::waveform_t Uflt**

  Input signal cache (second filter)
- **MHASignal::waveform_t Pu**

  Power of input signal delayline.
- **MHASignal::waveform_t fu**

  Filtered input signal.
- **MHASignal::waveform_t fuflt**

  Filtered input signal.
- **MHASignal::waveform_t fu_previous**
- **MHASignal::waveform_t y_previous**
- **MHASignal::waveform_t P_Sum**
- std::string **name_u_**
- std::string **name_d_**
- std::string **name_e_**
- int **n_no_update_**
- int **no_iter**
- **mha_wave_t s_E**

5.348.1 Constructor & Destructor Documentation

5.348.1.1 rt_nlms_t::rt_nlms_t (

algo_comm_t *iac,*
const std::string & *name,*
const mhaconfig_t & *cfg,*
unsigned int *ntaps_,*
const std::string & *name_u,*
const std::string & *name_d,*
const std::string & *name_e,*
const std::string & *name_f,*
const int *n_no_update*)

5.348.1.2 rt_nlms_t::~rt_nlms_t ( ) [inline]

5.348.2 Member Function Documentation

5.348.2.1 mha_wave_t * rt_nlms_t::process (

mha_wave_t * *sUD,*
mha_real_t *rho,*
mha_real_t *c,*
unsigned int *norm_type,*
unsigned int *estim_type,*
mha_real_t *lambda_smooth*)

5.348.2.2 void rt_nlms_t::insert ( )

5.348.3 Member Data Documentation

5.348.3.1 algo_comm_t rt_nlms_t::ac [private]

5.348.3.2 unsigned int rt_nlms_t::ntaps [private]

5.348.3.3 unsigned int rt_nlms_t::frames [private]

5.348.3.4 unsigned int rt_nlms_t::channels [private]

5.348.3.5 MHA_AC::waveform_t rt_nlms_t::F [private]

5.348.3.6 MHASignal::waveform_t rt_nlms_t::U [private]

Input signal cache.

5.348.3.7 MHASignal::waveform_t rt_nlms_t::Uflt [private]

Input signal cache (second filter)

5.348.3.8 MHASignal::waveform_t rt_nlms_t::Pu [private]

Power of input signal delayline.

5.348.3.9 MHASignal::waveform_t rt_nlms_t::fu [private]

Filtered input signal.

5.348.3.10 MHASignal::waveform_t rt_nlms_t::fuflt [private]

Filtered input signal.

5.348.3.11 MHASignal::waveform_t rt_nlms_t::fu_previous [private]

5.348.3.12 MHASignal::waveform_t rt_nlms_t::y_previous [private]

5.348.3.13 MHASignal::waveform_t rt_nlms_t::P_Sum [private]

5.348.3.14 std::string rt_nlms_t::name_u_ [private]

5.348.3.15 std::string rt_nlms_t::name_d_ [private]

5.348.3.16 std::string rt_nlms_t::name_e_ [private]

5.348.3.17 int rt_nlms_t::n_no_update_ [private]

5.348.3.18 int rt_nlms_t::no_iter [private]

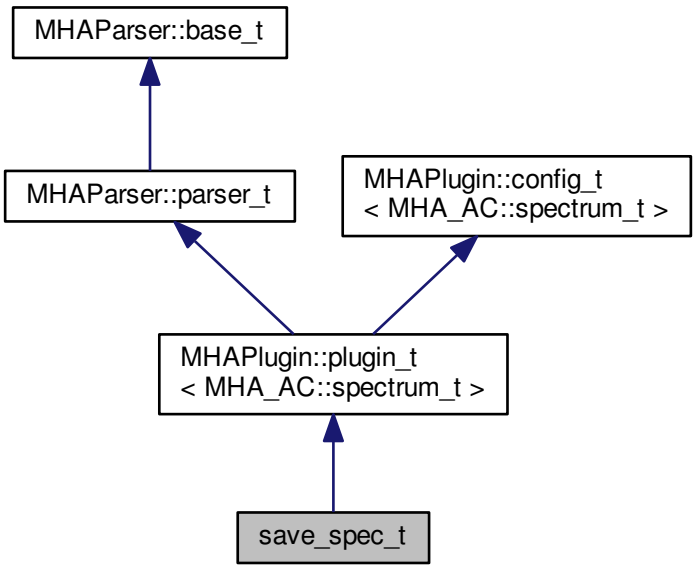
5.348.3.19 mha_wave_t rt_nlms_t::s_E [private]

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

- **nlms_wave.cpp**

5.349 save_spec_t Class Reference

Inheritance diagram for save_spec_t:



Public Member Functions

- **save_spec_t** (const **algo_comm_t** &iac, const std::string &ith, const std::string &ial)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)
- void **prepare** (**mhaconfig_t** &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.349.1 Constructor & Destructor Documentation

5.349.1.1 save_spec_t::save_spec_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*) [inline]

5.349.2 Member Function Documentation

5.349.2.1 mha_spec_t* save_spec_t::process (mha_spec_t * *s*) [inline]

5.349.2.2 void save_spec_t::prepare (mhaconfig_t & *tf*) [inline], [virtual]

Implements **MHAPlugin::plugin_t< MHA_AC::spectrum_t >** (p. 730).

5.349.3 Member Data Documentation

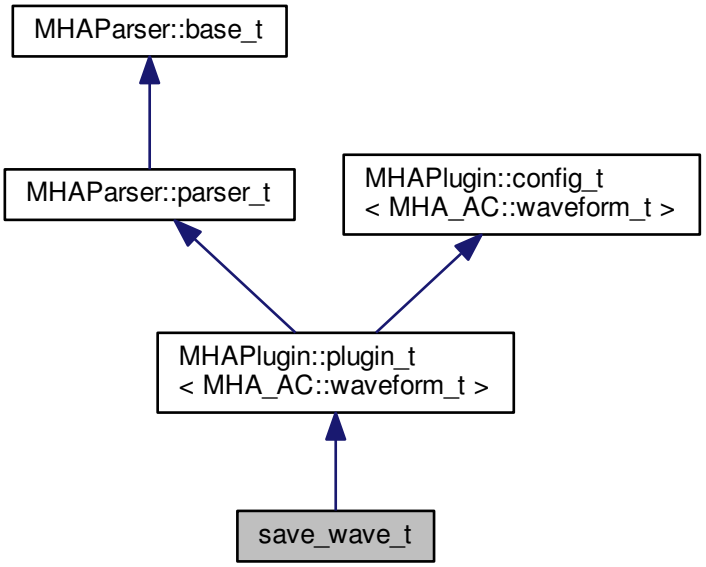
5.349.3.1 std::string save_spec_t::basename [private]

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

- **save_spec.cpp**

5.350 save_wave_t Class Reference

Inheritance diagram for save_wave_t:



Public Member Functions

- **save_wave_t** (const **algo_comm_t** &iac, const std::string &ith, const std::string &ial)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
- void **prepare** (**mhaconfig_t** &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.350.1 Constructor & Destructor Documentation

5.350.1.1 save_wave_t::save_wave_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*) [inline]

5.350.2 Member Function Documentation

5.350.2.1 mha_wave_t∗ save_wave_t::process (mha_wave_t ∗ *s*) [inline]

5.350.2.2 void save_wave_t::prepare (mhaconfig_t & *tf*) [inline], [virtual]

Implements **MHAPlugin::plugin_t**< **MHA_AC::waveform_t** > (p. 730).

5.350.3 Member Data Documentation

5.350.3.1 std::string save_wave_t::basename [private]

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

- **save_wave.cpp**

5.351 shadowfilter_begin::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft, int inch, int outch, **algo_comm_t** ac, std::string name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- **MHA_AC::spectrum_t in_spec_copy**
- **MHASignal::spectrum_t out_spec**
- **MHA_AC::int_t nch**
- **MHA_AC::int_t ntracks**

5.351.1 Constructor & Destructor Documentation

5.351.1.1 cfg_t::cfg_t (int *nfft,* int *inch,* int *outch,* algo_comm_t *ac,* std::string *name*)

5.351.2 Member Function Documentation

5.351.2.1 mha_spec_t * cfg_t::process (mha_spec_t * *s*)

5.351.3 Member Data Documentation

5.351.3.1 MHA_AC::spectrum_t shadowfilter_begin::cfg_t::in_spec_copy [private]

5.351.3.2 MHASignal::spectrum_t shadowfilter_begin::cfg_t::out_spec [private]

5.351.3.3 MHA_AC::int_t shadowfilter_begin::cfg_t::nch [private]

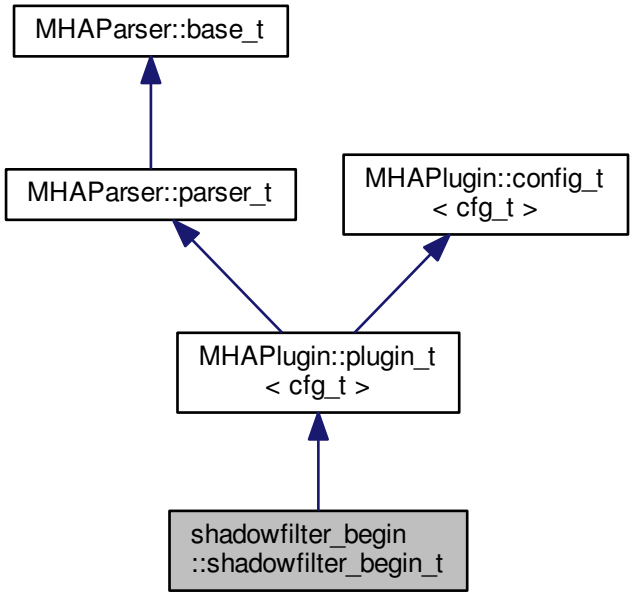
5.351.3.4 MHA_AC::int_t shadowfilter_begin::cfg_t::ntracks [private]

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

- **shadowfilter_begin.cpp**

5.352 shadowfilter_begin::shadowfilter_begin_t Class Reference

Inheritance diagram for shadowfilter_begin::shadowfilter_begin_t:



Public Member Functions

- **shadowfilter_begin_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- std::string **basename**
- **MHAParser::int_t nch**
- **MHAParser::int_t ntracks**

Additional Inherited Members

5.352.1 Constructor & Destructor Documentation

5.352.1.1 shadowfilter_begin::shadowfilter_begin_t::shadowfilter_begin_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.352.2 Member Function Documentation

5.352.2.1 mha_spec_t * shadowfilter_begin::shadowfilter_begin_t::process (mha_spec_t * *s*)

5.352.2.2 void shadowfilter_begin::shadowfilter_begin_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 730).

5.352.3 Member Data Documentation

5.352.3.1 std::string shadowfilter_begin::shadowfilter_begin_t::basename [private]

5.352.3.2 MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::nch [private]

5.352.3.3 MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::ntracks [private]

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

- **shadowfilter_begin.cpp**

5.353 shadowfilter_end::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft_, **algo_comm_t** ac_, std::string name_)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **algo_comm_t ac**
- std::string **name**
- int **nfft**
- int **ntracks**
- int **nch_out**
- **mha_spec_t in_spec**
- **MHASignal::spectrum_t out_spec**
- **MHA_AC::spectrum_t gains**

5.353.1 Constructor & Destructor Documentation

5.353.1.1 cfg_t::cfg_t (int *nfft_*, algo_comm_t *ac_*, std::string *name_*)

5.353.2 Member Function Documentation

5.353.2.1 mha_spec_t * cfg_t::process (mha_spec_t * *s*)

5.353.3 Member Data Documentation

5.353.3.1 algo_comm_t shadowfilter_end::cfg_t::ac [private]

5.353.3.2 std::string shadowfilter_end::cfg_t::name [private]

5.353.3.3 int shadowfilter_end::cfg_t::nfft [private]

5.353.3.4 int shadowfilter_end::cfg_t::ntracks [private]

5.353.3.5 int shadowfilter_end::cfg_t::nch_out [private]

5.353.3.6 mha_spec_t shadowfilter_end::cfg_t::in_spec [private]

5.353.3.7 MHASignal::spectrum_t shadowfilter_end::cfg_t::out_spec [private]

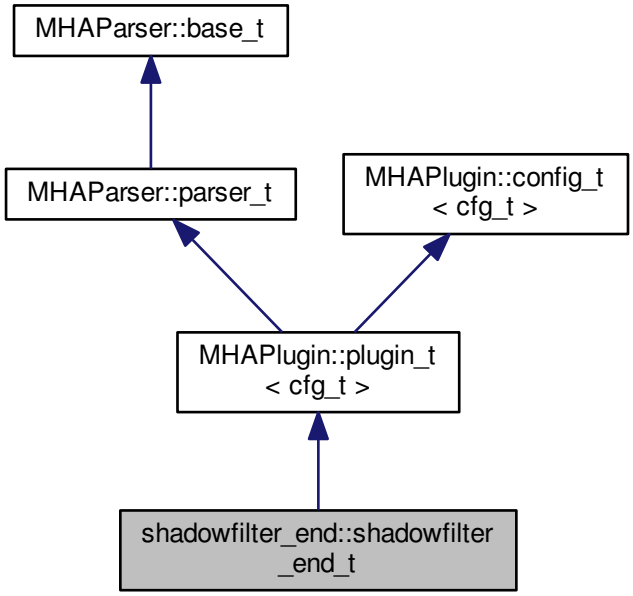
5.353.3.8 MHA_AC::spectrum_t shadowfilter_end::cfg_t::gains [private]

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

- **shadowfilter_end.cpp**

5.354 shadowfilter_end::shadowfilter_end_t Class Reference

Inheritance diagram for shadowfilter_end::shadowfilter_end_t:



Public Member Functions

- **shadowfilter_end_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- **MHAParser::string_t basename**

Additional Inherited Members

5.354.1 Constructor & Destructor Documentation

5.354.1.1 shadowfilter_end::shadowfilter_end_t::shadowfilter_end_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.354.2 Member Function Documentation

5.354.2.1 mha_spec_t * shadowfilter_end::shadowfilter_end_t::process (mha_spec_t * *s*)

5.354.2.2 void shadowfilter_end::shadowfilter_end_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 730).

5.354.3 Member Data Documentation

5.354.3.1 MHAParser::string_t shadowfilter_end::shadowfilter_end_t::basename [private]

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

- **shadowfilter_end.cpp**

5.355 sine_cfg_t Struct Reference

Public Member Functions

- **sine_cfg_t** (double sampling_rate, **mha_real_t** frequency, **mha_real_t** newlev, int _mix, const std::vector< int > &_channels)

Public Attributes

- double **phase_increment_div_2pi**
- double **amplitude**
- int **mix**
- const std::vector< int > **channels**

5.355.1 Constructor & Destructor Documentation

5.355.1.1 sine_cfg_t::sine_cfg_t (

double *sampling_rate,*
mha_real_t *frequency,*
mha_real_t *newlev,*
int *_mix,*
const std::vector< int > & *_channels*) [inline]

5.355.2 Member Data Documentation

5.355.2.1 double sine_cfg_t::phase_increment_div_2pi

5.355.2.2 double sine_cfg_t::amplitude

5.355.2.3 int sine_cfg_t::mix

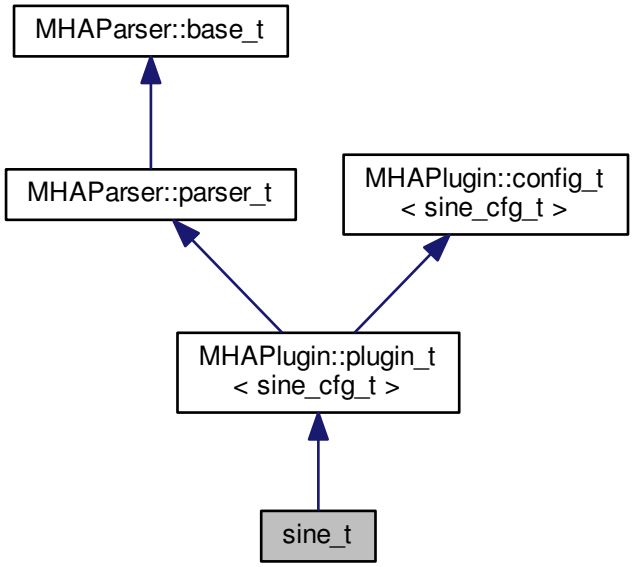
5.355.2.4 const std::vector<int> sine_cfg_t::channels

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

- **sine.cpp**

5.356 sine_t Class Reference

Inheritance diagram for sine_t:



Public Member Functions

- **sine_t** (const **algo_comm_t** &, const std::string &chain_name, const std::string &algo↩ _name)
- ∼**sine_t** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t lev**
- **MHAParser::float_t frequency**
- **MHAParser::kw_t mode**
- **MHAParser::vint_t channels**
- double **phase_div_2pi**
- **MHAEvents::patchbay_t**< **sine_t** > **patchbay**

Additional Inherited Members

5.356.1 Constructor & Destructor Documentation

5.356.1.1 sine_t::sine_t (

const **algo_comm_t** & *iac,*
const std::string & *chain_name,*
const std::string & *algo_name*)

5.356.1.2 sine_t::∼sine_t ( )

5.356.2 Member Function Documentation

5.356.2.1 mha_wave_t ∗ sine_t::process (

mha_wave_t ∗ *s*)

5.356.2.2 void sine_t::prepare (

mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t**< **sine_cfg_t** > (p. 730).

5.356.2.3 void sine_t::update_cfg (void) [private]

5.356.3 Member Data Documentation

5.356.3.1 MHAParser::float_t sine_t::lev [private]

5.356.3.2 MHAParser::float_t sine_t::frequency [private]

5.356.3.3 MHAParser::kw_t sine_t::mode [private]

5.356.3.4 MHAParser::vint_t sine_t::channels [private]

5.356.3.5 double sine_t::phase_div_2pi [private]

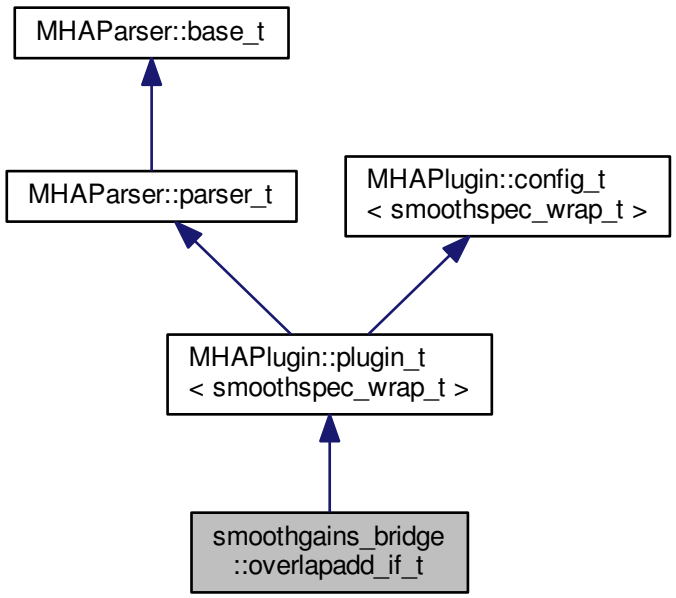
5.356.3.6 MHAEvents::patchbay_t<sine_t> sine_t::patchbay [private]

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

- **sine.cpp**

5.357 smoothgains_bridge::overlapadd_if_t Class Reference

Inheritance diagram for smoothgains_bridge::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- ~**overlapadd_if_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **overlapadd_if_t** > **patchbay**
- **MHAParser::kw_t mode**
- **MHAParser::window_t irswnd**
- **MHAParser::float_t epsilon**
- **MHAParser::mhapluginloader_t plugloader**
- std::string **algo**
- **mhaconfig_t cf_in**
- **mhaconfig_t cf_out**

Additional Inherited Members

5.357.1 Constructor & Destructor Documentation

5.357.1.1 smoothgains_bridge::overlapadd_if_t::overlapadd_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string & *ialg*)

5.357.1.2 smoothgains_bridge::overlapadd_if_t::~overlapadd_if_t ( )

5.357.2 Member Function Documentation

5.357.2.1 void smoothgains_bridge::overlapadd_if_t::prepare (**mhaconfig_t** & *t*) [virtual]

Implements **MHAPlugin::plugin_t**< **smoothspec_wrap_t** > (p. 730).

5.357.2.2 void smoothgains_bridge::overlapadd_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **smoothspec_wrap_t** > (p. 731).

5.357.2.3 mha_spec_t * smoothgains_bridge::overlapadd_if_t::process (mha_spec_t * *spec*)

5.357.2.4 void smoothgains_bridge::overlapadd_if_t::update ( ) [private]

5.357.3 Member Data Documentation

5.357.3.1 MHAEvents::patchbay_t<overlapadd_if_t> smoothgains_bridge::overlapadd_if_t::patchbay [private]

5.357.3.2 MHAParser::kw_t smoothgains_bridge::overlapadd_if_t::mode [private]

5.357.3.3 MHAParser::window_t smoothgains_bridge::overlapadd_if_t::irswnd [private]

5.357.3.4 MHAParser::float_t smoothgains_bridge::overlapadd_if_t::epsilon [private]

5.357.3.5 MHAParser::mhapluginloader_t smoothgains_bridge::overlapadd_if_t::plugloader [private]

5.357.3.6 std::string smoothgains_bridge::overlapadd_if_t::algo [private]

5.357.3.7 mhaconfig_t smoothgains_bridge::overlapadd_if_t::cf_in [private]

5.357.3.8 mhaconfig_t smoothgains_bridge::overlapadd_if_t::cf_out [private]

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

- **smoothgains_bridge.cpp**

5.358 smoothgains_bridge::smoothspec_wrap_t Class Reference

Public Member Functions

- **smoothspec_wrap_t** (**mhaconfig_t** spar_in, **mhaconfig_t** spar_out, const **MHAParser::kw_t** &mode, const **MHAParser::window_t** &irswnd, const **MHAParser::float_t** &epsilon)
- **mha_spec_t** * **proc_1** (**mha_spec_t** *)
- **mha_spec_t** * **proc_2** (**mha_spec_t** *)

Private Attributes

- **MHASignal::spectrum_t spec_in_copy**
  Copy of input spectrum for smoothspec.
- **MHAFilter::smoothspec_t smoothspec**
  Smoothspec calculator.
- bool **use_smoothspec**
- float **smoothspec_epsilon**

5.358.1 Constructor & Destructor Documentation

5.358.1.1 smoothgains_bridge::smoothspec_wrap_t::smoothspec_wrap_t (mhaconfig_t *spar_in*, mhaconfig_t *spar_out*, const MHAParser::kw_t & *mode*, const MHAParser::window_t & *irswnd*, const MHAParser::float_t & *epsilon*)

5.358.2 Member Function Documentation

5.358.2.1 mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_1 (mha_spec_t * *s*)

5.358.2.2 mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_2 (mha_spec_t * *s*)

5.358.3 Member Data Documentation

5.358.3.1 MHASignal::spectrum_t smoothgains_bridge::smoothspec_wrap_t::spec_in_copy [private]

Copy of input spectrum for smoothspec.

5.358.3.2 MHAFilter::smoothspec_t smoothgains_bridge::smoothspec_wrap_t::smoothspec [private]

Smoothspec calculator.

5.358.3.3 bool smoothgains_bridge::smoothspec_wrap_t::use_smoothspec [private]

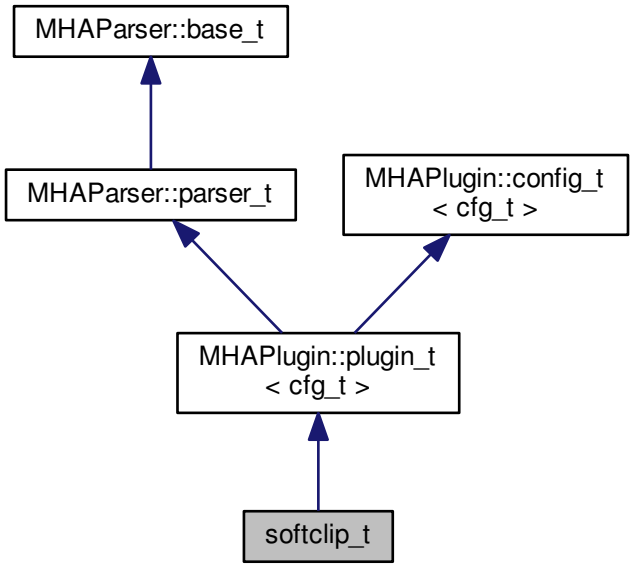
5.358.3.4 float smoothgains_bridge::smoothspec_wrap_t::smoothspec_epsilon [private]

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

- **smoothgains_bridge.cpp**

5.359 softclip_t Class Reference

Inheritance diagram for softclip_t:



Public Member Functions

- **softclip_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **update** ()

Private Attributes

- **mhaconfig_t tftype**
- **MHAParser::float_t attack**
- **MHAParser::float_t decay**
- **MHAParser::float_t start_limit**
- **MHAParser::float_t slope_db**
- **MHAEvents::patchbay_t**< **softclip_t** > **patchbay**

Additional Inherited Members

5.359.1 Constructor & Destructor Documentation

5.359.1.1 softclip_t::softclip_t (const algo_comm_t & *iac,* const std::string & *chain,* const std::string & *name*)

5.359.2 Member Function Documentation

5.359.2.1 mha_wave_t * softclip_t::process (mha_wave_t * *s*)

5.359.2.2 void softclip_t::prepare (mhaconfig_t & *tf*) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 730).

5.359.2.3 void softclip_t::update ( )

5.359.3 Member Data Documentation

5.359.3.1 mhaconfig_t softclip_t::tftype [private]

5.359.3.2 MHAParser::float_t softclip_t::attack [private]

5.359.3.3 MHAParser::float_t softclip_t::decay [private]

5.359.3.4 MHAParser::float_t softclip_t::start_limit [private]

5.359.3.5 MHAParser::float_t softclip_t::slope_db [private]

5.359.3.6 MHAEvents::patchbay_t<softclip_t> softclip_t::patchbay [private]

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

- **softclip.cpp**

5.360 softclipper_t Class Reference

Public Member Functions

- **softclipper_t** (const **softclipper_variables_t** &v, const **mhaconfig_t** &)
- **mha_real_t process** (**mha_wave_t** *)

Private Attributes

- **MHAFilter::o1flt_lowpass_t attack**
- **MHAFilter::o1flt_maxtrack_t decay**
- **MHAFilter::o1flt_lowpass_t clipmeter**
- **mha_real_t threshold**
- **mha_real_t hardlimit**
- **mha_real_t slope**
- bool **linear**

5.360.1 Constructor & Destructor Documentation

5.360.1.1 softclipper_t::softclipper_t (const softclipper_variables_t & *v*, const mhaconfig_t & *tf*)

5.360.2 Member Function Documentation

5.360.2.1 mha_real_t softclipper_t::process (mha_wave_t * *s*)

5.360.3 Member Data Documentation

5.360.3.1 MHAFilter::o1flt_lowpass_t softclipper_t::attack [private]

5.360.3.2 MHAFilter::o1flt_maxtrack_t softclipper_t::decay [private]

5.360.3.3 MHAFilter::o1flt_lowpass_t softclipper_t::clipmeter [private]

5.360.3.4 mha_real_t softclipper_t::threshold [private]

5.360.3.5 mha_real_t softclipper_t::hardlimit [private]

5.360.3.6 mha_real_t softclipper_t::slope [private]

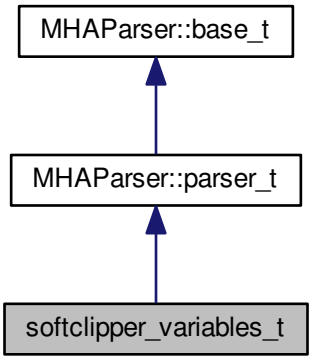
5.360.3.7 bool softclipper_t::linear [private]

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

- **transducers.cpp**

5.361 softclipper_variables_t Class Reference

Inheritance diagram for softclipper_variables_t:



Public Member Functions

- **softclipper_variables_t** ()

Public Attributes

- **MHAParser::float_t tau_attack**
- **MHAParser::float_t tau_decay**
- **MHAParser::float_t tau_clip**
- **MHAParser::float_t threshold**
- **MHAParser::float_t hardlimit**
- **MHAParser::float_t slope**
- **MHAParser::bool_t linear**
- **MHAParser::float_mon_t clipped**
- **MHAParser::float_t max_clipped**

Additional Inherited Members

5.361.1 Constructor & Destructor Documentation

5.361.1.1 softclipper_variables_t::softclipper_variables_t ( )

5.361.2 Member Data Documentation

5.361.2.1 MHAParser::float_t softclipper_variables_t::tau_attack

5.361.2.2 MHAParser::float_t softclipper_variables_t::tau_decay

5.361.2.3 MHAParser::float_t softclipper_variables_t::tau_clip

5.361.2.4 MHAParser::float_t softclipper_variables_t::threshold

5.361.2.5 MHAParser::float_t softclipper_variables_t::hardlimit

5.361.2.6 MHAParser::float_t softclipper_variables_t::slope

5.361.2.7 MHAParser::bool_t softclipper_variables_t::linear

5.361.2.8 MHAParser::float_mon_t softclipper_variables_t::clipped

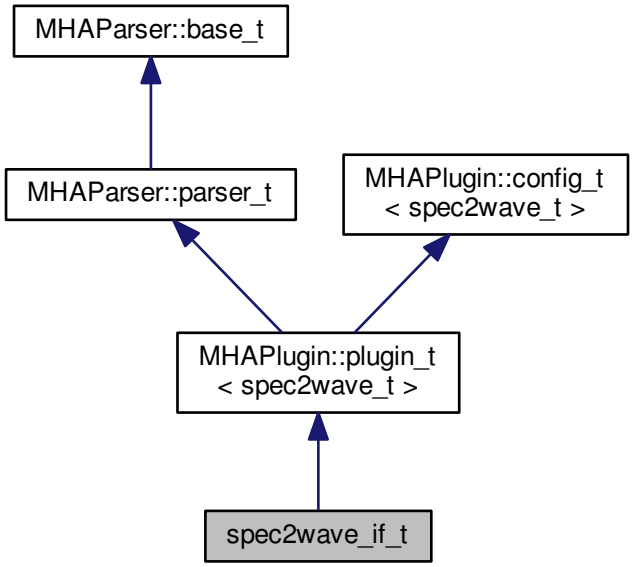
5.361.2.9 MHAParser::float_t softclipper_variables_t::max_clipped

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

- **transducers.cpp**

5.362 spec2wave_if_t Class Reference

Inheritance diagram for spec2wave_if_t:



Public Member Functions

- **spec2wave_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **spec2wave_if_t** > **patchbay**
- **MHAParser::float_t ramplen**
- **windowselector_t window_config**

Additional Inherited Members

5.362.1 Constructor & Destructor Documentation

5.362.1.1 spec2wave_if_t::spec2wave_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string &)

5.362.2 Member Function Documentation

5.362.2.1 void spec2wave_if_t::prepare (mhaconfig_t & *t*) [virtual]

Implements **MHAPlugin::plugin_t**< **spec2wave_t** > (p. 730).

5.362.2.2 mha_wave_t ∗ spec2wave_if_t::process (mha_spec_t ∗ *spec_in*)

5.362.2.3 void spec2wave_if_t::update ( ) [private]

5.362.3 Member Data Documentation

5.362.3.1 MHAEvents::patchbay_t<spec2wave_if_t> spec2wave_if_t::patchbay [private]

5.362.3.2 MHAParser::float_t spec2wave_if_t::ramplen [private]

5.362.3.3 windowselector_t spec2wave_if_t::window_config [private]

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

- **spec2wave.cpp**

5.363 spec2wave_t Class Reference

Public Member Functions

- **spec2wave_t** (unsigned int nfft_, unsigned int nwnd_, unsigned int nwndshift_, unsigned int nch, **mha_real_t** ramplen, const **MHAWindow::base_t** &postwin)
- ∼**spec2wave_t** ()
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- **mha_fft_t ft**
  FFT class.
- unsigned int **npad1**
  length of zero padding before window
- unsigned int **npad2**
  length of zero padding after window
- **hanning_ramps_t ramps**
- **MHASignal::waveform_t calc_out**
- **MHASignal::waveform_t out_buf**
- **MHASignal::waveform_t write_buf**
- **mha_real_t sc**
- unsigned int **nfft**
- unsigned int **nwndshift**
- **MHAWindow::base_t postwindow**

5.363.1 Constructor & Destructor Documentation

5.363.1.1 spec2wave_t::spec2wave_t (
unsigned int *nfft_,*
unsigned int *nwnd_,*
unsigned int *nwndshift_,*
unsigned int *nch,*
mha_real_t *ramplen,*
const **MHAWindow::base_t** & *postwin*)

5.363.1.2 spec2wave_t::∼spec2wave_t ( )

5.363.2 Member Function Documentation

5.363.2.1 mha_wave_t ∗ spec2wave_t::process (
mha_spec_t ∗ *spec_in*)

5.363.3 Member Data Documentation

5.363.3.1 mha_fft_t spec2wave_t::ft [private]

FFT class.

5.363.3.2 unsigned int spec2wave_t::npad1 [private]

length of zero padding before window

5.363.3.3 unsigned int spec2wave_t::npad2 [private]

length of zero padding after window

5.363.3.4 hanning_ramps_t spec2wave_t::ramps [private]

5.363.3.5 MHASignal::waveform_t spec2wave_t::calc_out [private]

5.363.3.6 MHASignal::waveform_t spec2wave_t::out_buf [private]

5.363.3.7 MHASignal::waveform_t spec2wave_t::write_buf [private]

5.363.3.8 mha_real_t spec2wave_t::sc [private]

5.363.3.9 unsigned int spec2wave_t::nfft [private]

5.363.3.10 unsigned int spec2wave_t::nwndshift [private]

5.363.3.11 MHAWindow::base_t spec2wave_t::postwindow [private]

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

- **spec2wave.cpp**

5.364 spec_fader_t Class Reference

Public Member Functions

- **spec_fader_t** (unsigned int ch, **mha_real_t fr**, **MHAParser::vfloat_t** &ng, **MHAParser**↩ **::float_t** &t)
- ∼**spec_fader_t** ()

Public Attributes

- unsigned int **nch**
- **mha_real_t** ∗ **gains**
- unsigned int **fr**

5.364.1 Constructor & Destructor Documentation

5.364.1.1 spec_fader_t::spec_fader_t (unsigned int *ch,* mha_real_t *fr,* MHAParser::vfloat_t & *ng,* MHAParser::float_t & *t*)

5.364.1.2 spec_fader_t::~spec_fader_t ( ) [inline]

5.364.2 Member Data Documentation

5.364.2.1 unsigned int spec_fader_t::nch

5.364.2.2 mha_real_t* spec_fader_t::gains

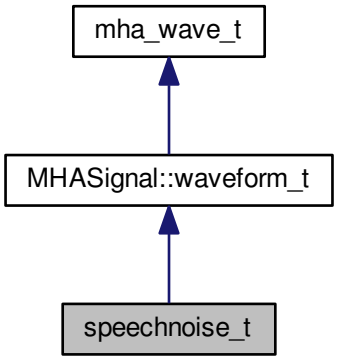
5.364.2.3 unsigned int spec_fader_t::fr

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

- **fader_spec.cpp**

5.365 speechnoise_t Class Reference

Inheritance diagram for speechnoise_t:



Public Types

Public Member Functions

- **speechnoise_t** (float duration, float srate, unsigned int **channels**, **speechnoise_t↩ ::noise_type_t** noise_type=**speechnoise_t::mha**)
- **speechnoise_t** (unsigned int length_samples, float srate, unsigned int **channels**, **speechnoise_t::noise_type_t** noise_type=**speechnoise_t::mha**)

Private Member Functions

- void **creator** (**speechnoise_t::noise_type_t** noise_type, float srate)

Additional Inherited Members

5.365.1 Member Enumeration Documentation

5.365.1.1 enum speechnoise_t::noise_type_t

Enumerator

mha

olnoise

LTASS_combined

LTASS_female

LTASS_male

white

pink

brown

TEN_SPL

TEN_SPL_250_8k

TEN_SPL_50_16k

sin125

sin250

sin500

sin1k

sin2k

sin4k

sin8k

5.365.2 Constructor & Destructor Documentation

5.365.2.1 speechnoise_t::speechnoise_t (
float *duration,*
float *srate,*
unsigned int *channels,*
speechnoise_t::noise_type_t *noise_type* = **speechnoise_t::mha**)

5.365.2.2 speechnoise_t::speechnoise_t (
unsigned int *length_samples,*
float *srate,*
unsigned int *channels,*
speechnoise_t::noise_type_t *noise_type* = **speechnoise_t::mha**)

5.365.3 Member Function Documentation

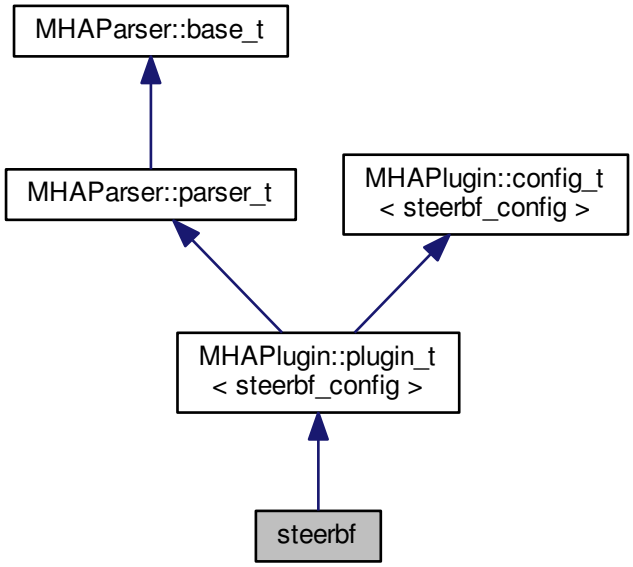
5.365.3.1 void speechnoise_t::creator (
speechnoise_t::noise_type_t *noise_type,*
float *srate*) [private]

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

- **speechnoise.h**
- **speechnoise.cpp**

5.366 steerbf Class Reference

Inheritance diagram for steerbf:



Public Member Functions

- **steerbf** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_↩ name)
  Constructs our plugin.
- ∼**steerbf** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
  Defers to configuration class.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::string_t bf_src**
- **parser_int_dyn angle_ind**
- **MHAParser::string_t angle_src**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **steerbf** > **patchbay**

Additional Inherited Members

5.366.1 Constructor & Destructor Documentation

5.366.1.1 steerbf::steerbf (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs our plugin.

5.366.1.2 steerbf::∼steerbf ( )

5.366.2 Member Function Documentation

5.366.2.1 mha_spec_t ∗ steerbf::process (mha_spec_t ∗ *signal*)

Defers to configuration class.

5.366.2.2 void steerbf::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< steerbf_config >** (p. 730).

5.366.2.3 void steerbf::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< steerbf_config >** (p. 731).

5.366.2.4 void steerbf::update_cfg (void) [private]

5.366.3 Member Data Documentation

5.366.3.1 MHAParser::string_t steerbf::bf_src

5.366.3.2 parser_int_dyn steerbf::angle_ind

5.366.3.3 MHAParser::string_t steerbf::angle_src

5.366.3.4 MHAEvents::patchbay_t<steerbf> steerbf::patchbay [private]

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

- **steerbf.h**
- **steerbf.cpp**

5.367 steerbf_config Class Reference

Public Member Functions

- **steerbf_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** in_cfg, **steerbf** ∗**steerbf**)
- ∼**steerbf_config** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- unsigned int **nchan**
- unsigned int **nfreq**
- **MHASignal::spectrum_t outSpec**
- **mha_spec_t bf_vec**
- unsigned int **nangle**
- **steerbf** * **_steerbf**
- **algo_comm_t** & **ac**
- std::string **bf_src_copy**

5.367.1 Constructor & Destructor Documentation

5.367.1.1 steerbf_config::steerbf_config (algo_comm_t & *ac*, const mhaconfig_t *in_cfg*, steerbf * *steerbf*)

5.367.1.2 steerbf_config::~steerbf_config ( )

5.367.2 Member Function Documentation

5.367.2.1 mha_spec_t * steerbf_config::process (mha_spec_t * *inSpec*)

5.367.3 Member Data Documentation

5.367.3.1 unsigned int steerbf_config::nchan [private]

5.367.3.2 unsigned int steerbf_config::nfreq [private]

5.367.3.3 MHASignal::spectrum_t steerbf_config::outSpec [private]

5.367.3.4 mha_spec_t steerbf_config::bf_vec [private]

5.367.3.5 unsigned int steerbf_config::nangle [private]

5.367.3.6 steerbf* steerbf_config::_steerbf [private]

5.367.3.7 algo_comm_t& steerbf_config::ac [private]

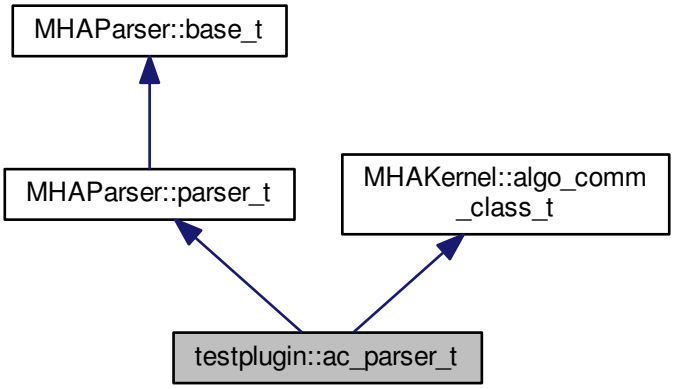
5.367.3.8 std::string steerbf_config::bf_src_copy [private]

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

- **steerbf.h**
- **steerbf.cpp**

5.368 testplugin::ac_parser_t Class Reference

Inheritance diagram for testplugin::ac_parser_t:



Public Types

Public Member Functions

- **ac_parser_t** ()
- void **do_insert_var** ()

  Insert variable into AC space.
- void **do_get_var** ()

Public Attributes

- **MHAParser::string_t insert_var**
- **MHAParser::string_t get_var**
- **MHAParser::kw_t data_type**
- **MHAParser::int_t num_entries**
- **MHAParser::int_t stride**
- **MHAParser::string_t char_data**
- **MHAParser::vint_t int_data**
- **MHAParser::vfloat_t float_data**
- **MHAParser::vcomplex_t complex_data**
- **MHAEvents::patchbay_t**< **ac_parser_t** > **patchbay**

Additional Inherited Members

5.368.1 Member Enumeration Documentation

5.368.1.1 enum testplugin::ac_parser_t::data_type_t

Enumerator

_MHA_AC_CHAR
_MHA_AC_INT
_MHA_AC_MHAREAL
_MHA_AC_FLOAT
_MHA_AC_DOUBLE
_MHA_AC_MHACOMPLEX
_unknown

5.368.2 Constructor & Destructor Documentation

5.368.2.1 testplugin::ac_parser_t::ac_parser_t ( ) [inline]

5.368.3 Member Function Documentation

5.368.3.1 void testplugin::ac_parser_t::do_insert_var ( ) [inline]

Insert variable into AC space.
This leaks memory by design, as the plugin is for testing only

5.368.3.2 void testplugin::ac_parser_t::do_get_var ( ) [inline]

5.368.4 Member Data Documentation

5.368.4.1 MHAParser::string_t testplugin::ac_parser_t::insert_var

5.368.4.2 MHAParser::string_t testplugin::ac_parser_t::get_var

5.368.4.3 MHAParser::kw_t testplugin::ac_parser_t::data_type

5.368.4.4 MHAParser::int_t testplugin::ac_parser_t::num_entries

5.368.4.5 MHAParser::int_t testplugin::ac_parser_t::stride

5.368.4.6 MHAParser::string_t testplugin::ac_parser_t::char_data

5.368.4.7 MHAParser::vint_t testplugin::ac_parser_t::int_data

5.368.4.8 MHAParser::vfloat_t testplugin::ac_parser_t::float_data

5.368.4.9 MHAParser::vcomplex_t testplugin::ac_parser_t::complex_data

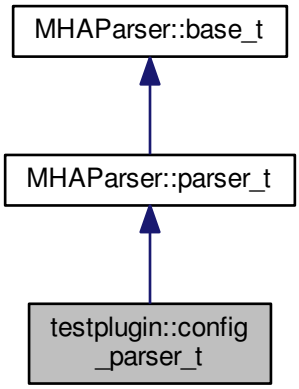
5.368.4.10 MHAEvents::patchbay_t<ac_parser_t> testplugin::ac_parser_t::patchbay

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

- **testplugin.cpp**

5.369 testplugin::config_parser_t Class Reference

Inheritance diagram for testplugin::config_parser_t:



Public Member Functions

- void **setlock** (const bool &b)
- **config_parser_t** ()
- **mhaconfig_t get** () const
- void **set** (**mhaconfig_t** c)

Public Attributes

- **MHAParser::int_t channels**
- **MHAParser::kw_t domain**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t wndlen**
- **MHAParser::int_t fftlen**
- **MHAParser::float_t srate**

Additional Inherited Members

5.369.1 Constructor & Destructor Documentation

5.369.1.1 testplugin::config_parser_t::config_parser_t ( ) [inline]

5.369.2 Member Function Documentation

5.369.2.1 void testplugin::config_parser_t::setlock (const bool & *b*) [inline]

5.369.2.2 mhaconfig_t testplugin::config_parser_t::get ( ) const [inline]

5.369.2.3 void testplugin::config_parser_t::set (mhaconfig_t *c*) [inline]

5.369.3 Member Data Documentation

5.369.3.1 MHAParser::int_t testplugin::config_parser_t::channels

5.369.3.2 MHAParser::kw_t testplugin::config_parser_t::domain

5.369.3.3 MHAParser::int_t testplugin::config_parser_t::fragsize

5.369.3.4 MHAParser::int_t testplugin::config_parser_t::wndlen

5.369.3.5 MHAParser::int_t testplugin::config_parser_t::fftlen

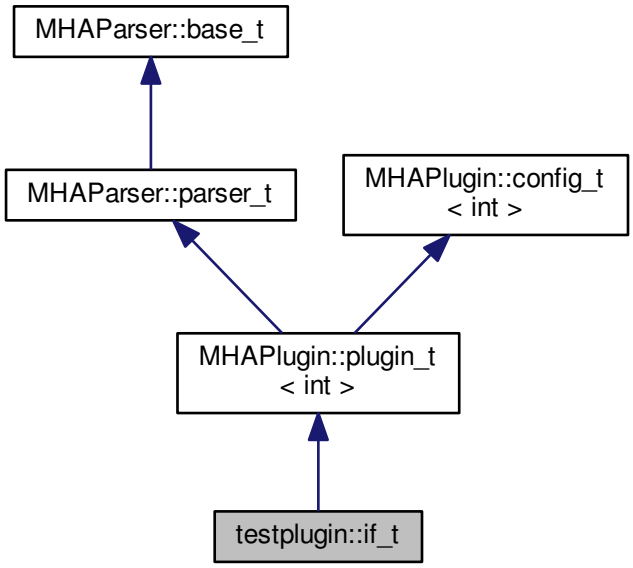
5.369.3.6 MHAParser::float_t testplugin::config_parser_t::srate

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

- **testplugin.cpp**

5.370 testplugin::if_t Class Reference

Inheritance diagram for testplugin::if_t:



Public Member Functions

- **if_t** (const **algo_comm_t** &iac, const std::string &ith, const std::string &ial)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s_in)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s_in)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **test_prepare** ()
- void **test_process** ()

Private Attributes

- **config_parser_t config_in**
- **config_parser_t config_out**
- **ac_parser_t ac**
- **signal_parser_t signal**
- **MHAParser::bool_t _prepare**
- **MHAEvents::patchbay_t**< **if_t** > **patchbay**
- **MHAParser::mhapluginloader_t plug**

Additional Inherited Members

5.370.1 Constructor & Destructor Documentation

5.370.1.1 testplugin::if_t::if_t (const algo_comm_t & *iac,* const std::string & *ith,* const std::string & *ial*)

5.370.2 Member Function Documentation

5.370.2.1 mha_spec_t* testplugin::if_t::process (mha_spec_t * *s_in*) [inline]

5.370.2.2 mha_wave_t* testplugin::if_t::process (mha_wave_t * *s_in*) [inline]

5.370.2.3 void testplugin::if_t::prepare (mhaconfig_t &) [inline], [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 730).

5.370.2.4 void testplugin::if_t::test_prepare ( ) [private]

5.370.2.5 void testplugin::if_t::test_process ( ) [private]

5.370.3 Member Data Documentation

5.370.3.1 config_parser_t testplugin::if_t::config_in [private]

5.370.3.2 config_parser_t testplugin::if_t::config_out [private]

5.370.3.3 ac_parser_t testplugin::if_t::ac [private]

5.370.3.4 signal_parser_t testplugin::if_t::signal [private]

5.370.3.5 MHAParser::bool_t testplugin::if_t::_prepare [private]

5.370.3.6 MHAEvents::patchbay_t<if_t> testplugin::if_t::patchbay [private]

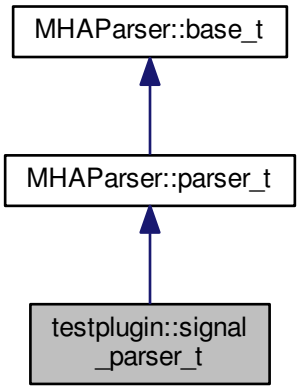
5.370.3.7 MHAParser::mhapluginloader_t testplugin::if_t::plug [private]

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

- **testplugin.cpp**

5.371 testplugin::signal_parser_t Class Reference

Inheritance diagram for testplugin::signal_parser_t:



Public Member Functions

- **signal_parser_t** ()

Public Attributes

- **MHAParser::mfloat_t input_wave**
- **MHAParser::mcomplex_t input_spec**
- **MHAParser::mfloat_mon_t output_wave**
- **MHAParser::mcomplex_mon_t output_spec**

Additional Inherited Members

5.371.1 Constructor & Destructor Documentation

5.371.1.1 testplugin::signal_parser_t::signal_parser_t ( ) [inline]

5.371.2 Member Data Documentation

5.371.2.1 MHAParser::mfloat_t testplugin::signal_parser_t::input_wave

5.371.2.2 MHAParser::mcomplex_t testplugin::signal_parser_t::input_spec

5.371.2.3 MHAParser::mfloat_mon_t testplugin::signal_parser_t::output_wave

5.371.2.4 MHAParser::mcomplex_mon_t testplugin::signal_parser_t::output_spec

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

- **testplugin.cpp**

5.372 timo_AC Class Reference

Public Member Functions

- **timo_AC** (**algo_comm_t** &ac, unsigned int fftlen, unsigned int nfreq, unsigned int nchan)
- void **copy** ()
- void **insert** ()

Public Attributes

- **MHA_AC::waveform_t gamma_post_AC**
- **MHA_AC::waveform_t xi_ml_AC**
- **MHA_AC::spectrum_t lambda_ml_AC**
- **MHA_AC::spectrum_t lambda_ml_ceps_AC**
- **MHA_AC::waveform_t lambda_ml_smooth_AC**
- **MHA_AC::waveform_t max_q_AC**
- **MHA_AC::waveform_t max_val_AC**
- **MHA_AC::waveform_t pitch_set_first_AC**
- **MHA_AC::waveform_t pitch_set_last_AC**
- **MHA_AC::waveform_t alpha_hat_AC**
- **MHA_AC::waveform_t alpha_frame_AC**
- **MHA_AC::spectrum_t lambda_ceps_AC**
- **MHA_AC::spectrum_t log_lambda_spec_AC**
- **MHA_AC::waveform_t lambda_spec_AC**
- **MHA_AC::waveform_t xi_est_AC**
- **MHA_AC::waveform_t gain_wiener_AC**
- **MHA_AC::waveform_t winF0_AC**
- **MHA_AC::waveform_t SPP**

5.372.1 Constructor & Destructor Documentation

5.372.1.1 timo_AC::timo_AC (algo_comm_t & *ac,* unsigned int *fftlen,* unsigned int *nfreq,* unsigned int *nchan*) [inline]

5.372.2 Member Function Documentation

5.372.2.1 void timo_AC::copy ( )

5.372.2.2 void timo_AC::insert ( )

5.372.3 Member Data Documentation

5.372.3.1 MHA_AC::waveform_t timo_AC::gamma_post_AC

5.372.3.2 MHA_AC::waveform_t timo_AC::xi_ml_AC

5.372.3.3 MHA_AC::spectrum_t timo_AC::lambda_ml_AC

5.372.3.4 MHA_AC::spectrum_t timo_AC::lambda_ml_ceps_AC

5.372.3.5 MHA_AC::waveform_t timo_AC::lambda_ml_smooth_AC

5.372.3.6 MHA_AC::waveform_t timo_AC::max_q_AC

5.372.3.7 MHA_AC::waveform_t timo_AC::max_val_AC

5.372.3.8 MHA_AC::waveform_t timo_AC::pitch_set_first_AC

5.372.3.9 MHA_AC::waveform_t timo_AC::pitch_set_last_AC

5.372.3.10 MHA_AC::waveform_t timo_AC::alpha_hat_AC

5.372.3.11 MHA_AC::waveform_t timo_AC::alpha_frame_AC

5.372.3.12 MHA_AC::spectrum_t timo_AC::lambda_ceps_AC

5.372.3.13 MHA_AC::spectrum_t timo_AC::log_lambda_spec_AC

5.372.3.14 MHA_AC::waveform_t timo_AC::lambda_spec_AC

5.372.3.15 MHA_AC::waveform_t timo_AC::xi_est_AC

5.372.3.16 MHA_AC::waveform_t timo_AC::gain_wiener_AC

5.372.3.17 MHA_AC::waveform_t timo_AC::winF0_AC

5.372.3.18 MHA_AC::waveform_t timo_AC::SPP

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

- **timoconfig.h**
- **timoconfig.cpp**

5.373 timo_params Class Reference

Public Member Functions

- **timo_params** (const **mhaconfig_t** &_in_cfg, float _xi_min_db, float _f0_low, float _f0_↩ high, float _delta_pitch, float _lambda_thresh, float _alpha_pitch, float _beta_const, float _kappa_const, float _prior_q, float _xi_opt_db, float _gain_min_db, std::vector< float > &_winF0, std::vector< float > &_alpha_const_vals, std::vector< float > &_alpha_const↩ _limits_hz, std::string &_noisePow_name)

Public Attributes

- const **mhaconfig_t in_cfg**
- float **xi_min_db**
- float **f0_low**
- float **f0_high**
- float **delta_pitch**
- float **lambda_thresh**
- float **alpha_pitch**
- float **beta_const**
- float **kappa_const**
- float **prior_q**
- float **xi_opt_db**
- float **gain_min_db**
- std::vector< float > **winF0**
- std::vector< float > **alpha_const_vals**
- std::vector< float > **alpha_const_limits_hz**
- std::string **noisePow_name**

5.373.1 Constructor & Destructor Documentation

5.373.1.1 timo_params::timo_params (

const mhaconfig_t & *_in_cfg,*
float *_xi_min_db,*
float *_f0_low,*
float *_f0_high,*
float *_delta_pitch,*
float *_lambda_thresh,*
float *_alpha_pitch,*
float *_beta_const,*
float *_kappa_const,*
float *_prior_q,*
float *_xi_opt_db,*
float *_gain_min_db,*
std::vector< float > & *_winF0,*
std::vector< float > & *_alpha_const_vals,*
std::vector< float > & *_alpha_const_limits_hz,*
std::string & *_noisePow_name*) `[inline]`

5.373.2 Member Data Documentation

5.373.2.1 const mhaconfig_t timo_params::in_cfg

5.373.2.2 float timo_params::xi_min_db

5.373.2.3 float timo_params::f0_low

5.373.2.4 float timo_params::f0_high

5.373.2.5 float timo_params::delta_pitch

5.373.2.6 float timo_params::lambda_thresh

5.373.2.7 float timo_params::alpha_pitch

5.373.2.8 float timo_params::beta_const

5.373.2.9 float timo_params::kappa_const

5.373.2.10 float timo_params::prior_q

5.373.2.11 float timo_params::xi_opt_db

5.373.2.12 float timo_params::gain_min_db

5.373.2.13 std::vector<float> timo_params::winF0

5.373.2.14 std::vector<float> timo_params::alpha_const_vals

5.373.2.15 std::vector<float> timo_params::alpha_const_limits_hz

5.373.2.16 std::string timo_params::noisePow_name

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

- **timoconfig.h**

5.374 timoConfig Class Reference

Public Member Functions

- **timoConfig** (**algo_comm_t** &**ac**, **timo_params** &**params**)
- ~**timoConfig** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **copy_AC** (**timo_AC** &**tAC**)

Private Attributes

- **algo_comm_t ac**
- **timo_params params**
- unsigned int **fftlen**
- **mha_fft_t mha_fft**
- unsigned int **nfreq**
- unsigned int **nchan**
- **timo_AC tAC**
- float **ola_powspec_scale**
- float **q_low**
- float **q_high**
- **MHASignal::waveform_t winF0**
- float **xi_min**
- float **gain_min**
- **MHASignal::waveform_t alpha_const**
- **MHASignal::waveform_t alpha_prev**
- **MHASignal::waveform_t noisePow**
- **MHASignal::waveform_t powSpec**
- **MHASignal::waveform_t gamma_post**
- **MHASignal::waveform_t xi_ml**
- **MHASignal::spectrum_t lambda_ml_full**
- **MHASignal::spectrum_t lambda_ml_ceps**
- **MHASignal::waveform_t lambda_ml_smooth**
- **MHASignal::waveform_t alpha_hat**
- **MHASignal::waveform_t alpha_frame**
- **MHASignal::spectrum_t lambda_ceps**
- **MHASignal::waveform_t lambda_ceps_prev**
- **MHASignal::spectrum_t log_lambda_spec**
- **MHASignal::waveform_t lambda_spec**
- **MHASignal::waveform_t xi_est**
- **MHASignal::waveform_t gain_wiener**
- **MHASignal::spectrum_t spec_out**
- double ∗ **max_val**
- int ∗ **max_q**
- int ∗ **pitch_set_first**
- int ∗ **pitch_set_last**
- float **priorFact**
- float **xiOpt**
- float **logGLRFact**
- float **GLRexp**
- **MHASignal::waveform_t GLR**

5.374.1 Constructor & Destructor Documentation

5.374.1.1 timoConfig::timoConfig (algo_comm_t & *ac,* timo_params & *params*)

5.374.1.2 timoConfig::∼timoConfig ( )

5.374.2 Member Function Documentation

5.374.2.1 mha_spec_t ∗ timoConfig::process (mha_spec_t ∗ *noisyFrame*)

5.374.2.2 void timoConfig::copy_AC (timo_AC & *tAC*) [private]

5.374.3 Member Data Documentation

5.374.3.1 algo_comm_t timoConfig::ac [private]

5.374.3.2 timo_params timoConfig::params [private]

5.374.3.3 unsigned int timoConfig::fftlen [private]

5.374.3.4 mha_fft_t timoConfig::mha_fft [private]

5.374.3.5 unsigned int timoConfig::nfreq [private]

5.374.3.6 unsigned int timoConfig::nchan [private]

5.374.3.7 timo_AC timoConfig::tAC [private]

5.374.3.8 float timoConfig::ola_powspec_scale [private]

5.374.3.9 float timoConfig::q_low [private]

5.374.3.10 float timoConfig::q_high [private]

5.374.3.11 MHASignal::waveform_t timoConfig::winF0 [private]

5.374.3.12 float timoConfig::xi_min [private]

5.374.3.13 float timoConfig::gain_min [private]

5.374.3.14 **MHASignal::waveform_t timoConfig::alpha_const** [private]

5.374.3.15 **MHASignal::waveform_t timoConfig::alpha_prev** [private]

5.374.3.16 **MHASignal::waveform_t timoConfig::noisePow** [private]

5.374.3.17 **MHASignal::waveform_t timoConfig::powSpec** [private]

5.374.3.18 **MHASignal::waveform_t timoConfig::gamma_post** [private]

5.374.3.19 **MHASignal::waveform_t timoConfig::xi_ml** [private]

5.374.3.20 **MHASignal::spectrum_t timoConfig::lambda_ml_full** [private]

5.374.3.21 **MHASignal::spectrum_t timoConfig::lambda_ml_ceps** [private]

5.374.3.22 **MHASignal::waveform_t timoConfig::lambda_ml_smooth** [private]

5.374.3.23 **MHASignal::waveform_t timoConfig::alpha_hat** [private]

5.374.3.24 **MHASignal::waveform_t timoConfig::alpha_frame** [private]

5.374.3.25 **MHASignal::spectrum_t timoConfig::lambda_ceps** [private]

5.374.3.26 **MHASignal::waveform_t timoConfig::lambda_ceps_prev** [private]

5.374.3.27 **MHASignal::spectrum_t timoConfig::log_lambda_spec** [private]

5.374.3.28 **MHASignal::waveform_t timoConfig::lambda_spec** [private]

5.374.3.29 **MHASignal::waveform_t timoConfig::xi_est** [private]

5.374.3.30 **MHASignal::waveform_t timoConfig::gain_wiener** [private]

5.374.3.31 **MHASignal::spectrum_t timoConfig::spec_out** [private]

5.374.3.32 **double* timoConfig::max_val** [private]

5.374.3.33 **int* timoConfig::max_q** [private]

5.374.3.34 **int* timoConfig::pitch_set_first** [private]

5.374.3.35 **int* timoConfig::pitch_set_last** [private]

5.374.3.36 **float timoConfig::priorFact** [private]

5.374.3.37 **float timoConfig::xiOpt** [private]

5.374.3.38 **float timoConfig::logGLRFact** [private]

5.374.3.39 **float timoConfig::GLRexp** [private]

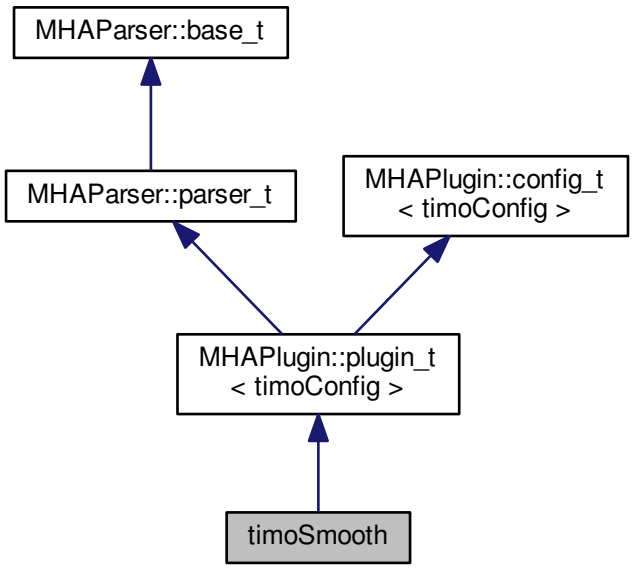
5.374.3.40 **MHASignal::waveform_t timoConfig::GLR** [private]

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

- **timoconfig.h**
- **timoconfig.cpp**

5.375 timoSmooth Class Reference

Inheritance diagram for timoSmooth:



Public Member Functions

- **timoSmooth** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)

  Constructs the beamforming plugin.
- ~**timoSmooth** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

  This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** (void)

Private Member Functions

- void **update_cfg** ()
- void **on_model_param_valuechanged** ()

Private Attributes

- **MHAParser::float_t xi_min_db**
- **MHAParser::float_t f0_low**
- **MHAParser::float_t f0_high**
- **MHAParser::float_t delta_pitch**
- **MHAParser::float_t lambda_thresh**
- **MHAParser::float_t alpha_pitch**
- **MHAParser::float_t beta_const**
- **MHAParser::float_t kappa_const**
- **MHAParser::float_t gain_min_db**
- **MHAParser::vfloat_t win_f0**
- **MHAParser::vfloat_t alpha_const_vals**
- **MHAParser::vfloat_t alpha_const_limits_hz**
- **MHAParser::string_t noisePow_name**
- **MHAParser::parser_t spp**
- **MHAParser::float_t prior_q**
- **MHAParser::float_t xi_opt_db**
- **MHAEvents::patchbay_t< timoSmooth > patchbay**
- bool **prepared**

Additional Inherited Members

5.375.1 Constructor & Destructor Documentation

5.375.1.1 timoSmooth::timoSmooth (algo_comm_t & *ac,* const std::string & *chain_name,* const std::string & *algo_name*)

Constructs the beamforming plugin.

5.375.1.2 timoSmooth::~timoSmooth ( )

5.375.2 Member Function Documentation

5.375.2.1 mha_spec_t * timoSmooth::process (mha_spec_t * *signal*)

This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin.

5.375.2.2 void timoSmooth::prepare (mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

This plugin checks that the input signal has the spectral domain and contains at least one channel

Parameters

signal_info	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Implements **MHAPlugin::plugin_t< timoConfig >** (p. 730).

5.375.2.3 void timoSmooth::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< timoConfig >** (p. 731).

5.375.2.4 void timoSmooth::update_cfg (void) [private]

5.375.2.5 void timoSmooth::on_model_param_valuechanged ( ) [private]

5.375.3 Member Data Documentation

5.375.3.1 MHAParser::float_t timoSmooth::xi_min_db [private]

5.375.3.2 MHAParser::float_t timoSmooth::f0_low [private]

5.375.3.3 MHAParser::float_t timoSmooth::f0_high [private]

5.375.3.4 MHAParser::float_t timoSmooth::delta_pitch [private]

5.375.3.5 MHAParser::float_t timoSmooth::lambda_thresh [private]

5.375.3.6 MHAParser::float_t timoSmooth::alpha_pitch [private]

5.375.3.7 MHAParser::float_t timoSmooth::beta_const [private]

5.375.3.8 MHAParser::float_t timoSmooth::kappa_const [private]

5.375.3.9 MHAParser::float_t timoSmooth::gain_min_db [private]

5.375.3.10 MHAParser::vfloat_t timoSmooth::win_f0 [private]

5.375.3.11 MHAParser::vfloat_t timoSmooth::alpha_const_vals [private]

5.375.3.12 MHAParser::vfloat_t timoSmooth::alpha_const_limits_hz [private]

5.375.3.13 MHAParser::string_t timoSmooth::noisePow_name [private]

5.375.3.14 MHAParser::parser_t timoSmooth::spp [private]

5.375.3.15 MHAParser::float_t timoSmooth::prior_q [private]

5.375.3.16 MHAParser::float_t timoSmooth::xi_opt_db [private]

5.375.3.17 MHAEvents::patchbay_t<timoSmooth> timoSmooth::patchbay [private]

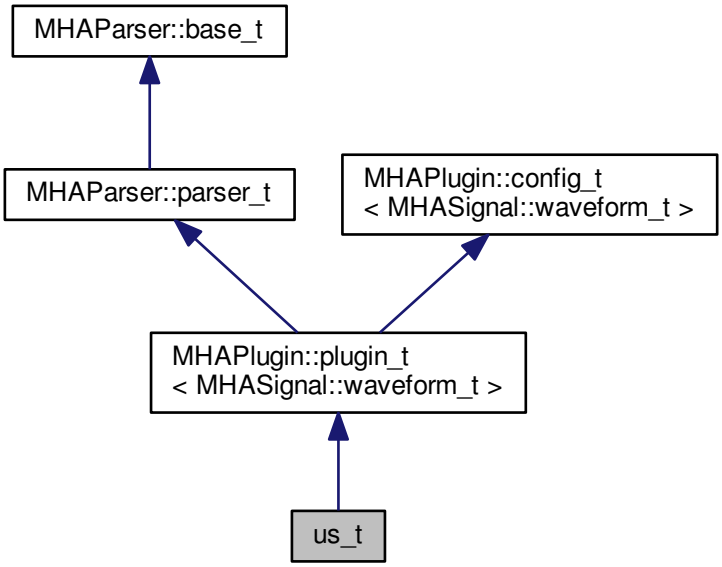
5.375.3.18 bool timoSmooth::prepared [private]

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

- **timosmooth.h**
- **timoSmooth.cpp**

5.376 us_t Class Reference

Inheritance diagram for us_t:



Public Member Functions

- **us_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Attributes

- **MHAParser::int_t ratio**
- **MHAFilter::iir_filter_t antialias**

Additional Inherited Members

5.376.1 Constructor & Destructor Documentation

5.376.1.1 us_t::us_t (

algo_comm_t *iac,*
std::string *,*
std::string)

5.376.2 Member Function Documentation

5.376.2.1 mha_wave_t ∗ us_t::process (

mha_wave_t ∗ *s*)

5.376.2.2 void us_t::prepare (

mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t**< **MHASignal::waveform_t** > (p. 730).

5.376.2.3 void us_t::release (

void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **MHASignal::waveform_t** > (p. 731).

5.376.3 Member Data Documentation

5.376.3.1 MHAParser::int_t us_t::ratio [private]

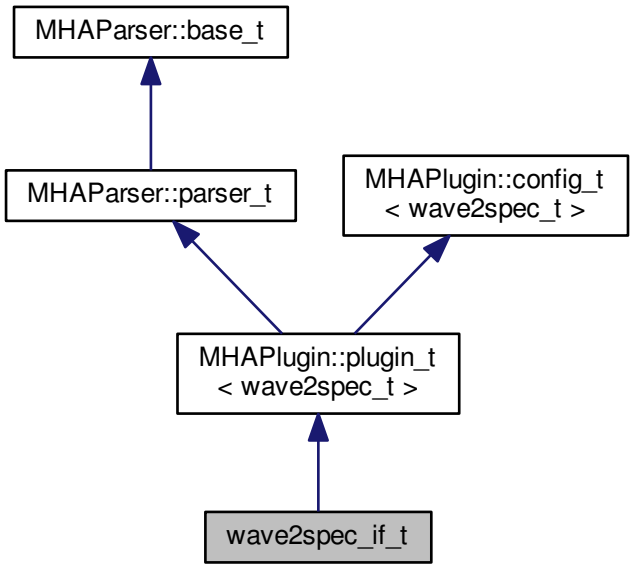
5.376.3.2 MHAFilter::iir_filter_t us_t::antialias [private]

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

- **upsample.cpp**

5.377 wave2spec_if_t Class Reference

Inheritance diagram for wave2spec_if_t:



Public Member Functions

- **wave2spec_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **process** (**mha_wave_t** ∗, **mha_spec_t** ∗∗)
- void **process** (**mha_wave_t** ∗, **mha_wave_t** ∗∗)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **wave2spec_if_t** > **patchbay**
- **MHAParser::int_t nfft**
- **MHAParser::int_t nwnd**
- **MHAParser::float_t wndpos**
- **windowselector_t window_config**
- **MHAParser::bool_t return_wave**
- std::string **algo**

Additional Inherited Members

5.377.1 Constructor & Destructor Documentation

5.377.1.1 wave2spec_if_t::wave2spec_if_t (const algo_comm_t & *iac,* const std::string & *,* const std::string & *ialg*)

5.377.2 Member Function Documentation

5.377.2.1 void wave2spec_if_t::prepare (mhaconfig_t & *t*) [virtual]

Implements **MHAPlugin::plugin_t< wave2spec_t >** (p. 730).

5.377.2.2 void wave2spec_if_t::process (mha_wave_t * *wave_in,* mha_spec_t ** *sout*)

5.377.2.3 void wave2spec_if_t::process (mha_wave_t * *wave_in,* mha_wave_t ** *sout*)

5.377.2.4 void wave2spec_if_t::update ( ) [private]

5.377.3 Member Data Documentation

5.377.3.1 MHAEvents::patchbay_t<wave2spec_if_t> wave2spec_if_t::patchbay [private]

5.377.3.2 MHAParser::int_t wave2spec_if_t::nfft [private]

5.377.3.3 MHAParser::int_t wave2spec_if_t::nwnd [private]

5.377.3.4 MHAParser::float_t wave2spec_if_t::wndpos [private]

5.377.3.5 windowselector_t wave2spec_if_t::window_config [private]

5.377.3.6 MHAParser::bool_t wave2spec_if_t::return_wave [private]

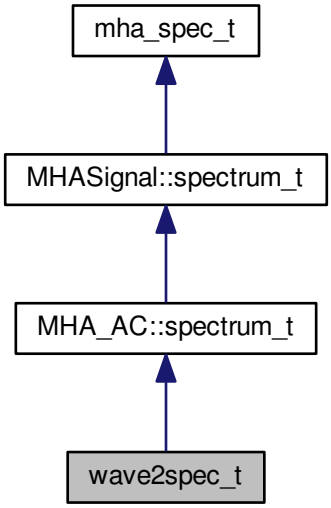
5.377.3.7 std::string wave2spec_if_t::algo [private]

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

- **wave2spec.cpp**

5.378 wave2spec_t Class Reference

Inheritance diagram for wave2spec_t:



Public Member Functions

- **wave2spec_t** (unsigned int nfft, unsigned int nwnd_, unsigned int nwndshift_, unsigned int nch, **mha_real_t** wndpos, const **MHAWindow::base_t** &**window**, **algo_comm_t ac**, std::string algo)
- **mha_spec_t** ∗ **process** (**mha_wave_t** ∗)
- ∼**wave2spec_t** ()

Private Member Functions

- void **calc_pre_wnd** (**MHASignal::waveform_t** &, const **MHASignal::waveform_t** &)

Private Attributes

- unsigned int **nwnd**
- unsigned int **nwndshift**
- **mha_fft_t ft**

  FFT class.
- unsigned int **npad1**

  length of zero padding before window
- unsigned int **npad2**

length of zero padding after window

- **MHAWindow::base_t window**
- **MHASignal::waveform_t calc_in**
- **MHASignal::waveform_t in_buf**
- **MHASignal::spectrum_t spec_in**

  non-interleaved, complex, fftlen

Additional Inherited Members

5.378.1 Constructor & Destructor Documentation

5.378.1.1 wave2spec_t::wave2spec_t (

unsigned int *nfft,*
unsigned int *nwnd_,*
unsigned int *nwndshift_,*
unsigned int *nch,*
mha_real_t *wndpos,*
const **MHAWindow::base_t** & *window,*
algo_comm_t *ac,*
std::string *algo*)

5.378.1.2 wave2spec_t::∼wave2spec_t ( )

5.378.2 Member Function Documentation

5.378.2.1 mha_spec_t ∗ wave2spec_t::process (

mha_wave_t ∗ *wave_in*)

5.378.2.2 void wave2spec_t::calc_pre_wnd (

MHASignal::waveform_t & *dest,*
const **MHASignal::waveform_t** & *src*) [private]

5.378.3 Member Data Documentation

5.378.3.1 unsigned int wave2spec_t::nwnd [private]

5.378.3.2 unsigned int wave2spec_t::nwndshift [private]

5.378.3.3 mha_fft_t wave2spec_t::ft [private]

FFT class.

5.378.3.4 unsigned int wave2spec_t::npad1 [private]

length of zero padding before window

5.378.3.5 unsigned int wave2spec_t::npad2 [private]

length of zero padding after window

5.378.3.6 MHAWindow::base_t wave2spec_t::window [private]

5.378.3.7 MHASignal::waveform_t wave2spec_t::calc_in [private]

5.378.3.8 MHASignal::waveform_t wave2spec_t::in_buf [private]

5.378.3.9 MHASignal::spectrum_t wave2spec_t::spec_in [private]

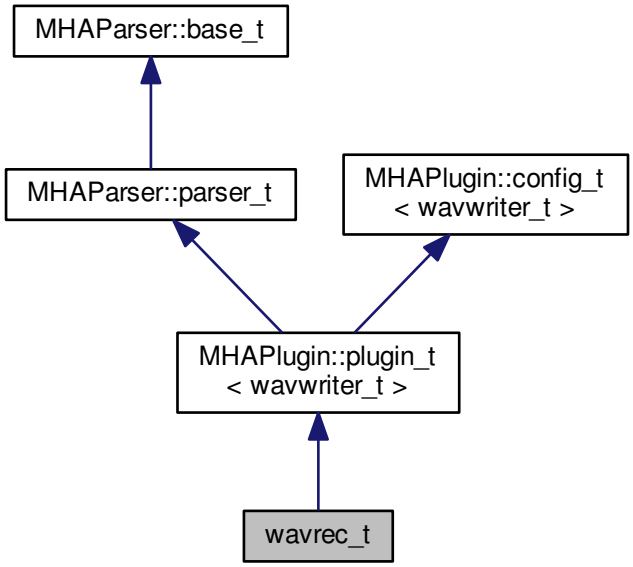
non-interleaved, complex, fftlen

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

- **wave2spec.cpp**

5.379 wavrec_t Class Reference

Inheritance diagram for wavrec_t:



Public Member Functions

- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &cf)
- void **release** ()
- **wavrec_t** (const **algo_comm_t** &iac, const std::string &, const std::string &)

Private Member Functions

- void **start_new_session** ()

Private Attributes

- **MHAParser::bool_t record**
- **MHAParser::int_t fifolen**
- **MHAParser::int_t minwrite**
- **MHAParser::string_t prefix**
- **MHAParser::bool_t use_date**
- **MHAEvents::patchbay_t**< **wavrec_t** > **patchbay**

Additional Inherited Members

5.379.1 Constructor & Destructor Documentation

5.379.1.1 wavrec_t::wavrec_t (

const **algo_comm_t** & *iac,*
const **std::string** & *,*
const **std::string** & *algo_name*)

5.379.2 Member Function Documentation

5.379.2.1 mha_wave_t ∗ wavrec_t::process (

mha_wave_t ∗ *s*)

5.379.2.2 void wavrec_t::prepare (

mhaconfig_t & *cf*) [virtual]

Implements **MHAPlugin::plugin_t**< **wavwriter_t** > (p. 730).

5.379.2.3 void wavrec_t::release (

void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **wavwriter_t** > (p. 731).

5.379.2.4 void wavrec_t::start_new_session ( ) [private]

5.379.3 Member Data Documentation

5.379.3.1 MHAParser::bool_t wavrec_t::record [private]

5.379.3.2 MHAParser::int_t wavrec_t::fifolen [private]

5.379.3.3 MHAParser::int_t wavrec_t::minwrite [private]

5.379.3.4 MHAParser::string_t wavrec_t::prefix [private]

5.379.3.5 MHAParser::bool_t wavrec_t::use_date [private]

5.379.3.6 MHAEvents::patchbay_t<wavrec_t> wavrec_t::patchbay [private]

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

- **wavrec.cpp**

5.380 wavwriter_t Class Reference

Public Member Functions

- **wavwriter_t** (bool active, const **mhaconfig_t** &cf, unsigned int fifosize, unsigned int minwrite, const std::string &prefix, bool use_date)
- ~**wavwriter_t** ()
- void **process** (**mha_wave_t** *)

Private Member Functions

- void **write_thread** ()

Static Private Member Functions

- static void * **write_thread** (void *this_)

Private Attributes

- bool **close_session**
- bool **act_**
- **mhaconfig_t cf_**
- SNDFILE * **sf**
- **mha_fifo_t< mha_real_t > fifo**
- unsigned int **minw_**
- pthread_t **writethread**
- float * **data**

5.380.1 Constructor & Destructor Documentation

5.380.1.1 wavwriter_t::wavwriter_t (
bool *active,*
const **mhaconfig_t** & *cf,*
unsigned int *fifosize,*
unsigned int *minwrite,*
const std::string & *prefix,*
bool *use_date*)

5.380.1.2 wavwriter_t::~wavwriter_t ( )

5.380.2 Member Function Documentation

5.380.2.1 void wavwriter_t::process (
mha_wave_t * *s*)

5.380.2.2 static void* wavwriter_t::write_thread (
void * *this_*) [inline], [static], [private]

5.380.2.3 void wavwriter_t::write_thread ( ) [private]

5.380.3 Member Data Documentation

5.380.3.1 bool wavwriter_t::close_session [private]

5.380.3.2 bool wavwriter_t::act_ [private]

5.380.3.3 mhaconfig_t wavwriter_t::cf_ [private]

5.380.3.4 SNDFILE* wavwriter_t::sf [private]

5.380.3.5 mha_fifo_t<mha_real_t> wavwriter_t::fifo [private]

5.380.3.6 unsigned int wavwriter_t::minw_ [private]

5.380.3.7 pthread_t wavwriter_t::writethread [private]

5.380.3.8 float* wavwriter_t::data [private]

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

- **wavrec.cpp**

5.381 windowselector_t Class Reference

A combination of mha parser variables to describe an overalapadd analysis window.

Public Member Functions

- **windowselector_t** (const std::string &default_type)
  constructor creates the mha parser variables that describe an overlapadd analysis window.
- ∼**windowselector_t** ()
  destructor frees window data that were allocated
- const **MHAWindow::base_t** & **get_window_data** (unsigned length)
  re-computes the window if required.
- void **insert_items** (**MHAParser::parser_t** ∗p)
  insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.

Public Attributes

- **MHAEvents::emitter_t updated**
  A collector event that fires when any of the window parameters managed here is written to.

Private Member Functions

- void **invalidate_window_data** ()
  invalidates any allocated window samples.
- void **update_parser** ()
  invoked when a parser parameter changes.

Private Attributes

- **MHAWindow::base_t** ∗ **wnd**
  Storage for the window data returned by ***get_window_data()*** *(*p. 958*)*
- **MHAParser::kw_t wndtype**
  parser variable for window type
- **MHAParser::float_t wndexp**
  parser variable for window exponent
- **MHAParser::vfloat_t userwnd**
  parser variable for user window samples to use
- **MHAEvents::patchbay_t**< **windowselector_t** > **patchbay**
  patchbay to watch for changes for the parser variables

5.381.1 Detailed Description

A combination of mha parser variables to describe an overalapadd analysis window.

Provides a method to get the window samples as an instance of **MHAWindow::base_t** (p. 835) when needed.

5.381.2 Constructor & Destructor Documentation

5.381.2.1 windowselector_t::windowselector_t (const std::string & *default_type*)

constructor creates the mha parser variables that describe an overlapadd analysis window.

Parameters

default_type	name of the default analysis window type. Must be one of: "rect", "bartlett", "hanning", "hamming", "blackman"

5.381.2.2 windowselector_t::~windowselector_t ( )

destructor frees window data that were allocated

5.381.3 Member Function Documentation

5.381.3.1 const MHAWindow::base_t & windowselector_t::get_window_data (unsigned *length*)

re-computes the window if required.

Parameters

length	the desired window length in samples return the window's samples as a constref to **MHAWindow::base_t** (p. 835) instance. The referenced instance lives until the window parameters are changed, or this **windowselector_t** (p. 957) instance is destroyed.

5.381.3.2 void windowselector_t::insert_items (MHAParser::parser_t * *p*)

insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.

Parameters

p	The configuration parser where to insert the window parameters. E.g. the plugin wave2spec's interface class.

5.381.3.3 void windowselector_t::invalidate_window_data ( ) [private]

invalidates any allocated window samples.

5.381.3.4 void windowselector_t::update_parser ( ) [private]

invoked when a parser parameter changes.

Calls **invalidate_window_data()** (p. 959) and emits the updated event.

5.381.4 Member Data Documentation

5.381.4.1 MHAEvents::emitter_t windowselector_t::updated

A collector event that fires when any of the window parameters managed here is written to.

5.381.4.2 MHAWindow::base_t* windowselector_t::wnd [private]

Storage for the window data returned by **get_window_data()** (p. 958)

5.381.4.3 MHAParser::kw_t windowselector_t::wndtype [private]

parser variable for window type

5.381.4.4 MHAParser::float_t windowselector_t::wndexp [private]

parser variable for window exponent

5.381.4.5 MHAParser::vfloat_t windowselector_t::userwnd [private]

parser variable for user window samples to use

5.381.4.6 MHAEvents::patchbay_t<windowselector_t> windowselector_t::patchbay [private]

patchbay to watch for changes for the parser variables

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

- **windowselector.h**
- **windowselector.cpp**

6 File Documentation

6.1 ac2lsl.cpp File Reference

Classes

- struct **ac2lsl::type_info**
- class **ac2lsl::save_var_base_t**
  Interface for ac to lsl bridge variable.
- class **ac2lsl::save_var_t< T >**
  Implementation for all ac to lsl bridges except complex types.
- class **ac2lsl::save_var_t< mha_complex_t >**
  *Template specialization of the **ac2lsl*** (p. 78) *bridge to take care of complex numbers.*
- class **ac2lsl::cfg_t**
  *Runtime configuration class of the **ac2lsl*** (p. 78) *plugin.*
- class **ac2lsl::ac2lsl_t**
  *Plugin class of **ac2lsl*** (p. 78).

Namespaces

- **ac2lsl**
  *All types for the **ac2lsl*** (p. 78) *plugins live in this namespace.*

Variables

- const std::map< int, type_info > **ac2lsl::types**

6.2 ac2osc.cpp File Reference

Classes

- class **ac2osc_t**
  Plugin class of the ac2osc plugin.

6.3 ac2wave.cpp File Reference

Classes

- class **ac2wave_t**
- class **ac2wave_if_t**

6.4 ac_monitor_type.cpp File Reference

6.5 ac_monitor_type.hh File Reference

Classes

- class **acmon::ac_monitor_t**
  A class for converting AC variables to Parser monitors of correct type.

Namespaces

- **acmon**
  Namespace for displaying ac variables as parser monitors.

6.6 acConcat_wave.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**acConcat_wave::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.6.1 Macro Definition Documentation

6.6.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &acConcat_wave::update_cfg)

6.6.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.7 acConcat_wave.h File Reference

Classes

- class **acConcat_wave_config**
- class **acConcat_wave**

6.8 acmon.cpp File Reference

Classes

- class **acmon::acmon_t**

Namespaces

- **acmon**
  Namespace for displaying ac variables as parser monitors.

6.9 acPooling_wave.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**acPooling_wave::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.9.1 Macro Definition Documentation

6.9.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &acPooling_wave::update_cfg)

6.9.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.10 acPooling_wave.h File Reference

Classes

- class **acPooling_wave_config**
- class **acPooling_wave**

6.11 acsave.cpp File Reference

Classes

- class **acsave::save_var_t**
- class **acsave::cfg_t**
- class **acsave::acsave_t**
- struct **acsave::mat4head_t**

Namespaces

- **acsave**

Macros

- #define **ACSAVE_FMT_TXT** 0
- #define **ACSAVE_SFMT_TXT** "txt"
- #define **ACSAVE_FMT_MAT4** 1
- #define **ACSAVE_SFMT_MAT4** "mat4"
- #define **ACSAVE_FMT_M** 2
- #define **ACSAVE_SFMT_M** "m"

6.11.1 Macro Definition Documentation

6.11.1.1 #define ACSAVE_FMT_TXT 0

6.11.1.2 #define ACSAVE_SFMT_TXT "txt"

6.11.1.3 #define ACSAVE_FMT_MAT4 1

6.11.1.4 #define ACSAVE_SFMT_MAT4 "mat4"

6.11.1.5 #define ACSAVE_FMT_M 2

6.11.1.6 #define ACSAVE_SFMT_M "m"

6.12 acSteer.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**acSteer**↩ **::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.12.1 Macro Definition Documentation

6.12.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &acSteer::update_cfg)

6.12.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.13 acSteer.h File Reference

Classes

- class **acSteer_config**
- class **acSteer**

6.14 acTransform_wave.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**acTransform_wave::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.14.1 Macro Definition Documentation

6.14.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &acTransform_wave::update_cfg)

6.14.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.15 acTransform_wave.h File Reference

Classes

- class **acTransform_wave_config**
- class **acTransform_wave**

6.16 addsndfile.cpp File Reference

Classes

- class **addsndfile::waveform_proxy_t**

  *Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t*** (p. 184).
- class **addsndfile::resampled_soundfile_t**

  Reads sound from file and resamples it if necessary and wanted.
- class **addsndfile::sndfile_t**
- class **addsndfile::level_adapt_t**
- class **addsndfile::addsndfile_if_t**

Namespaces

- **addsndfile**

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

Typedefs

- typedef **MHAPlugin::config_t**< level_adapt_t > **addsndfile::level_adaptor**
- typedef **MHAPlugin::plugin_t**< sndfile_t > **addsndfile::wave_reader**

Enumerations

Functions

- static unsigned **addsndfile::resampled_num_frames** (unsigned num_source_frames, float source_rate, float target_rate, addsndfile_resampling_mode_t resampling_mode)

6.16.1 Macro Definition Documentation

6.16.1.1 #define DEBUG(*x*) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

6.17 adm.cpp File Reference

Classes

- class **adm_rtconfig_t**
- class **adm_if_t**

Functions

- **MHASignal::waveform_t** * **adm_fir_lp** (unsigned int fs, unsigned f_pass, unsigned int f_stop, unsigned int order)
- **MHASignal::waveform_t** * **adm_fir_decomb** (unsigned int fs, float dist_m, unsigned int order)

6.17.1 Function Documentation

6.17.1.1 MHASignal::waveform_t* adm_fir_lp (unsigned int *fs,* unsigned *f_pass,* unsigned int *f_stop,* unsigned int *order*)

6.17.1.2 MHASignal::waveform_t* adm_fir_decomb (unsigned int *fs,* float *dist_m,* unsigned int *order*)

6.18 adm.hh File Reference

Classes

- class **ADM::Linearphase_FIR**< **F** >

  An efficient linear-phase fir filter implementation.
- class **ADM::Delay**< **F** >

  A delay-line class which can also do subsample-delays for a limited frequency range below fs/4.
- class **ADM::ADM**< **F** >

  Adaptive differential microphone, working for speech frequency range.

Namespaces

- **ADM**

Functions

- static double **ADM::subsampledelay_coeff** (double samples, double f_design, double fs=1.0)

  compute IIR coefficient for subsample delay

Variables

- const double **ADM::PI** = 3.14159265358979312
- const double **ADM::C** = 340
- const double **ADM::DELAY_FREQ** = 2000
- const double **ADM::START_BETA** = 0.5

6.19 altplugs.cpp File Reference

Classes

- class **mhaplug_cfg_t**
- class **altplugs_t**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.19.1 Macro Definition Documentation

6.19.1.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

6.20 analysemhaplugin.cpp File Reference

Functions

- std::string **strdom** (**mha_domain_t** d)
- void **print_ac** (**MHAKernel::algo_comm_class_t** &ac, std::string txt)
- int **main** (int argc, char ∗∗argv)

6.20.1 Function Documentation

6.20.1.1 std::string strdom (mha_domain_t *d*)

6.20.1.2 void print_ac (MHAKernel::algo_comm_class_t & *ac,* std::string *txt*)

6.20.1.3 int main (int *argc,* char ∗∗ *argv*)

6.21 analysispath.cpp File Reference

Classes

- class **analysepath_t**
- class **plug_t**
- class **analysispath_if_t**

Functions

- static void * **thread_start** (void *instance)

6.21.1 Function Documentation

6.21.1.1 static void* thread_start (void * *instance*) [static]

6.22 auditory_profile.cpp File Reference

6.23 auditory_profile.h File Reference

Classes

- class **AuditoryProfile::fmap_t**
  A class to store frequency dependent data (e.g., HTL and UCL).
- class **AuditoryProfile::profile_t**
  The Auditory Profile class.
- class **AuditoryProfile::profile_t::ear_t**
  Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.
- class **AuditoryProfile::parser_t**
  Class to make the auditory profile accessible through the parser interface.
- class **AuditoryProfile::parser_t::fmap_t**
- class **AuditoryProfile::parser_t::ear_t**

Namespaces

- **AuditoryProfile**
  Namespace for classes and functions around the auditory profile (e.g., audiogram handling)

6.24 browsemhaplugins.cpp File Reference

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x<<"="<<x << std::endl

Functions

- int **main** (int argc, char **argv)

6.24.1 Macro Definition Documentation

6.24.1.1 #define DEBUG(*x*) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x<<"="<<x << std::endl

6.24.2 Function Documentation

6.24.2.1 int main (int *argc,* char ** *argv*)

6.25 coherence.cpp File Reference

Classes

- class **coherence::vars_t**
- class **coherence::cohflt_t**
- class **coherence::cohflt_if_t**

Namespaces

- **coherence**

Functions

- void **coherence::getcipd** (**mha_complex_t** &c, **mha_real_t** &a, const **mha_complex_t** &xl, const **mha_complex_t** &xr)

6.26 combinechannels.cpp File Reference

Classes

- class **combc_t**
- class **combc_if_t**

6.27 compiler_id.cpp File Reference

6.28 compiler_id.hh File Reference

Macros

- #define **COMPILER_ID_VENDOR** "gcc"
- #define **COMPILER_ID_MAJOR** __GNUC__
- #define **COMPILER_ID_MINOR** __GNUC_MINOR__
- #define **COMPILER_ID_PATCH** __GNUC_PATCHLEVEL__
- #define **COMPILER_ID_VERSION_HELPER2**(x, y, z) #x "." #y "." #z
- #define **COMPILER_ID_VERSION_HELPER1**(x, y, z) **COMPILER_ID_VERSION_HE↩ LPER2**(x,y,z)
- #define **COMPILER_ID_VERSION**
- #define **COMPILER_ID COMPILER_ID_VENDOR** "-" **COMPILER_ID_VERSION** "-" C↩ OMPILER_ID_STANDARD

6.28.1 Macro Definition Documentation

6.28.1.1 #define COMPILER_ID_VENDOR "gcc"

6.28.1.2 #define COMPILER_ID_MAJOR __GNUC__

6.28.1.3 #define COMPILER_ID_MINOR __GNUC_MINOR__

6.28.1.4 #define COMPILER_ID_PATCH __GNUC_PATCHLEVEL__

6.28.1.5 #define COMPILER_ID_VERSION_HELPER2(*x,* *y,* *z*) #x "." #y "." #z

6.28.1.6 #define COMPILER_ID_VERSION_HELPER1(*x,* *y,* *z*) COMPILER_ID_VERSION_HELPER2(x,y,z)

6.28.1.7 #define COMPILER_ID_VERSION

6.28.1.8 #define COMPILER_ID COMPILER_ID_VENDOR "-" COMPILER_ID_VERSION "-" COMPILER_ID_STANDARD

6.29 complex_filter.cpp File Reference

6.30 complex_filter.h File Reference

Classes

- class **MHAFilter::complex_bandpass_t**
  Complex bandpass filter.
- class **MHAFilter::gamma_flt_t**
  Class for gammatone filter.
- class **MHAFilter::thirdoctave_analyzer_t**

Namespaces

- **MHAFilter**
  Namespace for IIR and FIR filter classes.

6.31 cpuload.cpp File Reference

Classes

- class **cpuload_t**

6.32 db.cpp File Reference

Classes

- class **db_t**
- class **db_if_t**

6.33 dc.cpp File Reference

Classes

- class **dc::wb_inhib_cfg_t**
- class **dc::wideband_inhib_vars_t**
- class **dc::dc_vars_t**
- class **dc::dc_vars_validator_t**
- class **dc::dc_t**
- class **dc::dc_if_t**

Namespaces

- **dc**

Macros

- #define **DUPVEC**(x) v.x.data = **MHASignal::dupvec_chk**(v.x.data,s)

Functions

- unsigned int **dc::get_audiochannels** (unsigned int totalchannels, std::string acname, **algo_comm_t** ac)

6.33.1 Macro Definition Documentation

6.33.1.1 #define DUPVEC(*x*) v.x.data = MHASignal::dupvec_chk(v.x.data,s)

6.34 dc_afterburn.cpp File Reference

Namespaces

- **DynComp**
  dynamic compression related classes and functions

Functions

- float **mylogf** (float x)

6.34.1 Function Documentation

6.34.1.1 float mylogf (float *x*)

6.35 dc_afterburn.h File Reference

Classes

- class **DynComp::dc_afterburn_vars_t**
  *Variables for **dc_afterburn_t** (p. 294) class.*
- class **DynComp::dc_afterburn_rt_t**
  Real-time class for after burn effect.
- class **DynComp::dc_afterburn_t**
  Afterburn class, to be defined as a member of compressors.

Namespaces

- **DynComp**
  dynamic compression related classes and functions

6.36 dc_simple.cpp File Reference

Classes

- class **dc_simple::dc_vars_t**
- class **dc_simple::dc_vars_validator_t**
- class **dc_simple::level_smoother_t**
- class **dc_simple::dc_t**
- class **dc_simple::dc_t::line_t**
- class **dc_simple::dc_if_t**

Namespaces

- **dc_simple**

Typedefs

- typedef **MHAPlugin::plugin_t**< dc_t > **dc_simple::DC**
- typedef **MHAPlugin::config_t**< level_smoother_t > **dc_simple::LEVEL**

Functions

- void **dc_simple::test_fail** (const std::vector< float > &v, unsigned int s, const std::string &name)
- std::vector< float > **dc_simple::force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)
- **mha_real_t dc_simple::not_zero** (**mha_real_t** x, const std::string &comment="")

6.37 delay.cpp File Reference

Classes

- class **delay::interface_t**

Namespaces

- **delay**

6.38 delaysum.cpp File Reference

Classes

- class **delaysum::delaysum_t**
  Runtime configuration of the delaysum plugin.
- class **delaysum::delaysum_if_t**
  Interface class for the delaysum plugin.

Namespaces

- **delaysum**
  This namespace contains the delaysum plugin.

6.39 doasvm_classification.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**doasvm_↩ classification::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.39.1 Macro Definition Documentation

6.39.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &doasvm_classification↩ ::update_cfg)

6.39.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.40 doasvm_classification.h File Reference

Classes

- class **doasvm_classification_config**
- class **doasvm_classification**

6.41 doasvm_feature_extraction.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**doasvm_↩ feature_extraction::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.41.1 Macro Definition Documentation

6.41.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &doasvm_feature_↩ extraction::update_cfg)

6.41.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.42 doasvm_feature_extraction.h File Reference

Classes

- class **doasvm_feature_extraction_config**
- class **doasvm_feature_extraction**

6.43 doc_appendix.h File Reference

6.44 doc_examples.h File Reference

6.45 doc_frameworks.h File Reference

6.46 doc_general.h File Reference

6.47 doc_kernel.h File Reference

6.48 doc_matlab.h File Reference

6.49 doc_mhamain.h File Reference

6.50 doc_parser.h File Reference

6.51 doc_plugif.cpp File Reference

6.52 doc_plugins.h File Reference

6.53 doc_system.h File Reference

6.54 doc_toolbox.h File Reference

6.55 downsample.cpp File Reference

Classes

- class **ds_t**

6.56 droptect.cpp File Reference

Classes

- class **droptect_t**
  Detect dropouts in a signal with a constant spectrum.

6.57 example1.cpp File Reference

Classes

- class **example1_t**
  This C++ class implements the simplest example plugin for the step-by-step tutorial.

6.58 example2.cpp File Reference

Classes

- class **example2_t**
  This C++ class implements the second example plugin for the step-by-step tutorial.

6.59 example3.cpp File Reference

Classes

- class **example3_t**
  A Plugin class using the openMHA Event mechanism.

6.60 example4.cpp File Reference

Classes

- class **example4_t**
  A Plugin class using the spectral signal.

6.61 example5.cpp File Reference

Classes

- class **example5_t**
- class **plugin_interface_t**

Macros

- #define **__declspec**(p)

6.61.1 Macro Definition Documentation

6.61.1.1 #define __declspec(*p*)

6.62 example6.cpp File Reference

Classes

- class **cfg_t**
- class **example6_t**

Macros

- #define **__declspec**(p)

6.62.1 Macro Definition Documentation

6.62.1.1 #define __declspec(*p*)

6.63 fader_spec.cpp File Reference

Classes

- class **spec_fader_t**
- class **fader_if_t**

6.64 fader_wave.cpp File Reference

Classes

- class **fader_wave::level_adapt_t**
- class **fader_wave::fader_wave_if_t**

Namespaces

- **fader_wave**

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

Typedefs

- typedef **MHAPlugin::plugin_t**< level_adapt_t > **fader_wave::level_adaptor**

6.64.1 Macro Definition Documentation

6.64.1.1 #define DEBUG(*x*) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

6.65 fftfbpow.cpp File Reference

Classes

- class **fftfbpow::fftfbpow_t**
  Run time configuration for the fftfbpow plugin.
- class **fftfbpow::fftfbpow_interface_t**
  Interface class for fftfbpow plugin.

Namespaces

- **fftfbpow**
  Namespace for the fftfbpow plugin.

6.66 fftfilterbank.cpp File Reference

Classes

- class **fftfilterbank::fftfb_plug_t**
- class **fftfilterbank::fftfb_interface_t**

Namespaces

- **fftfilterbank**

6.67 fshift.cpp File Reference

6.68 fshift.hh File Reference

Classes

- class **fshift::fshift_config_t**
  fshift runtime config class
- class **fshift::fshift_t**
  fshift plugin interface class

Namespaces

- **fshift**
  All types for the fshift plugin live in this namespace.

Functions

- int **fshift::fft_find_bin** (**mha_real_t** frequency, unsigned fftlen, **mha_real_t** srate)
  Finds bin number of FFT bin nearest to the given frequency.

6.69 fshift_hilbert.cpp File Reference

Classes

- class **fshift_hilbert::hilbert_shifter_t**
- class **fshift_hilbert::frequency_translator_t**

Namespaces

- **fshift_hilbert**
  All types for the hilbert frequency shifter live in this namespace.

6.70 gain.cpp File Reference

Classes

- class **gain::scaler_t**
- class **gain::gain_if_t**

Namespaces

- **gain**

6.71 gaintable.cpp File Reference

Functions

- std::vector< **mha_real_t** > **convert_f2logf** (const std::vector< **mha_real_t** > &vF)
- bool **isempty** (const std::vector< std::vector< **mha_real_t** > > &arg)

6.71.1 Function Documentation

6.71.1.1 std::vector<mha_real_t> convert_f2logf (const std::vector< mha_real_t > & *vF*)

6.71.1.2 bool isempty (const std::vector< std::vector< mha_real_t > > & *arg*)

6.72 gaintable.h File Reference

Classes

- class **DynComp::gaintable_t**
  Gain table class.

Namespaces

- **DynComp**
  dynamic compression related classes and functions

Functions

- **mha_real_t DynComp::interp1** (const std::vector< **mha_real_t** > &vX, const std::vector< **mha_real_t** > &vY, **mha_real_t** X)
  One-dimensional linear interpolation.
- **mha_real_t DynComp::interp2** (const std::vector< **mha_real_t** > &vX, const std::vector< **mha_real_t** > &vY, const std::vector< std::vector< **mha_real_t** > > &mZ, **mha_real_t** X, **mha_real_t** Y)
  Linear interpolation in a two-dimensional field.

6.73 generatemhaplugindoc.cpp File Reference

Classes

- class **latex_doc_t**

  Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUME↩ NTATION macro.

Functions

- std::string **conv2latex** (std::string s, bool iscolored=false)

  Escapes various character sequences in texts not intended to be processed by LaTeX for processing by LaTeX.
- static void **print_plugin_references** (const **std::set**< std::string > &all_categories, std↩ ::map< std::string, std::vector< std::string > > main_category_plugins, std::map< std↩ ::string, std::vector< std::string > > additional_category_plugins, std::ofstream &ofile, const std::string &category_macro)

  Function prints an overview of all categories and their associated plugins into the document.
- std::vector< std::string > **create_latex_doc** (std::map< std::string, std::string > &doc, const std::string &plugname, const std::string &plugin_macro)

  Loads the plugin, creates the latex documentation for the plugin, and adds the latex documentation for this plugin to the plugin's main category entry in doc.
- int **main** (int argc, char **argv)

6.73.1 Function Documentation

6.73.1.1 std::string conv2latex (std::string *s,* bool *iscolored* = `false`)

Escapes various character sequences in texts not intended to be processed by LaTeX for processing by LaTeX.

Focus is on correct display of symbols contained in these texts. E.g. the help texts of MHA configuration variables can be processed by this function. The contents of the MHAPLUGIN↩ _DOCUMENTATION is already in LaTeX format and should not be processed by this function.

Returns

A copy of s with various symbols escaped for LaTeX processing

Parameters

s	Text not ready for LaTeX
iscolored	if true, the complete returned text is surrounded with "\\color{monitorcolor}{" and "}"

6.73.1.2 static void print_plugin_references (

const std::set< std::string > & *all_categories,*
std::map< std::string, std::vector< std::string > > *main_category_plugins,*
std::map< std::string, std::vector< std::string > > *additional_category_plugins,*
std::ofstream & *ofile,*
const std::string & *category_macro*) [static]

Function prints an overview of all categories and their associated plugins into the document.

Parameters

all_categories	A sorted container with all category names
main_category_plugins	map of main categories to plugin names
additional_category_plugins	map of tags to plugin names
ofile	Latex document is produced by writing output to this stream

6.73.1.3 std::vector<std::string> create_latex_doc (

std::map< std::string, std::string > & *doc,*
const std::string & *plugname,*
const std::string & *plugin_macro*)

Loads the plugin, creates the latex documentation for the plugin, and adds the latex documentation for this plugin to the plugin's main category entry in doc.

Returns

the vector of all categories.

Parameters

doc	map of main categories to a string containint the documentation of all plugins in that categories. The documentation of the current plugin will be appended to the existing documentation of its main category. Will be created if non-existant.
plugname	Name of the MHA plugin to process
plugin_macro	name of the LaTeX section macro that documents a single plugin (e.g. "section", "subsection", "subsubsection", ...)

6.73.1.4 int main (

int *argc,*
char ** *argv*)

6.74 gtfb_analyzer.cpp File Reference

Gammatone Filterbank Analyzer Plugin.

Classes

- struct **gtfb_analyzer::gtfb_analyzer_cfg_t**
  Configuration for Gammatone Filterbank Analyzer.
- class **gtfb_analyzer::gtfb_analyzer_t**
  Gammatone Filterbank Analyzer Plugin.

Namespaces

- **gtfb_analyzer**

Functions

- static const **mha_complex_t** & **filter_complex** (const **mha_complex_t** &input, const **mha_complex_t** &coeff, **mha_complex_t** ∗states, unsigned orders)
  Filters a complex input sample with the given filter coefficient.
- static const **mha_complex_t** & **filter_real** (**mha_real_t** input, **mha_complex_t** &tmp↩ _complex, const **mha_complex_t** &coeff, **mha_complex_t** ∗states, unsigned orders, const **mha_complex_t** &normphase)
  Filters a real input sample with the given filter coefficient and applies the given normalization with phase correction.

6.74.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

6.74.2 Function Documentation

6.74.2.1 static const mha_complex_t& filter_complex (const mha_complex_t & *input,* const mha_complex_t & *coeff,* mha_complex_t ∗ *states,* unsigned *orders*) [inline], [static]

Filters a complex input sample with the given filter coefficient.

No normalization takes place. The implementation is tail-recursive and to exploit compiler optimization.

Parameters

input	The complex input sample
coeff	The complex filter coefficient
states	Pointer to the array of complex filter states.
orders	The filter order

Returns

A const ref to the filtered sample

6.74.2.2 static const mha_complex_t& filter_real (mha_real_t *input*, mha_complex_t & *tmp_complex*, const mha_complex_t & *coeff*, mha_complex_t * *states*, unsigned *orders*, const mha_complex_t & *normphase*) [inline], [static]

Filters a real input sample with the given filter coefficient and applies the given normalization with phase correction.

Parameters

input	The real input sample
tmp_complex	A reference to a **mha_complex_t** (p. 431) used for intermediate results. No assumptions should be made about the state of tmp_complex after the return of filter_real. This is an optimization to reduce the number of dtor/ctor calls of **mha_complex_t** (p. 431)
coeff	The complex filter coefficient
states	Pointer to the array of complex filter states.
orders	The filter order
normphase	Normalization coefficient including the phase correction

Returns

A const ref to the filtered sample

6.75 hann.cpp File Reference

Macros

- #define **PI** 3.14159265358979323846

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

6.75.1 Macro Definition Documentation

6.75.1.1 #define PI 3.14159265358979323846

6.75.2 Function Documentation

6.75.2.1 float* hannf (const unsigned int *N*)

6.75.2.2 double* hann (const unsigned int *N*)

6.76 hann.h File Reference

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

6.76.1 Function Documentation

6.76.1.1 float* hannf (const unsigned int *N*)

6.76.1.2 double* hann (const unsigned int *N*)

6.77 identity.cpp File Reference

Classes

- class **identity_t**

6.78 ifftshift.cpp File Reference

Functions

- void **ifftshift** (**mha_wave_t** *spec)

6.78.1 Function Documentation

6.78.1.1 void ifftshift (mha_wave_t * *spec*)

6.79 ifftshift.h File Reference

Functions

- void **ifftshift** (**mha_wave_t** *spec)

6.79.1 Function Documentation

6.79.1.1 void ifftshift (mha_wave_t * *spec*)

6.80 iirfilter.cpp File Reference

Classes

- class **iirfilter_t**

6.81 lpc.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**lpc::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

Functions

- void **Levinson2** (unsigned int P, const std::vector< **mha_real_t** > &R, std::vector< **mha_real_t** > &A)

6.81.1 Macro Definition Documentation

6.81.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &lpc::update_cfg)

6.81.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.81.2 Function Documentation

6.81.2.1 void Levinson2 (unsigned int *P,* const std::vector< mha_real_t > & *R,* std::vector< mha_real_t > & *A*)

6.82 lpc.h File Reference

Classes

- class **lpc_config**
- class **lpc**

6.83 lpc_bl_predictor.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**lpc_bl_predictor::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.83.1 Macro Definition Documentation

6.83.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &lpc_bl_predictor::update_cfg)

6.83.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.84 lpc_bl_predictor.h File Reference

Classes

- class **lpc_bl_predictor_config**
- class **lpc_bl_predictor**

Macros

- #define **EPSILON** 1e-10

6.84.1 Macro Definition Documentation

6.84.1.1 #define EPSILON 1e-10

6.85 lpc_burg-lattice.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**lpc_burglattice**↩ **::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.85.1 Macro Definition Documentation

6.85.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &lpc_burglattice::update↩ _cfg)

6.85.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.86 lpc_burg-lattice.h File Reference

Classes

- class **lpc_burglattice_config**
- class **lpc_burglattice**

Macros

- #define **EPSILON** 1e-10

6.86.1 Macro Definition Documentation

6.86.1.1 #define EPSILON 1e-10

6.87 matrixmixer.cpp File Reference

Classes

- class **matrixmixer::cfg_t**
- class **matrixmixer::matmix_t**

Namespaces

- **matrixmixer**

6.88 mha.cpp File Reference

Functions

- int **mhamain** (int argc, char ∗argv[ ])
- int **main** (int argc, char ∗argv[ ])

6.88.1 Function Documentation

6.88.1.1 int mhamain (int *argc,* char ∗ *argv[ ]*)

6.88.1.2 int main (int *argc,* char ∗ *argv[ ]*)

6.89 mha.hh File Reference

common types for MHA kernel, MHA framework applications and external plugins

Classes

- struct **mha_complex_t**

  Type for complex floating point values.
- struct **mha_complex_test_array_t**

  Several places in MHA rely on the fact that you can cast an array of ***mha_complex_t*** *(*p. 431*) c[] to an array of mha_real_t r[] with r[0] == c[0].re r[1] == c[0].im r[2] == c[1].re ...*
- struct **mha_real_test_array_t**
- struct **mha_direction_t**

  Channel source direction structure.
- struct **mha_channel_info_t**

  Channel information structure.
- struct **mha_wave_t**

  Waveform signal structure.
- struct **mha_spec_t**

  Spectrum signal structure.
- struct **mha_audio_descriptor_t**

  Description of an audio fragment (planned as a replacement of ***mhaconfig_t*** *(*p. 504*)).*

- struct **mha_audio_t**

  *An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 496) and **mha_spec_t** (p. 465)).*
- struct **mhaconfig_t**

  MHA prepare configuration structure.
- struct **comm_var_t**

  Algorithm communication variable structure.
- struct **algo_comm_t**

  A reference handle for algorithm communication variables.

Macros

- #define **MHA_CALLBACK_TEST**(x)

  Test macro to compare function type definition and declaration.
- #define **MHA_CALLBACK_TEST_PREFIX**(prefix, x)
- #define **MHA_XSTRF**(x) **MHA_STRF**(x)
- #define **MHA_STRF**(x) #x
- #define **MHA_VERSION_MAJOR** 4

  Major version number of MHA.
- #define **MHA_VERSION_MINOR** 8

  Minor version number of MHA.
- #define **MHA_VERSION_RELEASE** 0

  Release number of MHA.
- #define **MHA_VERSION_BUILD** 0

  Build number of MHA (currently unused)
- #define **MHA_STRUCT_SIZEMATCH** (unsigned int)((sizeof(**mha_real_t**)==4)+2∗(sizeof(**mha_complex_t**)==8)+4∗(sizeof(**mha_wave_t**)==8+2∗sizeof(void∗))+8∗(sizeof(**mha_spec_t**)==8+2∗sizeof(void∗))+16∗(sizeof(**mhaconfig_t**)==24))

  Test number for structure sizes.
- #define **MHA_VERSION** (unsigned int)((**MHA_STRUCT_SIZEMATCH** | (**MHA_VERSION_RELEASE** << 8) | (**MHA_VERSION_MINOR** << 16) | (**MHA_VERSION_MAJOR** << 24)))

  Full version number of MHA kernel.
- #define **MHA_VERSION_STRING MHA_XSTRF**(**MHA_VERSION_MAJOR**) "." **MHA_XSTRF**(**MHA_VERSION_MINOR**)

  Version string of MHA kernel (major.minor)
- #define **MHA_RELEASE_VERSION_STRING MHA_XSTRF**(**MHA_VERSION_MAJOR**) "." **MHA_XSTRF**(**MHA_VERSION_MINOR**) "." **MHA_XSTRF**(**MHA_VERSION_RELEASE**)

  Version string of MHA kernel (major.minor.release)
- #define **MHA_WAVEFORM** 0
- #define **MHA_SPECTRUM** 1
- #define **MHA_DOMAIN_MAX** 2
- #define **MHA_DOMAIN_UNKNOWN MHA_DOMAIN_MAX**
- #define **MHA_AC_UNKNOWN** 0
- #define **MHA_AC_CHAR** 1

- #define **MHA_AC_INT** 2
- #define **MHA_AC_MHAREAL** 3
- #define **MHA_AC_FLOAT** 4
- #define **MHA_AC_DOUBLE** 5
- #define **MHA_AC_MHACOMPLEX** 6
- #define **MHA_AC_VEC_FLOAT** 51
- #define **MHA_AC_USER** 1000

Typedefs

- typedef unsigned int **mha_domain_t**
- typedef float **mha_real_t**

  openMHA type for real numbers
- typedef void * **mha_fft_t**

  Handle for an FFT object.
- typedef struct **algo_comm_t algo_comm_t**
- typedef unsigned int(* **MHAGetVersion_t**) (void)
- typedef int(* **MHAInit_t**) (**algo_comm_t** algo_comm, const char *chain, const char *algo, void **h)
- typedef int(* **MHAPrepare_t**) (void *h, **mhaconfig_t** *cfg)
- typedef int(* **MHARelease_t**) (void *h)
- typedef void(* **MHADestroy_t**) (void *h)
- typedef int(* **MHASet_t**) (void *h, const char *cmd, char *retval, unsigned int len)
- typedef const char *(* **MHAStrError_t**) (void *h, int err)
- typedef int(* **MHAProc_wave2wave_t**) (void *h, **mha_wave_t** *sIn, **mha_wave_t** **s↩Out)
- typedef int(* **MHAProc_wave2spec_t**) (void *h, **mha_wave_t** *sIn, **mha_spec_t** **s↩Out)
- typedef int(* **MHAProc_spec2wave_t**) (void *h, **mha_spec_t** *sIn, **mha_wave_t** **s↩Out)
- typedef int(* **MHAProc_spec2spec_t**) (void *h, **mha_spec_t** *sIn, **mha_spec_t** **sOut)
- typedef const char *(* **MHAPluginDocumentation_t**) (void)
- typedef const char *(* **MHAPluginCategory_t**) (void)

6.89.1 Detailed Description

common types for MHA kernel, MHA framework applications and external plugins

6.89.2 Macro Definition Documentation

6.89.2.1 #define MHA_CALLBACK_TEST(*x*)

Test macro to compare function type definition and declaration.

6.89.2.2 #define MHA_CALLBACK_TEST_PREFIX(*prefix,* *x*)

6.89.2.3 #define MHA_XSTRF(*x*) MHA_STRF(x)

6.89.2.4 #define MHA_STRF(*x*) #x

6.89.2.5 #define MHA_VERSION_MAJOR 4

Major version number of MHA.

6.89.2.6 #define MHA_VERSION_MINOR 8

Minor version number of MHA.

6.89.2.7 #define MHA_VERSION_RELEASE 0

Release number of MHA.

6.89.2.8 #define MHA_VERSION_BUILD 0

Build number of MHA (currently unused)

6.89.2.9 #define MHA_STRUCT_SIZEMATCH (unsigned int)((sizeof(mha_real_t)==4)+2∗(sizeof(mha_complex_t)==8)+4∗(sizeof(mha_wave_t)==8+2∗sizeof(void∗))+8∗(sizeof(mha_spec_t)==8+2∗sizeof(void∗))+16∗(sizeof(mhaconfig_t)==24))

Test number for structure sizes.

6.89.2.10 #define MHA_VERSION (unsigned int)((MHA_STRUCT_SIZEMATCH | (MHA_VERSION_RELEASE << 8) | (MHA_VERSION_MINOR << 16) | (MHA_VERSION_MAJOR << 24)))

Full version number of MHA kernel.

6.89.2.11 #define MHA_VERSION_STRING MHA_XSTRF(MHA_VERSION_MAJOR) "." MHA_XSTRF(MHA_VERSION_MINOR)

Version string of MHA kernel (major.minor)

6.89.2.12 #define MHA_RELEASE_VERSION_STRING MHA_XSTRF(MHA_VE↩ RSION_MAJOR) "." MHA_XSTRF(MHA_VERSION_MINOR) "." MHA_XSTRF(MHA_VERSION_RELEASE)

Version string of MHA kernel (major.minor.release)

6.89.2.13 #define MHA_WAVEFORM 0

6.89.2.14 #define MHA_SPECTRUM 1

6.89.2.15 #define MHA_DOMAIN_MAX 2

6.89.2.16 #define MHA_DOMAIN_UNKNOWN MHA_DOMAIN_MAX

6.89.2.17 #define MHA_AC_UNKNOWN 0

6.89.2.18 #define MHA_AC_CHAR 1

6.89.2.19 #define MHA_AC_INT 2

6.89.2.20 #define MHA_AC_MHAREAL 3

6.89.2.21 #define MHA_AC_FLOAT 4

6.89.2.22 #define MHA_AC_DOUBLE 5

6.89.2.23 #define MHA_AC_MHACOMPLEX 6

6.89.2.24 #define MHA_AC_VEC_FLOAT 51

6.89.2.25 #define MHA_AC_USER 1000

6.89.3 Typedef Documentation

6.89.3.1 typedef unsigned int mha_domain_t

6.89.3.2 typedef struct algo_comm_t algo_comm_t

6.89.3.3 typedef unsigned int(∗ MHAGetVersion_t) (void)

6.89.3.4 typedef int(∗ MHAInit_t) (algo_comm_t algo_comm, const char ∗chain, const char ∗algo, void ∗∗h)

6.89.3.5 typedef int(∗ MHAPrepare_t) (void ∗h, mhaconfig_t ∗cfg)

6.89.3.6 typedef int(* MHARelease_t) (void *h)

6.89.3.7 typedef void(* MHADestroy_t) (void *h)

6.89.3.8 typedef int(* MHASet_t) (void *h, const char *cmd, char *retval, unsigned int len)

6.89.3.9 typedef const char*(* MHAStrError_t) (void *h, int err)

6.89.3.10 typedef int(* MHAProc_wave2wave_t) (void *h, mha_wave_t *sIn, mha_wave_t **sOut)

6.89.3.11 typedef int(* MHAProc_wave2spec_t) (void *h, mha_wave_t *sIn, mha_spec_t **sOut)

6.89.3.12 typedef int(* MHAProc_spec2wave_t) (void *h, mha_spec_t *sIn, mha_wave_t **sOut)

6.89.3.13 typedef int(* MHAProc_spec2spec_t) (void *h, mha_spec_t *sIn, mha_spec_t **sOut)

6.89.3.14 typedef const char*(* MHAPluginDocumentation_t) (void)

6.89.3.15 typedef const char*(* MHAPluginCategory_t) (void)

6.90 mha_algo_comm.cpp File Reference

Macros

- #define **AC_SUCCESS** 0
- #define **AC_INVALID_HANDLE** -1
- #define **AC_INVALID_NAME** -2
- #define **AC_STRING_TRUNCATED** -3
- #define **AC_INVALID_OUTPTR** -4
- #define **AC_TYPE_MISMATCH** -5
- #define **AC_DIM_MISMATCH** -6

Variables

- **algo_comm_t algo_comm_default**

6.90.1 Macro Definition Documentation

6.90.1.1 #define AC_SUCCESS 0

6.90.1.2 #define AC_INVALID_HANDLE -1

6.90.1.3 #define AC_INVALID_NAME -2

6.90.1.4 #define AC_STRING_TRUNCATED -3

6.90.1.5 #define AC_INVALID_OUTPTR -4

6.90.1.6 #define AC_TYPE_MISMATCH -5

6.90.1.7 #define AC_DIM_MISMATCH -6

6.90.2 Variable Documentation

6.90.2.1 algo_comm_t algo_comm_default

6.91 mha_algo_comm.h File Reference

Header file for Algorithm Communication.

Classes

- class **MHA_AC::spectrum_t**
  *Insert a **MHASignal::spectrum_t** (p. 800) class into the AC space.*
- class **MHA_AC::waveform_t**
  *Insert a **MHASignal::waveform_t** (p. 812) class into the AC space.*
- class **MHA_AC::int_t**
  Insert a integer variable into the AC space.
- class **MHA_AC::float_t**
  Insert a float point variable into the AC space.
- class **MHA_AC::double_t**
  Insert a double precision floating point variable into the AC space.
- class **MHA_AC::stat_t**
- class **MHA_AC::ac2matrix_helper_t**
- class **MHA_AC::ac2matrix_t**
  Copy AC variable to a matrix.
- class **MHA_AC::acspace2matrix_t**
  Copy all or a subset of all numeric AC variables into an array of matrixes.

Namespaces

- **MHA_AC**
  Functions and classes for Algorithm Communication (AC) support.

Functions

- **mha_spec_t MHA_AC::get_var_spectrum** (**algo_comm_t** ac, const std::string &name)
  Convert an AC variable into a spectrum.
- **mha_wave_t MHA_AC::get_var_waveform** (**algo_comm_t** ac, const std::string &name)
  Convert an AC variable into a waveform.
- int **MHA_AC::get_var_int** (**algo_comm_t** ac, const std::string &name)
  Return value of an integer scalar AC variable.
- float **MHA_AC::get_var_float** (**algo_comm_t** ac, const std::string &name)
  Return value of an floating point scalar AC variable.
- std::vector< float > **MHA_AC::get_var_vfloat** (**algo_comm_t** ac, const std::string &name)
  Return value of an floating point vector AC variable as standard vector of floats.

6.91.1 Detailed Description

Header file for Algorithm Communication.

6.92 mha_algo_comm.hh File Reference

Classes

- class **MHAKernel::comm_var_map_t**
- class **MHAKernel::algo_comm_class_t**

Namespaces

- **MHAKernel**

Macros

- #define **ALGO_COMM_ID_STR** "MFVK3jL5rmeus1XtggEI971aXCR/GU7RRehKz4k↩
  Qtrg="

Functions

- algo_comm_class_t * **MHAKernel::algo_comm_safe_cast** (void *)

Variables

- **algo_comm_t algo_comm_default**

6.92.1 Macro Definition Documentation

6.92.1.1 #define ALGO_COMM_ID_STR "MFVK3jL5rmeus1XtggEl971aXCR/GU7RRehKz4kQtrg="

6.92.2 Variable Documentation

6.92.2.1 algo_comm_t algo_comm_default

6.93 mha_defs.h File Reference

Preprocessor definitions common to all MHA components.

Macros

- #define **__MHA_FUN__** __FUNC__
- #define **CHECK_EXPR**(x) {if(!(x)){throw **MHA_Error**(__FILE__,__LINE__,"The expression \"" #x "\" is invalid.");}}
- #define **CHECK_VAR**(x) {if(!(x)){throw **MHA_Error**(__FILE__,__LINE__,"The variable \"" #x "\" is not defined.");}}
- #define **__declspec**(p)
- #define **M_PI** 3.14159265358979323846
  Define pi if it is not defined yet.
- #define **MIN**(a, b) (((a)<(b))?(a):(b))
  Macro for minimum function.
- #define **MAX**(a, b) (((a)>(b))?(a):(b))
  Macro for maximum function.
- #define **MHA_EAR_LEFT** 0
- #define **MHA_EAR_RIGHT** 1
- #define **MHA_EAR_MAX** 2

6.93.1 Detailed Description

Preprocessor definitions common to all MHA components.

This file contains all preprocessor and type definitions which are common to all Master Hearing Aid components.

6.93.2 Macro Definition Documentation

6.93.2.1 #define __MHA_FUN__ __FUNC__

6.93.2.2 #define CHECK_EXPR(*x*) {if(!(x)){throw MHA_Error(__FILE__,__LINE__,"The expression \"" #x "\" is invalid.");}}

6.93.2.3 #define CHECK_VAR(*x*) {if(!(x)){throw MHA_Error(__FILE__,__LINE__,"The variable \"" #x "\" is not defined.");}}

6.93.2.4 #define __declspec(*p*)

6.93.2.5 #define M_PI 3.14159265358979323846

Define pi if it is not defined yet.

6.93.2.6 #define MIN(*a,* *b*) (((a)<(b))?(a):(b))

Macro for minimum function.

6.93.2.7 #define MAX(*a,* *b*) (((a)>(b))?(a):(b))

Macro for maximum function.

6.93.2.8 #define MHA_EAR_LEFT 0

6.93.2.9 #define MHA_EAR_RIGHT 1

6.93.2.10 #define MHA_EAR_MAX 2

6.94 mha_errno.c File Reference

Macros

- #define **STRLEN** 0x1000

Functions

- const char * **mha_strerror** (int mhaerrno)
- void **mha_set_user_error** (const char *str)

Variables

- char **next_except_str** [**STRLEN**] = ""
- const char * **cstr_strerror** [**MHA_ERR_USER**]

6.94.1 Macro Definition Documentation

6.94.1.1 #define STRLEN 0x1000

6.94.2 Function Documentation

6.94.2.1 const char* mha_strerror (int *mhaerrno*)

6.94.2.2 void mha_set_user_error (const char * *str*)

6.94.3 Variable Documentation

6.94.3.1 char next_except_str[STRLEN] = ""

6.94.3.2 const char* cstr_strerror[MHA_ERR_USER]

6.95 mha_errno.h File Reference

Macros

- #define **MHA_ERR_SUCCESS** 0
- #define **MHA_ERR_UNKNOWN** 1
- #define **MHA_ERR_INVALID_HANDLE** 2
- #define **MHA_ERR_NULL** 3
- #define **MHA_ERR_VARRANGE** 4
- #define **MHA_ERR_VARFMT** 5
- #define **MHA_ERR_USER** 10000

Functions

- const char * **mha_strerror** (int mhaerrno)
- void **mha_set_user_error** (const char *str)

6.95.1 Macro Definition Documentation

6.95.1.1 #define MHA_ERR_SUCCESS 0

6.95.1.2 #define MHA_ERR_UNKNOWN 1

6.95.1.3 #define MHA_ERR_INVALID_HANDLE 2

6.95.1.4 #define MHA_ERR_NULL 3

6.95.1.5 #define MHA_ERR_VARRANGE 4

6.95.1.6 #define MHA_ERR_VARFMT 5

6.95.1.7 #define MHA_ERR_USER 10000

6.95.2 Function Documentation

6.95.2.1 const char* mha_strerror (int *mhaerrno*)

6.95.2.2 void mha_set_user_error (const char * *str*)

6.96 mha_error.cpp File Reference

Implementation of openMHA error handling.

Namespaces

- **mha_error_helpers**

Functions

- unsigned **mha_error_helpers::digits** (unsigned n)

  Compute number of decimal digits required to represent an unsigned integer.
- unsigned **mha_error_helpers::snprintf_required_length** (const char *formatstring,...)

  snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.
- void **mha_debug** (const char *fmt,...)

  Print an info message (stderr on Linux, OutputDebugString in Windows).

6.96.1 Detailed Description

Implementation of openMHA error handling.

This file forms a seperate library.

6.97 mha_error.hh File Reference

Classes

- class **MHA_Error**

  Error reporting exception class.

Namespaces

- **mha_error_helpers**

Macros

- #define **Getmsg**(e) ((e).get_msg())
- #define **MHA_ErrorMsg**(x) **MHA_Error**(__FILE__,__LINE__,"%s",x)

  Throw an openMHA error with a text message.
- #define **MHA_assert**(x) if(!(x)) throw **MHA_Error**(__FILE__,__LINE__,"\"%s\" is false.",#x)

  *Assertion macro, which throws an **MHA_Error** (p. 445).*
- #define **MHA_assert_equal**(a, b) if(a != b) throw **MHA_Error**(__FILE__,__LINE__↩ ,"\"%s == %s\" is false (%s = %g, %s = %g).",#a,#b,#a,(double)(a),#b,(double)(b))

  *Equality assertion macro, which throws an **MHA_Error** (p. 445) with the values.*

Functions

- void **mha_debug** (const char *fmt,...)

  Print an info message (stderr on Linux, OutputDebugString in Windows).
- unsigned **mha_error_helpers::digits** (unsigned n)

  Compute number of decimal digits required to represent an unsigned integer.
- unsigned **mha_error_helpers::snprintf_required_length** (const char *formatstring,...)

  snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.

6.97.1 Macro Definition Documentation

6.97.1.1 #define Getmsg(*e*) ((e).get_msg())

6.98 mha_event_emitter.h File Reference

Classes

- class **MHAEvents::connector_base_t**
- class **MHAEvents::emitter_t**
  Class for emitting openMHA events.

Namespaces

- **MHAEvents**
  Collection of event handling classes.

6.99 mha_events.cpp File Reference

6.100 mha_events.h File Reference

Classes

- class **MHAEvents::connector_t**< **receiver_t** >
- class **MHAEvents::patchbay_t**< **receiver_t** >
  Patchbay which connects any event emitter with any member function of the parameter class.

Namespaces

- **MHAEvents**
  Collection of event handling classes.

6.101 mha_fftfb.cpp File Reference

Classes

- class **MHAOvlFilter::barkscale::hz2bark_t**
- class **MHAOvlFilter::barkscale::bark2hz_t**

Namespaces

- **MHAOvlFilter**
  Namespace for overlapping FFT based filter bank classes and functions.
- **MHAOvlFilter::barkscale**
- **MHAOvlFilter::FreqScaleFun**
  Transform functions from linear scale in Hz to new frequency scales.
- **MHAOvlFilter::ShapeFun**
  Shape functions for overlapping filters.

Macros

- #define **BARKSCALE_ENTRIES** 50

Functions

- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2hz** (**mha_real_t** x)
  Dummy scale transformation Hz to Hz.
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2khz** (**mha_real_t** x)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2octave** (**mha_real_t** x)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2third_octave** (**mha_real_t** x)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark** (**mha_real_t** x)
  Transformation to bark scale.
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark_analytic** (**mha_real_t**)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb** (**mha_real_t**)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb_glasberg1990** (**mha_real_t**)
- **mha_real_t MHAOvlFilter::FreqScaleFun::hz2log** (**mha_real_t** x)
  Third octave frequency scale.
- **mha_real_t MHAOvlFilter::FreqScaleFun::inv_scale** (**mha_real_t**, **mha_real_t**(∗)(**mha_real_t**))
- **mha_real_t MHAOvlFilter::ShapeFun::rect** (**mha_real_t** x)
  Filter shape function for rectangular filters.
- **mha_real_t MHAOvlFilter::ShapeFun::linear** (**mha_real_t** x)
  Filter shape function for sawtooth filters.
- **mha_real_t MHAOvlFilter::ShapeFun::hann** (**mha_real_t** x)
  Filter shape function for hanning shaped filters.
- **mha_real_t MHAOvlFilter::ShapeFun::expflt** (**mha_real_t**)
- **mha_real_t MHAOvlFilter::ShapeFun::gauss** (**mha_real_t**)
- **mha_real_t filtershapefun** (**mha_real_t** f, **MHAOvlFilter::band_descriptor_t** b, **mha_real_t** plateau)

Variables

- **mha_real_t MHAOvlFilter::barkscale::vfreq** [**BARKSCALE_ENTRIES**]
- **mha_real_t MHAOvlFilter::barkscale::vbark** [**BARKSCALE_ENTRIES**]

6.101.1 Macro Definition Documentation

6.101.1.1 #define BARKSCALE_ENTRIES 50

6.101.2 Function Documentation

6.101.2.1 mha_real_t filtershapefun (mha_real_t *f,* MHAOvlFilter::band_descriptor_t *b,* mha_real_t *plateau*)

6.102 mha_fftfb.hh File Reference

Classes

- class **MHAOvlFilter::band_descriptor_t**
- class **MHAOvlFilter::scale_var_t**
- class **MHAOvlFilter::fscale_t**
- class **MHAOvlFilter::fscale_bw_t**
- class **MHAOvlFilter::fftfb_vars_t**

  Set of configuration variables for FFT-based overlapping filters.
- class **MHAOvlFilter::fspacing_t**

  Class for frequency spacing, used by filterbank shape generator class.
- class **MHAOvlFilter::fftfb_t**

  FFT based overlapping filter bank.
- class **MHAOvlFilter::overlap_save_filterbank_t**

  *A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 616).*
- class **MHAOvlFilter::overlap_save_filterbank_t::vars_t**
- class **MHAOvlFilter::overlap_save_filterbank_analytic_t**
- class **MHAOvlFilter::fftfb_ac_info_t**

Namespaces

- **MHAOvlFilter**

  Namespace for overlapping FFT based filter bank classes and functions.

Typedefs

- typedef **mha_real_t**(**MHAOvlFilter::scale_fun_t**) (**mha_real_t**)

6.103 mha_fifo.cpp File Reference

6.104 mha_fifo.h File Reference

Classes

- class **mha_fifo_t**< **T** >

  A FIFO class for blocksize adaptation Synchronization: None.
- class **mha_drifter_fifo_t**< **T** >

  A FIFO class for blocksize adaptation without Synchronization.
- class **mha_fifo_thread_platform_t**

  Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.
- class **mha_fifo_posix_threads_t**
- class **mha_fifo_thread_guard_t**

  Simple Mutex Guard Class.
- class **mha_fifo_lw_t**< **T** >

  This FIFO uses locks to synchronize access.
- class **mha_dblbuf_t**< **FIFO** >

  The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.
- class **mha_rt_fifo_element_t**< **T** >

  *Object wrapper for **mha_rt_fifo_t*** (p. 462).
- class **mha_rt_fifo_t**< **T** >

  Template class for thread safe, half real time safe fifo without explixit locks.

Macros

- #define **mha_fifo_thread_platform_implementation_t mha_fifo_posix_threads_t**

6.104.1 Macro Definition Documentation

6.104.1.1 #define mha_fifo_thread_platform_implementation_t mha_fifo_posix_threads_t

6.105 mha_filter.cpp File Reference

Functions

- std::vector< **mha_real_t** > **diff_coeffs** ()

6.105.1 Function Documentation

6.105.1.1 std::vector<mha_real_t> diff_coeffs ( )

6.106 mha_filter.hh File Reference

Header file for IIR filter classes.

Classes

- class **MHAFilter::filter_t**
  Generic IIR filter class.
- class **MHAFilter::diff_t**
  Differentiator class (non-normalized)
- class **MHAFilter::o1_ar_filter_t**
  First order attack-release lowpass filter.
- class **MHAFilter::o1flt_lowpass_t**
  First order low pass filter.
- class **MHAFilter::o1flt_maxtrack_t**
  First order maximum tracker.
- class **MHAFilter::o1flt_mintrack_t**
  First order minimum tracker.
- class **MHAFilter::iir_filter_state_t**
- class **MHAFilter::iir_filter_t**
  IIR filter class wrapper for integration into parser structure.
- class **MHAFilter::adapt_filter_state_t**
- class **MHAFilter::adapt_filter_param_t**
- class **MHAFilter::adapt_filter_t**
  Adaptive filter.
- class **MHAFilter::fftfilter_t**
  FFT based FIR filter implementation.
- class **MHAFilter::fftfilterbank_t**
  FFT based FIR filterbank implementation.
- struct **MHAFilter::transfer_function_t**
  a structure containing a source channel number, a target channel number, and an impulse response.
- struct **MHAFilter::transfer_matrix_t**
  A sparse matrix of transfer function partitionss.
- class **MHAFilter::partitioned_convolution_t**
  A filter class for partitioned convolution.
- struct **MHAFilter::partitioned_convolution_t::index_t**
  Bookkeeping class.
- class **MHAFilter::smoothspec_t**
  Smooth spectral gains, create a windowed impulse response.

- class **MHAFilter::resampling_filter_t**
  Hann shaped low pass filter for resampling.
- class **MHAFilter::polyphase_resampling_t**
  A class that performs polyphase resampling.
- class **MHAFilter::blockprocessing_polyphase_resampling_t**
  A class that does polyphase resampling and takes into account block processing.
- class **MHAFilter::iir_ord1_real_t**
  First order recursive filter.

Namespaces

- **MHAFilter**
  Namespace for IIR and FIR filter classes.

Functions

- void **MHAFilter::make_friendly_number** (**mha_real_t** &x)
- void **MHAFilter::make_friendly_number** (**mha_complex_t** &x)
- void **MHAFilter::make_friendly_number** (double &x)
- void **MHAFilter::o1_lp_coeffs** (const **mha_real_t** tau, const **mha_real_t** fs, **mha_real_t** &c1, **mha_real_t** &c2)
  Set first order filter coefficients from time constant and sampling rate.
- void **MHAFilter::butter_stop_ord1** (double *A, double *B, double f1, double f2, double fs)
  Setup a first order butterworth band stop filter.
- **MHASignal::waveform_t** * **MHAFilter::spec2fir** (const **mha_spec_t** *spec, const unsigned int fftlen, const **MHAWindow::base_t** &window, const bool minphase)
  Create a windowed impulse response/FIR filter coefficients from a spectrum.
- unsigned **MHAFilter::gcd** (unsigned a, unsigned b)
  greatest common divisor
- double **MHAFilter::sinc** (double x)
  sin(x)/x function, coping with x=0.
- std::pair< unsigned, unsigned > **MHAFilter::resampling_factors** (float source_sampling_rate, float target_sampling_rate, float factor=1.0f)
  Computes rational resampling factor from two sampling rates.

6.106.1 Detailed Description

Header file for IIR filter classes.

6.107 mha_generic_chain.cpp File Reference

Functions

- void **mhaconfig_compare** (**mhaconfig_t** req, **mhaconfig_t** avail, const char *cpref)

6.107.1 Function Documentation

6.107.1.1 void mhaconfig_compare (mhaconfig_t *req*, mhaconfig_t *avail*, const char * *cpref*)

6.108 mha_generic_chain.h File Reference

Classes

- class **mhachain::plugs_t**
- class **mhachain::chain_base_t**

Namespaces

- **mhachain**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.108.1 Macro Definition Documentation

6.108.1.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

6.109 mha_io_ifc.h File Reference

Typedefs

- typedef int(* **IOProcessEvent_t**) (void *handle, **mha_wave_t** *sIn, **mha_wave_t** **sOut)
  Event handler for signal stream.
- typedef void(* **IOStoppedEvent_t**) (void *handle, int proc_err, int io_err)
  Event handler for stop event.
- typedef void(* **IOStartedEvent_t**) (void *handle)
  Event handler for start event.
- typedef int(* **IOInit_t**) (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- typedef int(* **IOPrepare_t**) (void *handle, int num_inchannels, int num_outchannels)
- typedef int(* **IOStart_t**) (void *handle)
- typedef int(* **IOStop_t**) (void *handle)
- typedef int(* **IORelease_t**) (void *handle)
- typedef int(* **IOSetVar_t**) (void *handle, const char *cmd, char *retval, unsigned int len)
- typedef const char *(* **IOStrError_t**) (void *handle, int err)
- typedef void(* **IODestroy_t**) (void *handle)

6.109.1 Typedef Documentation

6.109.1.1 typedef int(∗ IOProcessEvent_t) (void ∗handle, mha_wave_t ∗sIn, mha_wave_t ∗∗sOut)

Event handler for signal stream.

This event handler needs to be realtime compatible. All signal path processing will be performed in this callback.

6.109.1.2 typedef void(∗ IOStoppedEvent_t) (void ∗handle, int proc_err, int io_err)

Event handler for stop event.

This event handler needs to be realtime compatible. The function must return immediatly.

6.109.1.3 typedef void(∗ IOStartedEvent_t) (void ∗handle)

Event handler for start event.

This event handler needs to be realtime compatible. The function must return immediatly.

6.109.1.4 typedef int(∗ IOInit_t) (int fragsize, float samplerate, IOProcessEvent_t proc_event, void ∗proc_handle, IOStartedEvent_t start_event, void ∗start_handle, IOStoppedEvent_t stop_event, void ∗stop_handle, void ∗∗handle)

6.109.1.5 typedef int(∗ IOPrepare_t) (void ∗handle, int num_inchannels, int num_outchannels)

6.109.1.6 typedef int(∗ IOStart_t) (void ∗handle)

6.109.1.7 typedef int(∗ IOStop_t) (void ∗handle)

6.109.1.8 typedef int(∗ IORelease_t) (void ∗handle)

6.109.1.9 typedef int(∗ IOSetVar_t) (void ∗handle, const char ∗cmd, char ∗retval, unsigned int len)

6.109.1.10 typedef const char∗(∗ IOStrError_t) (void ∗handle, int err)

6.109.1.11 typedef void(∗ IODestroy_t) (void ∗handle)

6.110 mha_multisrc.cpp File Reference

Namespaces

- **MHAMultiSrc**

  Collection of classes for selecting audio chunks from multiple sources.

6.111 mha_multisrc.h File Reference

Classes

- class **MHAMultiSrc::channel_t**
- class **MHAMultiSrc::channels_t**
- class **MHAMultiSrc::base_t**
  Base class for source selection.
- class **MHAMultiSrc::waveform_t**
- class **MHAMultiSrc::spectrum_t**

Namespaces

- **MHAMultiSrc**
  Collection of classes for selecting audio chunks from multiple sources.

6.112 mha_os.cpp File Reference

Functions

- bool **mha_hasenv** (const std::string &envvar)
  Checks if environment variable exists.
- std::string **mha_getenv** (const std::string &envvar)
  Get value of environment variable.
- void **mha_delenv** (const std::string &envvar)
  Deletes environment variable from process environment if it exists.
- int **mha_setenv** (const std::string &envvar, const std::string &**value**)
  Set value of environment variable.
- std::list< std::string > **mha_library_paths** ()
- std::list< std::string > **list_dir** (const std::string &path, const std::string &pattern)

6.112.1 Function Documentation

6.112.1.1 bool mha_hasenv (const std::string & *envvar*)

Checks if environment variable exists.

Parameters

envvar	Name of environment variable to check

Returns

true if the environment has a variable of this name

6.112.1.2 std::string mha_getenv (const std::string & *envvar*)

Get value of environment variable.

Parameters

envvar	Name of environment variable to retrieve

Returns

content of environment variable if it exists, empty string if the environment variable does not exist

6.112.1.3 void mha_delenv (const std::string & *envvar*)

Deletes environment variable from process environment if it exists.

Parameters

envvar	Name of environment variable to delete

6.112.1.4 int mha_setenv (const std::string & *envvar,* const std::string & *value*)

Set value of environment variable.

Parameters

envvar	Name of environment variable to set
value	New content for environment variable

Returns

error code: 0 on success, OS dependent error code on failure

6.112.1.5 std::list<std::string> mha_library_paths ( )

6.112.1.6 std::list<std::string> list_dir (const std::string & *path,* const std::string & *pattern*)

6.113 mha_os.h File Reference

Classes

- class **mha_stash_environment_variable_t**
  This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.
- class **dynamiclib_t**

Macros

- #define **mha_loadlib**(x) dlopen(x,RTLD_NOW)
- #define **mha_freelib**(x) dlclose(x)
- #define **mha_freelib_success**(x) (x == 0)
- #define **mha_getlibfun**(h, x) x ## _cb = (x ## _t)dlsym(h,#x)
- #define **mha_getlibfun_checked**(h, x) x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw **MHA_Error**(__FILE__,__LINE__,"Function " #x " is undefined.")
- #define **mha_loadlib_error**(x) dlerror()
- #define **mha_lib_extension** ".so"
- #define **mha_msleep**(milliseconds) usleep((milliseconds)*1000)
- #define **FMTsz** "%zu"
  printf modifier to print integers of type size_t
- #define **MHA_RESOLVE**(h, t) t ## _cb = (t ## _t)(h->resolve(#t))
- #define **MHA_RESOLVE_CHECKED**(h, t) t ## _cb = (t ## _t)(h->resolve_checked(#t))

Typedefs

- typedef void * **mha_libhandle_t**

Functions

- std::string **mha_getenv** (const std::string &envvar)
  Get value of environment variable.
- bool **mha_hasenv** (const std::string &envvar)
  Checks if environment variable exists.
- int **mha_setenv** (const std::string &envvar, const std::string &**value**)
  Set value of environment variable.
- void **mha_delenv** (const std::string &envvar)
  Deletes environment variable from process environment if it exists.
- std::list< std::string > **mha_library_paths** ()
- std::list< std::string > **list_dir** (const std::string &path, const std::string &pattern)
- void **mha_hton** (float *data, unsigned int len)
- void **mha_ntoh** (float *data, unsigned int len)
- void **mha_hton** (uint32_t *data, unsigned int len)
- void **mha_ntoh** (uint32_t *data, unsigned int len)
- void **mha_hton** (int32_t *data, unsigned int len)
- void **mha_ntoh** (int32_t *data, unsigned int len)

6.113.1 Macro Definition Documentation

6.113.1.1 #define mha_loadlib(*x*) dlopen(x,RTLD_NOW)

6.113.1.2 #define mha_freelib(*x*) dlclose(x)

6.113.1.3 #define mha_freelib_success(*x*) (x == 0)

6.113.1.4 #define mha_getlibfun(*h,* *x*) x ## _cb = (x ## _t)dlsym(h,#x)

6.113.1.5 #define mha_getlibfun_checked(*h,* *x*) x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw MHA_Error(__FILE__,__LINE__,"Function " #x " is undefined.")

6.113.1.6 #define mha_loadlib_error(*x*) dlerror()

6.113.1.7 #define mha_lib_extension ".so"

6.113.1.8 #define mha_msleep(*milliseconds*) usleep((milliseconds)∗1000)

6.113.1.9 #define FMTsz "%zu"

printf modifier to print integers of type size_t

6.113.1.10 #define MHA_RESOLVE(*h,* *t*) t ## _cb = (t ## _t)(h->resolve(#t))

6.113.1.11 #define MHA_RESOLVE_CHECKED(*h,* *t*) t ## _cb = (t ## _t)(h->resolve_checked(#t))

6.113.2 Typedef Documentation

6.113.2.1 typedef void∗ mha_libhandle_t

6.113.3 Function Documentation

6.113.3.1 std::string mha_getenv (const std::string & *envvar*)

Get value of environment variable.

Parameters

envvar	Name of environment variable to retrieve

Returns

content of environment variable if it exists, empty string if the environment variable does not exist

6.113.3.2 bool mha_hasenv (const std::string & *envvar*)

Checks if environment variable exists.

Parameters

envvar	Name of environment variable to check

Returns

true if the environment has a variable of this name

6.113.3.3 int mha_setenv (const std::string & *envvar,* const std::string & *value*)

Set value of environment variable.

Parameters

envvar	Name of environment variable to set
value	New content for environment variable

Returns

error code: 0 on success, OS dependent error code on failure

6.113.3.4 void mha_delenv (const std::string & *envvar*)

Deletes environment variable from process environment if it exists.

Parameters

envvar	Name of environment variable to delete

6.113.3.5 std::list<std::string> mha_library_paths ( )

6.113.3.6 std::list<std::string> list_dir (const std::string & *path*, const std::string & *pattern*)

6.113.3.7 void mha_hton (float * *data*, unsigned int *len*) [inline]

6.113.3.8 void mha_ntoh (float * *data*, unsigned int *len*) [inline]

6.113.3.9 void mha_hton (uint32_t * *data*, unsigned int *len*) [inline]

6.113.3.10 void mha_ntoh (uint32_t * *data*, unsigned int *len*) [inline]

6.113.3.11 void mha_hton (int32_t * *data*, unsigned int *len*) [inline]

6.113.3.12 void mha_ntoh (int32_t * *data*, unsigned int *len*) [inline]

6.114 mha_parser.cpp File Reference

Namespaces

- **MHAParser**
  Name space for the openMHA-Parser configuration language.
- **MHAParser::StrCnv**
  String converter namespace.

Macros

- #define **MHAPLATFORM** "undefined-linux"

Functions

- int **MHAParser::get_precision** ()
- int **MHAParser::StrCnv::num_brackets** (const std::string &s)

  Return number of brackets at beginning and end of string.
- int **MHAParser::StrCnv::bracket_balance** (const std::string &s)
- static std::ostream & **write_float** (std::ostream &o, const float &f)
- static std::string **parse_1_float** (const std::string &s, **mha_real_t** &v)

  This internal function parses a floating point number from the beginning of a string.
- static std::string **parse_1_complex** (const std::string &s, **mha_complex_t** &v)

  This internal function parses a complex number from the beginning of a string.

6.114.1 Macro Definition Documentation

6.114.1.1 #define MHAPLATFORM "undefined-linux"

6.114.2 Function Documentation

6.114.2.1 static std::ostream& write_float (std::ostream & *o,* const float & *f*) [inline], [static]

6.114.2.2 static std::string parse_1_float (const std::string & *s,* mha_real_t & *v*) [static]

This internal function parses a floating point number from the beginning of a string.

Parameters

s	The string to parse
v	The float variable to fill with a value

Returns

The rest of the string.

6.114.2.3 static std::string parse_1_complex (const std::string & *s,* mha_complex_t & *v*) [static]

This internal function parses a complex number from the beginning of a string.

Parameters

s	The string to parse
v	The complex variable to fill with a value

Returns

The rest of the string.

6.115 mha_parser.hh File Reference

Header file for the MHA-Parser script language.

Classes

- class **MHAParser::keyword_list_t**
  Keyword list class.
- class **MHAParser::expression_t**
- class **MHAParser::entry_t**
- class **MHAParser::base_t**
  Base class for all parser items.
- class **MHAParser::base_t::replace_t**
- class **MHAParser::parser_t**
  Parser node class.
- class **MHAParser::c_ifc_parser_t**
- class **MHAParser::monitor_t**
  Base class for monitors and variable nodes.
- class **MHAParser::variable_t**
  Base class for variable nodes.
- class **MHAParser::range_var_t**
  Base class for all variables with a numeric value range.
- class **MHAParser::kw_t**
  Variable with keyword list value.
- class **MHAParser::string_t**
  Variable with a string value.
- class **MHAParser::vstring_t**
  Vector variable with string values.
- class **MHAParser::bool_t**
  Variable with a boolean value ("yes"/"no")
- class **MHAParser::int_t**
  Variable with integer value.
- class **MHAParser::float_t**
  Variable with float value.

- class **MHAParser::complex_t**
  Variable with complex value.
- class **MHAParser::vint_t**
  Variable with vector<int> value.
- class **MHAParser::vfloat_t**
  Vector variable with float value.
- class **MHAParser::vcomplex_t**
  Vector variable with complex value.
- class **MHAParser::mfloat_t**
  Matrix variable with float value.
- class **MHAParser::mcomplex_t**
  Matrix variable with complex value.
- class **MHAParser::int_mon_t**
  Monitor variable with int value.
- class **MHAParser::bool_mon_t**
  Monitor with string value.
- class **MHAParser::string_mon_t**
  Monitor with string value.
- class **MHAParser::vstring_mon_t**
  Vector of monitors with string value.
- class **MHAParser::vint_mon_t**
  Vector of ints monitor.
- class **MHAParser::mint_mon_t**
  Matrix of ints monitor.
- class **MHAParser::vfloat_mon_t**
  Vector of floats monitor.
- class **MHAParser::mfloat_mon_t**
  Matrix of floats monitor.
- class **MHAParser::float_mon_t**
  Monitor with float value.
- class **MHAParser::complex_mon_t**
  Monitor with complex value.
- class **MHAParser::vcomplex_mon_t**
  Monitor with vector of complex values.
- class **MHAParser::mcomplex_mon_t**
  Matrix of complex numbers monitor.
- class **MHAParser::commit_t**< **receiver_t** >
  Parser variable with event-emission functionality.
- class **MHAParser::mhaconfig_mon_t**

Namespaces

- **MHAParser**
  Name space for the openMHA-Parser configuration language.
- **MHAParser::StrCnv**
  String converter namespace.

Macros

- #define **DEFAULT_RETSIZE** 0x100000
- #define **insert_member**(x) insert_item(#x,&x)
  Macro to insert a member variable into a parser.

Typedefs

- typedef std::string(base_t::* **MHAParser::opact_t**) (**expression_t** &)
- typedef std::string(base_t::* **MHAParser::query_t**) (const std::string &)
- typedef std::map< std::string, opact_t > **MHAParser::opact_map_t**
- typedef std::map< std::string, query_t > **MHAParser::query_map_t**
- typedef std::list< entry_t > **MHAParser::entry_map_t**
- typedef int(* **MHAParser::c_parse_cmd_t**) (void *, const char *, char *, unsigned int)
- typedef const char *(* **MHAParser::c_parse_err_t**) (void *, int)

Functions

- std::string **MHAParser::commentate** (const std::string &s)
- void **MHAParser::trim** (std::string &s)
- std::string **MHAParser::cfg_dump** (base_t *, const std::string &)
- std::string **MHAParser::cfg_dump_short** (base_t *, const std::string &)
- std::string **MHAParser::all_dump** (base_t *, const std::string &)
- std::string **MHAParser::mon_dump** (base_t *, const std::string &)
- std::string **MHAParser::all_ids** (base_t *, const std::string &, const std::string &="")
- void **MHAParser::strreplace** (std::string &, const std::string &, const std::string &)
  string replace function
- void **MHAParser::envreplace** (std::string &s)
- void **MHAParser::StrCnv::str2val** (const std::string &, bool &)
  Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, float &)
  Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, **mha_complex_t** &)
  Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, int &)
  Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, keyword_list_t &)
  Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, std::string &)
  Convert from string.
- template<class arg_t >
  void **MHAParser::StrCnv::str2val** (const std::string &s, std::vector< arg_t > &val)
  Converter for vector types.

- template<>
  void **MHAParser::StrCnv::str2val**< **mha_real_t** > (const std::string &s, std::vector< **mha_real_t** > &v)

  Converter for vector<mha_real_t> with Matlab-style expansion.
- template<class arg_t >
  void **MHAParser::StrCnv::str2val** (const std::string &s, std::vector< std::vector< arg_t > > &val)

  Converter for matrix types.
- std::string **MHAParser::StrCnv::val2str** (const bool &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const float &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const **mha_complex_t** &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const int &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const keyword_list_t &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::string &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< float > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< **mha_complex_t** > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< int > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< int > > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< std::string > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< float > > &)

  Convert to string.
- std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< **mha_↩ complex_t** > > &)

  Convert to string.

6.115.1 Detailed Description

Header file for the MHA-Parser script language.

6.115.2 Macro Definition Documentation

6.115.2.1 #define DEFAULT_RETSIZE 0x100000

6.115.2.2 #define insert_member(*x*) insert_item(#x,&x)

Macro to insert a member variable into a parser.

Parameters

x	Member variable to be inserted. Name of member variable will be used as configuration name.

See also **MHAParser::parser_t::insert_item()** (p. 690).

6.116 mha_plugin.hh File Reference

Header file for MHA C++ plugin class templates.

Classes

- class **MHAPlugin::cfg_chain_t**< **runtime_cfg_t** >
- class **MHAPlugin::config_t**< **runtime_cfg_t** >
  Template class for thread safe configuration.
- class **MHAPlugin::plugin_t**< **runtime_cfg_t** >
  The template class for C++ openMHA plugins.

Namespaces

- **MHAPlugin**
  Namespace for openMHA plugin class templates and thread-safe runtime configurations.

Macros

- #define **__declspec**(p)
- #define **WINAPI**
- #define **HINSTANCE** int
- #define **GITCOMMITHASH** "independent-plugin-build"
- #define **MHAPLUGIN_PROC_CALLBACK_PREFIX**(prefix, classname, indom, outdom)
- #define **MHAPLUGIN_INIT_CALLBACKS_PREFIX**(prefix, classname)
- #define **MHAPLUGIN_CALLBACKS_PREFIX**(prefix, classname, indom, outdom)
  C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION_PREFIX**(prefix, cat, doc)
- #define **MHAPLUGIN_PROC_CALLBACK**(plugname, classname, indom, outdom) **M**↩ **HAPLUGIN_PROC_CALLBACK_PREFIX**(MHA_STATIC_ ## plugname ## _↩ ,classname,indom,outdom)
- #define **MHAPLUGIN_INIT_CALLBACKS**(plugname, classname) **MHAPLUGIN_INIT**↩ **_CALLBACKS_PREFIX**(MHA_STATIC_ ## plugname ## _,classname)
- #define **MHAPLUGIN_CALLBACKS**(plugname, classname, indom, outdom) **MHAPLU**↩ **GIN_CALLBACKS_PREFIX**(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)
  C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION**(plugname, cat, doc) **MHAPLUGIN_DOCU**↩ **MENTATION_PREFIX**(MHA_STATIC_ ## plugname ## _,cat,doc)
  Wrapper macro for the plugin documentation interface.

Functions

- **__attribute__** ((unused)) static const char *mha_git_commit_hash
  store git commit hash in every binary plgin to support reproducible research

6.116.1 Detailed Description

Header file for MHA C++ plugin class templates.

This file defines useful macros and template classes for the development of MHA plugins. A set of macros wraps a C++ interface around the ANSI-C plugin interface. The plugin_t template class defines a corresponding C++ class with all required members. This class can make use of thread safe configurations (config_t).

6.116.2 Macro Definition Documentation

6.116.2.1 #define __declspec(
p)

6.116.2.2 #define WINAPI

6.116.2.3 #define HINSTANCE int

6.116.2.4 #define GITCOMMITHASH "independent-plugin-build"

6.116.2.5 #define MHAPLUGIN_PROC_CALLBACK_PREFIX(
prefix,
classname,
indom,
outdom)

6.116.2.6 #define MHAPLUGIN_INIT_CALLBACKS_PREFIX(
prefix,
classname)

6.116.2.7 #define MHAPLUGIN_DOCUMENTATION_PREFIX(
prefix,
cat,
doc)

6.116.2.8 #define MHAPLUGIN_PROC_CALLBACK(
plugname,
classname,
indom,
outdom) MHAPLUGIN_PROC_CALLBACK_PREFIX(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)

6.116.2.9 #define MHAPLUGIN_INIT_CALLBACKS(*plugname,* *classname*) MHAPLUGIN_INIT_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _,classname)

6.116.3 Function Documentation

6.116.3.1 __attribute__ ((unused)) const

store git commit hash in every binary plgin to support reproducible research

6.117 mha_profiling.c File Reference

Functions

- void **mha_tic** (**mha_tictoc_t** *t)
- void **mha_platform_tic** (**mha_platform_tictoc_t** *t)
- float **mha_toc** (**mha_tictoc_t** *t)
- float **mha_platform_toc** (**mha_platform_tictoc_t** *t)

6.117.1 Function Documentation

6.117.1.1 void mha_tic (mha_tictoc_t * *t*)

6.117.1.2 void mha_platform_tic (mha_platform_tictoc_t * *t*)

6.117.1.3 float mha_toc (mha_tictoc_t * *t*)

6.117.1.4 float mha_platform_toc (mha_platform_tictoc_t * *t*)

6.118 mha_profiling.h File Reference

Classes

- struct **mha_tictoc_t**

Typedefs

- typedef **mha_tictoc_t mha_platform_tictoc_t**

Functions

- void **mha_platform_tic** (**mha_platform_tictoc_t** ∗t)
- float **mha_platform_toc** (**mha_platform_tictoc_t** ∗t)

6.118.1 Typedef Documentation

6.118.1.1 typedef mha_tictoc_t mha_platform_tictoc_t

6.118.2 Function Documentation

6.118.2.1 void mha_platform_tic (mha_platform_tictoc_t ∗ *t*)

6.118.2.2 float mha_platform_toc (mha_platform_tictoc_t ∗ *t*)

6.119 mha_ruby.cpp File Reference

Typedefs

- typedef VALUE(∗ **rb_f_t**) (...)

Functions

- static void **mha_free** (void ∗mha)
- static VALUE **mha_alloc** (VALUE klass)
- static VALUE **mha_exit_request** (VALUE self)
- static VALUE **mha_parse** (VALUE self, VALUE request)
- void **Init_mha_ruby** ()

6.119.1 Typedef Documentation

6.119.1.1 typedef VALUE(∗ rb_f_t) (...)

6.119.2 Function Documentation

6.119.2.1 static void mha_free (void ∗ *mha*) [static]

6.119.2.2 static VALUE mha_alloc (VALUE *klass*) [static]

6.119.2.3 static VALUE mha_exit_request (VALUE *self*) [static]

6.119.2.4 static VALUE mha_parse (VALUE *self,* VALUE *request*) [static]

6.119.2.5 void Init_mha_ruby ( )

6.120 mha_signal.cpp File Reference

Classes

- class **MHASignal::hilbert_fftw_t**

Namespaces

- **MHASignal**
  Namespace for audio signal handling and processing classes.

Macros

- #define **MHA_ID_UINT_VECTOR** "MHASignal::uint_vector_t"
- #define **MHA_ID_MATRIX** "MHASignal::matrix_t"
- #define **ASSERT_EQUAL_DIM**(a, b)
- #define **ASSERT_EQUAL_DIM_PTR**(a, b)

Functions

- void **set_minabs** (**mha_spec_t** &self, const **mha_real_t** &m)
- **mha_wave_t** & **operator+=** (**mha_wave_t** &self, const **mha_real_t** &v)
  Addition operator.
- **mha_wave_t** & **operator∗=** (**mha_wave_t** &self, const **mha_real_t** &v)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &self, const **mha_real_t** &v)
  Element-wise multiplication operator.
- **mha_wave_t** & **operator∗=** (**mha_wave_t** &self, const **mha_wave_t** &v)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &self, const **mha_wave_t** &v)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &self, const **mha_spec_t** &v)
  Element-wise multiplication operator.
- **mha_spec_t** & **safe_div** (**mha_spec_t** &self, const **mha_spec_t** &v, **mha_real_t** eps)
  In-Place division with lower limit on divisor.

- **mha_spec_t** & **operator/=** (**mha_spec_t** &self, const **mha_spec_t** &v)
  Element-wise division operator.
- **mha_wave_t** & **operator/=** (**mha_wave_t** &self, const **mha_wave_t** &v)
  Element-wise division operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &self, const **mha_spec_t** &v)
  Addition operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &self, const **mha_real_t** &v)
  Addition operator.
- **mha_wave_t** & **operator+=** (**mha_wave_t** &self, const **mha_wave_t** &v)
  Addition operator.
- **mha_wave_t** & **operator-=** (**mha_wave_t** &self, const **mha_wave_t** &v)
  Subtraction operator.
- **mha_spec_t** & **operator-=** (**mha_spec_t** &self, const **mha_spec_t** &v)
  Subtraction operator.
- **mha_fft_t mha_fft_new** (unsigned int n)
  Create a new instance of an FFT object.
- void **mha_fft_free** (**mha_fft_t** h)
  Remove an FFT object.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out)
  Perform an FFT on each channel of input waveform signal.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out, bool swap)
  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out)
  Perform an inverse FFT on each channel of input spectrum.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out, unsigned int offset)
  Perform an inverse FFT on each channel of input spectrum.
- void **mha_fft_forward** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (forward).
- void **mha_fft_backward** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (backward).
- void **mha_fft_forward_scale** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (forward).
- void **mha_fft_backward_scale** (**mha_fft_t** h, **mha_spec_t** *sIn, **mha_spec_t** *sOut)
  Complex to complex FFT (backward).
- void **mha_fft_wave2spec_scale** (**mha_fft_t** h, const **mha_wave_t** *in, **mha_spec_t** *out)
  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave_scale** (**mha_fft_t** h, const **mha_spec_t** *in, **mha_wave_t** *out)
  Tranform spectrum into waveform segment.
- std::vector< float > **std_vector_float** (const **mha_wave_t** &w)
  *Converts a **mha_wave_t** (p. 496) structure into a std::vector<float> (interleaved order).*
- std::vector< std::vector< float > > **std_vector_vector_float** (const **mha_wave_t** &w)

*Converts a **mha_wave_t** (p. 496) structure into a std::vector< std::vector<float> > (outer vector represents channels).*

- std::vector< std::vector< **mha_complex_t** > > **std_vector_vector_complex** (const **mha_spec_t** &w)

  *Converts a **mha_spec_t** (p. 465) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).*
- static **mha_real_t intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** *sqfreq_response=0)
- void **integrate** (**mha_wave_t** &s)

  Numeric integration of a signal vector (real values)
- void **integrate** (**mha_spec_t** &s)

  Numeric integration of a signal vector (complex values)
- **mha_wave_t** & **operator^=** (**mha_wave_t** &self, const **mha_real_t** &arg)

  Exponent operator.
- **mha_wave_t range** (**mha_wave_t** s, unsigned int k0, unsigned int len)

  Return a time interval from a waveform chunk.
- **mha_spec_t channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)

  Return a channel interval from a spectrum.
- void **assign** (**mha_wave_t** self, const **mha_wave_t** &val)

  Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_spec_t** self, const **mha_spec_t** &val)

  Set all values of spectrum 'self' to 'val'.
- void **timeshift** (**mha_wave_t** &self, int shift)

  Time shift of waveform chunk.

6.120.1 Macro Definition Documentation

6.120.1.1 #define MHA_ID_UINT_VECTOR "MHASignal::uint_vector_t"

6.120.1.2 #define MHA_ID_MATRIX "MHASignal::matrix_t"

6.120.1.3 #define ASSERT_EQUAL_DIM(

a,
b)

6.120.1.4 #define ASSERT_EQUAL_DIM_PTR(

a,
b)

6.120.2 Function Documentation

6.120.2.1 void set_minabs (

mha_spec_t & *self,*
const **mha_real_t** & *m*)

6.120.2.2 mha_spec_t& safe_div (

mha_spec_t & *self,*
const **mha_spec_t** & *v,*
mha_real_t *eps*)

In-Place division with lower limit on divisor.

6.120.2.3 static mha_real_t intensity (const mha_spec_t & *s*, unsigned int *channel*, unsigned int *fftlen*, mha_real_t * *sqfreq_response* = 0) [static]

6.121 mha_signal.hh File Reference

Header file for audio signal handling and processing classes.

Classes

- class **MHASignal::spectrum_t**
  *a signal processing class for spectral data (based on **mha_spec_t** (p. 465))*
- class **MHASignal::waveform_t**
  *signal processing class for waveform data (based on **mha_wave_t** (p. 496))*
- class **MHASignal::doublebuffer_t**
  Double-buffering class.
- class **MHASignal::ringbuffer_t**
  A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.
- class **MHASignal::hilbert_t**
  Hilbert transformation of a waveform segment.
- class **MHASignal::minphase_t**
  Minimal phase function.
- class **MHASignal::stat_t**
- class **MHASignal::delay_wave_t**
  Delayline containing wave fragments.
- class **MHASignal::delay_spec_t**
- class **MHASignal::async_rmslevel_t**
  Class for asynchronous level metering.
- class **MHASignal::uint_vector_t**
  Vector of unsigned values, used for size and index description of n-dimensional matrixes.
- class **MHASignal::matrix_t**
  n-dimensional matrix with real or complex floating point values.
- class **MHASignal::schroeder_t**
  Schroeder tone complex class.
- class **MHASignal::quantizer_t**
  Simple simulation of fixpoint quantization.
- class **MHASignal::loop_wavefragment_t**
  Copy a fixed waveform fragment to a series of waveform fragments of other size.
- class **MHASignal::delay_t**
  Class to realize a simple delay of waveform streams.
- class **MHASignal::subsample_delay_t**
  implements subsample delay in spectral domain.

Namespaces

- **MHASignal**
  Namespace for audio signal handling and processing classes.

Macros

- #define **M_PI** 3.14159265358979323846
- #define **mha_round**(x) (int)((float)x+0.5)

Functions

- void **MHASignal::for_each** (**mha_wave_t** *s, **mha_real_t**(*fun)(**mha_real_t**))
  *Apply a function to each element of a **mha_wave_t** (p. 496).*
- **mha_real_t MHASignal::lin2db** (**mha_real_t** x)
  Conversion from linear scale to dB (no SPL reference)
- **mha_real_t MHASignal::db2lin** (**mha_real_t** x)
  Conversion from dB scale to linear (no SPL reference)
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x)
  Conversion from linear Pascal scale to dB SPL.
- **mha_real_t MHASignal::pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=1e-20f)
  Conversion from squared Pascal scale to dB SPL.
- **mha_real_t MHASignal::dbspl2pa** (**mha_real_t** x)
  Conversion from dB SPL to linear Pascal scale.
- **mha_real_t MHASignal::smp2sec** (**mha_real_t** n, **mha_real_t** srate)
  conversion from samples to seconds
- **mha_real_t MHASignal::sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
  conversion from seconds to samples
- **mha_real_t MHASignal::bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)
  conversion from fft bin index to frequency
- **mha_real_t MHASignal::freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)
  conversion from frequency to fft bin index
- **mha_real_t MHASignal::smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
  conversion from delay in samples to phase shift
- **mha_real_t MHASignal::rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)
  conversion from phase shift to delay in samples
- template<class elem_type >
  std::vector< elem_type > **MHASignal::dupvec** (std::vector< elem_type > vec, unsigned n)
  Duplicate last vector element to match desired size.
- template<class elem_type >
  std::vector< elem_type > **MHASignal::dupvec_chk** (std::vector< elem_type > vec, unsigned n)
  Duplicate last vector element to match desired size, check for dimension.

- bool **equal_dim** (const **mha_wave_t** &a, const **mha_wave_t** &b)
  Test for equal dimension of waveform structures.
- bool **equal_dim** (const **mha_wave_t** &a, const **mhaconfig_t** &b)
  Test for match of waveform dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_spec_t** &b)
  Test for equal dimension of spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mhaconfig_t** &b)
  Test for match of spectrum dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_wave_t** &a, const **mha_spec_t** &b)
  Test for equal dimension of waveform/spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_wave_t** &b)
  Test for equal dimension of waveform/spectrum structures.
- void **integrate** (**mha_wave_t** &s)
  Numeric integration of a signal vector (real values)
- void **integrate** (**mha_spec_t** &s)
  Numeric integration of a signal vector (complex values)
- unsigned int **mha_min_1** (unsigned int a)
- unsigned int **size** (const **mha_wave_t** &s)
  Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** &s)
  Return size of a spectrum structure.
- unsigned int **size** (const **mha_wave_t** ∗s)
  Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** ∗s)
  Return size of a spectrum structure.
- void **clear** (**mha_wave_t** &s)
  Set all values of waveform to zero.
- void **clear** (**mha_wave_t** ∗s)
  Set all values of waveform to zero.
- void **clear** (**mha_spec_t** &s)
  Set all values of spectrum to zero.
- void **clear** (**mha_spec_t** ∗s)
  Set all values of spectrum to zero.
- void **assign** (**mha_wave_t** self, **mha_real_t** val)
  Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_wave_t** self, const **mha_wave_t** &val)
  Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_spec_t** self, const **mha_spec_t** &val)
  Set all values of spectrum 'self' to 'val'.
- void **timeshift** (**mha_wave_t** &self, int shift)
  Time shift of waveform chunk.
- **mha_wave_t range** (**mha_wave_t** s, unsigned int k0, unsigned int len)
  Return a time interval from a waveform chunk.
- **mha_spec_t channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)
  Return a channel interval from a spectrum.

- **mha_real_t** & **value** (**mha_wave_t** ∗s, unsigned int fr, unsigned int ch)
  Access an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** ∗s, unsigned int fr, unsigned int ch)
  Constant access to an element of a waveform structure.
- **mha_real_t** & **value** (**mha_wave_t** ∗s, unsigned int k)
- **mha_complex_t** & **value** (**mha_spec_t** ∗s, unsigned int k)
- **mha_complex_t** & **value** (**mha_spec_t** ∗s, unsigned int fr, unsigned int ch)
  Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** ∗s, unsigned int fr, unsigned int ch)
  Constant access to an element of a spectrum.
- **mha_real_t** & **value** (**mha_wave_t** &s, unsigned int fr, unsigned int ch)
  Access to an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** &s, unsigned int fr, unsigned int ch)
  Constant access to an element of a waveform structure.
- **mha_complex_t** & **value** (**mha_spec_t** &s, unsigned int fr, unsigned int ch)
  Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** &s, unsigned int fr, unsigned int ch)
  Constant access to an element of a spectrum.
- std::vector< float > **std_vector_float** (const **mha_wave_t** &)
  *Converts a **mha_wave_t** (p. 496) structure into a std::vector<float> (interleaved order).*
- std::vector< std::vector< float > > **std_vector_vector_float** (const **mha_wave_t** &)
  *Converts a **mha_wave_t** (p. 496) structure into a std::vector< std::vector<float> > (outer vector represents channels).*
- std::vector< std::vector< **mha_complex_t** > > **std_vector_vector_complex** (const **mha_spec_t** &)
  *Converts a **mha_spec_t** (p. 465) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).*
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_real_t** &)
  Addition operator.
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_wave_t** &)
  Addition operator.
- **mha_wave_t** & **operator-=** (**mha_wave_t** &, const **mha_wave_t** &)
  Subtraction operator.
- **mha_spec_t** & **operator-=** (**mha_spec_t** &, const **mha_spec_t** &)
  Subtraction operator.
- **mha_wave_t** & **operator∗=** (**mha_wave_t** &, const **mha_real_t** &)
  Element-wise multiplication operator.
- **mha_wave_t** & **operator∗=** (**mha_wave_t** &, const **mha_wave_t** &)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &, const **mha_real_t** &)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &, const **mha_wave_t** &)
  Element-wise multiplication operator.
- **mha_spec_t** & **operator∗=** (**mha_spec_t** &, const **mha_spec_t** &)
  Element-wise multiplication operator.

- **mha_spec_t** & **operator/=** (**mha_spec_t** &, const **mha_spec_t** &)
  Element-wise division operator.
- **mha_wave_t** & **operator/=** (**mha_wave_t** &, const **mha_wave_t** &)
  Element-wise division operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &, const **mha_spec_t** &)
  Addition operator.
- **mha_spec_t** & **operator+=** (**mha_spec_t** &, const **mha_real_t** &)
  Addition operator.
- void **set_minabs** (**mha_spec_t** &, const **mha_real_t** &)
- **mha_spec_t** & **safe_div** (**mha_spec_t** &self, const **mha_spec_t** &v, **mha_real_t** eps)
  In-Place division with lower limit on divisor.
- **mha_wave_t** & **operator^=** (**mha_wave_t** &self, const **mha_real_t** &arg)
  Exponent operator.
- void **MHASignal::copy_channel** (**mha_spec_t** &self, const **mha_spec_t** &src, unsigned sch, unsigned dch)
  Copy one channel of a source signal.
- void **MHASignal::copy_channel** (**mha_wave_t** &self, const **mha_wave_t** &src, unsigned src_channel, unsigned dest_channel)
  Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen)
  Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** sqfreq_response[ ])
  Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs** (const **mha_spec_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel** (const **mha_wave_t** &s, unsigned int channel)
  Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s, unsigned int channel)
  Find maximal absolute value.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s)
  Find maximal absolute value.
- **mha_real_t MHASignal::max** (const **mha_wave_t** &s)
  Find maximal value.
- **mha_real_t MHASignal::min** (const **mha_wave_t** &s)
  Find minimal value.
- **mha_real_t MHASignal::sumsqr_channel** (const **mha_wave_t** &s, unsigned int channel)
  Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsqr_frame** (const **mha_wave_t** &s, unsigned int frame)
  Calculate sum over all channels of squared values.
- void **MHASignal::scale** (**mha_spec_t** *dest, const **mha_wave_t** *src)
- void **MHASignal::limit** (**mha_wave_t** &s, const **mha_real_t** &**min**, const **mha_real_t** &**max**)
  Limit the singal in the waveform buffer to the range [min, max].

- **mha_complex_t** & **set** (**mha_complex_t** &self, **mha_real_t** real, **mha_real_t** imag=0)

  *Assign real and imaginary parts to a **mha_complex_t** (p. 431) variable.*
- **mha_complex_t mha_complex** (**mha_real_t** real, **mha_real_t** imag=0)

  *Create a new **mha_complex_t** (p. 431) with specified real and imaginary parts.*
- **mha_complex_t** & **set** (**mha_complex_t** &self, const std::complex< **mha_real_t** > &**stdcomplex**)

  *Assign a **mha_complex_t** (p. 431) variable from a std::complex.*
- std::complex< **mha_real_t** > **stdcomplex** (const **mha_complex_t** &self)

  *Create a std::complex from **mha_complex_t** (p. 431).*
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t angle**)

  *replaces the value of the given **mha_complex_t** (p. 431) with exp(i∗b).*
- double **angle** (const **mha_complex_t** &self)

  Computes the angle of a complex number in the complex plane.
- **mha_complex_t** & **operator+=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Addition of two complex numbers, overwriting the first.
- **mha_complex_t operator+** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Addition of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator+=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Addition of a complex and a real number, overwriting the complex.
- **mha_complex_t operator+** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Addition of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **operator-=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Subtraction of two complex numbers, overwriting the first.
- **mha_complex_t operator-** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Subtraction of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator-=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Subtraction of a complex and a real number, overwriting the complex.
- **mha_complex_t operator-** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Subtraction of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **operator∗=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Multiplication of two complex numbers, overwriting the first.
- **mha_complex_t operator∗** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Multiplication of two complex numbers, result is a temporary object.
- **mha_complex_t** & **operator∗=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Multiplication of a complex and a real number, overwriting the complex.
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t angle**, **mha_real_t** factor)

  *replaces (!) the value of the given **mha_complex_t** (p. 431) with a ∗ exp(i∗b)*
- **mha_complex_t operator∗** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Multiplication of a complex and a real number, result is a temporary object.
- **mha_real_t abs2** (const **mha_complex_t** &self)

  Compute the square of the absolute value of a complex value.
- **mha_real_t abs** (const **mha_complex_t** &self)

Compute the absolute value of a complex value.

- **mha_complex_t** & **operator/=** (**mha_complex_t** &self, **mha_real_t** other_real)

  Division of a complex and a real number, overwriting the complex.
- **mha_complex_t operator/** (const **mha_complex_t** &self, **mha_real_t** other_real)

  Division of a complex and a real number, result is a temporary object.
- **mha_complex_t** & **safe_div** (**mha_complex_t** &self, const **mha_complex_t** &other, **mha_real_t** eps, **mha_real_t** eps2)

  Safe division of two complex numbers, overwriting the first.
- **mha_complex_t** & **operator/=** (**mha_complex_t** &self, const **mha_complex_t** &other)

  Division of two complex numbers, overwriting the first.
- **mha_complex_t operator/** (const **mha_complex_t** &self, const **mha_complex_t** &other)

  Division of two complex numbers, result is a temporary object.
- **mha_complex_t operator-** (const **mha_complex_t** &self)

  Unary minus on a complex results in a negative temporary object.
- bool **operator==** (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compare two complex numbers for equality.
- bool **operator!=** (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compare two complex numbers for inequality.
- void **conjugate** (**mha_complex_t** &self)

  *Replace (!) the value of this **mha_complex_t** (p. 431) with its conjugate.*
- void **conjugate** (**mha_spec_t** &self)

  *Replace (!) the value of this **mha_spec_t** (p. 465) with its conjugate.*
- **mha_complex_t _conjugate** (const **mha_complex_t** &self)

  Compute the cojugate of this complex value.
- void **reciprocal** (**mha_complex_t** &self)

  Replace the value of this complex with its reciprocal.
- **mha_complex_t _reciprocal** (const **mha_complex_t** &self)

  compute the reciprocal of this complex value.
- void **normalize** (**mha_complex_t** &self)

  Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle).
- void **normalize** (**mha_complex_t** &self, **mha_real_t** margin)

  Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle), with a safety margin.
- bool **almost** (const **mha_complex_t** &self, const **mha_complex_t** &other, **mha_real_t** times_epsilon=1e2)

  Compare two complex numbers for equality except for a small relative error.
- bool **operator**< (const **mha_complex_t** &x, const **mha_complex_t** &y)

  Compares the absolute values of two complex numbers.
- std::ostream & **operator**<< (std::ostream &o, const **mha_complex_t** &c)

  *ostream operator for **mha_complex_t** (p. 431)*
- std::istream & **operator**>> (std::istream &i, **mha_complex_t** &c)

  *preliminary istream operator for **mha_complex_t** (p. 431) without error checking*
- **mha_fft_t mha_fft_new** (unsigned int n)

  Create a new FFT handle.

- void **mha_fft_free** (**mha_fft_t** h)

  Destroy an FFT handle.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** ∗in, **mha_spec_t** ∗out)

  Tranform waveform segment into spectrum.
- void **mha_fft_wave2spec** (**mha_fft_t** h, const **mha_wave_t** ∗in, **mha_spec_t** ∗out, bool swaps)

  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** ∗in, **mha_wave_t** ∗out)

  Tranform spectrum into waveform segment.
- void **mha_fft_spec2wave** (**mha_fft_t** h, const **mha_spec_t** ∗in, **mha_wave_t** ∗out, unsigned int offset)

  Tranform spectrum into waveform segment.
- void **mha_fft_forward** (**mha_fft_t** h, **mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)

  Complex to complex FFT (forward).
- void **mha_fft_backward** (**mha_fft_t** h, **mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)

  Complex to complex FFT (backward).
- void **mha_fft_forward_scale** (**mha_fft_t** h, **mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)

  Complex to complex FFT (forward).
- void **mha_fft_backward_scale** (**mha_fft_t** h, **mha_spec_t** ∗sIn, **mha_spec_t** ∗sOut)

  Complex to complex FFT (backward).
- void **mha_fft_wave2spec_scale** (**mha_fft_t** h, const **mha_wave_t** ∗in, **mha_spec_↩ t** ∗out)

  Tranform waveform segment into spectrum.
- void **mha_fft_spec2wave_scale** (**mha_fft_t** h, const **mha_spec_t** ∗in, **mha_wave_↩ t** ∗out)

  Tranform spectrum into waveform segment.
- template<class elem_type >
  elem_type **MHASignal::kth_smallest** (elem_type array[ ], unsigned n, unsigned k)

  Fast search for the kth smallest element of an array.
- template<class elem_type >
  elem_type **MHASignal::median** (elem_type array[ ], unsigned n)

  Fast median search.
- template<class elem_type >
  elem_type **MHASignal::mean** (const std::vector< elem_type > &data, elem_type start↩ _val)

  Calculate average of elements in a vector.
- template<class elem_type >
  std::vector< elem_type > **MHASignal::quantile** (std::vector< elem_type > data, const std::vector< elem_type > &p)

  Calculate quantile of elements in a vector.
- void **MHASignal::saveas_mat4** (const **mha_spec_t** &data, const std::string &varname, FILE ∗fh)

  Save a openMHA spectrum as a variable in a Matlab4 file.
- void **MHASignal::saveas_mat4** (const **mha_wave_t** &data, const std::string &varname, FILE ∗fh)

  Save a openMHA waveform as a variable in a Matlab4 file.

- void **MHASignal::saveas_mat4** (const std::vector< **mha_real_t** > &data, const std::string &varname, FILE *fh)

  Save a float vector as a variable in a Matlab4 file.
- void **MHASignal::copy_permuted** (**mha_wave_t** *dest, const **mha_wave_t** *src)

  Copy contents of a waveform to a permuted waveform.

Variables

- unsigned long int **MHASignal::signal_counter** = 0

  Signal counter to produce signal ID strings.

6.121.1 Detailed Description

Header file for audio signal handling and processing classes.

The classes for waveform, spectrum and filterbank signals defined in this file are "intelligent" versions of the basic waveform, spectrum and filterbank structures used in the C function calls.

6.121.2 Macro Definition Documentation

6.121.2.1 #define M_PI 3.14159265358979323846

6.121.2.2 #define mha_round(*x*) (int)((float)x+0.5)

6.121.3 Function Documentation

6.121.3.1 unsigned int mha_min_1 (unsigned int *a*) [inline]

6.121.3.2 mha_real_t& value (mha_wave_t * *s*, unsigned int *k*) [inline]

6.121.3.3 mha_complex_t& value (mha_spec_t * *s*, unsigned int *k*) [inline]

6.121.3.4 void set_minabs (mha_spec_t & , const mha_real_t &)

6.121.3.5 mha_spec_t& safe_div (mha_spec_t & *self*, const mha_spec_t & *v*, mha_real_t *eps*)

In-Place division with lower limit on divisor.

6.121.3.6 std::ostream& operator<< (std::ostream & *o,* const mha_complex_t & *c*) [inline]

ostream operator for **mha_complex_t** (p. 431)

6.121.3.7 std::istream& operator>> (std::istream & *i,* mha_complex_t & *c*) [inline]

preliminary istream operator for **mha_complex_t** (p. 431) without error checking

6.122 mha_signal_fft.h File Reference

Classes

- class **MHASignal::fft_t**

Namespaces

- **MHASignal**
  Namespace for audio signal handling and processing classes.

6.123 mha_tablelookup.cpp File Reference

6.124 mha_tablelookup.hh File Reference

Header file for table lookup classes.

Classes

- class **MHATableLookup::table_t**
- class **MHATableLookup::linear_table_t**
  Class for interpolation with equidistant x values.
- class **MHATableLookup::xy_table_t**
  Class for interpolation with non-equidistant x values.

Namespaces

- **MHATableLookup**
  Namespace for table lookup classes.

6.124.1 Detailed Description

Header file for table lookup classes.

6.125 mha_tcp.cpp File Reference

Classes

- class **MHA_TCP::sock_init_t**

Namespaces

- **MHA_TCP**
  A Namespace for TCP helper classes.

Macros

- #define **INVALID_SOCKET** (-1)
- #define **SOCKET_ERROR** (-1)
- #define **closesocket**(fd) (close((fd)))
- #define **ASYNC_CONNECT_STARTED** EINPROGRESS

Typedefs

- typedef int **SOCKET**

Functions

- std::string **MHA_TCP::STRERROR** (int err)
  Portable conversion from error number to error string.
- std::string **MHA_TCP::HSTRERROR** (int err)
  Portable conversion from hostname error number to error string.
- int **MHA_TCP::N_ERRNO** ()
  Portable access to last network error number.
- int **MHA_TCP::H_ERRNO** ()
  Portable access to last hostname error number.
- int **MHA_TCP::G_ERRNO** ()
  Portable access to last non-network error number.
- static sockaddr_in **host_port_to_sock_addr** (const std::string &host, unsigned short port)
- static **SOCKET tcp_connect_to** (const std::string &host, unsigned short port)
- static **SOCKET tcp_connect_to_with_timeout** (const std::string &host, unsigned short port, **Timeout_Watcher** &timeout_watcher)
- static void * **thread_start_func** (void *thread)

Variables

- class **MHA_TCP::sock_init_t MHA_TCP::sock_initializer**

6.125.1 Macro Definition Documentation

6.125.1.1 #define INVALID_SOCKET (-1)

6.125.1.2 #define SOCKET_ERROR (-1)

6.125.1.3 #define closesocket(*fd*) (close((fd)))

6.125.1.4 #define ASYNC_CONNECT_STARTED EINPROGRESS

6.125.2 Typedef Documentation

6.125.2.1 typedef int SOCKET

6.125.3 Function Documentation

6.125.3.1 static sockaddr_in host_port_to_sock_addr (const std::string & *host,* unsigned short *port*) [static]

6.125.3.2 static SOCKET tcp_connect_to (const std::string & *host,* unsigned short *port*) [static]

6.125.3.3 static SOCKET tcp_connect_to_with_timeout (const std::string & *host,* unsigned short *port,* Timeout_Watcher & *timeout_watcher*) [static]

6.125.3.4 static void* thread_start_func (void * *thread*) [static]

6.126 mha_tcp.hh File Reference

Classes

- struct **MHA_TCP::OS_EVENT_TYPE**
- class **MHA_TCP::Wakeup_Event**
  A base class for asynchronous wakeup events.
- class **MHA_TCP::Async_Notify**

Portable Multiplexable cross-thread notification.

- class **MHA_TCP::Event_Watcher**
  OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.
- class **MHA_TCP::Timeout_Event**
- class **MHA_TCP::Timeout_Watcher**
  OS-independent event watcher with internal fixed-end-time timeout.
- class **MHA_TCP::Sockread_Event**
  Watch socket for incoming data.
- class **MHA_TCP::Sockwrite_Event**
- class **MHA_TCP::Sockaccept_Event**
- class **MHA_TCP::Connection**
  Connection *(p. 471) handles Communication between client and server, is used on both sides.*
- class **MHA_TCP::Server**
- class **MHA_TCP::Client**
  A portable class for a tcp client connections.
- class **MHA_TCP::Thread**
  A very simple class for portable threads.

Namespaces

- **MHA_TCP**
  A Namespace for TCP helper classes.

Macros

- #define **Sleep**(x) usleep((x)∗1000);

Typedefs

- typedef int **MHA_TCP::SOCKET**

Functions

- std::string **MHA_TCP::STRERROR** (int err)
  Portable conversion from error number to error string.
- std::string **MHA_TCP::HSTRERROR** (int err)
  Portable conversion from hostname error number to error string.
- int **MHA_TCP::N_ERRNO** ()
  Portable access to last network error number.
- int **MHA_TCP::H_ERRNO** ()
  Portable access to last hostname error number.
- int **MHA_TCP::G_ERRNO** ()
  Portable access to last non-network error number.
- double **MHA_TCP::dtime** ()
  Time access function for system's high resolution time, retrieve current time as double.
- double **MHA_TCP::dtime** (const struct timeval &tv)
  Time access function for unix' high resolution time, converts struct timeval to double.
- struct timeval **MHA_TCP::stime** (double d)
  Time access function for unix' high resolution time, converts time from double to struct timeval.

6.126.1 Macro Definition Documentation

6.126.1.1 #define Sleep(*x*) usleep((x)∗1000);

6.127 mha_toolbox.h File Reference

6.128 mha_windowparser.cpp File Reference

Variables

- float(∗ **wnd_funs** [ ])(float)

6.128.1 Variable Documentation

6.128.1.1 float(∗ wnd_funs[ ])(float)

6.129 mha_windowparser.h File Reference

Classes

- class **MHAWindow::base_t**
  Common base for window types.
- class **MHAWindow::fun_t**
  Generic window based on a generator function.
- class **MHAWindow::rect_t**
  Rectangular window.
- class **MHAWindow::bartlett_t**
  Bartlett window.
- class **MHAWindow::hanning_t**
  von-Hann window
- class **MHAWindow::hamming_t**
  Hamming window.
- class **MHAWindow::blackman_t**
  Blackman window.
- class **MHAWindow::user_t**
  User defined window.
- class **MHAParser::window_t**
  MHA configuration interface for a window function generator.

Namespaces

- **MHAWindow**
  Collection of Window types.
- **MHAParser**
  Name space for the openMHA-Parser configuration language.

Functions

- float **MHAWindow::rect** (float)
  Rectangular window function.
- float **MHAWindow::bartlett** (float)
  Bartlett window function.
- float **MHAWindow::hanning** (float)
  Hanning window function.
- float **MHAWindow::hamming** (float)
  Hamming window function.
- float **MHAWindow::blackman** (float)
  Blackman window function.

6.130 mhachain.cpp File Reference

Classes

- class **mhachain::mhachain_t**

Namespaces

- **mhachain**

6.131 mhafw_lib.cpp File Reference

6.132 mhafw_lib.h File Reference

Classes

- class **io_lib_t**
  Class for loading MHA sound IO module.
- class **fw_vars_t**
- class **fw_t**

6.133 MHAIOalsa.cpp File Reference

Classes

- class **alsa_base_t**
- class **alsa_dev_par_parser_t**

  Parser variables corresponding to one alsa device.
- class **alsa_t**< **T** >

  Our representation of one alsa device.
- class **io_alsa_t**

  MHA IO interface class for ALSA IO.

Macros

- #define **DBG**(x) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)
- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOalsa_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOalsa_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOalsa_IOStart
- #define **IOStop** MHA_STATIC_MHAIOalsa_IOStop
- #define **IORelease** MHA_STATIC_MHAIOalsa_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOalsa_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOalsa_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOalsa_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOalsa_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void ∗proc_↩ handle, **IOStartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_↩ event, void ∗stop_handle, void ∗∗handle)

  IO library initialization function, called by framework after loading this IO library into the MHA process.
- int **IOPrepare** (void ∗handle, int nch_in, int nch_out)

  IO library prepare function, called after the MHA prepared the processing plugins.
- int **IOStart** (void ∗handle)
- int **IOStop** (void ∗handle)
- int **IORelease** (void ∗handle)
- int **IOSetVar** (void ∗handle, const char ∗command, char ∗retval, unsigned int maxretlen)
- const char ∗ **IOStrError** (void ∗handle, int err)
- void **IODestroy** (void ∗handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.133.1 Macro Definition Documentation

6.133.1.1 #define DBG(*x*) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)

6.133.1.2 #define ERR_SUCCESS 0

6.133.1.3 #define ERR_IHANDLE -1

6.133.1.4 #define ERR_USER -1000

6.133.1.5 #define MAX_USER_ERR 0x500

6.133.1.6 #define IOInit MHA_STATIC_MHAIOalsa_IOInit

6.133.1.7 #define IOPrepare MHA_STATIC_MHAIOalsa_IOPrepare

6.133.1.8 #define IOStart MHA_STATIC_MHAIOalsa_IOStart

6.133.1.9 #define IOStop MHA_STATIC_MHAIOalsa_IOStop

6.133.1.10 #define IORelease MHA_STATIC_MHAIOalsa_IORelease

6.133.1.11 #define IOSetVar MHA_STATIC_MHAIOalsa_IOSetVar

6.133.1.12 #define IOStrError MHA_STATIC_MHAIOalsa_IOStrError

6.133.1.13 #define IODestroy MHA_STATIC_MHAIOalsa_IODestroy

6.133.1.14 #define dummy_interface_test MHA_STATIC_MHAIOalsa_dummy_interface_test

6.133.2 Function Documentation

6.133.2.1 int IOInit (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

IO library initialization function, called by framework after loading this IO library into the MHA process.

Gives plugin callback functions and callback handles to interact with the MHA framework.

Parameters

handle	output parameter. IO library returns pointer to void to the caller via this parameter. All other function calls from the MHA framework will use this handle.

6.133.2.2 int IOPrepare (void * *handle*, int *nch_in*, int *nch_out*)

IO library prepare function, called after the MHA prepared the processing plugins.

6.133.2.3 int IOStart (void * *handle*)

6.133.2.4 int IOStop (void * *handle*)

6.133.2.5 int IORelease (void * *handle*)

6.133.2.6 int IOSetVar (void * *handle*, const char * *command*, char * *retval*, unsigned int *maxretlen*)

6.133.2.7 const char* IOStrError (void * *handle*, int *err*)

6.133.2.8 void IODestroy (void * *handle*)

6.133.2.9 void dummy_interface_test (void)

6.133.3 Variable Documentation

6.133.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.134 MHAIOFile.cpp File Reference

Classes

- class **io_file_t**
  File IO.

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " << x << std::endl
- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOFile_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOFile_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOFile_IOStart
- #define **IOStop** MHA_STATIC_MHAIOFile_IOStop
- #define **IORelease** MHA_STATIC_MHAIOFile_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOFile_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOFile_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOFile_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOFile_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↩ handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↩ event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *command, char *retval, unsigned int maxretlen)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.134.1 Macro Definition Documentation

6.134.1.1 #define DEBUG(*x*) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " << x << std::endl

6.134.1.2 #define ERR_SUCCESS 0

6.134.1.3 #define ERR_IHANDLE -1

6.134.1.4 #define ERR_USER -1000

6.134.1.5 #define MAX_USER_ERR 0x500

6.134.1.6 #define IOInit MHA_STATIC_MHAIOFile_IOInit

6.134.1.7 #define IOPrepare MHA_STATIC_MHAIOFile_IOPrepare

6.134.1.8 #define IOStart MHA_STATIC_MHAIOFile_IOStart

6.134.1.9 #define IOStop MHA_STATIC_MHAIOFile_IOStop

6.134.1.10 #define IORelease MHA_STATIC_MHAIOFile_IORelease

6.134.1.11 #define IOSetVar MHA_STATIC_MHAIOFile_IOSetVar

6.134.1.12 #define IOStrError MHA_STATIC_MHAIOFile_IOStrError

6.134.1.13 #define IODestroy MHA_STATIC_MHAIOFile_IODestroy

6.134.1.14 #define dummy_interface_test MHA_STATIC_MHAIOFile_dummy_interface_test

6.134.2 Function Documentation

6.134.2.1 int IOInit (
int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.134.2.2 int IOPrepare (
void * *handle,*
int *nch_in,*
int *nch_out*)

6.134.2.3 int IOStart (
void * *handle*)

6.134.2.4 int IOStop (void * *handle*)

6.134.2.5 int IORelease (void * *handle*)

6.134.2.6 int IOSetVar (void * *handle,* const char * *command,* char * *retval,* unsigned int *maxretlen*)

6.134.2.7 const char* IOStrError (void * *handle,* int *err*)

6.134.2.8 void IODestroy (void * *handle*)

6.134.2.9 void dummy_interface_test (void)

6.134.3 Variable Documentation

6.134.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.135 MHAIOJack.cpp File Reference

Classes

- class **MHAIOJack::io_jack_t**
  Main class for JACK IO.

Namespaces

- **MHAIOJack**
  JACK IO.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOJack_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOJack_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOJack_IOStart
- #define **IOStop** MHA_STATIC_MHAIOJack_IOStop
- #define **IORelease** MHA_STATIC_MHAIOJack_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOJack_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOJack_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOJack_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOJack_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void ∗proc_↩ handle, **IOStartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_↩ event, void ∗stop_handle, void ∗∗handle)
- int **IOPrepare** (void ∗handle, int nch_in, int nch_out)
- int **IOStart** (void ∗handle)
- int **IOStop** (void ∗handle)
- int **IORelease** (void ∗handle)
- int **IOSetVar** (void ∗handle, const char ∗command, char ∗retval, unsigned int maxretlen)
- const char ∗ **IOStrError** (void ∗handle, int err)
- void **IODestroy** (void ∗handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

6.135.1 Macro Definition Documentation

6.135.1.1 #define ERR_SUCCESS 0

6.135.1.2 #define ERR_IHANDLE -1

6.135.1.3 #define ERR_USER -1000

6.135.1.4 #define MAX_USER_ERR 0x500

6.135.1.5 #define IOInit MHA_STATIC_MHAIOJack_IOInit

6.135.1.6 #define IOPrepare MHA_STATIC_MHAIOJack_IOPrepare

6.135.1.7 #define IOStart MHA_STATIC_MHAIOJack_IOStart

6.135.1.8 #define IOStop MHA_STATIC_MHAIOJack_IOStop

6.135.1.9 #define IORelease MHA_STATIC_MHAIOJack_IORelease

6.135.1.10 #define IOSetVar MHA_STATIC_MHAIOJack_IOSetVar

6.135.1.11 #define IOStrError MHA_STATIC_MHAIOJack_IOStrError

6.135.1.12 #define IODestroy MHA_STATIC_MHAIOJack_IODestroy

6.135.1.13 #define dummy_interface_test MHA_STATIC_MHAIOJack_dummy_interface_test

6.135.2 Function Documentation

6.135.2.1 int IOInit (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.135.2.2 int IOPrepare (

void * *handle,*
int *nch_in,*
int *nch_out*)

6.135.2.3 int IOStart (

void * *handle*)

6.135.2.4 int IOStop (

void * *handle*)

6.135.2.5 int IORelease (

void * *handle*)

6.135.2.6 int IOSetVar (
void * *handle,*
const char * *command,*
char * *retval,*
unsigned int *maxretlen*)

6.135.2.7 const char* IOStrError (
void * *handle,*
int *err*)

6.135.2.8 void IODestroy (
void * *handle*)

6.135.2.9 void dummy_interface_test (
void)

6.135.3 Variable Documentation

6.135.3.1 char user_err_msg[MAX_USER_ERR] = "" [static]

6.136 MHAIOJackdb.cpp File Reference

Classes

- class **MHAIOJackdb::io_jack_t**
  Main class for JACK IO.

Namespaces

- **MHAIOJackdb**

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOJackdb_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOJackdb_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOJackdb_IOStart
- #define **IOStop** MHA_STATIC_MHAIOJackdb_IOStop
- #define **IORelease** MHA_STATIC_MHAIOJackdb_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOJackdb_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOJackdb_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOJackdb_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOJackdb_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void ∗proc_↩ handle, **IOStartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_↩ event, void ∗stop_handle, void ∗∗handle)
- int **IOPrepare** (void ∗handle, int nch_in, int nch_out)
- int **IOStart** (void ∗handle)
- int **IOStop** (void ∗handle)
- int **IORelease** (void ∗handle)
- int **IOSetVar** (void ∗handle, const char ∗command, char ∗retval, unsigned int maxretlen)
- const char ∗ **IOStrError** (void ∗handle, int err)
- void **IODestroy** (void ∗handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

6.136.1 Macro Definition Documentation

6.136.1.1 #define ERR_SUCCESS 0

6.136.1.2 #define ERR_IHANDLE -1

6.136.1.3 #define ERR_USER -1000

6.136.1.4 #define MAX_USER_ERR 0x500

6.136.1.5 #define IOInit MHA_STATIC_MHAIOJackdb_IOInit

6.136.1.6 #define IOPrepare MHA_STATIC_MHAIOJackdb_IOPrepare

6.136.1.7 #define IOStart MHA_STATIC_MHAIOJackdb_IOStart

6.136.1.8 #define IOStop MHA_STATIC_MHAIOJackdb_IOStop

6.136.1.9 #define IORelease MHA_STATIC_MHAIOJackdb_IORelease

6.136.1.10 #define IOSetVar MHA_STATIC_MHAIOJackdb_IOSetVar

6.136.1.11 #define IOStrError MHA_STATIC_MHAIOJackdb_IOStrError

6.136.1.12 #define IODestroy MHA_STATIC_MHAIOJackdb_IODestroy

6.136.1.13 #define dummy_interface_test MHA_STATIC_MHAIOJackdb_dummy_interface_test

6.136.2 Function Documentation

6.136.2.1 int IOInit (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.136.2.2 int IOPrepare (

void * *handle,*
int *nch_in,*
int *nch_out*)

6.136.2.3 int IOStart (

void * *handle*)

6.136.2.4 int IOStop (

void * *handle*)

6.136.2.5 int IORelease (

void * *handle*)

6.136.2.6 int IOSetVar (

void * *handle,*
const char * *command,*
char * *retval,*
unsigned int *maxretlen*)

6.136.2.7 const char* IOStrError (

void * *handle,*
int *err*)

6.136.2.8 void IODestroy (

void * *handle*)

6.136.2.9 void dummy_interface_test (

void)

6.136.3 Variable Documentation

6.136.3.1 char user_err_msg[MAX_USER_ERR] = "" [static]

6.137 MHAIOParser.cpp File Reference

Classes

- class **io_parser_t**
  Main class for Parser IO.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOParser_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOParser_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOParser_IOStart
- #define **IOStop** MHA_STATIC_MHAIOParser_IOStop
- #define **IORelease** MHA_STATIC_MHAIOParser_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOParser_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOParser_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOParser_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOParser_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float, **IOProcessEvent_t** proc_event, void ∗proc_handle, **IO↩ StartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_event, void ∗stop_handle, void ∗∗handle)
- int **IOPrepare** (void ∗handle, int nch_in, int nch_out)
- int **IOStart** (void ∗handle)
- int **IOStop** (void ∗handle)
- int **IORelease** (void ∗handle)
- int **IOSetVar** (void ∗handle, const char ∗command, char ∗retval, unsigned int maxretlen)
- const char ∗ **IOStrError** (void ∗handle, int err)
- void **IODestroy** (void ∗handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.137.1 Macro Definition Documentation

6.137.1.1 #define ERR_SUCCESS 0

6.137.1.2 #define ERR_IHANDLE -1

6.137.1.3 #define ERR_USER -1000

6.137.1.4 #define MAX_USER_ERR 0x500

6.137.1.5 #define IOInit MHA_STATIC_MHAIOParser_IOInit

6.137.1.6 #define IOPrepare MHA_STATIC_MHAIOParser_IOPrepare

6.137.1.7 #define IOStart MHA_STATIC_MHAIOParser_IOStart

6.137.1.8 #define IOStop MHA_STATIC_MHAIOParser_IOStop

6.137.1.9 #define IORelease MHA_STATIC_MHAIOParser_IORelease

6.137.1.10 #define IOSetVar MHA_STATIC_MHAIOParser_IOSetVar

6.137.1.11 #define IOStrError MHA_STATIC_MHAIOParser_IOStrError

6.137.1.12 #define IODestroy MHA_STATIC_MHAIOParser_IODestroy

6.137.1.13 #define dummy_interface_test MHA_STATIC_MHAIOParser_dummy_interface_test

6.137.2 Function Documentation

6.137.2.1 int IOInit (

int *fragsize,*
float *,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.137.2.2 int IOPrepare (

void * *handle,*
int *nch_in,*
int *nch_out*)

6.137.2.3 int IOStart (void * *handle*)

6.137.2.4 int IOStop (void * *handle*)

6.137.2.5 int IORelease (void * *handle*)

6.137.2.6 int IOSetVar (void * *handle,* const char * *command,* char * *retval,* unsigned int *maxretlen*)

6.137.2.7 const char* IOStrError (void * *handle,* int *err*)

6.137.2.8 void IODestroy (void * *handle*)

6.137.2.9 void dummy_interface_test (void)

6.137.3 Variable Documentation

6.137.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.138 MHAIOPortAudio.cpp File Reference

Classes

- class **MHAIOPortAudio::device_info_t**
- class **MHAIOPortAudio::io_portaudio_t**

  Main class for Portaudio sound IO.

Namespaces

- **MHAIOPortAudio**

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOPortAudio_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOPortAudio_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOPortAudio_IOStart
- #define **IOStop** MHA_STATIC_MHAIOPortAudio_IOStop
- #define **IORelease** MHA_STATIC_MHAIOPortAudio_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOPortAudio_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOPortAudio_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOPortAudio_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOPortAudio_dummy_interface_↩ test

Functions

- static std::string **MHAIOPortAudio::parserFriendlyName** (const std::string &in)
- int **portaudio_callback** (const void ∗input, void ∗output, unsigned long frameCount, const PaStreamCallbackTimeInfo ∗timeInfo, PaStreamCallbackFlags statusFlags, void ∗user↩ Data)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void ∗proc_↩ handle, **IOStartedEvent_t** start_event, void ∗start_handle, **IOStoppedEvent_t** stop_↩ event, void ∗stop_handle, void ∗∗handle)
- int **IOPrepare** (void ∗handle, int nch_in, int nch_out)
- int **IOStart** (void ∗handle)
- int **IOStop** (void ∗handle)
- int **IORelease** (void ∗handle)
- int **IOSetVar** (void ∗handle, const char ∗command, char ∗retval, unsigned int maxretlen)
- const char ∗ **IOStrError** (void ∗handle, int err)
- void **IODestroy** (void ∗handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""
- PaStreamCallback **portaudio_callback**

6.138.1 Macro Definition Documentation

6.138.1.1 #define ERR_SUCCESS 0

6.138.1.2 #define ERR_IHANDLE -1

6.138.1.3 #define ERR_USER -1000

6.138.1.4 #define MAX_USER_ERR 0x500

6.138.1.5 #define IOInit MHA_STATIC_MHAIOPortAudio_IOInit

6.138.1.6 #define IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare

6.138.1.7 #define IOStart MHA_STATIC_MHAIOPortAudio_IOStart

6.138.1.8 #define IOStop MHA_STATIC_MHAIOPortAudio_IOStop

6.138.1.9 #define IORelease MHA_STATIC_MHAIOPortAudio_IORelease

6.138.1.10 #define IOSetVar MHA_STATIC_MHAIOPortAudio_IOSetVar

6.138.1.11 #define IOStrError MHA_STATIC_MHAIOPortAudio_IOStrError

6.138.1.12 #define IODestroy MHA_STATIC_MHAIOPortAudio_IODestroy

6.138.1.13 #define dummy_interface_test MHA_STATIC_MHAIOPortAudio_dummy_interface_test

6.138.2 Function Documentation

6.138.2.1 int portaudio_callback (

const void * *input,*
void * *output,*
unsigned long *frameCount,*
const PaStreamCallbackTimeInfo * *timeInfo,*
PaStreamCallbackFlags *statusFlags,*
void * *userData*)

6.138.2.2 int IOInit (

int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.138.2.3 int IOPrepare (
void * *handle,*
int *nch_in,*
int *nch_out*)

6.138.2.4 int IOStart (
void * *handle*)

6.138.2.5 int IOStop (
void * *handle*)

6.138.2.6 int IORelease (
void * *handle*)

6.138.2.7 int IOSetVar (
void * *handle,*
const char * *command,*
char * *retval,*
unsigned int *maxretlen*)

6.138.2.8 const char* IOStrError (
void * *handle,*
int *err*)

6.138.2.9 void IODestroy (
void * *handle*)

6.138.2.10 void dummy_interface_test (
void)

6.138.3 Variable Documentation

6.138.3.1 char user_err_msg[MAX_USER_ERR] = "" [static]

6.138.3.2 PaStreamCallback portaudio_callback

6.139 MHAIOTCP.cpp File Reference

Classes

- class **io_tcp_parser_t**
  The parser interface of the IOTCP library.
- class **io_tcp_sound_t**
  Sound data handling of io tcp library.
- union **io_tcp_sound_t::float_union**
  This union helps in conversion of floats from host byte order to network byte order and back again.
- class **io_tcp_fwcb_t**
  TCP sound-io library's interface to the framework callbacks.
- class **io_tcp_t**
  The tcp sound io library.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x2000
- #define **MHA_ErrorMsg2**(x, y) **MHA_Error**(__FILE__,__LINE__,(x),(y))
- #define **MHA_ErrorMsg3**(x, y, z) **MHA_Error**(__FILE__,__LINE__,(x),(y),(z))
- #define **MIN_TCP_PORT** 0
- #define **MIN_TCP_PORT_STR** "0"
- #define **MAX_TCP_PORT** 65535
- #define **MAX_TCP_PORT_STR** "65535"
- #define **IOInit** MHA_STATIC_MHAIOTCP_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOTCP_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOTCP_IOStart
- #define **IOStop** MHA_STATIC_MHAIOTCP_IOStop
- #define **IORelease** MHA_STATIC_MHAIOTCP_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIOTCP_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOTCP_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOTCP_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOTCP_dummy_interface_test

Functions

- static int **copy_error** (**MHA_Error** &e)
- static void * **thread_startup_function** (void *parameter)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↩ handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↩ event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int num_inchannels, int num_outchannels)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *cmd, char *retval, unsigned int len)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.139.1 Macro Definition Documentation

6.139.1.1 #define ERR_SUCCESS 0

6.139.1.2 #define ERR_IHANDLE -1

6.139.1.3 #define ERR_USER -1000

6.139.1.4 #define MAX_USER_ERR 0x2000

6.139.1.5 #define MHA_ErrorMsg2(

x,
y) **MHA_Error(__FILE__,__LINE__,(x),(y))**

6.139.1.6 #define MHA_ErrorMsg3(

x,
y,
z) **MHA_Error(__FILE__,__LINE__,(x),(y),(z))**

6.139.1.7 #define MIN_TCP_PORT 0

6.139.1.8 #define MIN_TCP_PORT_STR "0"

6.139.1.9 #define MAX_TCP_PORT 65535

6.139.1.10 #define MAX_TCP_PORT_STR "65535"

6.139.1.11 #define IOInit MHA_STATIC_MHAIOTCP_IOInit

6.139.1.12 #define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare

6.139.1.13 #define IOStart MHA_STATIC_MHAIOTCP_IOStart

6.139.1.14 #define IOStop MHA_STATIC_MHAIOTCP_IOStop

6.139.1.15 #define IORelease MHA_STATIC_MHAIOTCP_IORelease

6.139.1.16 #define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar

6.139.1.17 #define IOStrError MHA_STATIC_MHAIOTCP_IOStrError

6.139.1.18 #define IODestroy MHA_STATIC_MHAIOTCP_IODestroy

6.139.1.19 #define dummy_interface_test MHA_STATIC_MHAIOTCP_dummy_interface_test

6.139.2 Function Documentation

6.139.2.1 static int copy_error (MHA_Error & *e*) [static]

6.139.2.2 static void* thread_startup_function (void * *parameter*) [static]

6.139.2.3 int IOInit (
int *fragsize,*
float *samplerate,*
IOProcessEvent_t *proc_event,*
void * *proc_handle,*
IOStartedEvent_t *start_event,*
void * *start_handle,*
IOStoppedEvent_t *stop_event,*
void * *stop_handle,*
void ** *handle*)

6.139.2.4 int IOPrepare (
void * *handle,*
int *num_inchannels,*
int *num_outchannels*)

6.139.2.5 int IOStart (
void * *handle*)

6.139.2.6 int IOStop (
void * *handle*)

6.139.2.7 int IORelease (
void * *handle*)

6.139.2.8 int IOSetVar (
void * *handle,*
const char * *cmd,*
char * *retval,*
unsigned int *len*)

6.139.2.9 const char* IOStrError (
void * *handle,*
int *err*)

6.139.2.10 void IODestroy (
void * *handle*)

6.139.2.11 void dummy_interface_test (void)

6.139.3 Variable Documentation

6.139.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.140 mhajack.cpp File Reference

Functions

- static void **jack_error_handler** (const char *msg)
- static int **dummy_jack_proc_cb** (jack_nframes_t, void *)
- void **make_friendly_number** (jack_default_audio_sample_t &x)

Variables

- char **last_jack_err_msg** [**MAX_USER_ERR**] = ""
- int **last_jack_err** = 0

6.140.1 Function Documentation

6.140.1.1 static void jack_error_handler (const char * *msg*) [static]

6.140.1.2 static int dummy_jack_proc_cb (jack_nframes_t , void *) [static]

6.140.1.3 void make_friendly_number (jack_default_audio_sample_t & *x*) [inline]

6.140.2 Variable Documentation

6.140.2.1 char last_jack_err_msg[MAX_USER_ERR] = ""

6.140.2.2 int last_jack_err = 0

6.141 mhajack.h File Reference

Classes

- class **MHAJack::port_t**
  Class for one channel/port.
- class **MHAJack::client_t**
  Generic asynchronous JACK client.
- class **MHAJack::client_noncont_t**
  Generic client for synchronous playback and recording of waveform fragments.
- class **MHAJack::client_avg_t**
  Generic JACK client for averaging a system response across time.

Namespaces

- **MHAJack**

  Classes and functions for openMHA and JACK interaction.

Macros

- #define **MHAJACK_FW_STARTED** 1
- #define **MHAJACK_STOPPED** 2
- #define **MHAJACK_STARTING** 8
- #define **IO_ERROR_JACK** 11
- #define **IO_ERROR_MHAJACKLIB** 12
- #define **MAX_USER_ERR** 0x500

Functions

- void **MHAJack::io** (**mha_wave_t** ∗s_out, **mha_wave_t** ∗s_in, const std::string &name, const std::vector< std::string > &p_out, const std::vector< std::string > &p_in, float ∗srate=NULL, unsigned int ∗fragsize=NULL, bool use_jack_transport=false)

  Functional form of generic client for synchronous playback and recording of waveform fragments.
- std::vector< unsigned int > **MHAJack::get_port_capture_latency** (const std::vector< std::string > &ports)

  Return the JACK port latency of ports.
- std::vector< int > **MHAJack::get_port_capture_latency_int** (const std::vector< std↩ ::string > &ports)

  Return the JACK port latency of ports.
- std::vector< unsigned int > **MHAJack::get_port_playback_latency** (const std::vector< std::string > &ports)

  Return the JACK port latency of ports.
- std::vector< int > **MHAJack::get_port_playback_latency_int** (const std::vector< std↩ ::string > &ports)

Variables

- char **last_jack_err_msg** [**MAX_USER_ERR**]

6.141.1 Macro Definition Documentation

6.141.1.1 #define MHAJACK_FW_STARTED 1

6.141.1.2 #define MHAJACK_STOPPED 2

6.141.1.3 #define MHAJACK_STARTING 8

6.141.1.4 #define IO_ERROR_JACK 11

6.141.1.5 #define IO_ERROR_MHAJACKLIB 12

6.141.1.6 #define MAX_USER_ERR 0x500

6.141.2 Variable Documentation

6.141.2.1 char last_jack_err_msg[MAX_USER_ERR]

6.142 mhamain.cpp File Reference

Classes

- class **mhaserver_t**
  MHA Framework listening on TCP port for commands.

Macros

- #define **MAX_LINE_LENGTH** 0x100000
- #define **HELP_TEXT**
- #define **GREETING_TEXT**

Functions

- void **create_lock** (unsigned int p, std::string s)
- void **remove_lock** (unsigned int p)
- int **mhamain** (int argc, char ∗argv[ ])

6.142.1 Macro Definition Documentation

6.142.1.1 #define MAX_LINE_LENGTH 0x100000

6.142.1.2 #define HELP_TEXT

6.142.1.3 #define GREETING_TEXT

6.142.2 Function Documentation

6.142.2.1 void create_lock (unsigned int *p,* std::string *s*)

6.142.2.2 void remove_lock (unsigned int *p*)

6.142.2.3 int mhamain (int *argc,* char * *argv[]*)

6.143 mhapluginloader.cpp File Reference

6.144 mhapluginloader.h File Reference

Classes

- class **PluginLoader::config_file_splitter_t**
- class **PluginLoader::fourway_processor_t**

  This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.
- class **PluginLoader::mhapluginloader_t**
- class **MHAParser::mhapluginloader_t**

  Class to create a plugin loader in a parser, including the load logic.

Namespaces

- **PluginLoader**
- **MHAParser**

  Name space for the openMHA-Parser configuration language.

Functions

- const char * **PluginLoader::mhastrdomain** (**mha_domain_t**)
- void **PluginLoader::mhaconfig_compare** (const **mhaconfig_t** &req, const **mhaconfig_t** &avail, const std::string &pref="")

  *Compare two **mhaconfig_t** (p. 504) structures, and report differences as an error.*

6.145 mhasndfile.cpp File Reference

Functions

- void **write_wave** (const **mha_wave_t** &sig, const char *fname, const float &srate, const int &format)
- unsigned int **validator_channels** (std::vector< int > channel_map, unsigned int **channels**)
- unsigned int **validator_length** (unsigned int maxlen, unsigned int frames, unsigned int startpos)

6.145.1 Function Documentation

6.145.1.1 void write_wave (
const **mha_wave_t** & *sig,*
const char * *fname,*
const float & *srate,*
const int & *format*)

6.145.1.2 unsigned int validator_channels (
std::vector< int > *channel_map,*
unsigned int *channels*)

6.145.1.3 unsigned int validator_length (
unsigned int *maxlen,*
unsigned int *frames,*
unsigned int *startpos*)

6.146 mhasndfile.h File Reference

Classes

- class **MHASndFile::sf_t**
- class **MHASndFile::sf_wave_t**

Namespaces

- **MHASndFile**

Functions

- void **write_wave** (const **mha_wave_t** &sig, const char *fname, const float &srate=44100, const int &format=SF_FORMAT_WAV|SF_FORMAT_FLOAT|SF_ENDIAN_FILE)

6.146.1 Function Documentation

6.146.1.1 void write_wave (const mha_wave_t & *sig,* const char * *fname,* const float & *srate =* 44100*,* const int & *format =* SF_FORMAT_WAV|SF_FORMAT_FLOAT|SF_ENDIAN↩ _FILE)

6.147 multibandcompressor.cpp File Reference

Classes

- class **multibandcompressor::plugin_signals_t**
- class **multibandcompressor::fftfb_plug_t**
- class **multibandcompressor::interface_t**

Namespaces

- **multibandcompressor**

6.148 nlms_wave.cpp File Reference

Classes

- class **rt_nlms_t**
- class **nlms_t**

Macros

- #define **NORMALIZATION_TYPES** "[none default sum]"
- #define **NORM_NONE** 0
- #define **NORM_DEFAULT** 1
- #define **NORM_SUM** 2
- #define **ESTIMATION_TYPES** "[previous current]"
- #define **ESTIM_PREV** 0
- #define **ESTIM_CUR** 1

Functions

- void **make_friendly_number_by_limiting** (**mha_real_t** &x)

6.148.1 Macro Definition Documentation

6.148.1.1 #define NORMALIZATION_TYPES "[none default sum]"

6.148.1.2 #define NORM_NONE 0

6.148.1.3 #define NORM_DEFAULT 1

6.148.1.4 #define NORM_SUM 2

6.148.1.5 #define ESTIMATION_TYPES "[previous current]"

6.148.1.6 #define ESTIM_PREV 0

6.148.1.7 #define ESTIM_CUR 1

6.148.2 Function Documentation

6.148.2.1 void make_friendly_number_by_limiting (**mha_real_t** & ***x***) [inline]

6.149 noise.cpp File Reference

Classes

- class **cfg_t**
- class **noise_t**

6.150 noisePowProposedScale.cpp File Reference

Classes

- class **noisePowProposedScale::noisePowProposed**
- class **noisePowProposedScale::interface_t**

Namespaces

- **noisePowProposedScale**

Macros

- #define **POWSPEC_FACTOR** 0.0025

6.150.1 Macro Definition Documentation

6.150.1.1 #define POWSPEC_FACTOR 0.0025

6.151 overlapadd.cpp File Reference

Classes

- class **overlapadd::overlapadd_t**
- class **overlapadd::overlapadd_if_t**

Namespaces

- **overlapadd**

6.152 plingploing.cpp File Reference

Classes

- class **plingploing::plingploing_t**
  Run-time configuration of the plingploing music generator.
- class **plingploing::if_t**
  Plugin class of the plingploing music generator.

Namespaces

- **plingploing**
  All classes for the plingploing music generator live in this namespace.

Functions

- double **plingploing::drand** (double a, double b)

6.153 pluginbrowser.cpp File Reference

6.154 pluginbrowser.h File Reference

Classes

- class **plugindescription_t**
- class **pluginloader_t**
- class **pluginbrowser_t**

6.155 prediction_error.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**prediction_error::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

Functions

- void **make_friendly_number_by_limiting** (**mha_real_t** &x)

6.155.1 Macro Definition Documentation

6.155.1.1 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &prediction_error::update_cfg)

6.155.1.2 #define INSERT_PATCH(*var*) insert_member(var); PATCH_VAR(var)

6.155.2 Function Documentation

6.155.2.1 void make_friendly_number_by_limiting (mha_real_t & *x*) [inline]

6.156 prediction_error.h File Reference

Classes

- class **prediction_error_config**
- class **prediction_error**

6.157 resampling.cpp File Reference

Classes

- class **MHAPlugin_Resampling::resampling_t**
- class **MHAPlugin_Resampling::resampling_if_t**

Namespaces

- **MHAPlugin_Resampling**

6.158 rmslevel.cpp File Reference

Classes

- class **mon_t**
- class **rmslevel_t**
- class **rmslevel_if_t**

6.159 route.cpp File Reference

Classes

- class **route::process_t**
- class **route::interface_t**

Namespaces

- **route**

6.160 save_spec.cpp File Reference

Classes

- class **save_spec_t**

6.161 save_wave.cpp File Reference

Classes

- class **save_wave_t**

6.162 shadowfilter_begin.cpp File Reference

Classes

- class **shadowfilter_begin::cfg_t**
- class **shadowfilter_begin::shadowfilter_begin_t**

Namespaces

- **shadowfilter_begin**

6.163 shadowfilter_end.cpp File Reference

Classes

- class **shadowfilter_end::cfg_t**
- class **shadowfilter_end::shadowfilter_end_t**

Namespaces

- **shadowfilter_end**

6.164 sine.cpp File Reference

Classes

- struct **sine_cfg_t**
- class **sine_t**

6.165 smoothgains_bridge.cpp File Reference

Classes

- class **smoothgains_bridge::smoothspec_wrap_t**
- class **smoothgains_bridge::overlapadd_if_t**

Namespaces

- **smoothgains_bridge**

6.166 softclip.cpp File Reference

Classes

- class **cfg_t**
- class **softclip_t**

6.167 spec2wave.cpp File Reference

Classes

- class **hanning_ramps_t**
- class **spec2wave_t**
- class **spec2wave_if_t**

Functions

- unsigned int **max** (unsigned int a, unsigned int b)
- unsigned int **min** (unsigned int a, unsigned int b)

6.167.1 Function Documentation

6.167.1.1 unsigned int max (unsigned int *a,* unsigned int *b*) [inline]

6.167.1.2 unsigned int min (unsigned int *a,* unsigned int *b*) [inline]

6.168 speechnoise.cpp File Reference

Macros

- #define **NUM_ENTR_MHAORIG** 76
- #define **NUM_ENTR_LTASS** 25
- #define **NUM_ENTR_OLNOISE** 49

Functions

- float **fhz2bandno** (float x)
- float **erb_hz_f_hz** (float f_hz)
- float **hz2hz** (float x)

  Dummy scale transformation Hz to Hz.
- float **bandw_correction** (float f, float ldb)

Variables

- float **vMHAOrigSpec** [**NUM_ENTR_MHAORIG**] = {-1.473, 0, -4.939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13, -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.14, -27.55, -25.79, -25.89, -26.11, -27.↩ 48, -30.37, -33.13, -36.23, -36.64, -36.35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85, -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.↩ 53, -38.71, -38.7, -38.92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.↩ 43, -77.43}
- float **vMHAOrigFreq** [**NUM_ENTR_MHAORIG**] = {172.266,344.532,516.797,689.↩ 063,861.329,1033.59,1205.86,1378.13,1550.39,1722.66,1894.92,2067.19,2239.↩ 46,2411.72,2583.99,2756.25,2928.52,3100.78,3273.05,3445.32,3617.58,3789.85,3962.↩ 11,4134.38,4306.64,4478.91,4651.18,4823.44,4995.71,5167.97,5340.24,5512.51,5684.↩ 77,5857.04,6029.3,6201.57,6373.83,6546.1,6718.37,6890.63,7062.9,7235.16,7407.↩ 43,7579.69,7751.96,7924.23,8096.49,8268.76,8441.02,8613.29,8785.56,8957.82,9130.↩ 09,9302.35,9474.62,9646.88,9819.15,9991.42,10163.7,10335.9,10508.2,10680.↩ 5,10852.7,11025,11197.3,11369.5,11541.8,11714.1,11886.3,12058.6,12230.9,12403.↩ 1,12575.4,12747.7,12919.9,13092.2}
- float **vLTASS_freq** [**NUM_ENTR_LTASS**] = {63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000}
- float **vLTASS_combined_lev** [**NUM_ENTR_LTASS**] = {38.6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7}
- float **vLTASS_female_lev** [**NUM_ENTR_LTASS**] = {37.0,36.0,37.5,40.1,53.4,62.2,60.↩ 9,58.1,61.7,61.7,60.4,58,54.3,52.3,51.7,48.8,47.3,46.7,45.3,44.6,45.2,44.9,45.0,42.↩ 8,41.1}
- float **vLTASS_male_lev** [**NUM_ENTR_LTASS**] = {38.6,43.5,54.4,57.7,56.8,58.2,59.↩ 7,60.0,62.4,62.6,60.6,55.7,53.1,53.7,52.3,48.7,48.9,47.0,46.0,44.4,43.3,42.4,41.9,39.↩ 8,40.4}
- float **vOlnoiseFreq** [**NUM_ENTR_OLNOISE**] = {62.5,70.1539,78.7451,88.3884,99.↩ 2126,111.362,125,140.308,157.49,176.777,198.425,222.725,250,280.616,314.98,353.↩ 553,396.85,445.449,500,561.231,629.961,707.107,793.701,890.899,1000,1122.↩ 46,1259.92,1414.21,1587.4,1781.8,2000,2244.92,2519.84,2828.43,3174.8,3563.↩ 59,4000,4489.85,5039.68,5656.85,6349.6,7127.19,8000,8979.7,10079.4,11313.↩ 7,12699.2,14254.4,16000}
- float **vOlnoiseLev** [**NUM_ENTR_OLNOISE**] = {45.9042,38.044,48.9444,61.3697,67.↩ 6953,69.7451,71.6201,71.2431,65.2754,63.2547,70.2264,72.1434,73.4433,73.2659,69.↩ 8424,71.0132,70.9577,70.3492,68.691,64.8436,64.0435,64.2879,60.5889,60.6596,60.↩ 3727,61.2003,61.8477,61.1478,61.2312,58.6584,57.2892,56.8299,56.0191,53.3018,56.↩ 0525,54.3592,50.8823,55.992,54.6768,47.2616,46.9914,45.209,50.413,47.5848,43.↩ 3215,43.754,38.5773,-0.39427,5.74224}

6.168.1 Macro Definition Documentation

6.168.1.1 #define NUM_ENTR_MHAORIG 76

6.168.1.2 #define NUM_ENTR_LTASS 25

6.168.1.3 #define NUM_ENTR_OLNOISE 49

6.168.2 Function Documentation

6.168.2.1 float fhz2bandno (float *x*)

6.168.2.2 float erb_hz_f_hz (float *f_hz*)

6.168.2.3 float hz2hz (float *x*)

Dummy scale transformation Hz to Hz.

This function implements a dummy scale transformation (linear frequency scale).

Parameters

x	Input frequency in Hz

Returns

Frequency in Hz

6.168.2.4 float bandw_correction (float *f,* float *ldb*)

6.168.3 Variable Documentation

6.168.3.1 float vMHAOrigSpec[NUM_ENTR_MHAORIG] = {-1.473, 0, -4.939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13, -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.14, -27.55, -25.79, -25.89, -26.11, -27.48, -30.37, -33.13, -36.23, -36.64, -36.35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85, -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.53, -38.71, -38.7, -38.92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.43, -77.43}

6.168.3.2 float vMHAOrigFreq[NUM_ENTR_MHAORIG] = {172.266,344.532,516.797,689.063,861.↩ 329,1033.59,1205.86,1378.13,1550.39,1722.66,1894.92,2067.19,2239.46,2411.72,2583.99,2756.↩ 25,2928.52,3100.78,3273.05,3445.32,3617.58,3789.85,3962.11,4134.38,4306.64,4478.91,4651.↩ 18,4823.44,4995.71,5167.97,5340.24,5512.51,5684.77,5857.04,6029.3,6201.57,6373.83,6546.↩ 1,6718.37,6890.63,7062.9,7235.16,7407.43,7579.69,7751.96,7924.23,8096.49,8268.76,8441.↩ 02,8613.29,8785.56,8957.82,9130.09,9302.35,9474.62,9646.88,9819.15,9991.42,10163.7,10335.↩ 9,10508.2,10680.5,10852.7,11025,11197.3,11369.5,11541.8,11714.1,11886.3,12058.6,12230.↩ 9,12403.1,12575.4,12747.7,12919.9,13092.2}

6.168.3.3 float vLTASS_freq[NUM_ENTR_LTASS] = {63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000}

6.168.3.4 float vLTASS_combined_lev[NUM_ENTR_LTASS] = {38.6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7}

6.168.3.5 float vLTASS_female_lev[NUM_ENTR_LTASS] = {37.0,36.0,37.5,40.1,53.4,62.2,60.9,58.↩ 1,61.7,61.7,60.4,58,54.3,52.3,51.7,48.8,47.3,46.7,45.3,44.6,45.2,44.9,45.0,42.8,41.1}

6.168.3.6 float vLTASS_male_lev[NUM_ENTR_LTASS] = {38.6,43.5,54.4,57.7,56.8,58.2,59.7,60.0,62.↩ 4,62.6,60.6,55.7,53.1,53.7,52.3,48.7,48.9,47.0,46.0,44.4,43.3,42.4,41.9,39.8,40.4}

6.168.3.7 float vOlnoiseFreq[NUM_ENTR_OLNOISE] = {62.5,70.1539,78.7451,88.3884,99.2126,111.↩ 362,125,140.308,157.49,176.777,198.425,222.725,250,280.616,314.98,353.553,396.85,445.↩ 449,500,561.231,629.961,707.107,793.701,890.899,1000,1122.46,1259.92,1414.21,1587.4,1781.↩ 8,2000,2244.92,2519.84,2828.43,3174.8,3563.59,4000,4489.85,5039.68,5656.85,6349.6,7127.↩ 19,8000,8979.7,10079.4,11313.7,12699.2,14254.4,16000}

6.168.3.8 float vOlnoiseLev[NUM_ENTR_OLNOISE] = {45.9042,38.044,48.9444,61.3697,67.6953,69.↩ 7451,71.6201,71.2431,65.2754,63.2547,70.2264,72.1434,73.4433,73.2659,69.8424,71.0132,70.↩ 9577,70.3492,68.691,64.8436,64.0435,64.2879,60.5889,60.6596,60.3727,61.2003,61.8477,61.↩ 1478,61.2312,58.6584,57.2892,56.8299,56.0191,53.3018,56.0525,54.3592,50.8823,55.992,54.↩ 6768,47.2616,46.9914,45.209,50.413,47.5848,43.3215,43.754,38.5773,-0.39427,5.74224}

6.169 speechnoise.h File Reference

Classes

- class **speechnoise_t**

6.170 split.cpp File Reference

Classes

- class **MHAPlugin_Split::uni_processor_t**

  An interface to a class that sports a process method with no parameters and no return value.
- class **MHAPlugin_Split::thread_platform_t**

  Basic interface for encapsulating thread creation, thread priority setting, and synchronization on any threading platform (i.e., pthreads or win32threads).
- class **MHAPlugin_Split::dummy_threads_t**

  Dummy specification of a thread platform: This class implements everything in a single thread.
- class **MHAPlugin_Split::posix_threads_t**

  Posix threads specification of thread platform.
- class **MHAPlugin_Split::domain_handler_t**

  Handles domain-specific partial input and output signal.
- class **MHAPlugin_Split::splitted_part_t**

  *The **splitted_part_t** (p. 751) instance manages the plugin that performs processing on the reduced set of channels.*
- class **MHAPlugin_Split::split_t**

  Implements split plugin.

Namespaces

- **MHAPlugin_Split**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**
  This define modifies the definition of MHAPLUGIN_CALLBACKS and friends.
- #define **posixthreads** 1
- #define **default_thread_platform_string** "posix"
- #define **default_thread_platform_type** posix_threads_t

Enumerations

6.170.1 Detailed Description

Source code for the split plugin. The split plugin splits the audio signal by channel. The splitted paths execute in parallel.

6.170.2 Macro Definition Documentation

6.170.2.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

This define modifies the definition of MHAPLUGIN_CALLBACKS and friends.

The output signal is transferred through a second parameter to the process method, enabling all four domain transformations in a single plugin.

6.170.2.2 #define posixthreads 1

6.170.2.3 #define default_thread_platform_string "posix"

6.170.2.4 #define default_thread_platform_type posix_threads_t

6.171 steerbf.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**steerbf**↩ **::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.171.1 Macro Definition Documentation

6.171.1.1 #define PATCH_VAR(

var) patchbay.connect(&var.valuechanged, this, &steerbf::update_cfg)

6.171.1.2 #define INSERT_PATCH(

var) insert_member(var); PATCH_VAR(var)

6.172 steerbf.h File Reference

Classes

- class **parser_int_dyn**
- class **steerbf_config**
- class **steerbf**

6.173 testalsadevice.c File Reference

Functions

- int **main** (int argc, char ∗∗argv)

6.173.1 Function Documentation

6.173.1.1 int main (

int *argc,*
char ∗∗ *argv*)

6.174 testplugin.cpp File Reference

Classes

- class **testplugin::config_parser_t**
- class **testplugin::ac_parser_t**
- class **testplugin::signal_parser_t**
- class **testplugin::if_t**

Namespaces

- **testplugin**

6.175 timoconfig.cpp File Reference

Macros

- #define **LPSCALE** (5.2429e+007)
- #define **POWSPEC_FACTOR** 0.0025
- #define **OVERLAP_FACTOR** 2
- #define **EPSILON** (1e-10)
- #define **CHANLOOP** for (unsigned int c=0; c<nchan; ++c)

6.175.1 Macro Definition Documentation

6.175.1.1 #define LPSCALE (5.2429e+007)

6.175.1.2 #define POWSPEC_FACTOR 0.0025

6.175.1.3 #define OVERLAP_FACTOR 2

6.175.1.4 #define EPSILON (1e-10)

6.175.1.5 #define CHANLOOP for (unsigned int c=0; c<nchan; ++c)

6.176 timoconfig.h File Reference

Classes

- class **timo_AC**
- class **timo_params**
- class **timoConfig**

6.177 timoSmooth.cpp File Reference

Macros

- #define **INSERT_VAR**(var) insert_item(#var, &var)
- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &**timoSmooth**↩**::on_model_param_valuechanged**)
- #define **INSERT_PATCH**(var) **INSERT_VAR**(var); **PATCH_VAR**(var)

6.177.1 Macro Definition Documentation

6.177.1.1 #define INSERT_VAR(*var*) insert_item(#var, &var)

6.177.1.2 #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, &timoSmooth::on_model_param_valuechanged)

6.177.1.3 #define INSERT_PATCH(*var*) INSERT_VAR(var); PATCH_VAR(var)

6.178 timosmooth.h File Reference

Classes

- class **timoSmooth**

6.179 transducers.cpp File Reference

Classes

- class **softclipper_variables_t**
- class **softclipper_t**
- class **calibrator_variables_t**
- class **calibrator_runtime_layer_t**
- class **calibrator_t**
- class **bbcalib_interface_t**

Typedefs

- typedef **MHAPlugin::config_t**< **MHASignal::async_rmslevel_t** > **rmslevelmeter**
- typedef **MHAPlugin::plugin_t**< **calibrator_runtime_layer_t** > **rtcalibrator**

Functions

- **speechnoise_t::noise_type_t kw_index2type** (unsigned int idx)
- std::vector< int > **vint_0123n1** (unsigned int n)

6.179.1 Typedef Documentation

6.179.1.1 typedef MHAPlugin::config_t<MHASignal::async_rmslevel_t> rmslevelmeter

6.179.1.2 typedef MHAPlugin::plugin_t<calibrator_runtime_layer_t> rtcalibrator

6.179.2 Function Documentation

6.179.2.1 speechnoise_t::noise_type_t kw_index2type (unsigned int *idx*)

6.179.2.2 std::vector<int> vint_0123n1 (unsigned int *n*)

6.180 upsample.cpp File Reference

Classes

- class **us_t**

6.181 wave2spec.cpp File Reference

Classes

- class **wave2spec_t**
- class **wave2spec_if_t**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.181.1 Macro Definition Documentation

6.181.1.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

6.182 wavrec.cpp File Reference

Classes

- class **wavwriter_t**
- class **wavrec_t**

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x << std::endl

6.182.1 Macro Definition Documentation

6.182.1.1 #define DEBUG(*x*) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x << std::endl

6.183 windowselector.cpp File Reference

6.184 windowselector.h File Reference

Classes

- class **windowselector_t**

  A combination of mha parser variables to describe an overalapadd analysis window.

Index