The Open Master Hearing Aid (openMHA)

4.18.0

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. 30)
- Signal processing plugins
- Audio input-output (IO) plugins (see **io_file_t** (p. 714), **MHAIOJack** (p. 114), **io_parser_t** (p. 725), **io_tcp_parser_t** (p. 735))
- **The openMHA toolbox library** (p. 30)

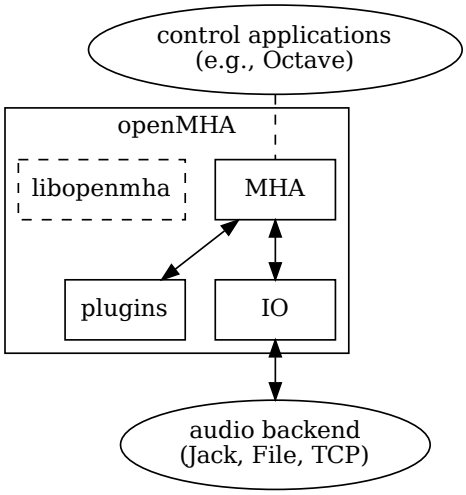


Figure 1 openMHA structure

The openMHA command line application (MHA) (p. 30) 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. 114) 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 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. 23) 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 openMHA 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: E.g. plugins can compute the rms (root mean squared) sound pressure in Pascal of the current block by computing $\mathrm{rms} = \sqrt{\sum_i x_i^2}$, where x_i refer to the samples of the audio signal in a single audio channel, and the corresponding free-field sound pressure level L in dB SPL FF as $\mathrm{L} = 20\log_{10}(rms/20\mu\mathrm{Pa})$. ($20\mu\mathrm{Pa}$ is the sound pressure at 0dB SPL FF).

A derived convention is employed in the spectral domain for STFT signals: The sum of the squared magnitudes of all spectral bins computes rms of the signal in the current STFT block: $\mathrm{rms} = \sqrt{\sum_i |X_i|^2}$, the X_i refer to the complex values of the STFT spectral bins. The STFT bins

of the negative frequencies are not stored, since they contain the complex conjugate values of the corresponding positive frequencies. When summing over all bins to compute the rms as above, care must be taken to also account for the negative frequencies. Note that the bins corresponding to 0Hz and to the Nyquist frequency have no corresponding negative frequency bin. The sound pressure level in dB can be computed from the rms in the same way as in the time domain.

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 Deprecated List

Member MHAParser::base_t::base_t (p. 1176) (const base_t (p. 1172) &)
Copying parser nodes makes little sense, avoid wherever possible

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. 23)): 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. 16) and references therein).

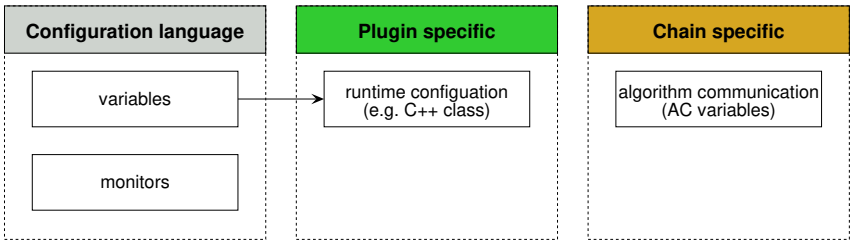


Figure 2 Variable types in the openMHA

Variables that describe physical facts to the MHA user should be given in SI units, e.g. meters for distances (not centimeters or inches), seconds for times (not milliseconds or minutes) etc for reasons of uniformity and simplicity of handling derived units.

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 the openMHA application manual.

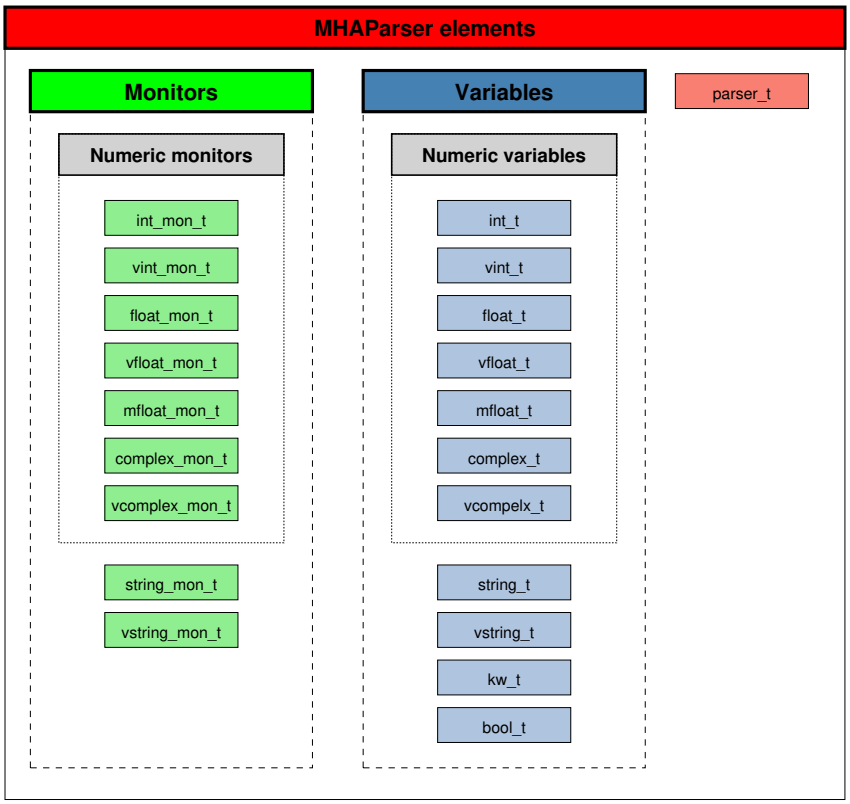


Figure 3 MHAParser elements

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

Plugins are C++ code that is compiled and linked against the openMHA library. The compiler needs be instructed on how to find the openMHA headers and library and to link against the openMHA library. There are two possible options: One can compile openMHA and then create a copy of an example plugin directory and customize from there. See COMPILATION.md for more information on how to compile openMHA.

On Ubuntu is is also possible to install the libopenmha-dev package and include config.mk into the user's Makefile. Example 21 provides an example plugin and Makefile for this scenario.

On the personal hearing lab (PHL) running mahalia, the required packages for compiling open↩ MHA plugins are already installed since version 4.17.0-r1. Example 21 can also be used here.

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.

3.2.1 example1.cpp

The example plugin file **example1.cpp** (p. 1762) 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. 1298) exports several methods, but only two of them need a non-empty implementation:

- `prepare()`
  - The `prepare()` method in the parent class `mha_plugin_t<>` is a pure virtual method and needs an implementation so that the plugin class can be instantiated.
- `process()`
  - The `process()` method is called whenever a new block of audio arrives and needs signal processing by this plugin.

```
#include "mha_plugin.hh"
class example1_t : public MHAPlugin::plugin_t<int> {
public:
  example1_t(MHA_AC::algo_comm_t & iac, const std::string & configured_name)
      : MHAPlugin::plugin_t<int>("",iac)
  {(void)configured_name;/* ignore 2nd parameter */}
  void release(void)
  {/* Do nothing in release */}
```

Every plugin implementation should include the '**mha_plugin.hh** (p. 1832)' 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. 1832).

The class **example1_t** (p. 552) inherits from the class **MHAPlugin::plugin_t** (p. 1298), which in turn inherits from **MHAParser::parser_t** (p. 1246) – the configuration language interface in the method "parse". Our plugin class therefore inherits the "parse" method from **MHAParser::parser_t** (p. 1246), which integrates the plugin into the global openMHA configuration tree.

The constructor has to accept two parameters of types `algo_comm_t` and `std::string`, respectively. 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.

3.2.1.1 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 dimensions, see **mhaconfig_t** (p. 996).

The `prepare()` method of the plugin is called before the signal processing starts, when the input signal dimensions 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.

3.2.1.2 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. 985).

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. 985) 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).

3.2.1.3 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. 1298) to be compatible with the C interface wrapper.

This macro also catches C++ exceptions of type **MHA_Error** (p. 906), 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. 906) 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. 906). 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.

3.2.2 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(MHA_AC::algo_comm_t & iac, const std::string & configured_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. 1298) 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.

3.2.2.1 Constructor

```
example2_t::example2_t(MHA_AC::algo_comm_t & iac,
                       const std::string & configured_name)
    : MHAPlugin::plugin_t<int>("This plugin multiplies the sound signal"
                               " in one audio channel by a factor",iac),
      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);
    (void)configured_name; // Ignore 2nd parameter
}
```

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. 1249) method.

3.2.2.2 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. 996).

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.

3.2.2.3 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,[".

3.2.2.4 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. 1212) and **MHAParser::float_t** (p. 1207) 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.

3.2.3 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.

3.2.3.1 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.

3.2.3.2 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(MHA_AC::algo_comm_t & iac, const std::string & configured_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.

3.2.3.3 Example 3 constructor

```
      example3_t::example3_t(MHA_AC::algo_comm_t & iac, const std::string &)
    : MHAPlugin::plugin_t<int>("This plugin multiplies the sound signal"
                               " in one audio channel by a factor",iac),
      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 a monitor variable does not require a parameter for setting the initial value. The only 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.

3.2.3.4 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.

3.2.3.5 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.

3.2.3.6 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.

3.2.3.7 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.

3.2.4 example4.cpp

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

3.2.4.1 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.

3.2.4.2 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. 937) instance stores the complex (**mha_complex_t** (p. 886)) spectral signal for positive frequences only (since the waveform signal is always real). The num_frames member of **mha_spec_t** (p. 937) actually denotes the number of STFT bins.

Please note that different from **mha_wave_t** (p. 985), a multichannel signal in **mha_spec_t** (p. 937) is stored non-interleaved in the signal buffer.

Some arithmetic operations are defined on struct **mha_complex_t** (p. 886) to facilitate efficient complex computations. The $*=$ operator used here (defined for real and for complex arguments) is one of them.

3.2.4.3 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.

3.2.5 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. 11)). To handle these situations, a special C++ template class **MHAPlugin::plugin_t** (p. 1298) 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. 566) is the parameter of the template class **MHAPlugin::plugin_t** (p. 1298). 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. 1298), 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. 1006) instance.

```
class plugin_interface_t : public MHAPlugin::plugin_t<example5_t> {
public:
    plugin_interface_t(MHA_AC::algo_comm_t & iac,
                       const std::string & configured_name);
    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. 906) 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. 906) 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 %u (only %u channels configured).",
                        channel,numchannels);
}
```

In this example, the run time configuration class **`example5_t`** (p. 566) 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. 1006) instance.

```
plugin_interface_t::plugin_interface_t(MHA_AC::algo_comm_t & iac,
                                       const std::string &)
    : MHAPlugin::plugin_t<example5_t>("example plugin scaling a spectral signal",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. 906) 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. 1296) stores the configuration in a FiFo queue of runtime configurations. Once they are inserted in the FiFo, the **MHAPlugin::plugin_t** (p. 1298) 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. 1835) defines all ANSI-C interface functions and passes them to the corresponding C++ class member functions (partly defined by the **MHAPlugin::plugin_t** (p. 1298) template class). All exceptions of type **MHA_Error** (p. 906) are caught and transformed into an appropriate error code and error message.

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

3.2.6 example6.cpp

This example is the same as the previous one, except that 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. 23).

```
example6_t::example6_t(MHA_AC::algo_comm_t & iac, const std::string &)
    : MHAPlugin::plugin_t<cfg_t>("Example rms level meter plugin",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("example6_rmslev", &rmsdb );
}
```

3.2.7 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 GDB debugger to inspect the **`example6_t::prepare()`** (p. 568) and **`example6_t::process()`** (p. 568) routines of **example6.cpp** (p. 18) 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. 568)) 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) 2005-2021 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)
```

3.2.8 Writing unit tests for openMHA plugins

This section introduces how to test a plugin with C++ unit tests using the Googletest framework. In order to execute the tests, navigate to the openMHA root directory and run `make unit-tests` in your terminal. Afterwards you may execute `make unit-tests` in the plugin directory in order to only execute the very test you are working on.

3.2.8.1 example7 As an example, unit tests for plugin **`example7.cpp`** (p. 1764) are written, which is functionally the same as plugin **`example1.cpp`** (p. 1762) (see section **example1.cpp** (p. 7)). In order to write unit tests for your plugin it must have its class/function declarations in a header file (.hh) so you can include it in the unit test file. The class/function definitions are contained in the respective source file (.cpp).
The unit tests are written using a test fixture class (here: `example7_testing`) which will be inherited by the individual tests (`TEST_F`). This enables us to use the members in `example7_testing` in multiple tests without the need for redundant declarations.

```
#include "mha_algo_comm.hh"
#include "mha_signal.hh"
#include "example7.hh"
#include <gtest/gtest.h>
class example7_testing : public ::testing::Test {
public:
  MHA_AC::algo_comm_class_t acspace;
  MHA_AC::algo_comm_t & ac = {acspace};
  mhaconfig_t signal_properties {
      .channels = 2U,
      .domain = MHA_WAVEFORM,
      .fragsize = 10U,
      .wndlen = 0U,
      .fftlen = 0U,
      .srate = 44100.0f
  };
  example7_t ex7 = {ac,"algo"};
  MHASignal::waveform_t wave_input{signal_properties.fragsize,signal_properties.channels};
};
```

The test fixture class is derived from the **::testing::Test** class declared in `gtest.h`. The constructor of **`example7_t`** (p. 570) needs three parameters, namely a handle to the algorithm communication variable space and two strings. A container for audio signals for repeatedly passing blocks of the input signal to the plugin under test is also allocated by the test fixture class. It is defined as an instance of **`MHASignal::waveform_t`** (p. 1414) with the name `wave_input` and its values are zero upon initialization.

```
TEST_F(example7_testing,test_state_methods){
  EXPECT_FALSE(ex7.is_prepared());
  ex7.prepare_(signal_properties);
  acspace.set_prepared(true);
  EXPECT_TRUE(ex7.is_prepared());
  acspace.set_prepared(false);
  ex7.release_();
  EXPECT_FALSE(ex7.is_prepared());
}
```

The first test checks whether the state methods work as expected. Next to the actual processing there are often certain variables in each individual openMHA plugin that need to be allocated beforehand or wiped from memory afterwards. The methods that are used to do this are `prepare()` and `release()`. In order to assert that they were called and that we switched states accordingly we use the methods `prepare_()` and `release_()` (Note: the underscore!) that are defined in the plugin base class `mha_plugin_t<>`. These methods keep track of the state, call `prepare()` and `release()` and do additional bookkeeping. To ensure that the state methods work as expected the Googletest methods `EXPECT_FALSE` and `EXPECT_TRUE` are used.

```
TEST_F(example7_testing,test_functionality){
  ex7.prepare_(signal_properties);
  acspace.set_prepared(true);
  wave_input.assign(1.0f);
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,4,0));
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,5,0));
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,4,1));
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,5,1));
  ex7.process(&wave_input);
  EXPECT_FLOAT_EQ(0.1f,value(wave_input,4,0));
  EXPECT_FLOAT_EQ(0.1f,value(wave_input,5,0));
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,4,1));
  EXPECT_FLOAT_EQ(1.0f,value(wave_input,5,1));
  acspace.set_prepared(false);
  ex7.release_();
}
```

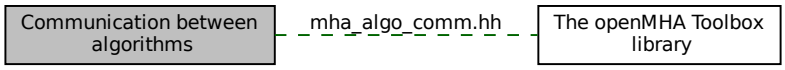
In this test the goal is to assess the main feature of the plugin (**`example7_t`** (p. 570)), which is the same as in **`example1_t`** (p. 552), namely altering the signal's first channel by a constant factor of 0.1. The variable `wave_input` is the signal that will be processed by the plugin. In order to assert the success, the elements in `wave_input` are set to a constant value of 1, because they are 0 upon initialization. During the `process()` function all elements of the first channel of `wave_input` are multiplied by the factor 0.1. Before `process()` is called the value assigned to `wave_input` is checked via the method `EXPECT_FLOAT_EQ` provided by Googletest. The values of `wave_input` are retrieved by the method **`value()`** (p. 45) by passing the desired sample position and channel number as second and third input parameter, respectively. Here, we checked the values of two frames in each channel to show the difference before and after processing; the frame indices were chosen randomly. After calling `process()`, the values contained in `wave_input` are checked again to make sure that the plugin worked as intended.

3.3 The MHA Framework interface

3.4 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.hh**

  Header file for Algorithm Communication.

Classes

- struct **MHA_AC::comm_var_t**

  Algorithm communication variable structure.
- class **MHA_AC::spectrum_t**

  Convenience class for inserting a spectrum into the AC space.
- class **MHA_AC::waveform_t**

  Convenience class for inserting a waveform (a block of time-domain audio signal) 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.
- class **MHA_AC::scalar_t**< **numeric_t, MHA_AC_TYPECODE** >

  Template for convenience classes for inserting a numeric scalar into the AC space.

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.

3.4.1 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) is passed at initialisation time to the constructor of each plugin class **constructor** (p. 1298). This handle contains a reference handle, algo_comm_t::handle, and a number of function pointers, algo_comm_t::insert_var etc.. An algorithm communication variable is accessed through objects of type comm_var_t.

For openMHA users, openMHA provides generic plugins to inspect and store AC variables of numeric types:

- plugin acmon mirrors AC variables of numeric types in readonly configuration variables (called monitors),
- plugin acsave stores AC variables into Matlab or text files. Plugin developers may also want to use these plugins to inspect any AC variables published by their own plugins during testing.

As a developer of openMHA plugin(s), please observe the following best practices in plugins using AC variables:

1. Plugins publishing AC variables:
   - insert all variables during prepare()
   - re-insert all variables during each process()
   - memory used for storing AC variable values is allocated and owned by the publishing plugin and needs to remain valid until the next call to process() or release() of the same plugin.
2. Plugins consuming AC variable published by other plugins:
   - poll required variables (and check validity) again during each process() before accessing their values.

3.4.2 Function Documentation

3.4.2.1 get_var_spectrum()

```
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 only valid during the current call of the plugin's process() method and should not be stored for later reuse.

The stride of the AC variable is used as the number of spectral bins per channel. The complex values of the spectrum are not copied, the `buf` pointer of the returned spectrum points to the original memory of the AC variable.

Parameters

ac	AC handle
name	Name of the variable

Returns

Spectrum structure

3.4.2.2 get_var_waveform()

```
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 block of waveform signal. The waveform variable only valid during the current call of the plugin's process() method and should not be stored for later reuse.

The stride of the AC variable is used as the number of audio channels. The single-precision floating-point sample values are not copied, the `buf` pointer of the returned waveform points to the original memory of the AC variable.

Parameters

ac	AC handle
name	Name of the variable

Returns

waveform structure

3.4.2.3 get_var_int() `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

3.4.2.4 get_var_float() `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

3.4.2.5 get_var_vfloat() `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.

Because this function allocates memory for the return value, it should not be called during signal processing, but only from the plugin prepare() method.

Parameters

ac	AC handle
name	Name of the variable

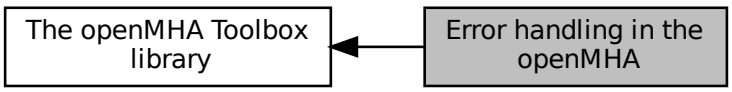
Returns

Variable value

3.5 Error handling in the openMHA

Errors are reported to the user via the **MHA_Error** (p. 906) 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. 906).*
- #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. 906) with the values.*

Functions

- void **mha_debug** (const char *fmt,...) __attribute__((__format__(printf

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

3.5.1 Detailed Description

Errors are reported to the user via the **MHA_Error** (p. 906) exception.

3.5.2 Macro Definition Documentation

3.5.2.1 MHA_ErrorMsg `#define MHA_ErrorMsg( x ) MHA_Error(__FILE__,__LINE__,"%s",x)`

Throw an openMHA error with a text message.

Parameters

x	Text message.

3.5.2.2 MHA_assert `#define MHA_assert(`

```
x ) if(!(x)) throw  MHA_Error(__FILE__,__LINE__,"\"%s\" is false.",#x)
```

Assertion macro, which throws an **MHA_Error** (p. 906).

Parameters

x	Boolean expression which should be true.

3.5.2.3 MHA_assert_equal `#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. 906) with the values.

Parameters

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

3.5.3 Function Documentation

3.5.3.1 mha_debug() `void mha_debug (`

```
const char * fmt,
 ...  )
```

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

3.6 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. 1827).

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. 1827).

All required classes and functions for parser access are declared in the namespace **MHAParser** (p. 125). The plugin class should be derived from the class **MHAParser::parser_t** (p. 1246) (or **MHAPlugin::plugin_t** (p. 1298)), 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. 1249).

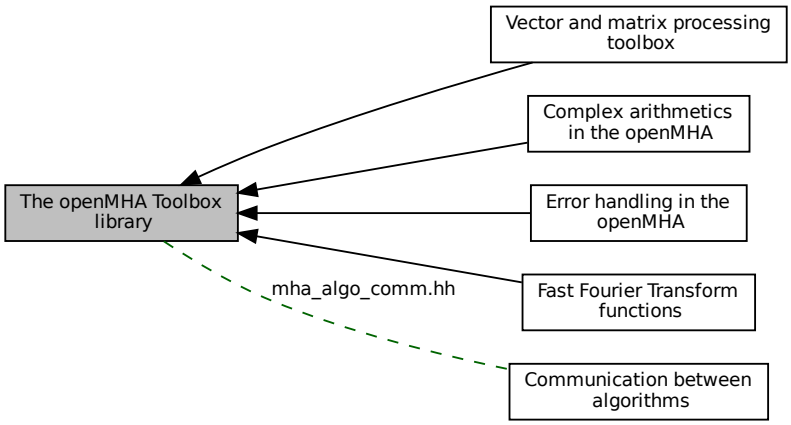
The openMHA Plugin template class **MHAPlugin::plugin_t** (p. 1298) together with the Plugin macro **MHAPLUGIN_CALLBACKS** (p. 1835) provide the callback mappings and correct inheritance. If your plugin is based on that template class, you simply have to use the insert_item command to give access to your variables, everything else is managed internally.

A complete list of all openMHA script items is given in the description of the **MHAParser** (p. 125) namespace.

3.7 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. 906) exception.*
- **Vector and matrix processing toolbox**

  *The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 139), and many functions and operators for use with the structures **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937).*
- **Complex arithmetics in the openMHA**
- **Fast Fourier Transform functions**

Files

- file **mha_algo_comm.hh**

  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.

3.7.1 Detailed Description

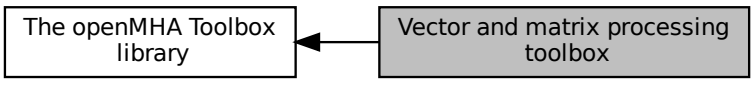
The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins.

It contains the openMHA script language classes.

3.8 Vector and matrix processing toolbox

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 139), and many functions and operators for use with the structures **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937).

Collaboration diagram for Vector and matrix processing toolbox:



Classes

- struct **mha_wave_t**
  Waveform signal structure.
- struct **mha_audio_descriptor_t**
  *Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 996)).*
- struct **mha_audio_t**
  *An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937)).*
- class **MHASignal::spectrum_t**
  *a signal processing class for spectral data (based on **mha_spec_t** (p. 937))*
- class **MHASignal::waveform_t**
  *signal processing class for waveform data (based on **mha_wave_t** (p. 985))*
- 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.
- **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. 985) *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. 985) *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. 937) *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=nullptr)

  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. 937) with its conjugate.*

3.8.1 Detailed Description

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 139), and many functions and operators for use with the structures **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937).

3.8.2 Typedef Documentation

3.8.2.1 mha_real_t `typedef float` **`mha_real_t`**

openMHA type for real numbers

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

3.8.3 Function Documentation

3.8.3.1 range() `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.

3.8.3.2 channels() `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.

3.8.3.3 bin2freq()

```
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

3.8.3.4 freq2bin()

```
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

3.8.3.5 smp2rad()

```
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.

3.8.3.6 rad2smp()

```
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.

3.8.3.7 dupvec() `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.

3.8.3.8 dupvec_chk() `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.

3.8.3.9 equal_dim() [1/6]

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

Test for equal dimension of waveform structures.

3.8.3.10 equal_dim() [2/6]

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

Test for match of waveform dimension with mhaconfig structure.

3.8.3.11 equal_dim() [3/6]

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

Test for equal dimension of spectrum structures.

3.8.3.12 equal_dim() [4/6]

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

Test for match of spectrum dimension with mhaconfig structure.

3.8.3.13 equal_dim() [5/6]

```
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. 985) use interleaved data order, while spectrum structures **mha_spec_t** (p. 937) use non-interleaved.

3.8.3.14 equal_dim() [6/6]

```
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. 985) use interleaved data order, while spectrum structures **mha_spec_t** (p. 937) use non-interleaved.

3.8.3.15 integrate() [1/2] void integrate (mha_wave_t & s)

Numeric integration of a signal vector (real values)

Parameters

s	Input signal vector

3.8.3.16 integrate() [2/2] void integrate (mha_spec_t & s)

Numeric integration of a signal vector (complex values)

Parameters

s	Input signal vector

3.8.3.17 size() [1/4] unsigned int size (const mha_wave_t & s) [inline]

Return size of a waveform structure.

3.8.3.18 size() [2/4] unsigned int size (const mha_spec_t & s) [inline]

Return size of a spectrum structure.

3.8.3.19 size() [3/4] unsigned int size (const mha_wave_t * s) [inline]

Return size of a waveform structure.

3.8.3.20 size() [4/4] unsigned int size (const mha_spec_t * s) [inline]

Return size of a spectrum structure.

3.8.3.21 clear() [1/4] void clear (mha_wave_t & s) [inline]

Set all values of waveform to zero.

3.8.3.22 clear() [2/4] void clear (mha_wave_t * s) [inline]

Set all values of waveform to zero.

3.8.3.23 clear() [3/4] void clear (mha_spec_t & s) [inline]

Set all values of spectrum to zero.

3.8.3.24 clear() [4/4] void clear (mha_spec_t * s) [inline]

Set all values of spectrum to zero.

3.8.3.25 assign() [1/3] 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.

3.8.3.26 assign() [2/3] 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.

3.8.3.27 assign() [3/3] 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.

3.8.3.28 timeshift() `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

3.8.3.29 value() [1/8] `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

3.8.3.30 value() [2/8] 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

3.8.3.31 value() [3/8] 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

3.8.3.32 value() [4/8] 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

3.8.3.33 value() [5/8] 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

3.8.3.34 value() [6/8] 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

3.8.3.35 value() [7/8] 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

3.8.3.36 value() [8/8] 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

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

Converts a **mha_wave_t** (p. 985) structure into a std::vector<float> (interleaved order).

Warning

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

3.8.3.38 std_vector_vector_float() std::vector<std::vector<float> > std_vector_↩ vector_float (const mha_wave_t & w)

Converts a **mha_wave_t** (p. 985) 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.

3.8.3.39 std_vector_vector_complex() std::vector<std::vector< mha_complex_t> > std_vector_vector_complex (

```
const mha_spec_t & w )
```

Converts a **mha_spec_t** (p. 937) 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.

3.8.3.40 operator+=() [1/4] mha_wave_t& operator+= (

```
mha_wave_t & self,
const mha_real_t & v )
```

Addition operator.

3.8.3.41 operator+=() [2/4] mha_wave_t& operator+= (

```
mha_wave_t & self,
const mha_wave_t & v )
```

Addition operator.

3.8.3.42 operator-=() [1/2] mha_wave_t& operator-= (

```
mha_wave_t & self,
const mha_wave_t & v )
```

Subtraction operator.

3.8.3.43 operator-=() [2/2] mha_spec_t& operator-= (

```
mha_spec_t & self,
const mha_spec_t & v )
```

Subtraction operator.

3.8.3.44 operator*=() [1/5]

```
mha_wave_t& operator*= (
    mha_wave_t & self,
    const  mha_real_t & v )
```

Element-wise multiplication operator.

3.8.3.45 operator*=() [2/5]

```
mha_wave_t& operator*= (
    mha_wave_t & self,
    const  mha_wave_t & v )
```

Element-wise multiplication operator.

3.8.3.46 operator*=() [3/5]

```
mha_spec_t& operator*= (
    mha_spec_t & self,
    const  mha_real_t & v )
```

Element-wise multiplication operator.

3.8.3.47 operator*=() [4/5]

```
mha_spec_t& operator*= (
    mha_spec_t & self,
    const  mha_wave_t & v )
```

Element-wise multiplication operator.

3.8.3.48 operator*=() [5/5]

```
mha_spec_t& operator*= (
    mha_spec_t & self,
    const  mha_spec_t & v )
```

Element-wise multiplication operator.

3.8.3.49 operator/=() [1/2]

```
mha_spec_t& operator/= (
    mha_spec_t & self,
    const mha_spec_t & v )
```

Element-wise division operator.

3.8.3.50 operator/=() [2/2]

```
mha_wave_t& operator/= (
    mha_wave_t & self,
    const mha_wave_t & v )
```

Element-wise division operator.

3.8.3.51 operator+=() [3/4]

```
mha_spec_t& operator+= (
    mha_spec_t & self,
    const mha_spec_t & v )
```

Addition operator.

3.8.3.52 operator+=() [4/4]

```
mha_spec_t& operator+= (
    mha_spec_t & self,
    const mha_real_t & v )
```

Addition operator.

3.8.3.53 operator^=()

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

Exponent operator.

Warning

This overwrites the xor operator!

3.8.3.54 copy_channel() [1/2]

```
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

3.8.3.55 copy_channel() [2/2]

```
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

3.8.3.56 rmslevel() [1/2]

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

Return RMS level of a spectrum channel.

Computes the RMS level of the signal in Pascal in the given channel.

Takes into account the the negative frequency bins that are not stored (**Central Calibration** (p. 3)).

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

3.8.3.57 colored_intensity()

```
mha_real_t MHASignal::colored_intensity (
    const  mha_spec_t & s,
    unsigned int channel,
    unsigned int fftlen,
     mha_real_t * sqfreq_response = nullptr )
```

Colored spectrum intensity.

computes the squared sum of the spectrum after filtering with the frequency response. Takes into account the negative frequency bins that are not stored (**Central Calibration** (p. 3)).

Parameters

s	Input spectrum
channel	Channel number to be tested
fftlen	FFT length (to correctly count the level of the Nyquis bin)
sqfreq_response	An array with one squared weighting factor for every fft bin. Array length must be equal to s->num_frames. nullptr can be given for equal weighting of all frequencies.

Returns

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

3.8.3.58 maxabs() [1/3] 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

3.8.3.59 rmslevel() [2/2] 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

3.8.3.60 maxabs() [2/3] 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

3.8.3.61 maxabs() [3/3] mha_real_t MHASignal::maxabs (const mha_wave_t & s)

Find maximal absolute value.

Parameters

s	Input signal

Returns

maximum absolute value

3.8.3.62 max() mha_real_t MHASignal::max (const mha_wave_t & s)

Find maximal value.

Parameters

s	Input signal

Returns

maximum absolute value

3.8.3.63 min() `mha_real_t MHASignal::min ( const mha_wave_t & s )`

Find minimal value.

Parameters

s	Input signal

Returns

maximum absolute value

3.8.3.64 sumsqr_channel() `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

$\sum x^2$

3.8.3.65 sumsqr_frame() `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

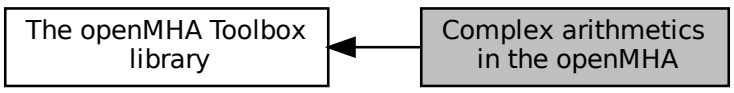
$\sum x^2$

3.8.3.66 conjugate() `void conjugate ( mha_spec_t & self ) [inline]`

Replace (!) the value of this **mha_spec_t** (p. 937) with its conjugate.

3.9 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. 886) variable.*
- **mha_complex_t** **mha_complex** (**mha_real_t** real, **mha_real_t** imag=0)
  *Create a new **mha_complex_t** (p. 886) 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. 886) 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. 886).*
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t** **angle**)
  *replaces the value of the given **mha_complex_t** (p. 886) 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. 886) 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. 886) 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.

3.9.1 Detailed Description

3.9.2 Function Documentation

3.9.2.1 set() [1/2]

```
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. 886) variable.

Parameters

self	The **mha_complex_t** (p. 886) variable whose value is about to change.
real	The new real part.
imag	The new imaginary part.

Returns

A reference to the changed variable.

3.9.2.2 mha_complex()

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

Create a new **mha_complex_t** (p. 886) with specified real and imaginary parts.

Parameters

real	The real part.
imag	The imaginary part.

Returns

The new value.

3.9.2.3 set() [2/2] mha_complex_t& set (mha_complex_t & self, const std::complex< mha_real_t > & stdcomplex) [inline]

Assign a **mha_complex_t** (p. 886) variable from a std::complex.

Parameters

self	The **mha_complex_t** (p. 886) variable whose value is about to change.
stdcomplex	The new complex value.

Returns

A reference to the changed variable.

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

Create a std::complex from **mha_complex_t** (p. 886).

3.9.2.5 expi() [1/2] mha_complex_t& expi (mha_complex_t & self, mha_real_t angle) [inline]

replaces the value of the given **mha_complex_t** (p. 886) with exp(i∗b).

Parameters

self	The **mha_complex_t** (p. 886) variable whose value is about to change.
angle	The angle in the complex plane [rad].

Returns

A reference to the changed variable.

3.9.2.6 angle() `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.

3.9.2.7 operator+=() [1/2] **`mha_complex_t`**`& operator+= (`

`mha_complex_t` `&` *`self`*`,`
`const` **`mha_complex_t`** `&` *`other`* `) [inline]`

Addition of two complex numbers, overwriting the first.

3.9.2.8 operator+() [1/2] **`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.

3.9.2.9 operator+=() [2/2] **`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.

3.9.2.10 operator+() [2/2]

```
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.

3.9.2.11 operator-=() [1/2]

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

Subtraction of two complex numbers, overwriting the first.

3.9.2.12 operator-() [1/3]

```
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.

3.9.2.13 operator-=() [2/2]

```
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.

3.9.2.14 operator-() [2/3]

```
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.

3.9.2.15 operator∗=() [1/2]

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

Multiplication of two complex numbers, overwriting the first.

3.9.2.16 operator∗() [1/2]

```
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.

3.9.2.17 operator∗=() [2/2]

```
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.

3.9.2.18 expi() [2/2]

```
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. 886) with a ∗ exp(i∗b)

Parameters

self	The **mha_complex_t** (p. 886) 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.

3.9.2.19 operator*() [2/2]

```
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.

3.9.2.20 abs2()

```
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.

3.9.2.21 abs()

```
mha_real_t abs (
    const mha_complex_t & self ) [inline]
```

Compute the absolute value of a complex value.

Returns

The absolute value of self.

3.9.2.22 operator/=() [1/2]

```
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.

3.9.2.23 operator/() [1/2]

```
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.

3.9.2.24 safe_div()

```
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.

3.9.2.25 operator/=() [2/2]

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

Division of two complex numbers, overwriting the first.

3.9.2.26 operator/() [2/2]

```
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.

3.9.2.27 operator-() [3/3]

```
mha_complex_t operator- (
    const  mha_complex_t & self ) [inline]
```

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

3.9.2.28 operator==()

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

Compare two complex numbers for equality.

3.9.2.29 operator"!=() bool operator!= (const mha_complex_t & x, const mha_complex_t & y) [inline]

Compare two complex numbers for inequality.

3.9.2.30 conjugate() void conjugate (mha_complex_t & self) [inline]

Replace (!) the value of this **mha_complex_t** (p. 886) with its conjugate.

3.9.2.31 _conjugate() 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.

3.9.2.32 reciprocal() void reciprocal (mha_complex_t & self) [inline]

Replace the value of this complex with its reciprocal.

3.9.2.33 _reciprocal() 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.

3.9.2.34 normalize() [1/2] void normalize (mha_complex_t & self) [inline]

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

3.9.2.35 normalize() [2/2] 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.

3.9.2.36 almost() 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

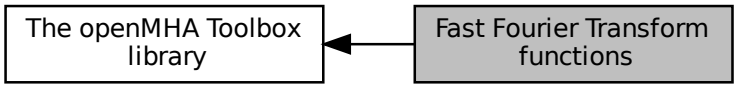
true if the relative difference is below times_epsilon $*$ std::numeric_limits<mha_real_t>↩ ::epsilon

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

Compares the absolute values of two complex numbers.

3.10 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.

3.10.1 Detailed Description

3.10.2 Typedef Documentation

3.10.2.1 mha_fft_t `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.

3.10.3 Function Documentation

3.10.3.1 mha_fft_new() `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

3.10.3.2 mha_fft_free() `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

3.10.3.3 mha_fft_wave2spec() [1/2] `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)

3.10.3.4 mha_fft_wave2spec() [2/2]

```
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.

3.10.3.5 mha_fft_spec2wave() [1/2]

```
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)

3.10.3.6 mha_fft_spec2wave() [2/2]

```
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.

3.10.3.7 mha_fft_forward()

```
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. 937) 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.

3.10.3.8 mha_fft_backward()

```
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. 937) 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.

3.10.3.9 mha_fft_forward_scale()

```
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. 937) 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.

3.10.3.10 mha_fft_backward_scale()

```
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. 937) 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.

3.10.3.11 mha_fft_wave2spec_scale()

```
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.

3.10.3.12 mha_fft_spec2wave_scale()

```
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

- struct **type_info**
- 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.*
- class **cfg_t**
  *Runtime configuration class of the **ac2lsl** (p. 78) plugin.*
- class **ac2lsl_t**
  *Plugin class of **ac2lsl** (p. 78).*

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

4.2 ac2wave Namespace Reference

Namespace containing all code for the **ac2wave** (p. 79) plugin.

Classes

- class **ac2wave_t**
  ac2wave *(p. 79) real-time configuration class*
- class **ac2wave_if_t**
  ac2wave *(p. 79) plugin interface class*

4.2.1 Detailed Description

Namespace containing all code for the **ac2wave** (p. 79) plugin.

4.3 ac2xdf Namespace Reference

Classes

- class **ac2xdf_rt_t**
- class **ac2xdf_if_t**
  Plugin interface class of plugin ***ac2xdf*** *(p. 79).*
- class **output_file_t**
  output_file_t *(p. 211) represents one XDF output file.*
- class **acwriter_base_t**
  Base class for all ***acwriter_t*** *(p. 205)'s.*
- class **acwriter_t**

Functions

- std::string **to_iso8601** (time_t tm)

Variables

- const std::unordered_map< std::type_index, std::string > **types**

4.3.1 Function Documentation

4.3.1.1 to_iso8601() `std::string ac2xdf::to_iso8601 ( time_t tm )`

4.3.2 Variable Documentation

4.3.2.1 types `const std::unordered_map<std::type_index, std::string> ac2xdf::types`

4.4 ac_proc Namespace Reference

Classes

- class **interface_t**

4.5 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**
  acmon plugin interface class

4.5.1 Detailed Description

Namespace for displaying ac variables as parser monitors.

Namespace containing the acmon plugin interface class.

4.6 acsave Namespace Reference

Classes

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

4.7 addsndfile Namespace Reference

Classes

- class **waveform_proxy_t**

  *Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. 286).*
- class **resampled_soundfile_t**

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

Typedefs

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

Enumerations

- enum **addsndfile_resampling_mode_t** { **DONT_RESAMPLE_PERMISSIVE** , **DONT_RESAMPLE_STRICT** , **DO_RESAMPLE** }

  Specifies the resampling mode in resampled_soundfile_t.

Functions

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

4.7.1 Typedef Documentation

4.7.1.1 level_adaptor typedef MHAPlugin::config_t< level_adapt_t> addsndfile↩ ::level_adaptor

4.7.1.2 wave_reader typedef MHAPlugin::plugin_t< sndfile_t> addsndfile::wave_↩ reader

4.7.2 Enumeration Type Documentation

4.7.2.1 addsndfile_resampling_mode_t enum addsndfile::addsndfile_resampling_↩ mode_t

Specifies the resampling mode in **resampled_soundfile_t** (p. 286).

Enumerator

DONT_RESAMPLE_PERMISSIVE	
DONT_RESAMPLE_STRICT	Do not resample, if the sample rate of the sample rate of the sound file, raise an err
DO_RESAMPLE	Resample.

4.7.3 Function Documentation

4.7.3.1 resampled_num_frames() static unsigned addsndfile::resampled_num_frames (

```
            unsigned num_source_frames,
            float source_rate,
            float target_rate,
             addsndfile_resampling_mode_t resampling_mode )  [static]
```

4.8 ADM Namespace Reference

Classes

- class **Linearphase_FIR**

  An efficient linear-phase fir filter implementation.
- class **Delay**

  A delay-line class.
- class **ADM**

  Adaptive differential microphone, working for speech frequency range.

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

4.8.1 Function Documentation

4.8.1.1 subsampledelay_coeff()

```
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

4.8.2 Variable Documentation

4.8.2.1 PI

```
const double ADM::PI = 3.14159265358979312
```

4.8.2.2 C

```
const double ADM::C = 340
```

4.8.2.3 DELAY_FREQ

```
const double ADM::DELAY_FREQ = 2000
```

4.8.2.4 START_BETA

```
const double ADM::START_BETA = 0.5
```

4.9 audiometerbackend Namespace Reference

Classes

- class **lnn3rdoct_t**
- class **sine_t**
- class **signal_gen_t**
- class **level_adapt_t**
- class **audiometer_if_t**

Typedefs

- typedef **MHAPlugin::config_t**< **level_adapt_t** > **level_adaptor**
- typedef **MHAPlugin::plugin_t**< **signal_gen_t** > **generator**

Functions

- static unsigned int **gcd** (unsigned int a, unsigned int b)
- **MHASignal::waveform_t return_sig** (unsigned int sigtype, unsigned int fs, unsigned int f)

4.9.1 Typedef Documentation

4.9.1.1 level_adaptor

`typedef MHAPlugin::config_t< level_adapt_t> audiometerbackend::level_adaptor`

4.9.1.2 generator

`typedef MHAPlugin::plugin_t< signal_gen_t> audiometerbackend::generator`

4.9.2 Function Documentation

4.9.2.1 gcd()

```
static unsigned int audiometerbackend::gcd (
    unsigned int a,
    unsigned int b ) [inline], [static]
```

4.9.2.2 return_sig()

```
MHASignal::waveform_t audiometerbackend::return_sig (
    unsigned int sigtype,
    unsigned int fs,
    unsigned int f )
```

4.10 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 **profile_t**
  The Auditory Profile class.
- class **parser_t**
  Class to make the auditory profile accessible through the parser interface.

4.10.1 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. 354). Until a complete definition is available, only the currently needed elements are implemented.

4.11 coherence Namespace Reference

Classes

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

Functions

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

4.11.1 Function Documentation

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

4.12 cpuload Namespace Reference

Classes

- class **cpuload_cfg_t**
- class **cpuload_if_t**

4.13 dbasync_native Namespace Reference

Classes

- class **delay_check_t**
- class **dbasync_t**
- class **db_if_t**

Enumerations

- enum { **INVALID_THREAD_PRIORITY** = 999999999 }

Functions

- static void * **thread_start** (void *instance)
- static unsigned **gcd** (unsigned a, unsigned b)

4.13.1 Enumeration Type Documentation

4.13.1.1 anonymous enum `anonymous enum`

Enumerator

INVALID_↩ THREAD_↩ PRIORITY	

4.13.2 Function Documentation

4.13.2.1 thread_start() `static void* dbasync_native::thread_start ( void * instance ) [static]`

4.13.2.2 gcd() `static unsigned dbasync_native::gcd ( unsigned a, unsigned b ) [inline], [static]`

4.14 dc Namespace Reference

Namespace containing all classes of the `dc` plugin which performs dynamic compression.

Classes

- class **dc_vars_t**
  Collection of configuration variables of the `dc` plugin.
- class **dc_vars_validator_t**
  Consistency checker.
- class **dc_t**
  Runtime configuration class of dynamic compression plugin `dc`.
- class **dc_if_t**
  Plugin interface class of the dynamic compression plugin `dc`.

4.14.1 Detailed Description

Namespace containing all classes of the `dc` plugin which performs dynamic compression.

4.15 dc_simple Namespace Reference

Classes

- class **dc_vars_t**
  *class for **dc_simple** (p. 89) plugin which registers variables to **MHAParser** (p. 125).*
- class **dc_vars_validator_t**
  Helper class to check sizes of configuration variable vectors.
- class **level_smoother_t**
  Class which computes smoothed input levels on individual bands, using an attack and release filter, which are a first order low pass filter and a maximum tracker filter, respectively.
- class **dc_t**
  *Runtime config class for **dc_simple** (p. 89) plugin.*
- class **dc_if_t**
  *interface class for **dc_simple** (p. 89)*

Typedefs

- typedef **MHAPlugin::plugin_t**< **dc_t** > **DC**
  Define alternate name for runtime_cfg_t.
- typedef **MHAPlugin::config_t**< **level_smoother_t** > **LEVEL**
  Define alternate name for config_t.

Functions

- void **test_fail** (const std::vector< float > &v, unsigned int s, const std::string &name)
  Checks size of vector.
- std::vector< float > **force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)
  Creates a copy of vector v with s elements, provided that \v has either s elements or 1 elements.
- **mha_real_t not_zero** (**mha_real_t** x, const std::string &comment)
  Helper function to throw an error if x is 0.

4.15.1 Typedef Documentation

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

Define alternate name for runtime_cfg_t.

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

Define alternate name for config_t.

4.15.2 Function Documentation

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

Checks size of vector.

Parameters

in	*v*	The vector to check the size of.
in	*s*	Expected size of vector v.
in	*name*	Name of vector to include in error message when size does not match.

Exceptions

MHA_Error *(p. 906)*	if the size of v is neither s nor 1.

4.15.2.2 force_resize()

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

Creates a copy of vector `v` with `s` elements, provided that \v has either s elements or 1 elements.

Parameters

in	*v*	The vector to copy elements from.
in	*s*	The desired number of elements in the output vector.
in	*name*	Name of vector to include in error message when input size does not match expectation.

Returns

A copy of `v` with `s` elements.

Exceptions

MHA_Error *(p. 906)*	if size of `v` is neither `s` nor 1.

4.15.2.3 not_zero()

```
mha_real_t dc_simple::not_zero (
    mha_real_t x,
    const std::string & comment )
```

Helper function to throw an error if `x` is 0.

Parameters

in	*x*	The value to check.
in	*comment*	Optional explanation for error message.

Exceptions

MHA_Error *(p. 906)*	if x == 0.

4.16 delay Namespace Reference

Classes

- class **interface_t**

4.17 delaysum Namespace Reference

This namespace contains the delaysum plugin.

Classes

- class **delaysum_wave_t**
  Runtime configuration of the delaysum_wave plugin.
- class **delaysum_wave_if_t**
  Interface class for the delaysum plugin.

4.17.1 Detailed Description

This namespace contains the delaysum plugin.

4.18 delaysum_spec Namespace Reference

Classes

- class **delaysum_t**
- class **delaysum_spec_if_t**

4.19 double2acvar Namespace Reference

Classes

- class **double2acvar_t**
  Plugin interface class for ***double2acvar*** *(p. 92).*

4.20 DynComp Namespace Reference

dynamic compression related classes and functions

Classes

- class **dc_afterburn_vars_t**

  *Variables for **dc_afterburn_t** (p. 538) class.*
- 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 **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.

4.20.1 Detailed Description

dynamic compression related classes and functions

4.20.2 Function Documentation

4.20.2.1 interp1()

```
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.

4.20.2.2 interp2()

```
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.

4.21 equalize Namespace Reference

Classes

- class **cfg_t**
- class **freqgains_t**

4.22 fader_wave Namespace Reference

Classes

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

Typedefs

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

4.22.1 Typedef Documentation

4.22.1.1 level_adaptor `typedef MHAPlugin::plugin_t< level_adapt_t> fader_wave↩ ::level_adaptor`

4.23 fftfbpow Namespace Reference

Namespace for the fftfbpow plugin.

Classes

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

4.23.1 Detailed Description

Namespace for the fftfbpow plugin.

4.24 fftfilter Namespace Reference

Classes

- class **fftfilter_t**
- class **interface_t**

Functions

- unsigned int **irs_length** (const **MHAParser::mfloat_t** &irs)
- unsigned int **irs_validator** (const **MHAParser::mfloat_t** &irs, const unsigned int & **channels**, const unsigned int &fragsize, const unsigned int &fftlen)

4.24.1 Function Documentation

4.24.1.1 irs_length() `unsigned int fftfilter::irs_length ( const MHAParser::mfloat_t & irs )`

Return the length of the longest vector in irs.

Parameters

irs	"Matrix" of floats parser variable

Returns

length of the longest vector in irs, or 1 if all vectors are empty

4.24.1.2 irs_validator()

```
unsigned int fftfilter::irs_validator (
    const  MHAParser::mfloat_t & irs,
    const unsigned int & channels,
    const unsigned int & fragsize,
    const unsigned int & fftlen )
```

Validity checks. Throws Error if parameters are invalid: Number of channels must be $>$ 0, fftlen must be $>=$ fragsize. The number of rows in irs has to match the number of channels, or has to be exactly 1. None of the row vectors may be empty. The longest supported impulse response is (fftlen - fragsize + 1). Impulse responses longer than (fftlen - fragsize + 1) would cause temporal aliasing.

Parameters

irs	The matrix containing the impulse responses (one response per channel, or the same response for every channels) as set by the parser.
channels	The number of prepared audio channels for this MHA plugin.
fragsize	The block size (samples per channel) for waveform audio data
fftlen	FFT length used for filtering.

Returns

the length of the longest impulse response vector in irs.

4.25 fftfilterbank Namespace Reference

Classes

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

4.26 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.

4.26.1 Detailed Description

All types for the fshift plugin live in this namespace.

4.26.2 Function Documentation

4.26.2.1 fft_find_bin()

```
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

4.27 fshift_hilbert Namespace Reference

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

Classes

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

4.27.1 Detailed Description

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

4.28 gain Namespace Reference

Classes

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

4.29 gsc_adaptive_stage Namespace Reference

Classes

- class **gsc_adaptive_stage**
- class **gsc_adaptive_stage_if**
  Plugin interface class.

Variables

- constexpr **mha_real_t** **DELT** =1e-12
  Small constant to ensure no division by zero occurs.

4.29.1 Variable Documentation

4.29.1.1 DELT `constexpr mha_real_t gsc_adaptive_stage::DELT =1e-12 [constexpr]`

Small constant to ensure no division by zero occurs.

4.30 gtfb_analyzer Namespace Reference

Classes

- struct **gtfb_analyzer_cfg_t**
  Configuration for Gammatone Filterbank Analyzer.
- class **gtfb_analyzer_t**
  Gammatone Filterbank Analyzer Plugin.

4.31 level_matching Namespace Reference

Classes

- class **channel_pair**
- class **level_matching_config_t**
- class **level_matching_t**

4.32 lsl2ac Namespace Reference

Classes

- class **save_var_base_t**
- class **save_var_t**
  LSL to AC bridge variable.
- class **save_var_t**< **std::string** >
  Specialication for marker streams.
- class **cfg_t**
  *Runtime configuration class of the **lsl2ac*** (p. 100) *plugin.*
- class **lsl2ac_t**
  *Plugin class of **lsl2ac*** (p. 100).

Enumerations

- enum class **overrun_behavior** { **Discard** =0 , **Ignore** }

4.32.1 Enumeration Type Documentation

4.32.1.1 overrun_behavior `enum lsl2ac::overrun_behavior [strong]`

Enumerator

Discard	
Ignore	

4.33 matlab_wrapper Namespace Reference

Namespace where all classes of the matlab wrapper plugin live.

Classes

- struct **types**
- struct **types**< **MHA_WAVEFORM** >
- struct **types**< **MHA_SPECTRUM** >
- class **matlab_wrapper_rt_cfg_t**

  Thin wrapper around the emxArray containing the user defined configuration variables.
- class **callback**

  Utility class connecting a user_config_t instance to its corresponding configuration parser.
- class **matlab_wrapper_t**

  Matlab wraper plugin interface class.

4.33.1 Detailed Description

Namespace where all classes of the matlab wrapper plugin live.

4.34 matrixmixer Namespace Reference

Classes

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

4.35 mconv Namespace Reference

Classes

- class **MConv**

4.36 MHA_AC Namespace Reference

Classes

- struct **comm_var_t**
  Algorithm communication variable structure.
- class **spectrum_t**
  Convenience class for inserting a spectrum into the AC space.
- class **waveform_t**
  Convenience class for inserting a waveform (a block of time-domain audio signal) into the AC space.
- class **stat_t**
- 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 **comm_var_map_t**
  Storage class for the AC variable space.
- class **algo_comm_t**
  Algorithm communication variable space interface.
- class **algo_comm_class_t**
  AC variable space implementation.
- class **scalar_t**
  Template for convenience classes for inserting a numeric scalar into the AC space.

Typedefs

- typedef **scalar_t**< int, **MHA_AC_INT** > **int_t**
  Convenience class for inserting an integer variable into the AC space.
- typedef **scalar_t**< float, **MHA_AC_FLOAT** > **float_t**
  Convenience class for inserting a single-precision floating-point variable into the AC space.
- typedef **scalar_t**< double, **MHA_AC_DOUBLE** > **double_t**
  Convenience class for inserting a double-precision floating-point variable 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.

4.36.1 Typedef Documentation

4.36.1.1 int_t typedef scalar_t<int, MHA_AC_INT> MHA_AC::int_t

Convenience class for inserting an integer variable into the AC space.

4.36.1.2 float_t typedef scalar_t<float, MHA_AC_FLOAT> MHA_AC::float_t

Convenience class for inserting a single-precision floating-point variable into the AC space.

4.36.1.3 double_t typedef scalar_t<double, MHA_AC_DOUBLE> MHA_AC::double_t

Convenience class for inserting a double-precision floating-point variable into the AC space.

4.37 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.

4.37.1 Function Documentation

4.37.1.1 digits() 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.

4.37.1.2 snprintf_required_length() 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

4.38 MHA_TCP Namespace Reference

A Namespace for TCP helper classes.

Classes

- class **sock_init_t**
- struct **OS_EVENT_TYPE**
- class **Wakeup_Event**
  A base class for asynchronous wakeup events.
- class **Async_Notify**
  Portable Multiplexable cross-thread notification.
- class **Event_Watcher**

OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.

- class **Timeout_Event**
- class **Timeout_Watcher**

  OS-independent event watcher with internal fixed-end-time timeout.
- class **Sockread_Event**

  Watch socket for incoming data.
- class **Sockwrite_Event**
- class **Sockaccept_Event**
- class **Connection**

  Connection *(p. 946) handles Communication between client and server, is used on both sides.*
- class **Server**
- class **Client**

  A portable class for a tcp client connections.
- class **Thread**

  A very simple class for portable threads.

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**

4.38.1 Detailed Description

A Namespace for TCP helper classes.

4.38.2 Typedef Documentation

4.38.2.1 SOCKET `typedef int` **MHA_TCP::SOCKET**

4.38.3 Function Documentation

4.38.3.1 STRERROR() `std::string MHA_TCP::STRERROR ( int err )`

Portable conversion from error number to error string.

4.38.3.2 HSTRERROR() `std::string MHA_TCP::HSTRERROR ( int err )`

Portable conversion from hostname error number to error string.

4.38.3.3 N_ERRNO() `int MHA_TCP::N_ERRNO ( )`

Portable access to last network error number.

4.38.3.4 H_ERRNO() `int MHA_TCP::H_ERRNO ( )`

Portable access to last hostname error number.

4.38.3.5 G_ERRNO() int MHA_TCP::G_ERRNO ( )

Portable access to last non-network error number.

4.38.3.6 dtime() [1/2] double MHA_TCP::dtime ( )

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

4.38.3.7 dtime() [2/2] double MHA_TCP::dtime (const struct timeval & *tv*)

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

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

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

4.38.4 Variable Documentation

4.38.4.1 sock_initializer class **MHA_TCP::sock_init_t** MHA_TCP::sock_initializer

4.39 mha_tcp Namespace Reference

namespace for network communication classes of MHA

Classes

- class **buffered_socket_t**

  An asio TCP socket with an associated streambuf buffer for receiving lines of text, as well as string buffers for sending responses.
- class **server_t**

  Class for accepting TCP connections from clients.

4.39.1 Detailed Description

namespace for network communication classes of MHA

4.40 mhachain Namespace Reference

Classes

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

4.41 MHAEvents Namespace Reference

Collection of event handling classes.

Classes

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

4.41.1 Detailed Description

Collection of event handling classes.

4.42 MHAFilter Namespace Reference

Namespace for IIR and FIR filter classes.

Classes

- class **complex_bandpass_t**
  Complex bandpass filter.
- class **gamma_flt_t**
  Class for gammatone filter.
- class **thirdoctave_analyzer_t**
- class **filter_t**
  Generic IIR filter class.
- class **diff_t**
  Differentiator class (non-normalized)
- 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 **iir_filter_state_t**
- class **iir_filter_t**
  IIR filter class wrapper for integration into parser structure.
- class **adapt_filter_state_t**
- class **adapt_filter_param_t**
- class **adapt_filter_t**
  Adaptive filter.
- class **fftfilter_t**
  FFT based FIR filter implementation.
- class **fftfilterbank_t**
  FFT based FIR filterbank implementation.
- 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.
- class **partitioned_convolution_t**
  A filter class for partitioned convolution.
- class **smoothspec_t**
  Smooth spectral gains, create a windowed impulse response.
- class **resampling_filter_t**
  Hann shaped low pass filter for resampling.
- class **polyphase_resampling_t**
  A class that performs polyphase resampling.
- class **blockprocessing_polyphase_resampling_t**
  A class that does polyphase resampling and takes into account block processing.
- class **iir_ord1_real_t**
  First order recursive filter.

Functions

- template<typename T , typename std::enable_if< std::is_floating_point< T >::value, T >::type * = nullptr> void **make_friendly_number** (T &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.
- std::vector< float > **fir_lp** (float f_pass_, float f_stop_, float fs_, unsigned order_)

  Setup a nth order fir low pass 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.

4.42.1 Detailed Description

Namespace for IIR and FIR filter classes.

4.42.2 Function Documentation

4.42.2.1 make_friendly_number() `template<typename T , typename std::enable_if< std::is_floating_point< T >::value, T >::type * = nullptr> void MHAFilter::make_friendly_number ( T & x ) [inline]`

4.42.2.2 o1_lp_coeffs()

```
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

4.42.2.3 butter_stop_ord1()

```
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

4.42.2.4 fir_lp()

```
std::vector< float > MHAFilter::fir_lp (
    float f_pass_,
    float f_stop_,
    float fs_,
    unsigned order_ )
```

Setup a nth order fir low pass filter.

This function calculates the filter coefficients of a nth order fir low pass filter filter. Frequency arguments above the nyquist frequency are accepted but the spectral response is truncated at the nyquist frequency

Returns

vector containing filter coefficients

Precondition

f_pass_ must be smaller or equal to f_stop_.

Parameters

*f_pass_*	Upper passband frequency
f_stop-	Lower stopband frequency
*fs_*	sample frequency

4.42.2.5 spec2fir()

```
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.

4.42.2.6 gcd()

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

greatest common divisor

4.42.2.7 sinc()

```
double MHAFilter::sinc (
    double x )
```

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

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

4.42.2.8 resampling_factors()

```
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. 906)*	if no rational resampling factor can be found.

4.43 MHAIOJack Namespace Reference

JACK IO.

Classes

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

4.43.1 Detailed Description

JACK IO.

4.44 MHAIOJackdb Namespace Reference

Classes

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

4.45 MHAIOPortAudio Namespace Reference

Classes

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

Functions

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

4.45.1 Function Documentation

4.45.1.1 parserFriendlyName()

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

4.46 mhaioutils Namespace Reference

Functions

- template<typename T >
  T **to_int_clamped** (float val)

4.46.1 Function Documentation

4.46.1.1 to_int_clamped()

```
template<typename T >
T mhaioutils::to_int_clamped (
            float val )
```

4.47 MHAJack Namespace Reference

Classes and functions for openMHA and JACK interaction.

Classes

- class **port_t**
  Class for one channel/port.
- class **client_t**
  Generic asynchronous JACK client.
- class **client_noncont_t**
  Generic client for synchronous playback and recording of waveform fragments.
- class **client_avg_t**
  Generic JACK client for averaging a system response across time.

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)

4.47.1 Detailed Description

Classes and functions for openMHA and JACK interaction.

4.47.2 Function Documentation

4.47.2.1 io()

```
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.

4.47.2.2 get_port_capture_latency()

```
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)

4.47.2.3 get_port_capture_latency_int()

```
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)

4.47.2.4 get_port_playback_latency()

```
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)

4.47.2.5 get_port_playback_latency_int()

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

4.48 MHAMultiSrc Namespace Reference

Collection of classes for selecting audio chunks from multiple sources.

Classes

- class **channel_t**
- class **channels_t**
- class **base_t**

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

4.48.1 Detailed Description

Collection of classes for selecting audio chunks from multiple sources.

4.49 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 **scale_var_t**
- class **fscale_t**
- class **fscale_bw_t**
- class **fftfb_vars_t**
  Set of configuration variables for FFT-based overlapping filters.
- class **fspacing_t**
  Class for frequency spacing, used by filterbank shape generator class.
- class **fftfb_t**
  FFT based overlapping filter bank.
- class **overlap_save_filterbank_t**
  *A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1148).*
- class **overlap_save_filterbank_analytic_t**
- class **fftfb_ac_info_t**

Typedefs

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

4.49.1 Detailed Description

Namespace for overlapping FFT based filter bank classes and functions.

4.49.2 Typedef Documentation

4.49.2.1 scale_fun_t `typedef mha_real_t() MHAOvlFilter::scale_fun_t( mha_real_t)`

4.50 MHAOvlFilter::barkscale Namespace Reference

Classes

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

Variables

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

4.50.1 Variable Documentation

4.50.1.1 vfreq `mha_real_t MHAOvlFilter::barkscale::vfreq [extern]`

4.50.1.2 vbark `mha_real_t MHAOvlFilter::barkscale::vbark [extern]`

4.51 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**))

4.51.1 Detailed Description

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

4.51.2 Function Documentation

4.51.2.1 hz2hz() `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

4.51.2.2 hz2khz() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2khz ( mha_real_t x )`

4.51.2.3 hz2octave() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2octave ( mha_real_t x )`

4.51.2.4 hz2third_octave() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2third_octave ( mha_real_t x )`

4.51.2.5 hz2bark() `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

4.51.2.6 hz2bark_analytic() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark_analytic ( mha_real_t x )`

4.51.2.7 hz2erb() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb ( mha_real_t x )`

4.51.2.8 hz2erb_glasberg1990() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2erb_glasberg1990 ( mha_real_t x )`

4.51.2.9 hz2log()

```
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

4.51.2.10 inv_scale()

```
mha_real_t MHAOvlFilter::FreqScaleFun::inv_scale (
        mha_real_t y,
        mha_real_t(*)( mha_real_t) fun )
```

4.52 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**)

4.52.1 Detailed Description

Shape functions for overlapping filters.

4.52.2 Function Documentation

4.52.2.1 rect() `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]

4.52.2.2 linear() `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]

4.52.2.3 hann() `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]

4.52.2.4 expflt() `mha_real_t MHAOvlFilter::ShapeFun::expflt ( mha_real_t x )`

4.52.2.5 gauss() `mha_real_t MHAOvlFilter::ShapeFun::gauss ( mha_real_t x )`

4.53 MHAParser Namespace Reference

Name space for the openMHA-Parser configuration language.

Namespaces

- **StrCnv**

  String converter namespace.

Classes

- class **keyword_list_t**
  Keyword list class.
- class **expression_t**
- class **entry_t**
- class **base_t**
  Base class for all parser items.
- class **parser_t**
  Parser node class.
- class **c_ifc_parser_t**
- class **monitor_t**
  Base class for monitors and variable nodes.
- class **variable_t**
  Base class for variable nodes.
- class **range_var_t**
  Base class for all variables with a numeric value range.
- class **kw_t**
  Variable with keyword list value.
- class **string_t**
  Variable with a string value.
- class **vstring_t**
  Vector variable with string values.
- class **bool_t**
  Variable with a boolean value ("yes"/"no")
- class **int_t**
  Variable with integer value.
- class **float_t**
  Variable with float value.
- class **complex_t**
  Variable with complex value.
- class **vint_t**
  Variable with vector<int> value.
- class **vfloat_t**
  Vector variable with float value.
- class **vcomplex_t**
  Vector variable with complex value.
- class **mint_t**
  Matrix variable with int value.
- class **mfloat_t**
  Matrix variable with float value.
- class **mcomplex_t**
  Matrix variable with complex value.
- class **int_mon_t**
  Monitor variable with int value.

- class **bool_mon_t**

  Monitor with string value.
- class **string_mon_t**

  Monitor with string value.
- class **vstring_mon_t**

  Vector of monitors with string value.
- class **vint_mon_t**

  Vector of ints monitor.
- class **mint_mon_t**

  Matrix of ints monitor.
- class **vfloat_mon_t**

  Vector of floats monitor.
- class **mfloat_mon_t**

  Matrix of floats monitor.
- class **float_mon_t**

  Monitor with float value.
- class **complex_mon_t**

  Monitor with complex value.
- class **vcomplex_mon_t**

  Monitor with vector of complex values.
- class **mcomplex_mon_t**

  Matrix of complex numbers monitor.
- class **commit_t**

  Parser variable with event-emission functionality.
- class **mhaconfig_mon_t**
- class **window_t**

  MHA configuration interface for a window function generator.
- class **mhapluginloader_t**

  Class to create a plugin loader in a parser, including the load logic.

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)

4.53.1 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. 30).

4.53.2 List of valid MHAParser items

- **Sub-parser**: **parser_t** (p. 1246)
- **Variables**:
  Numeric variables: **int_t** (p. 1212), **vint_t** (p. 1277), **float_t** (p. 1207), **vfloat_t** (p. 1272), **mfloat_t** (p. 1229)
  Other variables: **string_t** (p. 1260), **vstring_t** (p. 1282), **kw_t** (p. 1219), **bool_t** (p. 1190)
- **Monitors**:
  Numeric monitors: **int_mon_t** (p. 1210), **vint_mon_t** (p. 1275), **float_mon_t** (p. 1205), **vfloat_mon_t** (p. 1270)
  mfloat_mon_t (p. 1227)
  mcomplex_mon_t (p. 1223)
  Other monitors: **bool_mon_t** (p. 1188), **string_mon_t** (p. 1258), **vstring_mon_↩ t** (p. 1280)

Members can be inserted into the configuration namespace by using MHAParser::insert_item() or the **insert_member()** (p. 1831) macro.

4.53.3 Typedef Documentation

4.53.3.1 opact_t

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

4.53.3.2 query_t

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

4.53.3.3 opact_map_t

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

4.53.3.4 query_map_t

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

4.53.3.5 entry_map_t

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

4.53.3.6 c_parse_cmd_t

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

4.53.3.7 c_parse_err_t

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

4.53.4 Function Documentation

4.53.4.1 get_precision()

```
int MHAParser::get_precision ( )
```

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

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

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

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

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

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

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

4.53.4.9 strreplace()

```
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

4.53.4.10 envreplace()

```
void MHAParser::envreplace (
    std::string & s )
```

4.54 MHAParser::StrCnv Namespace Reference

String converter namespace.

Functions

- int **num_brackets** (const std::string &s)
  count number of brackets
- 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.

4.54.1 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.

4.54.2 Function Documentation

4.54.2.1 num_brackets() `int MHAParser::StrCnv::num_brackets ( const std::string & s )`

count number of brackets

Return number of brackets according to layer depth (vector:2, matrix:4, etc)

Parameters

s	String

Returns

Number of brackets, or -1 for empty string, or -2 for invalid brackets

4.54.2.2 bracket_balance() `int MHAParser::StrCnv::bracket_balance ( const std::string & s )`

4.54.2.3 str2val() [1/8] `void MHAParser::StrCnv::str2val ( const std::string & s, bool & v )`

Convert from string.

4.54.2.4 str2val() [2/8] `void MHAParser::StrCnv::str2val ( const std::string & s, float & v )`

Convert from string.

4.54.2.5 str2val() [3/8]

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

Convert from string.

4.54.2.6 str2val() [4/8]

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

Convert from string.

4.54.2.7 str2val() [5/8]

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

Convert from string.

4.54.2.8 str2val() [6/8]

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

Convert from string.

4.54.2.9 str2val() [7/8]

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

Converter for vector types.

4.54.2.10 str2val< mha_real_t >() 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.

4.54.2.11 str2val() [8/8] template<class arg_t >

```
void MHAParser::StrCnv::str2val (
            const std::string & s,
            std::vector< std::vector< arg_t > > & val )
```

Converter for matrix types.

4.54.2.12 val2str() [1/13] std::string MHAParser::StrCnv::val2str (

```
            const bool & v )
```

Convert to string.

4.54.2.13 val2str() [2/13] std::string MHAParser::StrCnv::val2str (

```
            const float & v )
```

Convert to string.

4.54.2.14 val2str() [3/13] std::string MHAParser::StrCnv::val2str (

```
            const mha_complex_t & v )
```

Convert to string.

4.54.2.15 val2str() [4/13] std::string MHAParser::StrCnv::val2str (

```
            const int & v )
```

Convert to string.

4.54.2.16 val2str() [5/13] std::string MHAParser::StrCnv::val2str (const keyword_list_t & v)

Convert to string.

4.54.2.17 val2str() [6/13] std::string MHAParser::StrCnv::val2str (const std::string & v)

Convert to string.

4.54.2.18 val2str() [7/13] std::string MHAParser::StrCnv::val2str (const std::vector< float > & v)

Convert to string.

4.54.2.19 val2str() [8/13] std::string MHAParser::StrCnv::val2str (const std::vector< mha_complex_t > & v)

Convert to string.

4.54.2.20 val2str() [9/13] std::string MHAParser::StrCnv::val2str (const std::vector< int > & v)

Convert to string.

4.54.2.21 val2str() [10/13] std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< int > > & v)

Convert to string.

4.54.2.22 val2str() [11/13] std::string MHAParser::StrCnv::val2str (const std::vector< std::string > & v)

Convert to string.

4.54.2.23 val2str() [12/13] std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< float > > & v)

Convert to string.

4.54.2.24 val2str() [13/13] std::string MHAParser::StrCnv::val2str (const std::vector< std::vector< **mha_complex_t** > > & v)

Convert to string.

4.55 MHAPlugin Namespace Reference

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

Classes

- class **cfg_node_t**

  A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.
- class **config_t**

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

  The template class for C++ openMHA plugins.

4.55.1 Detailed Description

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

4.56 MHAPlugin_Resampling Namespace Reference

Classes

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

4.57 MHAPlugin_Split Namespace Reference

Classes

- class **uni_processor_t**

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

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

  *The **splitted_part_t** (p. 1331) instance manages the plugin that performs processing on the reduced set of channels.*
- class **split_t**

  Implements split plugin.

Enumerations

- enum { **INVALID_THREAD_PRIORITY** = 999999999 }

  Invalid thread priority.

4.57.1 Detailed Description

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

4.57.2 Enumeration Type Documentation

4.57.2.1 anonymous enum `anonymous enum`

Invalid thread priority.

Enumerator

INVALID_↩ THREAD_↩ PRIORITY	

4.58 MHASignal Namespace Reference

Namespace for audio signal handling and processing classes.

Classes

- class **hilbert_fftw_t**
- class **spectrum_t**

  *a signal processing class for spectral data (based on **mha_spec_t** (p. 937))*
- class **waveform_t**

  *signal processing class for waveform data (based on **mha_wave_t** (p. 985))*
- class **doublebuffer_t**

  Double-buffering class.
- class **ringbuffer_t**

  A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.
- class **hilbert_t**

  Hilbert transformation of a waveform segment.
- class **minphase_t**

  Minimal phase function.
- class **stat_t**
- class **delay_wave_t**

  Delayline containing wave fragments.
- class **delay_spec_t**
- class **async_rmslevel_t**

  Class for asynchronous level metering.
- class **uint_vector_t**

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

  n-dimensional matrix with real or complex floating point values.
- class **schroeder_t**

  Schroeder tone complex class.
- class **quantizer_t**

Simple simulation of fixpoint quantization.

- class **loop_wavefragment_t**

  Copy a fixed waveform fragment to a series of waveform fragments of other size.
- class **delay_t**

  Class to realize a simple delay of waveform streams.
- class **subsample_delay_t**

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

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. 985)*.*
- **mha_real_t** **lin2db** (**mha_real_t** x, **mha_real_t** eps)

  Conversion from linear scale to dB (no SPL reference)
- **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** **sq2db** (**mha_real_t** x, **mha_real_t** eps=0.0f)

  conversion from squared values to dB (no SPL reference)
- **mha_real_t** **db2sq** (**mha_real_t** x)

  conversion from dB to squared values (no SPL reference)
- **mha_real_t** **pa2dbspl** (**mha_real_t** x, **mha_real_t** eps)

  Conversion from linear Pascal scale to dB SPL.
- **mha_real_t** **pa2dbspl** (**mha_real_t** x)

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

  Conversion from dB SPL to linear Pascal scale.
- **mha_real_t** **pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=0.0f)

  Conversion from squared Pascal scale to dB SPL.
- **mha_real_t** **dbspl2pa2** (**mha_real_t** x)

  conversion from dB SPL to squared 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=nullptr)

  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.

4.58.1 Detailed Description

Namespace for audio signal handling and processing classes.

4.58.2 Function Documentation

4.58.2.1 for_each()

```
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. 985).

Parameters

s	Pointer to a **mha_wave_t** (p. 985) structure
fun	Function to be applied (one argument)

4.58.2.2 lin2db() [1/2]

```
mha_real_t MHASignal::lin2db (
        mha_real_t x,
        mha_real_t eps ) [inline]
```

Conversion from linear scale to dB (no SPL reference)

Parameters

x	Linear input
eps	minimum linear value (if x $<$ eps --> convert eps instead), eps $<$ 0 not allowed

Returns

NaN if x $<$ 0 (log not defined for negative)

Exceptions

MHA_Error *(p. 906)*	if eps $<$ 0

4.58.2.3 lin2db() [2/2]

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

Conversion from linear scale to dB (no SPL reference)

Parameters

x	Linear input.

Returns

NaN if x $<$ 0 (log not defined for negative)

4.58.2.4 db2lin() `mha_real_t MHASignal::db2lin ( mha_real_t x ) [inline]`

Conversion from dB scale to linear (no SPL reference)

Parameters

x	dB input.

4.58.2.5 sq2db() `mha_real_t MHASignal::sq2db ( mha_real_t x, mha_real_t eps = 0.0f ) [inline]`

conversion from squared values to dB (no SPL reference)

Parameters

x	squared value input
eps	minimum squared value (if x $<$ eps --> convert eps instead), eps $<$ 0 not allowed

Returns

NaN if x $<$ 0 (log not defined for negative)

Exceptions

MHA_Error *(p. 906)*	if eps $<$ 0

4.58.2.6 db2sq() `mha_real_t MHASignal::db2sq ( mha_real_t x ) [inline]`

conversion from dB to squared values (no SPL reference)

Parameters

x	dB input

4.58.2.7 pa2dbspl() [1/2] mha_real_t MHASignal::pa2dbspl (mha_real_t *x*, mha_real_t *eps*) [inline]

Conversion from linear Pascal scale to dB SPL.

Parameters

x	Linear input
eps	minimum pascal value (if x < eps --> convert eps instead),

Precondition

eps >= 0

Returns

NaN if x < 0 (logarithm not defined for negative numbers)

Exceptions

MHA_Error *(p. 906)*	if eps < 0

4.58.2.8 pa2dbspl() [2/2] mha_real_t MHASignal::pa2dbspl (mha_real_t *x*) [inline]

Conversion from linear Pascal scale to dB SPL.

Parameters

x	Linear input

Returns

NaN if x $<$ 0 (log not defined for negative)

4.58.2.9 dbspl2pa() `mha_real_t MHASignal::dbspl2pa ( mha_real_t x ) [inline]`

Conversion from dB SPL to linear Pascal scale.

Parameters

x	Linear input.

4.58.2.10 pa22dbspl() `mha_real_t MHASignal::pa22dbspl ( mha_real_t x, mha_real_t eps = 0.0f ) [inline]`

Conversion from squared Pascal scale to dB SPL.

Parameters

x	squared pascal input
eps	minimum squared-pascal value (if x $<$ eps --> convert eps instead), eps $<$ 0 not allowed

Returns

NaN if x $<$ 0 (log not defined for negative)

Exceptions

MHA_Error *(p. 906)*	if eps < 0

4.58.2.11 dbspl2pa2()

```
mha_real_t MHASignal::dbspl2pa2 (
    mha_real_t x ) [inline]
```

conversion from dB SPL to squared Pascal scale

Parameters

x	dB SPL input

4.58.2.12 smp2sec()

```
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

4.58.2.13 sec2smp()

```
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

4.58.2.14 scale()

```
void MHASignal::scale (
    mha_spec_t * dest,
    const mha_wave_t * src )
```

4.58.2.15 limit()

```
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

4.58.2.16 kth_smallest() 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

4.58.2.17 median() 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

4.58.2.18 mean() 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

4.58.2.19 quantile() `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.

4.58.2.20 saveas_mat4() [1/3]

```
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.

Parameters

varname	Matlab variable name (Matlab4 limitations on maximal length are not checked).
fh	File handle to Matlab4 file.

4.58.2.21 saveas_mat4() [2/3]

```
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.

4.58.2.22 saveas_mat4() [3/3]

```
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.

4.58.2.23 copy_permuted() `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.

4.58.3 Variable Documentation

4.58.3.1 signal_counter `unsigned long int MHASignal::signal_counter = 0 [extern]`

Signal counter to produce signal ID strings.

4.59 MHASndFile Namespace Reference

Classes

- class **sf_t**
- class **sf_wave_t**

4.60 MHATableLookup Namespace Reference

Namespace for table lookup classes.

Classes

- class **table_t**
- class **linear_table_t**
  Class for interpolation with equidistant x values.
- class **xy_table_t**
  Class for interpolation with non-equidistant x values.

4.60.1 Detailed Description

Namespace for table lookup classes.

4.61 MHAUtils Namespace Reference

Functions

- bool **is_multiple_of** (const unsigned big, const unsigned small)
- bool **is_power_of_two** (const unsigned n)
- bool **is_multiple_of_by_power_of_two** (const unsigned big, const unsigned small)
- std::string **strip** (const std::string &line)
- std::string **remove** (const std::string &str_, char c)
- bool **is_denormal** (**mha_real_t** x)

  Get the normal-ness of a mha_real_t.
- bool **is_denormal** (const **mha_complex_t** &x)

  Get the normal-ness of a complex number.
- bool **is_denormal** (const std::complex< **mha_real_t** > &x)

  Get the normal-ness of a complex number.
- **mha_real_t** **spl2hl** (**mha_real_t** f)

  Get the offset of between dB(SPL) and dB(HL) for a given frequency according to ISO 389-7:2005 (freefield); e.g.

4.61.1 Function Documentation

4.61.1.1 is_multiple_of()

```
bool MHAUtils::is_multiple_of (
    const unsigned big,
    const unsigned small ) [inline]
```

4.61.1.2 is_power_of_two()

```
bool MHAUtils::is_power_of_two (
    const unsigned n ) [inline]
```

4.61.1.3 is_multiple_of_by_power_of_two()

```
bool MHAUtils::is_multiple_of_by_power_of_two (
            const unsigned big,
            const unsigned small ) [inline]
```

4.61.1.4 strip()

```
std::string MHAUtils::strip (
            const std::string & line ) [inline]
```

4.61.1.5 remove()

```
std::string MHAUtils::remove (
            const std::string & str_,
            char c ) [inline]
```

4.61.1.6 is_denormal() [1/3]

```
bool MHAUtils::is_denormal (
            mha_real_t x ) [inline]
```

Get the normal-ness of a mha_real_t.

Returns true iff x is not equal to zero and the absolute value of x is smaller than the minimum positive normalized value of mha_real_t.

Parameters

x	A mha_real_t floating point number

Returns

True if x is denormal, false otherwise

4.61.1.7 is_denormal() [2/3] bool MHAUtils::is_denormal (const **mha_complex_t** & *x*) [inline]

Get the normal-ness of a complex number.

Overload for **mha_complex_t** (p. 886). Returns true iff one or both of real and imaginary part are denormal

Parameters

x	[in] A **mha_complex_t** (p. 886) number

Returns

True if at least one component of x is denormal, false otherwise

4.61.1.8 is_denormal() [3/3] bool MHAUtils::is_denormal (const std::complex< **mha_real_t** > & *x*) [inline]

Get the normal-ness of a complex number.

Overload for std::complex Returns true iff one or both of real and imaginary part are denormal.

Parameters

x	[in] A **mha_complex_t** (p. 886) number

Returns

True if at least one component of x is denormal, false otherwise

4.61.1.9 spl2hl() `mha_real_t MHAUtils::spl2hl ( mha_real_t f )`

Get the offset of between dB(SPL) and dB(HL) for a given frequency according to ISO 389-7↩ :2005 (freefield); e.g.

an intensity of 22.1 dB(SPL) at 125 Hz is equivalent to 0 dB(HL), so spl2hl(125)=-22.1. Interpolation between mesh points is linear. The correction values for frequencies above 16 kHz are extrapolated."

Parameters

in	*f*	The frequency in Hz for which the offset shall be returned

Returns

The offet between dB(SPL) and dB(HL) at frequency f

Exceptions

MHA_Error *(p. 906)*	if f<0

4.62 MHAWindow Namespace Reference

Collection of Window types.

Classes

- class **base_t**
  Common base for window types.
- class **fun_t**
  Generic window based on a generator function.
- class **rect_t**
  Rectangular window.
- class **bartlett_t**
  Bartlett window.
- class **hanning_t**
  von-Hann window
- class **hamming_t**
  Hamming window.
- class **blackman_t**
  Blackman 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.

4.62.1 Detailed Description

Collection of Window types.

4.62.2 Function Documentation

4.62.2.1 rect() `float MHAWindow::rect ( float x )`

Rectangular window function.

4.62.2.2 bartlett() `float MHAWindow::bartlett ( float x )`

Bartlett window function.

4.62.2.3 hanning() `float MHAWindow::hanning ( float x )`

Hanning window function.

4.62.2.4 hamming() `float MHAWindow::hamming ( float x )`

Hamming window function.

4.62.2.5 blackman() `float MHAWindow::blackman ( float x )`

Blackman window function.

4.63 multibandcompressor Namespace Reference

Classes

- class **plugin_signals_t**
- class **fftfb_plug_t**
- class **interface_t**

4.64 noise_psd_estimator Namespace Reference

Classes

- class **noise_psd_estimator_t**
- class **noise_psd_estimator_if_t**

4.65 overlapadd Namespace Reference

Classes

- class **overlapadd_t**
- class **overlapadd_if_t**

4.66 plingploing Namespace Reference

All classes for the plingploing music generator live in this namespace.

Classes

- class **plingploing_t**

  Run-time configuration of the plingploing music generator.
- class **if_t**

  Plugin class of the plingploing music generator.

Functions

- double **drand** (double a, double b)

4.66.1 Detailed Description

All classes for the plingploing music generator live in this namespace.

4.66.2 Function Documentation

4.66.2.1 drand() `double plingploing::drand (`
`double a,`
`double b )`

4.67 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. 996) structures, and report differences as an error.*

4.67.1 Function Documentation

4.67.1.1 mhastrdomain() const char * PluginLoader::mhastrdomain (mha_domain_t d)

4.67.1.2 mhaconfig_compare() void PluginLoader::mhaconfig_compare (const mhaconfig_t & req, const mhaconfig_t & avail, const std::string & pref = "")

Compare two **mhaconfig_t** (p. 996) structures, and report differences as an error.

Parameters

req	Expected **mhaconfig_t** (p. 996) structure
avail	Available **mhaconfig_t** (p. 996) structure
pref	Prefix for error messages

4.68 plugins Namespace Reference

Namespaces

- **hoertech**

4.69 plugins::hoertech Namespace Reference

Namespaces

- **acrec**

4.70 plugins::hoertech::acrec Namespace Reference

Classes

- class **acwriter_t**
  acwriter_t *(p. 1537) decouples signal processing from writing to disk.*
- class **acrec_t**
  Plugin interface class of plugin acrec.

Functions

- std::string **to_iso8601** (time_t tm)

4.70.1 Function Documentation

4.70.1.1 to_iso8601() `std::string plugins::hoertech::acrec::to_iso8601 ( time_t tm )`

4.71 rmslevel Namespace Reference

Classes

- class **rmslevel_if_t**
  Rmslevel plugin.

Enumerations

- enum class **UNIT** { **SPL** =0 , **HL** =1 }

4.71.1 Enumeration Type Documentation

4.71.1.1 UNIT enum rmslevel::UNIT [strong]

Enumerator

SPL	
HL	

4.72 rohBeam Namespace Reference

Classes

- struct **configOptions**
- class **rohConfig**
- class **rohBeam**

Functions

- double **j0** (double x)
  Cylindrical bessel function of the first kind of order 0.

Variables

- auto **scalarify** =[ ](auto t){return t(0);}
- constexpr float **CONST_C** = 343.0115f
- constexpr int **refL** = 0
- constexpr int **refR** = 3

4.72.1 Function Documentation

4.72.1.1 j0() double rohBeam::j0 (double *x*)

Cylindrical bessel function of the first kind of order 0.

Parameters

x	the argument of the function

Returns

j0(x)

4.72.2 Variable Documentation

4.72.2.1 scalarify `auto rohBeam::scalarify =[](auto t){return t(0);}`

4.72.2.2 CONST_C `constexpr float rohBeam::CONST_C = 343.0115f [constexpr]`

4.72.2.3 refL `constexpr int rohBeam::refL = 0 [constexpr]`

4.72.2.4 refR `constexpr int rohBeam::refR = 3 [constexpr]`

4.73 route Namespace Reference

Classes

- class **process_t**
- class **interface_t**

4.74 shadowfilter_begin Namespace Reference

Classes

- class **cfg_t**
- class **shadowfilter_begin_t**

4.75 shadowfilter_end Namespace Reference

Classes

- class **cfg_t**
- class **shadowfilter_end_t**

4.76 smooth_cepstrum Namespace Reference

Classes

- class **smooth_params**
- class **smooth_cepstrum_t**
- class **smooth_cepstrum_if_t**

4.77 smoothgains_bridge Namespace Reference

Classes

- class **smoothspec_wrap_t**
- class **overlapadd_if_t**

4.78 testplugin Namespace Reference

Classes

- class **config_parser_t**
- class **ac_parser_t**
- class **signal_parser_t**
- class **if_t**

4.79 trigger2lsl Namespace Reference

namespace for **trigger2lsl** (p. 166) plugin

Classes

- class **trigger2lsl_rt_t**
  *real-time configuration class for **trigger2lsl** (p. 166) plugin*
- class **trigger2lsl_if_t**
  *Plugin interface class of plugin **trigger2lsl** (p. 166).*

4.79.1 Detailed Description

namespace for **trigger2lsl** (p. 166) plugin

4.80 wave2lsl Namespace Reference

All types for the **wave2lsl** (p. 166) plugins live in this namespace.

Classes

- class **cfg_t**
  *Runtime configuration class of the **wave2lsl** (p. 166) plugin.*
- class **wave2lsl_t**
  *Plugin class of **wave2lsl** (p. 166).*

4.80.1 Detailed Description

All types for the **wave2lsl** (p. 166) plugins live in this namespace.

4.81 windnoise Namespace Reference

namespace for plugin windnoise which detects and cancels wind noise

Classes

- class **cfg_t**
  Runtime config class for windnoise plugin.
- class **if_t**
  interface class for windnoise plugin

4.81.1 Detailed Description

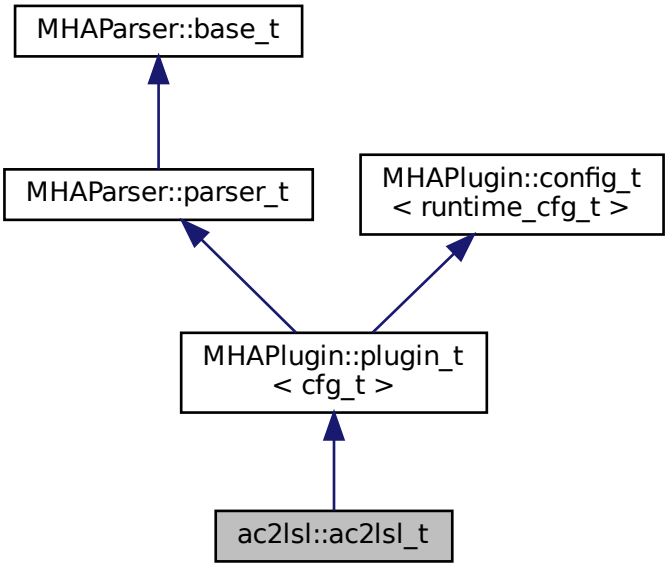
namespace for plugin windnoise which detects and cancels wind noise

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** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)

  *Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 170).*
- **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

- 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**
- **MHAParser::float_t nominal_srate**
- **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_t()

```
ac2lsl::ac2lsl_t::ac2lsl_t (
        MHA_AC::algo_comm_t & iac,
        const std::string & configured_name )
```

5.1.3 Member Function Documentation

5.1.3.1 prepare()

```
void ac2lsl::ac2lsl_t::prepare (
        mhaconfig_t & ) [virtual]
```

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 170).

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 1301).

5.1.3.2 process() [1/3]

```
mha_wave_t* ac2lsl::ac2lsl_t::process (
        mha_wave_t * s ) [inline]
```

Processing fct for waveforms.

Calls **process(void)** (p. 169).

5.1.3.3 process() [2/3]

```
mha_spec_t* ac2lsl::ac2lsl_t::process (
        mha_spec_t * s ) [inline]
```

Processing fct for spectra.

Calls **process(void)** (p. 169).

5.1.3.4 process() [3/3]

```
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. 172).

5.1.3.5 release() void ac2lsl::ac2lsl_t::release (void) [virtual]

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1302).

5.1.3.6 update() void ac2lsl::ac2lsl_t::update ( ) [private]

Construct new runtime configuration.

5.1.4 Member Data Documentation

5.1.4.1 vars MHAParser::vstring_t ac2lsl::ac2lsl_t::vars [private]

5.1.4.2 source_id MHAParser::string_t ac2lsl::ac2lsl_t::source_id [private]

5.1.4.3 rt_strict MHAParser::bool_t ac2lsl::ac2lsl_t::rt_strict [private]

5.1.4.4 activate MHAParser::bool_t ac2lsl::ac2lsl_t::activate [private]

5.1.4.5 skip MHAParser::int_t ac2lsl::ac2lsl_t::skip [private]

5.1.4.6 nominal_srate `MHAParser::float_t ac2lsl::ac2lsl_t::nominal_srate [private]`

5.1.4.7 patchbay `MHAEvents::patchbay_t< ac2lsl_t> ac2lsl::ac2lsl_t::patchbay [private]`

5.1.4.8 is_first_run `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** (**MHA_AC::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 **MHA_AC::comm_var_t** &v)
- void **check_and_send** ()

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 **MHA_AC::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::cfg_t (
    MHA_AC::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 create_or_replace_var()

```
void cfg_t::create_or_replace_var (
    const std::string & name,
    const  MHA_AC::comm_var_t & v ) [private]
```

5.2.3.2 check_and_send()

```
void cfg_t::check_and_send ( ) [private]
```

5.2.3.3 process() void cfg_t::process (void)

5.2.4 Member Data Documentation

5.2.4.1 varlist 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 skipcnt unsigned ac2lsl::cfg_t::skipcnt [private]

Counter of frames to skip.

5.2.4.3 skip const unsigned ac2lsl::cfg_t::skip [private]

Number of frames to skip after each send.

5.2.4.4 srate const double ac2lsl::cfg_t::srate [private]

Sampling rate of the stream.

5.2.4.5 source_id const std::string ac2lsl::cfg_t::source_id [private]

User configurable source id.

5.2.4.6 ac `const MHA_AC::algo_comm_t& ac2lsl::cfg_t::ac [private]`

Handle to the ac space.

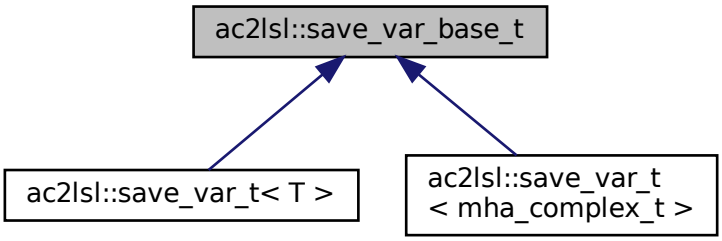
The documentation for this class was generated from the following file:

- **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** (unsigned num_entries)=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 ~**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 ~save_var_base_t() `virtual ac2lsl::save_var_base_t::~save_var_base_t ( ) [virtual], [default]`

5.3.3 Member Function Documentation

5.3.3.1 send_frame() `virtual void ac2lsl::save_var_base_t::send_frame ( unsigned num_entries ) [pure virtual]`

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 182), and **ac2lsl::save_var_t< T >** (p. 179).

5.3.3.2 get_buf_address() `virtual void* ac2lsl::save_var_base_t::get_buf_address ( ) const [pure virtual], [noexcept]`

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 181), and **ac2lsl::save_var_t< T >** (p. 178).

5.3.3.3 set_buf_address() `virtual void ac2lsl::save_var_base_t::set_buf_address ( void * data ) [pure virtual]`

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 181), and **ac2lsl::save_var_t< T >** (p. 178).

5.3.3.4 info() `virtual lsl::stream_info ac2lsl::save_var_base_t::info ( ) const [pure virtual], [noexcept]`

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 182), and **ac2lsl::save_var_t< T >** (p. 178).

5.3.3.5 data_type() `virtual unsigned ac2lsl::save_var_base_t::data_type ( ) const [pure virtual], [noexcept]`

Implemented in **ac2lsl::save_var_t< mha_complex_t >** (p. 182), and **ac2lsl::save_var_t< T >** (p. 178).

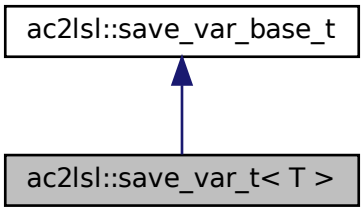
The documentation for this class was generated from the following file:

- **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_↩ channels_, 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 noexcept override
  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 noexcept override
  Get stream info object from stream outlet.
- virtual unsigned **data_type** () const noexcept override
  Get data type id according MHA convention.
- virtual ~**save_var_t** ()=default
- virtual void **send_frame** (unsigned num_entries) 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 save_var_t() template<typename T >

```
ac2lsl::save_var_t< T >::  save_var_t (
            const std::string & name_,
            const std::string & type_,
            unsigned num_channels_,
            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 ~save_var_t() template<typename T >

virtual **ac2lsl::save_var_t**< T >::~ **save_var_t** ( ) [virtual], [default]

5.4.3 Member Function Documentation

5.4.3.1 get_buf_address() 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. 175).

5.4.3.2 set_buf_address() 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. 175).

5.4.3.3 info() 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. 175).

5.4.3.4 data_type() 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. 175).

5.4.3.5 send_frame() template<typename T >

virtual void **ac2lsl::save_var_t**< T >::send_frame (
unsigned *num_entries*) [inline], [override], [virtual]

Send a frame to lsl.

Implements **ac2lsl::save_var_base_t** (p. 175).

5.4.4 Member Data Documentation

5.4.4.1 stream template<typename T >

lsl::stream_outlet **ac2lsl::save_var_t**< T >::stream [private]

LSL stream outlet.

Interface to lsl

5.4.4.2 buf template<typename T >

T* **ac2lsl::save_var_t**< T >::buf [private]

Pointer to data buffer of the ac variable.

5.4.4.3 data_type_ template<typename T >

const unsigned **ac2lsl::save_var_t**< T >::data_type_ [private]

Data type id according to MHA convention.

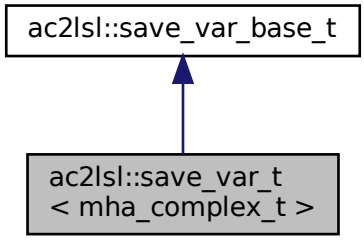
The documentation for this class was generated from the following file:

- **ac2lsl.cpp**

5.5 ac2lsl::save_var_t< mha_complex_t > Class 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_, unsigned num_↩ channels_, 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 noexcept override
- virtual void **set_buf_address** (void *data) override
- virtual lsl::stream_info **info** () const noexcept override

  Get buffer address as void pointer.
- virtual unsigned **data_type** () const noexcept override

  Cast the input pointer to the appropriate type and set the buffer address.
- virtual ~**save_var_t** ()=default
- virtual void **send_frame** (unsigned num_entries) 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 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 save_var_t() ac2lsl::save_var_t< mha_complex_t >::save_var_t (

```
const std::string & name_,
const std::string & type_,
unsigned num_channels_,
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 ~save_var_t() virtual ac2lsl::save_var_t< mha_complex_t >::~save_var_t ( ) [virtual], [default]

5.5.3 Member Function Documentation

5.5.3.1 get_buf_address() virtual void* ac2lsl::save_var_t< mha_complex_t >::get_buf_address ( ) const [inline], [override], [virtual], [noexcept]

Implements **ac2lsl::save_var_base_t** (p. 175).

5.5.3.2 set_buf_address() virtual void ac2lsl::save_var_t< mha_complex_t >::set_buf_address (void * *data*) [inline], [override], [virtual]

Implements **ac2lsl::save_var_base_t** (p. 175).

5.5.3.3 info() 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. 175).

5.5.3.4 data_type() 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. 175).

5.5.3.5 send_frame() virtual void ac2lsl::save_var_t< mha_complex_t >::send_frame (unsigned *num_entries*) [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. 175).

5.5.4 Member Data Documentation

5.5.4.1 stream `lsl::stream_outlet ac2lsl::save_var_t< mha_complex_t >::stream [private]`

LSL stream outlet.

Interface to lsl

5.5.4.2 buf `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 name `const std::string ac2lsl::type_info::name`

5.6.1.2 format `const lsl::channel_format_t ac2lsl::type_info::format`

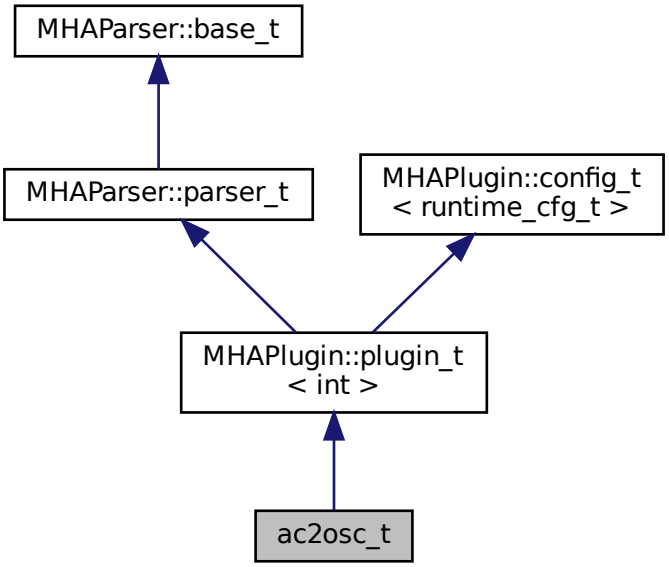
The documentation for this struct was generated from the following file:

- **ac2lsl.cpp**

5.7 ac2osc_t Class Reference

Plugin class of the ac2osc plugin.

Inheritance diagram for ac2osc_t:



Public Member Functions

- **ac2osc_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  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?
- std::unique_ptr< **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::ac2osc_t (`

```
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

C'tor of plugin class.

5.7.3 Member Function Documentation

5.7.3.1 prepare() `void ac2osc_t::prepare (`

`mhaconfig_t & cf ) [virtual]`

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.7.3.2 process() [1/3] `mha_wave_t* ac2osc_t::process (`

`mha_wave_t * s ) [inline]`

Processing fct for waveforms.

Calls **process(void)** (p. 186).

5.7.3.3 process() [2/3] `mha_spec_t* ac2osc_t::process (`

`mha_spec_t * s ) [inline]`

Processing fct for spectra.

Calls **process(void)** (p. 186).

5.7.3.4 process() [3/3] `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 release() `void ac2osc_t::release (`

`void ) [virtual]`

Release frees osc related memory, does cleanup.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.7.3.6 send_osc_float() void ac2osc_t::send_osc_float ( ) [private]

5.7.3.7 update_mode() void ac2osc_t::update_mode ( ) [private]

Start/Stop sending of messages.

5.7.4 Member Data Documentation

5.7.4.1 host MHAParser::string_t ac2osc_t::host [private]

OSC server host name.

5.7.4.2 port MHAParser::string_t ac2osc_t::port [private]

OSC server port.

5.7.4.3 ttl MHAParser::int_t ac2osc_t::ttl [private]

Time-to-live of UDP packages.

5.7.4.4 vars MHAParser::vstring_t ac2osc_t::vars [private]

List of AC variables to be saved, empty for all.

5.7.4.5 mode `MHAParser::kw_t ac2osc_t::mode [private]`

Record mode.

5.7.4.6 skip `MHAParser::int_t ac2osc_t::skip [private]`

number of frames to skip after sending

5.7.4.7 rt_strict `MHAParser::bool_t ac2osc_t::rt_strict [private]`

abort if used in real-time thread?

5.7.4.8 acspace `std::unique_ptr< MHA_AC::acspace2matrix_t> ac2osc_t::acspace [private]`

5.7.4.9 patchbay `MHAEvents::patchbay_t< ac2osc_t> ac2osc_t::patchbay [private]`

5.7.4.10 b_record `bool ac2osc_t::b_record [private]`

5.7.4.11 rtmem `uint8_t* ac2osc_t::rtmem [private]`

5.7.4.12 framerate `float ac2osc_t::framerate [private]`

5.7.4.13 skipcnt `int ac2osc_t::skipcnt [private]`

5.7.4.14 lo_addr `lo_address ac2osc_t::lo_addr [private]`

5.7.4.15 is_first_run `bool ac2osc_t::is_first_run [private]`

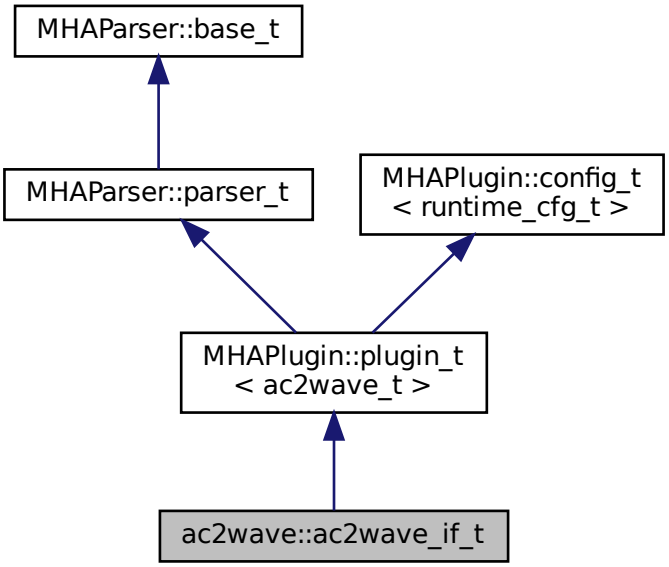
The documentation for this class was generated from the following file:

- **ac2osc.cpp**

5.8 ac2wave::ac2wave_if_t Class Reference

ac2wave (p. 79) plugin interface class

Inheritance diagram for ac2wave::ac2wave_if_t:



Public Member Functions

- **ac2wave_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor.
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)
  process callback for spectral input
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  process callback for waveform input
- void **prepare** (**mhaconfig_t** &tf)
  Prepare callback.
- void **release** ()
  Release callback.

Private Member Functions

- void **update** ()
  Update the real-time configuration after a configuration change.

Private Attributes

- **MHAParser::string_t name**
  AC variable name.
- **MHAParser::float_t gain_in**
  Linear gain for main input signal.
- **MHAParser::float_t gain_ac**
  Linear gain for AC input signal.
- **MHAParser::int_t delay_in**
  Delay of main input signal in fragments.
- **MHAParser::int_t delay_ac**
  Delay of AC input signal in fragments.
- std::unique_ptr< **MHASignal::waveform_t** > **zeros**
  Zero signal, used for Spectral inputs.
- **MHAEvents::patchbay_t**< **ac2wave_if_t** > **patchbay**
  Patchbay for connecting the configuration parsers to the update callback.

Additional Inherited Members

5.8.1 Detailed Description

ac2wave (p. 79) plugin interface class

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ac2wave_if_t() `ac2wave::ac2wave_if_t::ac2wave_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor.

Parameters

iac	Handle to the AC space
configured_name	Assigned name of the plugin within the configuration tree

5.8.3 Member Function Documentation

5.8.3.1 process() [1/2] `mha_wave_t * ac2wave::ac2wave_if_t::process (`

```
mha_spec_t * )
```

process callback for spectral input

5.8.3.2 process() [2/2] `mha_wave_t * ac2wave::ac2wave_if_t::process (`

```
mha_wave_t * s )
```

process callback for waveform input

Parameters

s	input Pointer to input waveform

Returns

Pointer to output waveform. Points to an internal buffer of s_in_delayed

5.8.3.3 prepare() `void ac2wave::ac2wave_if_t::prepare ( mhaconfig_t & tf ) [virtual]`

Prepare callback.

Changes the output domain to waveform, using the input dimensions in tf.

Parameters

tf	Input signal configuration

Implements **MHAPlugin::plugin_t< ac2wave_t >** (p. 1301).

5.8.3.4 release() `void ac2wave::ac2wave_if_t::release ( void ) [inline], [virtual]`

Release callback.

Reimplemented from **MHAPlugin::plugin_t< ac2wave_t >** (p. 1302).

5.8.3.5 update() `void ac2wave::ac2wave_if_t::update ( ) [private]`

Update the real-time configuration after a configuration change.

5.8.4 Member Data Documentation

5.8.4.1 name `MHAParser::string_t ac2wave::ac2wave_if_t::name [private]`

AC variable name.

5.8.4.2 gain_in

MHAParser::float_t ac2wave::ac2wave_if_t::gain_in [private]

Linear gain for main input signal.

5.8.4.3 gain_ac

MHAParser::float_t ac2wave::ac2wave_if_t::gain_ac [private]

Linear gain for AC input signal.

5.8.4.4 delay_in

MHAParser::int_t ac2wave::ac2wave_if_t::delay_in [private]

Delay of main input signal in fragments.

5.8.4.5 delay_ac

MHAParser::int_t ac2wave::ac2wave_if_t::delay_ac [private]

Delay of AC input signal in fragments.

5.8.4.6 zeros

std::unique_ptr< **MHASignal::waveform_t**> ac2wave::ac2wave_if_t::zeros [private]

Zero signal, used for Spectral inputs.

5.8.4.7 patchbay

MHAEvents::patchbay_t< **ac2wave_if_t**> ac2wave::ac2wave_if_t↩ ::patchbay [private]

Patchbay for connecting the configuration parsers to the update callback.

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

- **ac2wave.cpp**

5.9 ac2wave::ac2wave_t Class Reference

ac2wave (p. 79) real-time configuration class

Public Member Functions

- **ac2wave_t** (unsigned int frames_, unsigned int channels_, **MHA_AC::algo_comm_t** &ac_, std::string name_, float gain_in_, float gain_ac_, unsigned int delay_in_, unsigned int delay_ac_)
  Constructor of the real-time configuration.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
  Process callback.

Private Attributes

- unsigned int **frames**
  frames_ Number of frames in the input signal
- unsigned int **channels**
  Number of channels in the input signal.
- **mha_wave_t** **w**
  AC input waveform.
- **MHA_AC::algo_comm_t** & **ac**
  Handle to AC space.
- std::string **name**
  Name of the input AC variable.
- **MHASignal::delay_wave_t** **delay_in**
  Delay, in fragments, for the input signal.
- **MHASignal::delay_wave_t** **delay_ac**
  Delay, in fragments, for the AC input.
- **mha_real_t** **gain_in**
  Linear gain for the input signal.
- **mha_real_t** **gain_ac**
  Linear gain for the AC input.

5.9.1 Detailed Description

ac2wave (p. 79) real-time configuration class

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ac2wave_t()

```
ac2wave::ac2wave_t::ac2wave_t (
        unsigned int frames_,
        unsigned int channels_,
        MHA_AC::algo_comm_t & ac_,
        std::string name_,
        float gain_in_,
        float gain_ac_,
        unsigned int delay_in_,
        unsigned int delay_ac_ )
```

Constructor of the real-time configuration.

Parameters

*frames_*	Number of frames in the input signal
*channels_*	Number of channels in the input signal
*ac_*	Handle to AC space
*name_*	Name of the AC variable to be mixed in
*gain_in_*	Linear gain for the input signal
*gain_ac_*	Linear gain for the AC input
*delay_in_*	Delay, in fragments, for the input signal
*delay_ac_*	Delay, in fragments, for the AC input

5.9.3 Member Function Documentation

5.9.3.1 process()

```
mha_wave_t * ac2wave::ac2wave_t::process (
        mha_wave_t * s )
```

Process callback.

Parameters

s	Pointer to input waveform

Returns

Pointer to output waveform

5.9.4 Member Data Documentation

5.9.4.1 frames `unsigned int ac2wave::ac2wave_t::frames [private]`

frames_ Number of frames in the input signal

5.9.4.2 channels `unsigned int ac2wave::ac2wave_t::channels [private]`

Number of channels in the input signal.

5.9.4.3 w **`mha_wave_t`** `ac2wave::ac2wave_t::w [private]`

AC input waveform.

5.9.4.4 ac **`MHA_AC::algo_comm_t`**`& ac2wave::ac2wave_t::ac [private]`

Handle to AC space.

5.9.4.5 name `std::string ac2wave::ac2wave_t::name [private]`

Name of the input AC variable.

5.9.4.6 delay_in **`MHASignal::delay_wave_t`** `ac2wave::ac2wave_t::delay_in [private]`

Delay, in fragments, for the input signal.

5.9.4.7 delay_ac `MHASignal::delay_wave_t ac2wave::ac2wave_t::delay_ac [private]`

Delay, in fragments, for the AC input.

5.9.4.8 gain_in `mha_real_t ac2wave::ac2wave_t::gain_in [private]`

Linear gain for the input signal.

5.9.4.9 gain_ac `mha_real_t ac2wave::ac2wave_t::gain_ac [private]`

Linear gain for the AC input.

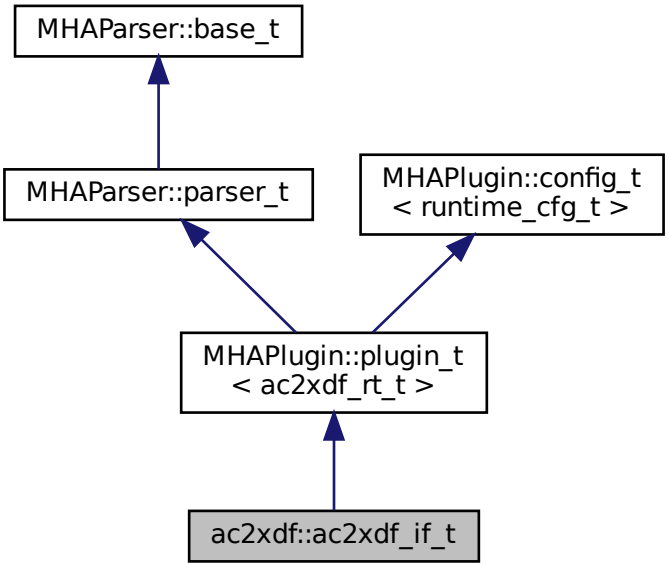
The documentation for this class was generated from the following file:

- **ac2wave.cpp**

5.10 ac2xdf::ac2xdf_if_t Class Reference

Plugin interface class of plugin **ac2xdf** (p. 79).

Inheritance diagram for ac2xdf::ac2xdf_if_t:



Public Member Functions

- template<class mha_signal_t >
  mha_signal_t * **process** (mha_signal_t *s)
  Process callback.
- void **prepare** (**mhaconfig_t** &)
  Prepare callback.
- void **release** ()
  Ensure recorded data is flushed to disk.
- **ac2xdf_if_t** (**algo_comm_t** &iac, const std::string &)
  Plugin interface constructor.

Private Member Functions

- void **start_new_session** ()
  Configuration callback called whenever configuaration variable "record" is written to.

Private Attributes

- **MHAParser::bool_t record**
- **MHAParser::int_t fifolen**
- **MHAParser::int_t minwrite**
- **MHAParser::string_t prefix**
- **MHAParser::vstring_t varnames**
- **MHAParser::vfloat_t nominal_sampling_rates**
- **MHAParser::bool_t use_date**
- **MHAEvents::patchbay_t< ac2xdf_if_t > patchbay**

Additional Inherited Members

5.10.1 Detailed Description

Plugin interface class of plugin **ac2xdf** (p. 79).

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ac2xdf_if_t()

```
ac2xdf::ac2xdf_if_t::ac2xdf_if_t (
        algo_comm_t & iac,
    const std::string &  ) [inline]
```

Plugin interface constructor.

Parameters

iac	Algorithm communication variable space.

5.10.3 Member Function Documentation

5.10.3.1 process()

```
template<class mha_signal_t >
mha_signal_t* ac2xdf::ac2xdf_if_t::process (
        mha_signal_t * s ) [inline]
```

Process callback.

Pushes the data from one AC variable into the fifo.

Returns

the unmodified input signal.

Parameters

s	input signal. The audio signal is not used or modified.

5.10.3.2 prepare() `void ac2xdf::ac2xdf_if_t::prepare ( mhaconfig_t & ) [inline], [virtual]`

Prepare callback.

ac2xdf (p. 79) does not modify the signal parameters.

Parameters

cf	The signal parameters.

Implements **MHAPlugin::plugin_t< ac2xdf_rt_t >** (p. 1301).

5.10.3.3 release() `void ac2xdf::ac2xdf_if_t::release ( void ) [inline], [virtual]`

Ensure recorded data is flushed to disk.

Reimplemented from **MHAPlugin::plugin_t< ac2xdf_rt_t >** (p. 1302).

5.10.3.4 start_new_session() `void ac2xdf::ac2xdf_if_t::start_new_session ( ) [inline], [private]`

Configuration callback called whenever configuaration variable "record" is written to.

Always flush the old file if there's one.

5.10.4 Member Data Documentation

5.10.4.1 record `MHAParser::bool_t ac2xdf::ac2xdf_if_t::record [private]`

5.10.4.2 fifolen

MHAParser::int_t ac2xdf::ac2xdf_if_t::fifolen [private]

5.10.4.3 minwrite

MHAParser::int_t ac2xdf::ac2xdf_if_t::minwrite [private]

5.10.4.4 prefix

MHAParser::string_t ac2xdf::ac2xdf_if_t::prefix [private]

5.10.4.5 varnames

MHAParser::vstring_t ac2xdf::ac2xdf_if_t::varnames [private]

5.10.4.6 nominal_sampling_rates

MHAParser::vfloat_t ac2xdf::ac2xdf_if_t::nominal_sampling_rates [private]

5.10.4.7 use_date

MHAParser::bool_t ac2xdf::ac2xdf_if_t::use_date [private]

5.10.4.8 patchbay

MHAEvents::patchbay_t< ac2xdf_if_t> ac2xdf::ac2xdf_if_t::patchbay [private]

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

- **ac2xdf.cpp**

5.11 ac2xdf::ac2xdf_rt_t Class Reference

Public Member Functions

- **ac2xdf_rt_t** (const std::string prefix, bool use_date, bool active, unsigned fifosize, unsigned minwrite, const std::vector< std::string > &varnames, const std::vector< **mha_real_t** > &sampling_rates, **algo_comm_t** &iac)
- void **process** ()
- void **exit_request** ()

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
- std::vector< std::unique_ptr< **acwriter_base_t** > > **vars**
- std::unique_ptr< **output_file_t** > **outfile**

5.11.1 Constructor & Destructor Documentation

5.11.1.1 ac2xdf_rt_t()

```
ac2xdf::ac2xdf_rt_t::ac2xdf_rt_t (
    const std::string prefix,
    bool use_date,
    bool active,
    unsigned fifosize,
    unsigned minwrite,
    const std::vector< std::string > & varnames,
    const std::vector< mha_real_t > & sampling_rates,
    algo_comm_t & iac ) [inline]
```

5.11.2 Member Function Documentation

5.11.2.1 process()

```
void ac2xdf::ac2xdf_rt_t::process (
    void ) [inline]
```

5.11.2.2 exit_request()

```
void ac2xdf::ac2xdf_rt_t::exit_request ( ) [inline]
```

5.11.3 Member Data Documentation

5.11.3.1 ac

```
MHA_AC::algo_comm_t& ac2xdf::ac2xdf_rt_t::ac [private]
```

5.11.3.2 vars `std::vector<std::unique_ptr< acwriter_base_t> > ac2xdf::ac2xdf_rt_↩ t::vars [private]`

5.11.3.3 outfile `std::unique_ptr< output_file_t> ac2xdf::ac2xdf_rt_t::outfile [private]`

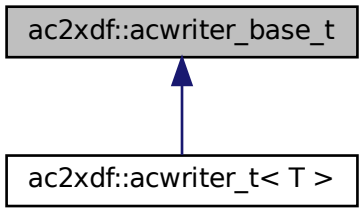
The documentation for this class was generated from the following file:

- **ac2xdf.cpp**

5.12 ac2xdf::acwriter_base_t Class Reference

Base class for all **acwriter_t** (p. 205)'s.

Inheritance diagram for ac2xdf::acwriter_base_t:



Public Member Functions

- virtual void **process** (**comm_var_t** &)=0
  Place the data present in the algorithm communication variable into the fifo for output to disk.
- virtual void **exit_request** ()=0
  Terminate output thread. Returns after exit thread has joined.
- virtual const char * **get_varname** () const =0
  getter for ac variable name
- virtual ~**acwriter_base_t** ()=default

5.12.1 Detailed Description

Base class for all **acwriter_t** (p. 205)'s.

This class decouples signal processing from writing to disk. There's one acwriter per AC variable to be written to disk. Each instance of acwriter spawns its own writer thread and has its own internal FIFO to safely move the samples of the AC variable out of the processing thread. All acwriters share an output file. It's not problematic when an acwriter has to wait for write access because the waiting does happen in its own thread, not in the audio thread.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 ~acwriter_base_t() `virtual ac2xdf::acwriter_base_t::~acwriter_base_t ( ) [virtual], [default]`

5.12.3 Member Function Documentation

5.12.3.1 process() `virtual void ac2xdf::acwriter_base_t::process ( comm_var_t & ) [pure virtual]`

Place the data present in the algorithm communication variable into the fifo for output to disk.

Implemented in **ac2xdf::acwriter_t< T >** (p. 208).

5.12.3.2 exit_request() `virtual void ac2xdf::acwriter_base_t::exit_request ( ) [pure virtual]`

Terminate output thread. Returns after exit thread has joined.

Implemented in **ac2xdf::acwriter_t< T >** (p. 208).

5.12.3.3 get_varname() `virtual const char* ac2xdf::acwriter_base_t::get_varname ( ) const [pure virtual]`

getter for ac variable name

Returns

name as char* as needed by get_var

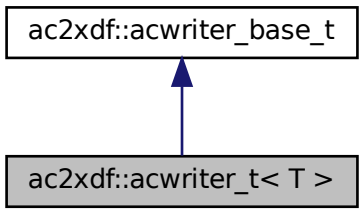
Implemented in **ac2xdf::acwriter_t< T >** (p. 208).

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

- **ac2xdf.hh**

5.13 ac2xdf::acwriter_t< T > Class Template Reference

Inheritance diagram for ac2xdf::acwriter_t< T >:



Public Member Functions

- **acwriter_t** (bool **active**, unsigned fifosize, unsigned minwrite, const std::string & **varname**, double **sampling_rate**, **output_file_t** * **outfile**, uint32_t **stream_id**)
  Constructor allocates fifo and disk output buffer.
- **acwriter_t** (const **acwriter_t** &)=delete
- **acwriter_t** (**acwriter_t** &&)=delete
- virtual ~**acwriter_t** ()=default
  Deallocates memory but does not terminate the write_thread.
- void **process** (**comm_var_t** &s) override
- void **exit_request** () override
  Terminate output thread.
- const char * **get_varname** () const override

Private Member Functions

- void **write_thread** ()

  Main method of the disk writer thread.

Private Attributes

- std::atomic< bool > **close_session**

  cross-thread-synchronization.
- const bool **active**

  The writer thread and the output file will only be created when active is true.
- std::unique_ptr< **mha_fifo_lf_t**< T > > **fifo**

  Fifo for decoupling signal processing thread from disk writer thread.
- const unsigned int **disk_write_threshold_min_num_samples**

  Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.
- std::thread **writethread**

  The thread that writes to disk.
- std::unique_ptr< T[ ]> **diskbuffer**

  Intermediate buffer to receive data from fifo and store on disk.
- **output_file_t** * **outfile**

  Ouput file.
- unsigned **num_channels** = 0U

  Number of channels of AC variable using stride.
- bool **is_num_channels_known** = false

  The number of channels is determined during the first process callback.
- bool **is_complex** = false

  If the AC variable is of complex valued type or not.
- unsigned int **data_type** = **MHA_AC_UNKNOWN**

  Data type of the ac variable. Used to protect against data type change during processing.
- const std::string **varname**

  The name of the ac variable to publish.
- const uint32_t **stream_id**
- bool **is_stream_initialized** =false
- double **sampling_rate**

5.13.1 Constructor & Destructor Documentation

5.13.1.1 acwriter_t() [1/3]

```
template<typename T >
ac2xdf::acwriter_t< T >::  acwriter_t (
            bool active,
            unsigned fifosize,
            unsigned minwrite,
            const std::string & varname,
            double sampling_rate,
             output_file_t * outfile,
            uint32_t stream_id )
```

Constructor allocates fifo and disk output buffer.

It spawns a new thread for writing data to disk when active==true. In order to terminate the thread, method exit_request **must** be called before this object is destroyed.

Parameters

active	Only write data to disk when this is true.
fifosize	Capacity of both the fifo pipeline and of the disk buffer.
minwrite	Wait for a fifo fill count of at least minwrite doubles before flushing the contents of the fifo to disk. Fifo is also flushed before this object is destroyed.
varname	Name of AC variable to save into file. Can be accessed through getter method **get_varname()** (p. 208). Stored here to avoid races between processing thread and configuration thread.
outfile	Handle to the output file
stream_id	Numerical id of the stream.

5.13.1.2 acwriter_t() [2/3]

```
template<typename T >
ac2xdf::acwriter_t< T >::  acwriter_t (
            const  acwriter_t< T > &  ) [delete]
```

5.13.1.3 acwriter_t() [3/3]

```
template<typename T >
ac2xdf::acwriter_t< T >::  acwriter_t (
             acwriter_t< T > &&  ) [delete]
```

5.13.1.4 ~acwriter_t() template<typename T >

virtual **ac2xdf::acwriter_t**< T >::~ **acwriter_t** ( ) [virtual], [default]

Deallocates memory but does not terminate the write_thread.

write_thread must be terminated before the destructor executes by calling exit_request.

5.13.2 Member Function Documentation

5.13.2.1 process() template<typename T >

void **ac2xdf::acwriter_t**< T >::process (
comm_var_t & *s*) [override], [virtual]

Implements **ac2xdf::acwriter_base_t** (p. 204).

5.13.2.2 exit_request() template<typename T >

void **ac2xdf::acwriter_t**< T >::exit_request [override], [virtual]

Terminate output thread.

Implements **ac2xdf::acwriter_base_t** (p. 204).

5.13.2.3 get_varname() template<typename T >

const char* **ac2xdf::acwriter_t**< T >::get_varname ( ) const [inline], [override], [virtual]

Implements **ac2xdf::acwriter_base_t** (p. 204).

5.13.2.4 write_thread() template<typename T >

void **ac2xdf::acwriter_t**< T >::write_thread [private]

Main method of the disk writer thread.

Periodically wakes up and checks if data needs to be written to disk.

5.13.3 Member Data Documentation

5.13.3.1 close_session `template<typename T >`

`std::atomic<bool>` **ac2xdf::acwriter_t**< T >::close_session [private]

cross-thread-synchronization.

write_thread() (p. 208) terminates after this is set to true by **exit_request()** (p. 208).

5.13.3.2 active `template<typename T >`

`const bool` **ac2xdf::acwriter_t**< T >::active [private]

The writer thread and the output file will only be created when active is true.

5.13.3.3 fifo `template<typename T >`

`std::unique_ptr<` **mha_fifo_lf_t**<T> > **ac2xdf::acwriter_t**< T >::fifo [private]

Fifo for decoupling signal processing thread from disk writer thread.

5.13.3.4 disk_write_threshold_min_num_samples `template<typename T >`

`const unsigned int` **ac2xdf::acwriter_t**< T >::disk_write_threshold_min_num_samples [private]

Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.

5.13.3.5 writethread `template<typename T >`

`std::thread` **ac2xdf::acwriter_t**< T >::writethread [private]

The thread that writes to disk.

5.13.3.6 diskbuffer template<typename T >

std::unique_ptr<T[]> **ac2xdf::acwriter_t**< T >::diskbuffer [private]

Intermediate buffer to receive data from fifo and store on disk.

5.13.3.7 outfile template<typename T >

output_file_t* **ac2xdf::acwriter_t**< T >::outfile [private]

Ouput file.

5.13.3.8 num_channels template<typename T >

unsigned **ac2xdf::acwriter_t**< T >::num_channels = 0U [private]

Number of channels of AC variable using stride.

If the number of channels changes during processing, an exception is thrown.

5.13.3.9 is_num_channels_known template<typename T >

bool **ac2xdf::acwriter_t**< T >::is_num_channels_known = false [private]

The number of channels is determined during the first process callback.

is_num_channels_known is set to true after the first process callback.

5.13.3.10 is_complex template<typename T >

bool **ac2xdf::acwriter_t**< T >::is_complex = false [private]

If the AC variable is of complex valued type or not.

If this changes during processing, then an exception is thrown.

5.13.3.11 data_type template<typename T >

unsigned int **ac2xdf::acwriter_t**< T >::data_type = **MHA_AC_UNKNOWN** [private]

Data type of the ac variable. Used to protect against data type change during processing.

5.13.3.12 varname template<typename T >

const std::string **ac2xdf::acwriter_t**< T >::varname [private]

The name of the ac variable to publish.

5.13.3.13 stream_id template<typename T >

const uint32_t **ac2xdf::acwriter_t**< T >::stream_id [private]

5.13.3.14 is_stream_initialized template<typename T >

bool **ac2xdf::acwriter_t**< T >::is_stream_initialized =false [private]

5.13.3.15 sampling_rate template<typename T >

double **ac2xdf::acwriter_t**< T >::sampling_rate [private]

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

- **ac2xdf.hh**
- **ac2xdf.cpp**

5.14 ac2xdf::output_file_t Class Reference

output_file_t (p. 211) represents one XDF output file.

Public Member Functions

- **output_file_t** (const std::string &prefix, bool use_date)
  Constructor.
- void **initialize_stream** (uint32_t stream_id, const std::string &varname, const std::string &channel_format, unsigned num_channels, double sampling_rate=0)
  Initialize stream.
- template<typename T = double>
  void **write** (uint32_t stream_id, const T *buf, std::size_t frames, std::size_t **channels**)
  Write data chunk to the stream with id stream_id.
- void **close_stream** (uint32_t stream_id)
  Close stream with id stream_id by writing stream footer.

Private Attributes

- std::mutex **write_lock**

  Mutex to protect write access to the output file.
- std::unique_ptr< XDFWriter > **outfile**

  XDFWriter. Handles the translation into the xdf format and disk writes.

5.14.1 Detailed Description

output_file_t (p. 211) represents one XDF output file.

It wraps around the XDFWriter class, which handles the conversion of a stream into the bits and bytes on disk. Access to the output file protected by a lock. There's usually one output file per plugin instance shared by all acwriters.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 output_file_t() `ac2xdf::output_file_t::output_file_t (`

```
const std::string & prefix,
bool use_date )
```

Constructor.

Parameters

prefix	Path and start of output file name. Will be extended with file name extension ".xdf".
use_date	When true, the current date and time will be appended to the output file name before the file name extension.

5.14.3 Member Function Documentation

5.14.3.1 initialize_stream()

```
void ac2xdf::output_file_t::initialize_stream (
        uint32_t stream_id,
        const std::string & varname,
        const std::string & channel_format,
        unsigned num_channels,
        double sampling_rate = 0 )
```

Initialize stream.

Writes a minimal stream header and a boundary chunk

Parameters

stream_id	Numerical stream id.
varname	Human-readable stream id. Gets saved as stream name in metadata
channel_format	Data type of the stream, gets written into the channel_format metadata. must be one of {"int8", "int16", "int32", "int64","float32" ,"double64" , "strin
num_channels	Number of channels in stream to be written in metadata
sampling_rate	Nominal sampling rate in Hz. To be written in metadata. Zero means irregular rate

5.14.3.2 write()

```
template<typename T >
template void ac2xdf::output_file_t::write<  mha_real_t > (
        uint32_t stream_id,
        const T * buf,
        std::size_t frames,
        std::size_t channels )
```

Write data chunk to the stream with id stream_id.

Parameters

stream_id	The stream id.
buf	Pointer to buffer containing frames entries. The caller retains ownership of buf.
Pointer	to buffer containing frames * **channels** (p. 37) values. Interleaved storage: The first **channels** (p. 37) values in memory contain the values of the first frame, etc. The caller retains ownership of buf.
frames	Number of entries per channel in buf.
channels	Number of channels in buf.

5.14.3.3 close_stream() `void ac2xdf::output_file_t::close_stream ( uint32_t stream_id )`

Close stream with id stream_id by writing stream footer.

Parameters

stream_id	Numeric ID of the stream to be closed

5.14.4 Member Data Documentation

5.14.4.1 write_lock `std::mutex ac2xdf::output_file_t::write_lock [private]`

Mutex to protect write access to the output file.

5.14.4.2 outfile `std::unique_ptr<XDFWriter> ac2xdf::output_file_t::outfile [private]`

XDFWriter. Handles the translation into the xdf format and disk writes.

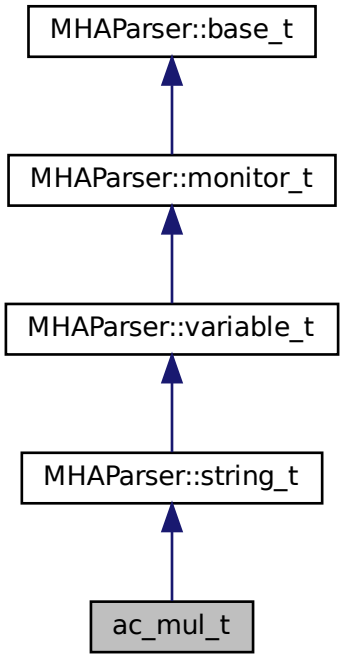
The documentation for this class was generated from the following files:

- **ac2xdf.hh**
- **ac2xdf.cpp**

5.15 ac_mul_t Class Reference

The class which implements the **ac_mul_t** (p. 215) plugin.

Inheritance diagram for ac_mul_t:



Public Member Functions

- **ac_mul_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)

  Plugin constructor.
- void **prepare_** (**mhaconfig_t** &)

  *Prepare method, called **prepare_()** (p. 217) with trailing underscore because **ac_mul_t** (p. 215) does not inherit from plugin_t<>.*
- void **release_** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **scan_syntax** ()
- void **get_arg_type_and_dimension** ()
- void **get_arg_type_and_dimension** (const std::string &, **val_type_t** &, unsigned int &, unsigned int &)
- void **process** ()
- void **process_rr** ()
- void **process_rc** ()
- void **process_cr** ()
- void **process_cc** ()

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
- std::string **algo**
- **arg_type_t** **argt**
- std::string **str_a**
- std::string **str_b**
- **MHA_AC::waveform_t** ∗ **res_r**
- **MHA_AC::spectrum_t** ∗ **res_c**
- unsigned int **num_frames**
- unsigned int **num_channels**

Additional Inherited Members

5.15.1 Detailed Description

The class which implements the **ac_mul_t** (p. 215) plugin.

Different from most other plugins, the ac_mul plugin's interface class does not inherit from plugin_t<>, but from **MHAParser::string_t** (p. 1260). This way, it does not get inserted into the MHA configuration tree as a parser node wich can have multiple variables, but as a string variable.

The **ac_mul_t** (p. 215) variable multiplies two AC variables element-wise.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ac_mul_t() ac_mul_t::ac_mul_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Plugin constructor.

5.15.3 Member Function Documentation

5.15.3.1 prepare_() void ac_mul_t::prepare_ (

```
mhaconfig_t & )
```

Prepare method, called **prepare_()** (p. 217) with trailing underscore because **ac_mul_t** (p. 215) does not inherit from plugin_t<>.

Leaves signal dimensions unchanged. The AC variables contained in the string expression must exist at this point.

5.15.3.2 release_() void ac_mul_t::release_ ( )

5.15.3.3 process() [1/3] mha_wave_t * ac_mul_t::process (

```
mha_wave_t * s )
```

5.15.3.4 process() [2/3] mha_spec_t * ac_mul_t::process (

```
mha_spec_t * s )
```

5.15.3.5 scan_syntax() void ac_mul_t::scan_syntax ( ) [private]

5.15.3.6 get_arg_type_and_dimension() [1/2] void ac_mul_t::get_arg_type_and_↩ dimension ( ) [private]

5.15.3.7 get_arg_type_and_dimension() [2/2] void ac_mul_t::get_arg_type_and_↩ dimension (

```
const std::string & name,
 val_type_t & vt,
unsigned int & num_frames,
unsigned int & num_channels ) [private]
```

5.15.3.8 process() [3/3] void ac_mul_t::process (

```
void ) [private]
```

5.15.3.9 process_rr() void ac_mul_t::process_rr ( ) [private]

5.15.3.10 process_rc() void ac_mul_t::process_rc ( ) [private]

5.15.3.11 process_cr() void ac_mul_t::process_cr ( ) [private]

5.15.3.12 process_cc() void ac_mul_t::process_cc ( ) [private]

5.15.4 Member Data Documentation

5.15.4.1 ac

```
MHA_AC::algo_comm_t& ac_mul_t::ac [private]
```

5.15.4.2 algo

```
std::string ac_mul_t::algo [private]
```

5.15.4.3 argt

```
arg_type_t ac_mul_t::argt [private]
```

5.15.4.4 str_a

```
std::string ac_mul_t::str_a [private]
```

5.15.4.5 str_b

```
std::string ac_mul_t::str_b [private]
```

5.15.4.6 res_r

```
MHA_AC::waveform_t* ac_mul_t::res_r [private]
```

5.15.4.7 res_c

```
MHA_AC::spectrum_t* ac_mul_t::res_c [private]
```

5.15.4.8 num_frames

```
unsigned int ac_mul_t::num_frames [private]
```

5.15.4.9 num_channels

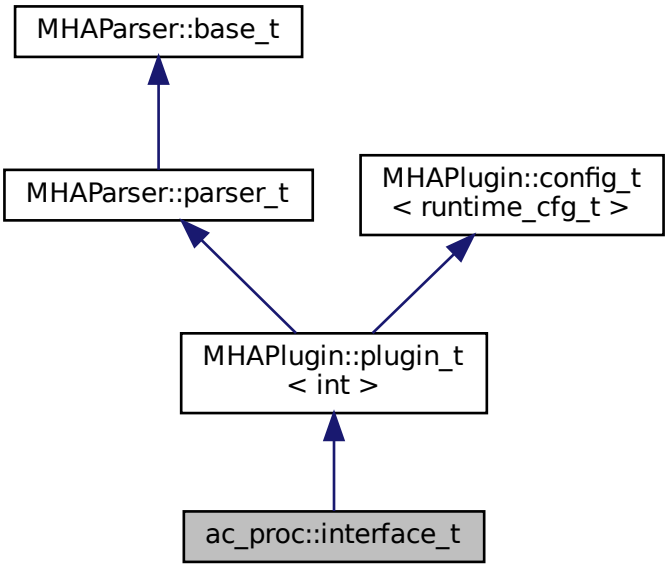
```
unsigned int ac_mul_t::num_channels [private]
```

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

- **ac_mul.hh**
- **ac_mul.cpp**

5.16 ac_proc::interface_t Class Reference

Inheritance diagram for ac_proc::interface_t:



Public Member Functions

- **interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- std::string **algo**
- **MHAParser::mhapluginloader_t plug**
- **MHAParser::string_t input**
- **MHAParser::bool_t permute**
- **MHA_AC::waveform_t** ∗ **s_out**
- **MHASignal::waveform_t** ∗ **s_in_perm**
- bool **b_permute**
- **mha_wave_t s_in**

Additional Inherited Members

5.16.1 Constructor & Destructor Documentation

5.16.1.1 interface_t() ac_proc::interface_t::interface_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

Default values are set and MHA configuration variables registered into the parser.

Parameters

iac	algorithm communication handle
configured_name	algorithm name

5.16.2 Member Function Documentation

5.16.2.1 prepare() void ac_proc::interface_t::prepare (mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.16.2.2 release() void ac_proc::interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.16.2.3 process() [1/3] void ac_proc::interface_t::process (void)

5.16.2.4 process() [2/3] mha_spec_t * ac_proc::interface_t::process (mha_spec_t * s)

5.16.2.5 process() [3/3] mha_wave_t * ac_proc::interface_t::process (mha_wave_t * s)

5.16.3 Member Data Documentation

5.16.3.1 algo std::string ac_proc::interface_t::algo [private]

5.16.3.2 plug MHAParser::mhapluginloader_t ac_proc::interface_t::plug [private]

5.16.3.3 input MHAParser::string_t ac_proc::interface_t::input [private]

5.16.3.4 permute MHAParser::bool_t ac_proc::interface_t::permute [private]

5.16.3.5 s_out MHA_AC::waveform_t* ac_proc::interface_t::s_out [private]

5.16.3.6 s_in_perm MHASignal::waveform_t* ac_proc::interface_t::s_in_perm [private]

5.16.3.7 b_permute `bool ac_proc::interface_t::b_permute [private]`

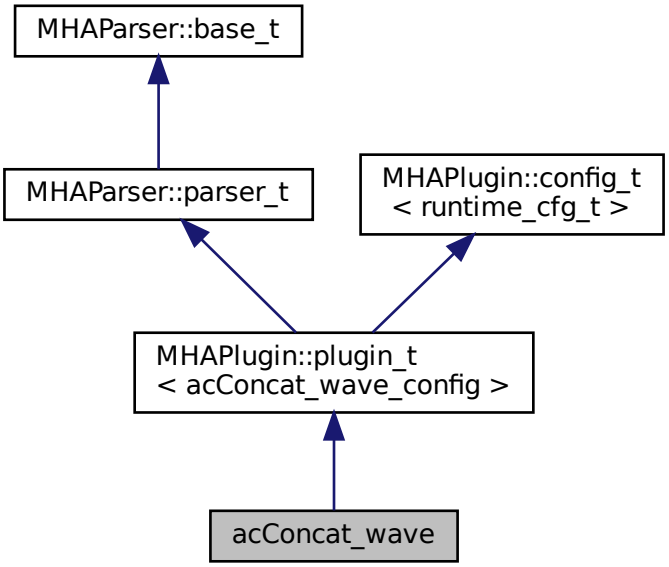
5.16.3.8 s_in `mha_wave_t ac_proc::interface_t::s_in [private]`

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

- **ac_proc.cpp**

5.17 acConcat_wave Class Reference

Inheritance diagram for acConcat_wave:



Public Member Functions

- **acConcat_wave** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.17.1 Constructor & Destructor Documentation

5.17.1.1 acConcat_wave() `acConcat_wave::acConcat_wave (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.17.1.2 ~acConcat_wave() `acConcat_wave::~acConcat_wave ( )`

5.17.2 Member Function Documentation

5.17.2.1 process() `mha_wave_t * acConcat_wave::process ( mha_wave_t * signal )`

Checks for the most recent configuration and defers processing to it.

5.17.2.2 prepare() `void acConcat_wave::prepare ( mhaconfig_t & ) [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. 1301).

5.17.2.3 release() `void acConcat_wave::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< acConcat_wave_config >** (p. 1302).

5.17.2.4 update_cfg() `void acConcat_wave::update_cfg ( ) [private]`

5.17.3 Member Data Documentation

5.17.3.1 num_AC `MHAParser::int_t acConcat_wave::num_AC`

5.17.3.2 prefix_names_AC **MHAParser::string_t** acConcat_wave::prefix_names_AC

5.17.3.3 samples_AC **MHAParser::vint_t** acConcat_wave::samples_AC

5.17.3.4 name_con_AC **MHAParser::string_t** acConcat_wave::name_con_AC

5.17.3.5 numchannels **MHAParser::int_t** acConcat_wave::numchannels

5.17.3.6 patchbay **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**

5.18 acConcat_wave_config Class Reference

Public Member Functions

- **acConcat_wave_config** (**MHA_AC::algo_comm_t** & **ac**, **acConcat_wave** *_concat)
- ~**acConcat_wave_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Public Attributes

- **MHA_AC::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**

5.18.1 Constructor & Destructor Documentation

5.18.1.1 acConcat_wave_config()

```
acConcat_wave_config::acConcat_wave_config (
    MHA_AC::algo_comm_t & ac,
    acConcat_wave * _concat )
```

5.18.1.2 ~acConcat_wave_config()

```
acConcat_wave_config::~acConcat_wave_config ( )
```

5.18.2 Member Function Documentation

5.18.2.1 process()

```
mha_wave_t * acConcat_wave_config::process (
    mha_wave_t * wave )
```

5.18.3 Member Data Documentation

5.18.3.1 ac

```
MHA_AC::algo_comm_t& acConcat_wave_config::ac
```

5.18.3.2 strNames_AC

```
std::vector<std::string> acConcat_wave_config::strNames_AC
```

5.18.3.3 numSamples_AC

```
std::vector<int> acConcat_wave_config::numSamples_AC
```

5.18.3.4 vGCC `mha_wave_t acConcat_wave_config::vGCC`

5.18.3.5 vGCC_con `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**

5.19 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_, **MHA_AC↩ ::algo_comm_t** &ac, bool use_matrix)
  Converts AC variable to parser monitor.
- void **getvar** (**MHA_AC::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::string_mon_t mon_string**
- **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

5.19.1 Detailed Description

A class for converting AC variables to Parser monitors of correct type.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 ac_monitor_t()

```
acmon::ac_monitor_t::ac_monitor_t (
    MHAParser::parser_t & parent,
    const std::string & name_,
    MHA_AC::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.

5.19.3 Member Function Documentation

5.19.3.1 getvar() `void acmon::ac_monitor_t::getvar (` **`MHA_AC::algo_comm_t`** `&` *`ac`* `)`

Update values of monitor.

Parameters

ac	Handle to algorithm communication space

5.19.4 Member Data Documentation

5.19.4.1 name `std::string acmon::ac_monitor_t::name`

name of AC variable and parser monitor

5.19.4.2 dimstr `std::string acmon::ac_monitor_t::dimstr`

columns x rows

5.19.4.3 mon **`MHAParser::vfloat_mon_t`** `acmon::ac_monitor_t::mon`

Monitor used for real vectors.

5.19.4.4 mon_mat **`MHAParser::mfloat_mon_t`** `acmon::ac_monitor_t::mon_mat`

Monitor used for real matrices.

5.19.4.5 mon_complex **MHAParser::vcomplex_mon_t** acmon::ac_monitor_t::mon_complex

monitor used for complex vectors

5.19.4.6 mon_mat_complex **MHAParser::mcomplex_mon_t** acmon::ac_monitor_t::mon_↩ mat_complex

monitor used for complex matrices

5.19.4.7 mon_string **MHAParser::string_mon_t** acmon::ac_monitor_t::mon_string

5.19.4.8 p_parser **MHAParser::parser_t&** acmon::ac_monitor_t::p_parser

parent parser to insert monitor into

5.19.4.9 use_mat bool acmon::ac_monitor_t::use_mat [private]

if true, use matrix monitor, else use vector monitor

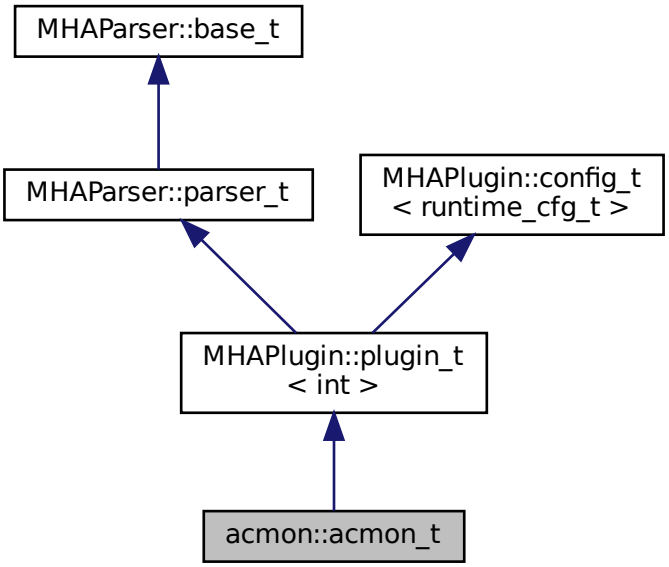
The documentation for this class was generated from the following files:

- **ac_monitor_type.hh**
- **ac_monitor_type.cpp**

5.20 acmon::acmon_t Class Reference

acmon plugin interface class

Inheritance diagram for acmon::acmon_t:



Public Member Functions

- **acmon_t** (**MHA_AC::algo_comm_t** &, const std::string &configured_name)
  Plugin interface constructor.
- ∼**acmon_t** ()=default
  Default destructor.
- void **prepare** (**mhaconfig_t** &)
  Prepare callback.
- void **release** ()
  Do-nothing release.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
  Process callback for frequency domain.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Process callback for time domain.

Private Member Functions

- void **save_vars** ()
  Save the current value of the AC variables into their corresponding monitors.
- void **update_recmode** ()
  Set/Re-Set the recording mode from continous to snapshot or vice versa.

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
  Handle to the AC space.
- **MHAParser::vstring_mon_t varlist**
  Monitor variable containing the AC variable names.
- **MHAParser::vstring_mon_t dimensions**
  Monitor variable containing the dimension strings of the AC variables.
- **MHAParser::kw_t dispmode**
  Configuration variable for the display mode.
- **MHAParser::kw_t recmode**
  Configuration variable for the record mode.
- std::vector< std::unique_ptr< **ac_monitor_t** > > **vars**
  Vector containing pointers to the ac to monitor bridge variables.
- **MHAEvents::patchbay_t**< **acmon_t** > **patchbay**
  Patchbay.
- std::string **algo**
  String saving the configured name of the plugin.
- bool **b_cont**
  Recording mode.
- bool **b_snapshot**
  Snapshot flag.

Additional Inherited Members

5.20.1 Detailed Description

acmon plugin interface class

5.20.2 Constructor & Destructor Documentation

5.20.2.1 acmon_t() acmon::acmon_t::acmon_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

Plugin interface constructor.

Parameters

configured_name	Assigned name of the plugin within the configuration tree

5.20.2.2 ~acmon_t() acmon::acmon_t::~acmon_t ( ) [default]

Default destructor.

5.20.3 Member Function Documentation

5.20.3.1 prepare() void acmon::acmon_t::prepare (mhaconfig_t &) [virtual]

Prepare callback.

Initializes the AC to monitor bridges and variable list, leaves the signal dimensions untouched

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.20.3.2 release() void acmon::acmon_t::release (void) [inline], [virtual]

Do-nothing release.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.20.3.3 process() [1/2] mha_spec_t * acmon::acmon_t::process (mha_spec_t * s)

Process callback for frequency domain.

Calls **save_vars()** (p. 235) and returns the signal unmodified.

5.20.3.4 process() [2/2] mha_wave_t * acmon::acmon_t::process (mha_wave_t * s)

Process callback for time domain.

Calls **save_vars()** (p. 235) and returns the signal unmodified.

5.20.3.5 save_vars() void acmon::acmon_t::save_vars ( ) [private]

Save the current value of the AC variables into their corresponding monitors.

5.20.3.6 update_recmode() void acmon::acmon_t::update_recmode ( ) [private]

Set/Re-Set the recording mode from continous to snapshot or vice versa.

5.20.4 Member Data Documentation

5.20.4.1 ac MHA_AC::algo_comm_t& acmon::acmon_t::ac [private]

Handle to the AC space.

5.20.4.2 varlist MHAParser::vstring_mon_t acmon::acmon_t::varlist [private]

Monitor variable containing the AC variable names.

5.20.4.3 dimensions `MHAParser::vstring_mon_t acmon::acmon_t::dimensions [private]`

Monitor variable containing the dimension strings of the AC variables.

5.20.4.4 dispmode `MHAParser::kw_t acmon::acmon_t::dispmode [private]`

Configuration variable for the display mode.

5.20.4.5 recmode `MHAParser::kw_t acmon::acmon_t::recmode [private]`

Configuration variable for the record mode.

5.20.4.6 vars `std::vector<std::unique_ptr< ac_monitor_t> > acmon::acmon_t::vars [private]`

Vector containing pointers to the ac to monitor bridge variables.

5.20.4.7 patchbay `MHAEvents::patchbay_t< acmon_t> acmon::acmon_t::patchbay [private]`

Patchbay.

5.20.4.8 algo `std::string acmon::acmon_t::algo [private]`

String saving the configured name of the plugin.

5.20.4.9 b_cont `bool acmon::acmon_t::b_cont [private]`

Recording mode.

True for continous mode, false for snapshot mode

5.20.4.10 b_snapshot `bool acmon::acmon_t::b_snapshot [private]`

Snapshot flag.

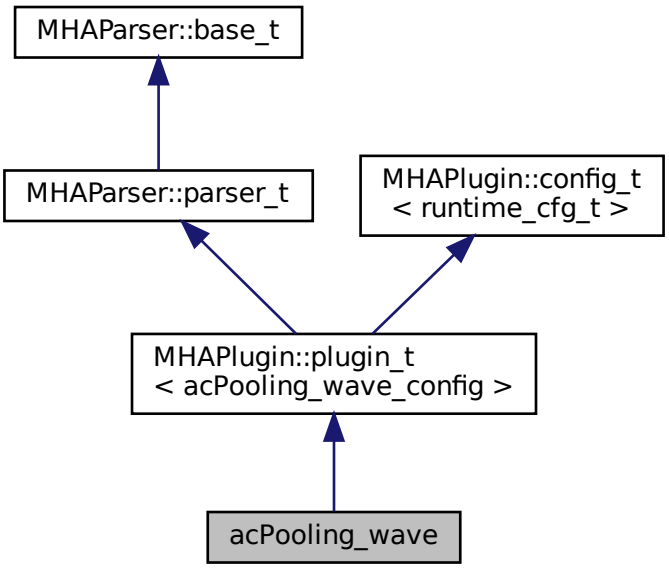
Set to true to request a snapshot in the next **process()** (p. 234) call

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

- **acmon.cpp**

5.21 acPooling_wave Class Reference

Inheritance diagram for acPooling_wave:



Public Member Functions

- **acPooling_wave** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.21.1 Constructor & Destructor Documentation

5.21.1.1 acPooling_wave() acPooling_wave::acPooling_wave (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.21.1.2 ~acPooling_wave() acPooling_wave::~acPooling_wave ( )

5.21.2 Member Function Documentation

5.21.2.1 process() `mha_wave_t * acPooling_wave::process ( mha_wave_t * signal )`

Checks for the most recent configuration and defers processing to it.

5.21.2.2 prepare() `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. 1301).

5.21.2.3 release() `void acPooling_wave::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< acPooling_wave_config >** (p. 1302).

5.21.2.4 update_cfg() `void acPooling_wave::update_cfg ( ) [private]`

5.21.3 Member Data Documentation

5.21.3.1 numsamples **MHAParser::int_t** acPooling_wave::numsamples

5.21.3.2 pooling_wndlen **MHAParser::int_t** acPooling_wave::pooling_wndlen

5.21.3.3 pooling_type **MHAParser::kw_t** acPooling_wave::pooling_type

5.21.3.4 upper_threshold **MHAParser::float_t** acPooling_wave::upper_threshold

5.21.3.5 lower_threshold **MHAParser::float_t** acPooling_wave::lower_threshold

5.21.3.6 neighbourhood **MHAParser::int_t** acPooling_wave::neighbourhood

5.21.3.7 alpha **MHAParser::float_t** acPooling_wave::alpha

5.21.3.8 p_name **MHAParser::string_t** acPooling_wave::p_name

5.21.3.9 p_biased_name **MHAParser::string_t** acPooling_wave::p_biased_name

5.21.3.10 pool_name **MHAParser::string_t** acPooling_wave::pool_name

5.21.3.11 max_pool_ind_name **MHAParser::string_t** acPooling_wave::max_pool_ind_↩ name

5.21.3.12 like_ratio_name **MHAParser::string_t** acPooling_wave::like_ratio_name

5.21.3.13 prob_bias **MHAParser::vfloat_t** acPooling_wave::prob_bias

5.21.3.14 patchbay **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**

5.22 acPooling_wave_config Class Reference

Public Member Functions

- **acPooling_wave_config** (**MHA_AC::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

- **MHA_AC::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**

5.22.1 Constructor & Destructor Documentation

5.22.1.1 acPooling_wave_config()

```
acPooling_wave_config::acPooling_wave_config (
    MHA_AC::algo_comm_t & ac,
    const mhaconfig_t in_cfg,
    acPooling_wave * _pooling )
```

5.22.1.2 ~acPooling_wave_config()

```
acPooling_wave_config::~acPooling_wave_config (
)
```

5.22.2 Member Function Documentation

5.22.2.1 process()

```
mha_wave_t * acPooling_wave_config::process (
    mha_wave_t * wave )
```

5.22.2.2 insert() void acPooling_wave_config::insert ( )

5.22.3 Member Data Documentation

5.22.3.1 ac MHA_AC::algo_comm_t& acPooling_wave_config::ac

5.22.3.2 raw_p_name std::string acPooling_wave_config::raw_p_name

5.22.3.3 p MHA_AC::waveform_t acPooling_wave_config::p

5.22.3.4 p_biased MHA_AC::waveform_t acPooling_wave_config::p_biased

5.22.3.5 p_max MHA_AC::waveform_t acPooling_wave_config::p_max

5.22.3.6 like_ratio MHA_AC::waveform_t acPooling_wave_config::like_ratio

5.22.3.7 c mha_wave_t acPooling_wave_config::c

5.22.3.8 pooling_ind unsigned int acPooling_wave_config::pooling_ind

5.22.3.9 pooling_option `unsigned int acPooling_wave_config::pooling_option`

5.22.3.10 pooling_size `unsigned int acPooling_wave_config::pooling_size`

5.22.3.11 up_thresh `float acPooling_wave_config::up_thresh`

5.22.3.12 low_thresh `float acPooling_wave_config::low_thresh`

5.22.3.13 neigh `int acPooling_wave_config::neigh`

5.22.3.14 alpha `float acPooling_wave_config::alpha`

5.22.3.15 pool **`MHASignal::waveform_t`** `acPooling_wave_config::pool`

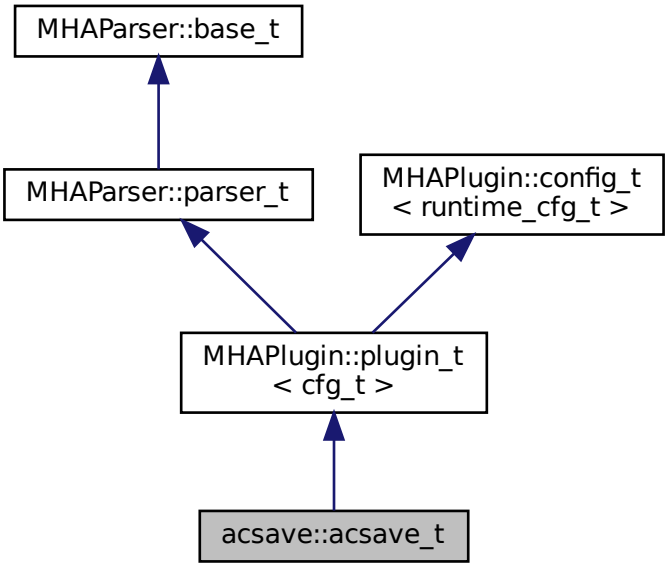
5.22.3.16 prob_bias_func **`MHASignal::waveform_t`** `acPooling_wave_config::prob_bias↩_func`

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

- **acPooling_wave.h**
- **acPooling_wave.cpp**

5.23 acsave::acsave_t Class Reference

Inheritance diagram for acsave::acsave_t:



Public Member Functions

- **acsave_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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 **algo**
- bool **b_prepared**
- bool **b_flushed**
- **MHAEvents::patchbay_t< acsave_t > patchbay**

Additional Inherited Members

5.23.1 Member Typedef Documentation

5.23.1.1 varlist_t `typedef std::vector< save_var_t* > acsave::acsave_t::varlist_t [private]`

5.23.2 Constructor & Destructor Documentation

5.23.2.1 acsave_t() `acsave::acsave_t::acsave_t ( MHA_AC::algo_comm_t & iac, const std::string & configured_name )`

5.23.3 Member Function Documentation

5.23.3.1 prepare() `void acsave::acsave_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.23.3.2 release() void acsave::acsave_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1302).

5.23.3.3 process() [1/3] mha_spec_t * acsave::acsave_t::process (mha_spec_t * *s*)

5.23.3.4 process() [2/3] mha_wave_t * acsave::acsave_t::process (mha_wave_t * *s*)

5.23.3.5 event_start_recording() void acsave::acsave_t::event_start_recording ( )

5.23.3.6 event_stop_and_flush() void acsave::acsave_t::event_stop_and_flush ( )

5.23.3.7 process() [3/3] void acsave::acsave_t::process (void) [private]

5.23.4 Member Data Documentation

5.23.4.1 bflush MHAParser::bool_t acsave::acsave_t::bflush [private]

5.23.4.2 fileformat MHAParser::kw_t acsave::acsave_t::fileformat [private]

5.23.4.3 fname

`MHAParser::string_t acsave::acsave_t::fname [private]`

5.23.4.4 reclen

`MHAParser::float_t acsave::acsave_t::reclen [private]`

5.23.4.5 variables

`MHAParser::vstring_t acsave::acsave_t::variables [private]`

5.23.4.6 varlist

`varlist_t acsave::acsave_t::varlist [private]`

5.23.4.7 algo

`std::string acsave::acsave_t::algo [private]`

5.23.4.8 b_prepared

`bool acsave::acsave_t::b_prepared [private]`

5.23.4.9 b_flushed

`bool acsave::acsave_t::b_flushed [private]`

5.23.4.10 patchbay

`MHAEvents::patchbay_t< acsave_t> acsave::acsave_t::patchbay [private]`

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

- **acsave.cpp**

5.24 acsave::cfg_t Class Reference

Public Member Functions

- **cfg_t** (**MHA_AC::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

- **MHA_AC::algo_comm_t** & **ac**
- unsigned int **nvars**
- **save_var_t** ** **varlist**
- unsigned int **rec_frames**
- unsigned int **max_frames**

5.24.1 Constructor & Destructor Documentation

5.24.1.1 cfg_t() `cfg_t::cfg_t (`

```
    MHA_AC::algo_comm_t & iac,
    unsigned int imax_frames,
    std::vector< std::string > & var_names )
```

5.24.1.2 ~cfg_t() `cfg_t::~cfg_t ( )`

5.24.2 Member Function Documentation

5.24.2.1 store_frame() `void cfg_t::store_frame ( )`

This function is called in the processing thread.

5.24.2.2 flush_data() `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

5.24.3 Member Data Documentation

5.24.3.1 ac `MHA_AC::algo_comm_t& acsave::cfg_t::ac [private]`

5.24.3.2 nvars `unsigned int acsave::cfg_t::nvars [private]`

5.24.3.3 varlist `save_var_t** acsave::cfg_t::varlist [private]`

5.24.3.4 rec_frames `unsigned int acsave::cfg_t::rec_frames [private]`

5.24.3.5 max_frames `unsigned int acsave::cfg_t::max_frames [private]`

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

- **acsave.cpp**

5.25 acsave::mat4head_t Struct Reference

Public Attributes

- int32_t **t**
- int32_t **rows**
- int32_t **cols**
- int32_t **imag**
- int32_t **namelen**

5.25.1 Member Data Documentation

5.25.1.1 t `int32_t acsave::mat4head_t::t`

5.25.1.2 rows `int32_t acsave::mat4head_t::rows`

5.25.1.3 cols `int32_t acsave::mat4head_t::cols`

5.25.1.4 imag `int32_t acsave::mat4head_t::imag`

5.25.1.5 namelen `int32_t acsave::mat4head_t::namelen`

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

- **acsave.cpp**

5.26 acsave::save_var_t Class Reference

Public Member Functions

- **save_var_t** (const std::string &, int, **MHA_AC::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**
- **MHA_AC::algo_comm_t** & **ac**
- unsigned int **framecnt**
- bool **b_complex**

5.26.1 Constructor & Destructor Documentation

5.26.1.1 save_var_t() `acsave::save_var_t::save_var_t (`

```
const std::string & nm,
int n,
 MHA_AC::algo_comm_t & iac )
```

5.26.1.2 ~save_var_t() `acsave::save_var_t::~save_var_t ( )`

5.26.2 Member Function Documentation

5.26.2.1 store_frame() void acsave::save_var_t::store_frame ( )

5.26.2.2 save_txt() void acsave::save_var_t::save_txt (

```
FILE * fh,
unsigned int writeframes )
```

5.26.2.3 save_mat4() void acsave::save_var_t::save_mat4 (

```
FILE * fh,
unsigned int writeframes )
```

5.26.2.4 save_m() void acsave::save_var_t::save_m (

```
FILE * fh,
unsigned int writeframes )
```

5.26.3 Member Data Documentation

5.26.3.1 data double* acsave::save_var_t::data

5.26.3.2 name std::string acsave::save_var_t::name [private]

5.26.3.3 nframes `unsigned int acsave::save_var_t::nframes [private]`

5.26.3.4 ndim `unsigned int acsave::save_var_t::ndim [private]`

5.26.3.5 maxframe `unsigned int acsave::save_var_t::maxframe [private]`

5.26.3.6 ac `MHA_AC::algo_comm_t& acsave::save_var_t::ac [private]`

5.26.3.7 framecnt `unsigned int acsave::save_var_t::framecnt [private]`

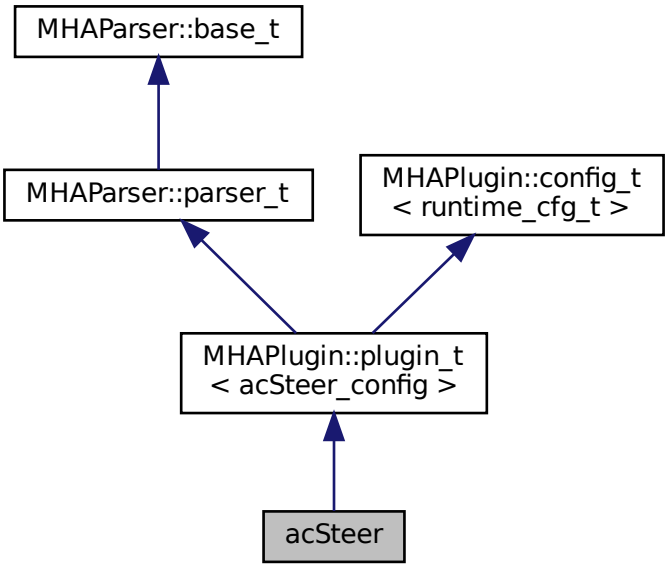
5.26.3.8 b_complex `bool acsave::save_var_t::b_complex [private]`

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

- **acsave.cpp**

5.27 acSteer Class Reference

Inheritance diagram for acSteer:



Public Member Functions

- **acSteer** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.27.1 Constructor & Destructor Documentation

5.27.1.1 acSteer() `acSteer::acSteer (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.27.1.2 ~acSteer() `acSteer::~acSteer ( )`

5.27.2 Member Function Documentation

5.27.2.1 process() `mha_spec_t * acSteer::process (`

```
mha_spec_t * signal )
```

Thos method is a NOOP.

5.27.2.2 prepare() 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. 1301).

5.27.2.3 release() void acSteer::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< acSteer_config >** (p. 1302).

5.27.2.4 update_cfg() void acSteer::update_cfg ( ) [private]

5.27.3 Member Data Documentation

5.27.3.1 steerFile MHAParser::string_t acSteer::steerFile

5.27.3.2 acSteerName1 MHAParser::string_t acSteer::acSteerName1

5.27.3.3 acSteerName2 MHAParser::string_t acSteer::acSteerName2

5.27.3.4 nsteerchan `MHAParser::int_t acSteer::nsteerchan`

5.27.3.5 nrefmic `MHAParser::int_t acSteer::nrefmic`

5.27.3.6 patchbay `MHAEvents::patchbay_t< acSteer> acSteer::patchbay [private]`

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

- **acSteer.h**
- **acSteer.cpp**

5.28 acSteer_config Class Reference

Public Member Functions

- **acSteer_config** (**MHA_AC::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**

5.28.1 Constructor & Destructor Documentation

5.28.1.1 acSteer_config()

```
acSteer_config::acSteer_config (
    MHA_AC::algo_comm_t & ac,
    const  mhaconfig_t in_cfg,
    acSteer * acSteer )
```

5.28.1.2 ~acSteer_config()

```
acSteer_config::~acSteer_config ( )
```

5.28.2 Member Function Documentation

5.28.2.1 insert()

```
void acSteer_config::insert ( )
```

5.28.3 Member Data Documentation

5.28.3.1 nchan

```
unsigned int acSteer_config::nchan
```

5.28.3.2 nfreq

```
unsigned int acSteer_config::nfreq
```

5.28.3.3 nsteerchan

```
unsigned int acSteer_config::nsteerchan
```

5.28.3.4 nrefmic

```
unsigned int acSteer_config::nrefmic
```

5.28.3.5 nangle `unsigned int acSteer_config::nangle`

5.28.3.6 specSteer1 `MHA_AC::spectrum_t acSteer_config::specSteer1`

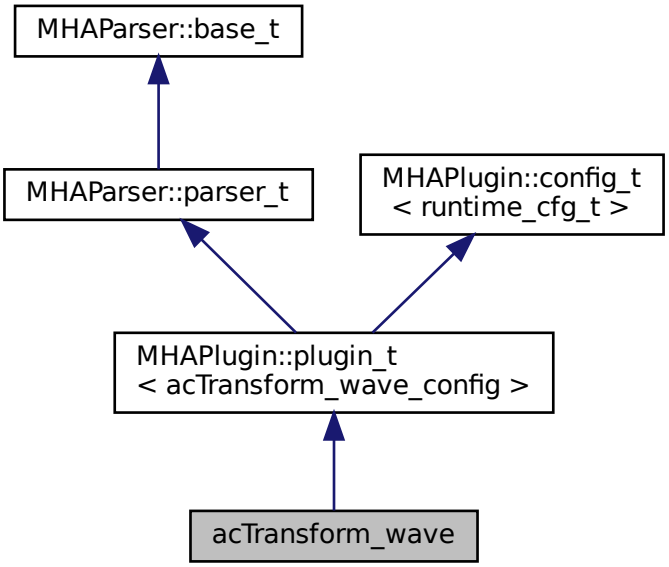
5.28.3.7 specSteer2 `MHA_AC::spectrum_t acSteer_config::specSteer2`

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

- **acSteer.h**
- **acSteer.cpp**

5.29 acTransform_wave Class Reference

Inheritance diagram for acTransform_wave:



Public Member Functions

- **acTransform_wave** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ 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

5.29.1 Constructor & Destructor Documentation

5.29.1.1 acTransform_wave() `acTransform_wave::acTransform_wave ( MHA_AC::algo_comm_t & iac, const std::string & configured_name )`

Constructs our plugin.

5.29.1.2 ~acTransform_wave() acTransform_wave::~acTransform_wave ( )

5.29.2 Member Function Documentation

5.29.2.1 process() mha_wave_t * acTransform_wave::process (mha_wave_t * signal)

Checks for the most recent configuration and defers processing to it.

5.29.2.2 prepare() 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. 1301).

5.29.2.3 release() void acTransform_wave::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< acTransform_wave_config >** (p. 1302).

5.29.2.4 update_cfg() void acTransform_wave::update_cfg ( ) [private]

5.29.3 Member Data Documentation

5.29.3.1 ang_name `MHAParser::string_t acTransform_wave::ang_name`

5.29.3.2 raw_p_name `MHAParser::string_t acTransform_wave::raw_p_name`

5.29.3.3 raw_p_max_name `MHAParser::string_t acTransform_wave::raw_p_max_name`

5.29.3.4 rotated_p_name `MHAParser::string_t acTransform_wave::rotated_p_name`

5.29.3.5 rotated_p_max_name `MHAParser::string_t acTransform_wave::rotated_p_max_name`

5.29.3.6 numsamples `MHAParser::int_t acTransform_wave::numsamples`

5.29.3.7 to_from `MHAParser::bool_t acTransform_wave::to_from`

5.29.3.8 patchbay `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**

5.30 acTransform_wave_config Class Reference

Public Member Functions

- **acTransform_wave_config** (**MHA_AC::algo_comm_t** & **ac**, **acTransform_wave** *↩ _transform)
- ~**acTransform_wave_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **insert_ac_variables** ()

  Insert or reinsert AC variables rotated_p, rotated_i.

Public Attributes

- **MHA_AC::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**

5.30.1 Constructor & Destructor Documentation

5.30.1.1 acTransform_wave_config() `acTransform_wave_config::acTransform_wave_↩ config (`

```
            MHA_AC::algo_comm_t & ac,
            acTransform_wave * _transform )
```

5.30.1.2 ~acTransform_wave_config() `acTransform_wave_config::~acTransform_wave↩ _config ( )`

5.30.2 Member Function Documentation

5.30.2.1 process()

```
mha_wave_t * acTransform_wave_config::process (
        mha_wave_t * wave )
```

5.30.2.2 insert_ac_variables()

```
void acTransform_wave_config::insert_ac_variables ( )
```

Insert or reinsert AC variables rotated_p, rotated_i.

5.30.3 Member Data Documentation

5.30.3.1 ac

```
MHA_AC::algo_comm_t& acTransform_wave_config::ac
```

5.30.3.2 ang_name

```
std::string acTransform_wave_config::ang_name
```

5.30.3.3 raw_p_name

```
std::string acTransform_wave_config::raw_p_name
```

5.30.3.4 raw_p_max_name

```
std::string acTransform_wave_config::raw_p_max_name
```

5.30.3.5 rotated_p

```
MHA_AC::waveform_t acTransform_wave_config::rotated_p
```

5.30.3.6 rotated_i

```
MHA_AC::int_t acTransform_wave_config::rotated_i
```

5.30.3.7 offset `unsigned int acTransform_wave_config::offset`

5.30.3.8 resolution `unsigned int acTransform_wave_config::resolution`

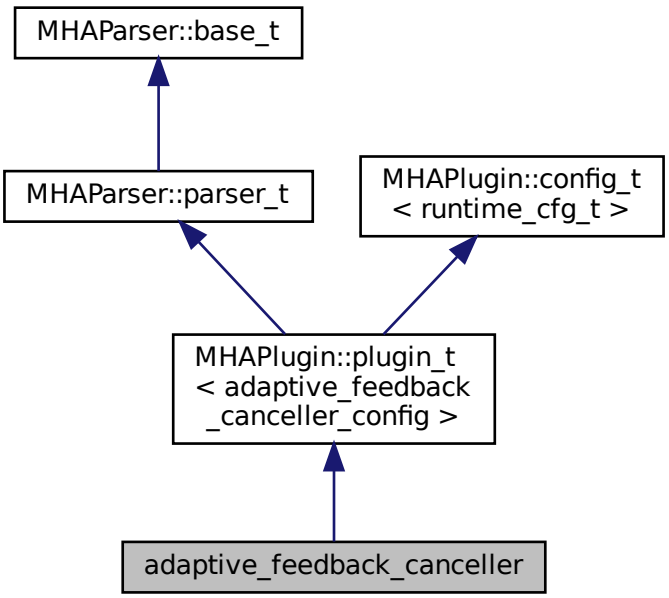
5.30.3.9 to_from `unsigned int acTransform_wave_config::to_from`

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

5.31 adaptive_feedback_canceller Class Reference

Inheritance diagram for adaptive_feedback_canceller:



Public Member Functions

- **adaptive_feedback_canceller** (**MHA_AC::algo_comm_t** & **ac**, const std::string &configured_name)

  Constructs our plugin.
- ∼**adaptive_feedback_canceller** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

  Plugin preparation.
- void **release** ()

Public Attributes

- **MHAParser::float_t stepsize**
- **MHAParser::float_t min_const**
- **MHAParser::int_t filter_length**
- **MHAParser::mhapluginloader_t plugloader**

  Plugin loader that loads the plugin needed to simulate the hearing aid in the forward processing path.
- **MHAParser::int_t fragsize**

  Fragsize for computing adaptive_feedback_canceller::delay_roundtrip and adaptive_↩ feedback_canceller::delay_update, defaults to the MHA's fragsize.
- **MHAParser::vint_t measured_roundtrip_latency**

  The roundtrip latency describes the timespan between playing back and receiving the same signal, including delays induced by soundcard buffering and the transducers.
- **MHAParser::bool_t use_lpc_decorr**

  Boolean defining whether decorrelation using LPC-filtering of microphone and loudspeaker signal should be performed.
- **MHAParser::int_t lpc_order**
- **MHAParser::vint_t delay_forward_path**
- **MHAParser::int_t blocks_no_update**
- **MHAParser::bool_t debug_mode**

  debug_mode *(p. 270) == true provides more variables in the AC-space that are useful for debugging, including FBfilter_estim_ac, ERRsig_ac, current_power_ac, estim_err_ac*

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **adaptive_feedback_canceller** > **patchbay**

Additional Inherited Members

5.31.1 Constructor & Destructor Documentation

5.31.1.1 adaptive_feedback_canceller() `adaptive_feedback_canceller::adaptive_feedback_canceller (`

```
MHA_AC::algo_comm_t & ac,
const std::string & configured_name )
```

Constructs our plugin.

5.31.1.2 ~adaptive_feedback_canceller() `adaptive_feedback_canceller::~adaptive_feedback_canceller ( )`

5.31.2 Member Function Documentation

5.31.2.1 process() `mha_wave_t * adaptive_feedback_canceller::process (`

```
mha_wave_t * signal )
```

5.31.2.2 prepare() `void adaptive_feedback_canceller::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< adaptive_feedback_canceller_config >** (p. 1301).

5.31.2.3 release() void adaptive_feedback_canceller::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< adaptive_feedback_canceller_config >** (p. 1302).

5.31.2.4 update_cfg() void adaptive_feedback_canceller::update_cfg ( ) [private]

5.31.3 Member Data Documentation

5.31.3.1 stepsize MHAParser::float_t adaptive_feedback_canceller::stepsize

5.31.3.2 min_const MHAParser::float_t adaptive_feedback_canceller::min_const

5.31.3.3 filter_length MHAParser::int_t adaptive_feedback_canceller::filter_length

5.31.3.4 plugloader MHAParser::mhapluginloader_t adaptive_feedback_canceller::plugloader

Plugin loader that loads the plugin needed to simulate the hearing aid in the forward processing path.

5.31.3.5 fragsize MHAParser::int_t adaptive_feedback_canceller::fragsize

Fragsize for computing adaptive_feedback_canceller::delay_roundtrip and adaptive_feedback_canceller::delay_update, defaults to the MHA's fragsize.

It is assumed that the soundcard's and the MHA's fragsize are equal. In special cases, the user can set this variable to an individual value.

5.31.3.6 measured_roundtrip_latency **MHAParser::vint_t** adaptive_feedback_canceller↩ ::measured_roundtrip_latency

The roundtrip latency describes the timespan between playing back and receiving the same signal, including delays induced by soundcard buffering and the transducers.

Can be measured via jack_iodelay. It is used to compute delay_roundtrip and delay_update for computations in the backward path.

5.31.3.7 use_lpc_decorr **MHAParser::bool_t** adaptive_feedback_canceller::use_lpc_↩ decorr

Boolean defining whether decorrelation using LPC-filtering of microphone and loudspeaker signal should be performed.

NOTE: The algorithm did not yet implement the decorrelation via LPC coefficients. If you set this variable to 'true' in the current state, the plugin will not work.

5.31.3.8 lpc_order **MHAParser::int_t** adaptive_feedback_canceller::lpc_order

5.31.3.9 delay_forward_path **MHAParser::vint_t** adaptive_feedback_canceller::delay↩ _forward_path

5.31.3.10 blocks_no_update **MHAParser::int_t** adaptive_feedback_canceller::blocks↩ _no_update

5.31.3.11 debug_mode **MHAParser::bool_t** adaptive_feedback_canceller::debug_mode

debug_mode (p. 270) == true provides more variables in the AC-space that are useful for debugging, including FBfilter_estim_ac, ERRsig_ac, current_power_ac, estim_err_ac

5.31.3.12 patchbay `MHAEvents::patchbay_t< adaptive_feedback_canceller> adaptive_feedback_canceller::patchbay [private]`

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

- **adaptive_feedback_canceller.h**
- **adaptive_feedback_canceller.cpp**

5.32 adaptive_feedback_canceller_config Class Reference

This is the runtime configuration, the main processing will be done in this class.

Public Member Functions

- **adaptive_feedback_canceller_config** (**algo_comm_t** &ac, const **mhaconfig_t** in_cfg, **adaptive_feedback_canceller** *afc)
- ~**adaptive_feedback_canceller_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *MICsig)

  *The **process()** (p. 273) method contains the actual AFC algorithm, the output is stored in LSsig_output.*
- void **insert** ()

  Insert all AC-variables into the AC-space.

Private Attributes

- const unsigned int **ntaps**

  Length of the estimated filter.
- const unsigned int **frames**

  Length of a block in samples (fragsize)
- const unsigned int **channels**

  Number of channels.
- const int **n_no_update_**

  Number of blocks after startup where the feedback filter is not updated.
- int **no_update_count**

  *Index counting no-update-blocks up to n_no_update_ to check in **process()** (p. 273) whether a filter update shall be performed or not.*
- const **mha_real_t** **fragsize**

  *Fragsize for computing **delay_roundtrip** (p. 276) and **delay_update** (p. 276), defaults to the MHA's fragsize.*
- const **mha_real_t** **stepsize**

  Normalized stepsize of the NLMS-Algorithm.
- const **mha_real_t** **min_const**

Minimum constant to prevent division by zero in the NLMS-Algorithm.

- **MHASignal::waveform_t forward_sig**

  Copy of error signal that is channeled into the forward path processing.
- **MHASignal::waveform_t LSsig_initializer**

  Signal consisting of zeros, it is only used to initialize the loudspeaker signal (LSsig) for the first iteration (before forward_path_proc.process() is called for the first time).
- **mha_wave_t** * **LSsig**

  Pointer to the loudspeaker (LS) signal.
- **MHASignal::waveform_t LSsig_output**
- **MHASignal::delay_t delay_forward_path**

  Delay line for additional decorrelation in the forward path.
- **MHAParser::mhapluginloader_t** & **forward_path_proc**

  Pluginloader to load the plugins that represent the hearing aid processing in the forward path.
- **MHASignal::delay_t delay_roundtrip**

  *Delay line equal to the measured roundtrip latency - **fragsize** (p. 274).*
- **MHASignal::delay_t delay_update**
- std::vector< **MHAFilter::filter_t** > **FBfilter_estim**
- **MHA_AC::waveform_t FBfilter_estim_ac**
- **MHASignal::waveform_t FBsig_estim**
- **MHASignal::waveform_t ERRsig**
- **MHA_AC::waveform_t ERRsig_ac**
- bool **use_lpc_decorr**
- std::vector< **MHAFilter::filter_t** > **lpc_filter**
- **MHASignal::waveform_t white_LSsig**

  Loudspeaker signal, whitened by filtering with LPC coefficients.
- **MHASignal::waveform_t white_LSsig_smpl**

  Single sample (for each channel) of white_LSsig that will be written to rb_white_LSsig next.
- **MHASignal::ringbuffer_t rb_white_LSsig**

  *Ringbuffer containing the values used to determine the power of **white_LSsig** (p. 277) in each channel (see **channels** (p. 274)) and update **FBfilter_estim** (p. 276).*
- std::vector< **mha_real_t** > **current_power**
- **MHASignal::waveform_t white_MICsig**
- **MHASignal::waveform_t white_FBsig_estim**
- **MHASignal::waveform_t white_ERRsig**
- bool **debug_mode**

  When executing the code for debug purposes this flag can be set to true in order to capture variable states and provide them as AC variables to the user.
- **MHA_AC::waveform_t current_power_ac**
- **MHA_AC::waveform_t estim_err_ac**

  *AC-variable publishing the state of 'estim_err' if **debug_mode** (p. 278) == true 'estim_err' is a variable local to this plugin's **process()** (p. 273) method.*

5.32.1 Detailed Description

This is the runtime configuration, the main processing will be done in this class.

During runtime AC variables are published by this class, mainly for debbugging purposes.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 adaptive_feedback_canceller_config() adaptive_feedback_canceller_config↩ ::adaptive_feedback_canceller_config (

```
algo_comm_t & ac,
const mhaconfig_t in_cfg,
adaptive_feedback_canceller * afc )
```

Parameters

ac	Algorithm communication variable space
in_cfg	MHA signal dimensions for this plugin
adaptive_feedback_canceller *(p. 266)*	Pointer to plugin interface instance, used to ext configuration values.

5.32.2.2 ∼adaptive_feedback_canceller_config() adaptive_feedback_canceller_↩ config::∼adaptive_feedback_canceller_config ( )

5.32.3 Member Function Documentation

5.32.3.1 process() mha_wave_t * adaptive_feedback_canceller_config::process (

```
mha_wave_t * MICsig )
```

The **process()** (p. 273) method contains the actual AFC algorithm, the output is stored in LSsig_output.

Parameters

MICsig	The microphone signal which contains the target signal + the feedback signal. The input signal is not altered in place.

Returns

A pointer to LSsig_output which contains the loudspeaker signal that is actually channeled to the output.

5.32.3.2 insert() `void adaptive_feedback_canceller_config::insert ( )`

Insert all AC-variables into the AC-space.

5.32.4 Member Data Documentation

5.32.4.1 ntaps `const unsigned int adaptive_feedback_canceller_config::ntaps [private]`

Length of the estimated filter.

5.32.4.2 frames `const unsigned int adaptive_feedback_canceller_config::frames [private]`

Length of a block in samples (fragsize)

5.32.4.3 channels `const unsigned int adaptive_feedback_canceller_config::channels [private]`

Number of channels.

This plugin assumes that the number of input channels is equal to the number of output channels. Each input is paired with an output and one pair is called a channel. This is necessary because the AFC needs input and output information to work.

5.32.4.4 n_no_update_ `const int adaptive_feedback_canceller_config::n_no_update_ [private]`

Number of blocks after startup where the feedback filter is not updated.

5.32.4.5 no_update_count `int adaptive_feedback_canceller_config::no_update_count [private]`

Index counting no-update-blocks up to n_no_update_ to check in **process()** (p. 273) whether a filter update shall be performed or not.

5.32.4.6 fragsize

`const` **`mha_real_t`** `adaptive_feedback_canceller_config::fragsize [private]`

Fragsize for computing **delay_roundtrip** (p. 276) and **delay_update** (p. 276), defaults to the MHA's fragsize.

It is assumed that the soundcard's and the MHA's fragsize are equal. In special cases, the user can set this variable to an individual value.

5.32.4.7 stepsize

`const` **`mha_real_t`** `adaptive_feedback_canceller_config::stepsize [private]`

Normalized stepsize of the NLMS-Algorithm.

5.32.4.8 min_const

`const` **`mha_real_t`** `adaptive_feedback_canceller_config::min_const [private]`

Minimum constant to prevent division by zero in the NLMS-Algorithm.

5.32.4.9 forward_sig

`MHASignal::waveform_t` `adaptive_feedback_canceller_config↩ ::forward_sig [private]`

Copy of error signal that is channeled into the forward path processing.

5.32.4.10 LSsig_initializer

`MHASignal::waveform_t` `adaptive_feedback_canceller_↩ config::LSsig_initializer [private]`

Signal consisting of zeros, it is only used to initialize the loudspeaker signal (LSsig) for the first iteration (before forward_path_proc.process() is called for the first time).

5.32.4.11 LSsig

`mha_wave_t``*` `adaptive_feedback_canceller_config::LSsig [private]`

Pointer to the loudspeaker (LS) signal.

The destination of this pointer will be altered by the plugin loaded by the pluginloader in the forward path processing.

5.32.4.12 LSsig_output

`MHASignal::waveform_t adaptive_feedback_canceller_config::LSsig_output [private]`

5.32.4.13 delay_forward_path

`MHASignal::delay_t adaptive_feedback_canceller_config::delay_forward_path [private]`

Delay line for additional decorrelation in the forward path.

5.32.4.14 forward_path_proc

`MHAParser::mhapluginloader_t& adaptive_feedback_canceller_config::forward_path_proc [private]`

Pluginloader to load the plugins that represent the hearing aid processing in the forward path.

5.32.4.15 delay_roundtrip

`MHASignal::delay_t adaptive_feedback_canceller_config::delay_roundtrip [private]`

Delay line equal to the measured roundtrip latency - **fragsize** (p. 274).

The roundtrip latency describes the timespan between playing back and receiving the same signal, including delays induced by soundcard buffering and the transducers. Can be measured via jack_iodelay. It is used on the loudspeaker signal LSsig before the filtering with FBfilter_estim, to achieve about the same temporal alignment to MICsig as the alignment of the target signal and the true feedback signal.

5.32.4.16 delay_update

`MHASignal::delay_t adaptive_feedback_canceller_config::delay_update [private]`

5.32.4.17 FBfilter_estim

`std::vector< MHAFilter::filter_t> adaptive_feedback_canceller_config::FBfilter_estim [private]`

5.32.4.18 FBfilter_estim_ac

MHA_AC::waveform_t adaptive_feedback_canceller_config↩
::FBfilter_estim_ac [private]

5.32.4.19 FBsig_estim

MHASignal::waveform_t adaptive_feedback_canceller_config↩
::FBsig_estim [private]

5.32.4.20 ERRsig

MHASignal::waveform_t adaptive_feedback_canceller_config::↩
ERRsig [private]

5.32.4.21 ERRsig_ac

MHA_AC::waveform_t adaptive_feedback_canceller_config::↩
ERRsig_ac [private]

5.32.4.22 use_lpc_decorr

bool adaptive_feedback_canceller_config::use_lpc_decorr
[private]

5.32.4.23 lpc_filter

std::vector< **MHAFilter::filter_t**> adaptive_feedback_canceller↩
_config::lpc_filter [private]

5.32.4.24 white_LSsig

MHASignal::waveform_t adaptive_feedback_canceller_config↩
::white_LSsig [private]

Loudspeaker signal, whitened by filtering with LPC coefficients.

If **use_lpc_decorr** (p. 277) is false then this is just a copy of **LSsig** (p. 275) before the delays.

5.32.4.25 white_LSsig_smpl `MHASignal::waveform_t adaptive_feedback_canceller_config::white_LSsig_smpl [private]`

Single sample (for each channel) of white_LSsig that will be written to rb_white_LSsig next.

For the filter estimation we have to use a ringbuffer with the size of filter_length to buffer **white_LSsig** (p. 277). For each filter tap we have to update the buffer and it is only possible to update a ringbuffer with a whole waveform_t. That is why we have to use this single sample variable.

5.32.4.26 rb_white_LSsig `MHASignal::ringbuffer_t adaptive_feedback_canceller_config::rb_white_LSsig [private]`

Ringbuffer containing the values used to determine the power of **white_LSsig** (p. 277) in each channel (see **channels** (p. 274)) and update **FBfilter_estim** (p. 276).

The ringbuffer has a size equal to filter_length.

5.32.4.27 current_power `std::vector< mha_real_t> adaptive_feedback_canceller_config::current_power [private]`

5.32.4.28 white_MICsig `MHASignal::waveform_t adaptive_feedback_canceller_config::white_MICsig [private]`

5.32.4.29 white_FBsig_estim `MHASignal::waveform_t adaptive_feedback_canceller_config::white_FBsig_estim [private]`

5.32.4.30 white_ERRsig `MHASignal::waveform_t adaptive_feedback_canceller_config::white_ERRsig [private]`

5.32.4.31 debug_mode `bool adaptive_feedback_canceller_config::debug_mode [private]`

When executing the code for debug purposes this flag can be set to true in order to capture variable states and provide them as AC variables to the user.

The following variables are captured: **FBfilter_estim_ac** (p. 276), **ERRsig_ac** (p. 277), **current_power_ac** (p. 278), **estim_err_ac** (p. 279)

5.32.4.32 current_power_ac `MHA_AC::waveform_t adaptive_feedback_canceller_config::current_power_ac [private]`

5.32.4.33 estim_err_ac `MHA_AC::waveform_t adaptive_feedback_canceller_config::estim_err_ac [private]`

AC-variable publishing the state of 'estim_err' if **debug_mode** (p. 278) == true 'estim_err' is a variable local to this plugin's **process()** (p. 273) method.

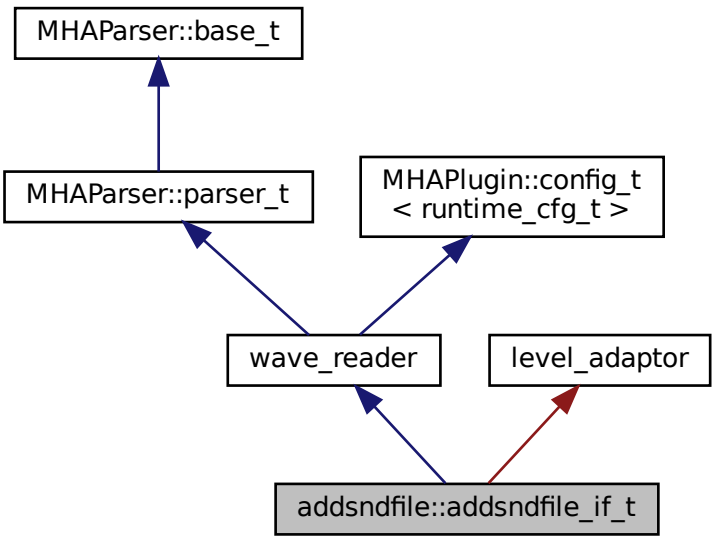
It represents a part of the NLMS equation, namely the **stepsize** (p. 275) normalized by **current_power** (p. 278) times **ERRsig** (p. 277).

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

- **adaptive_feedback_canceller.h**
- **adaptive_feedback_canceller.cpp**

5.33 addsndfile::addsndfile_if_t Class Reference

Inheritance diagram for addsndfile::addsndfile_if_t:



Public Member Functions

- **addsndfile_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.33.1 Constructor & Destructor Documentation

5.33.1.1 addsndfile_if_t()

```
addsndfile::addsndfile_if_t::addsndfile_if_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.33.2 Member Function Documentation

5.33.2.1 process()

```
mha_wave_t * addsndfile::addsndfile_if_t::process (
    mha_wave_t * s )
```

5.33.2.2 prepare()

```
void addsndfile::addsndfile_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1301).

5.33.2.3 release()

```
void addsndfile::addsndfile_if_t::release (
    void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1302).

5.33.2.4 update()

```
void addsndfile::addsndfile_if_t::update ( ) [private]
```

5.33.2.5 change_mode()

```
void addsndfile::addsndfile_if_t::change_mode ( ) [private]
```

5.33.2.6 set_level()

```
void addsndfile::addsndfile_if_t::set_level ( ) [private]
```

5.33.2.7 scan_dir() void addsndfile::addsndfile_if_t::scan_dir ( ) [private]

5.33.3 Member Data Documentation

5.33.3.1 filename MHAParser::string_t addsndfile::addsndfile_if_t::filename [private]

5.33.3.2 path MHAParser::string_t addsndfile::addsndfile_if_t::path [private]

5.33.3.3 loop MHAParser::bool_t addsndfile::addsndfile_if_t::loop [private]

5.33.3.4 level MHAParser::float_t addsndfile::addsndfile_if_t::level [private]

5.33.3.5 levelmode MHAParser::kw_t addsndfile::addsndfile_if_t::levelmode [private]

5.33.3.6 resamplingmode MHAParser::kw_t addsndfile::addsndfile_if_t::resamplingmode [private]

5.33.3.7 channels MHAParser::vint_t addsndfile::addsndfile_if_t::channels [private]

5.33.3.8 mode

MHAParser::kw_t addsndfile::addsndfile_if_t::mode [private]

5.33.3.9 ramplen

MHAParser::float_t addsndfile::addsndfile_if_t::ramplen [private]

5.33.3.10 startpos

MHAParser::int_t addsndfile::addsndfile_if_t::startpos [private]

5.33.3.11 mapping

MHAParser::vint_mon_t addsndfile::addsndfile_if_t::mapping [private]

5.33.3.12 numchannels

MHAParser::int_mon_t addsndfile::addsndfile_if_t::numchannels [private]

5.33.3.13 mhachannels

MHAParser::int_mon_t addsndfile::addsndfile_if_t::mhachannels [private]

5.33.3.14 active

MHAParser::int_mon_t addsndfile::addsndfile_if_t::active [private]

5.33.3.15 search_pattern

MHAParser::string_t addsndfile::addsndfile_if_t::search↩ _pattern [private]

5.33.3.16 search_result

MHAParser::vstring_mon_t addsndfile::addsndfile_if_t↩ ::search_result [private]

5.33.3.17 uint_mode `unsigned int addsndfile::addsndfile_if_t::uint_mode [private]`

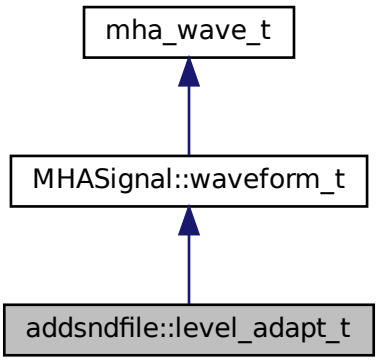
5.33.3.18 patchbay `MHAEvents::patchbay_t< addsndfile_if_t> addsndfile::addsndfile_if_t::patchbay [private]`

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

- **addsndfile.cpp**

5.34 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

5.34.1 Constructor & Destructor Documentation

5.34.1.1 level_adapt_t() 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_*)

5.34.2 Member Function Documentation

5.34.2.1 update_frame() void addsndfile::level_adapt_t::update_frame ( )

5.34.2.2 get_level() **mha_real_t** addsndfile::level_adapt_t::get_level ( ) const [inline]

5.34.2.3 can_update() bool addsndfile::level_adapt_t::can_update ( ) const [inline]

5.34.3 Member Data Documentation

5.34.3.1 ilen `unsigned int addsndfile::level_adapt_t::ilen [private]`

5.34.3.2 pos `unsigned int addsndfile::level_adapt_t::pos [private]`

5.34.3.3 wnd **`MHAWindow::fun_t`** `addsndfile::level_adapt_t::wnd [private]`

5.34.3.4 l_new **`mha_real_t`** `addsndfile::level_adapt_t::l_new [private]`

5.34.3.5 l_old **`mha_real_t`** `addsndfile::level_adapt_t::l_old [private]`

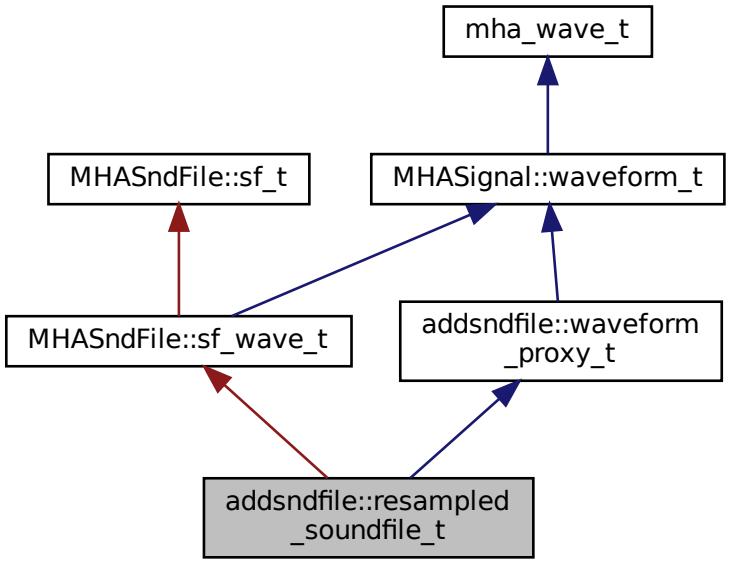
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

5.35 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

5.35.1 Detailed Description

Reads sound from file and resamples it if necessary and wanted.

Sound data can then be used by addsndfile.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 resampled_soundfile_t() 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 a point of the addsndfile plugin
resampling_mode	DONT_RESAMPLE_STRICT: Do not resample, ju use the samples from the sound file at the current sample rate, even if the sample rate of the sound differs. DONT_RESAMPLE_PERMISSIVE: Do no resample, if the sample rate of the MHA differs fro sample rate of the sound file, raise an error. DO_RESAMPLE: Resample.

Exceptions

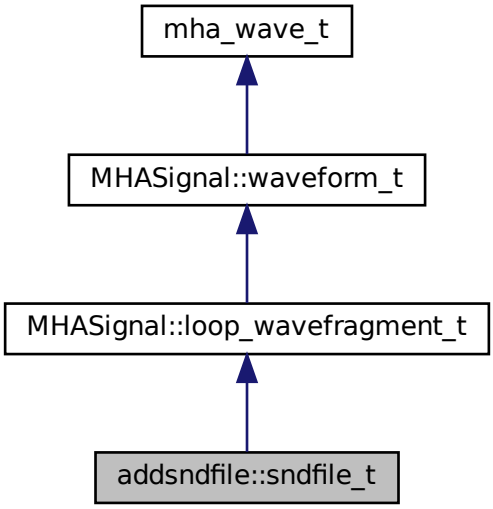
MHA_Error *(p. 906)*	If the sampling rate of the file does not match the sampling rate of the MHA and DONT_RESAMPLE_STRICT was requested. If resampling failed (e.g. due to non-rational quotien MHA sampling rate and sound file sampling rate).

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

- **addsndfile.cpp**

5.36 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

5.36.1 Constructor & Destructor Documentation

5.36.1.1 sndfile_t()

```
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 )
```

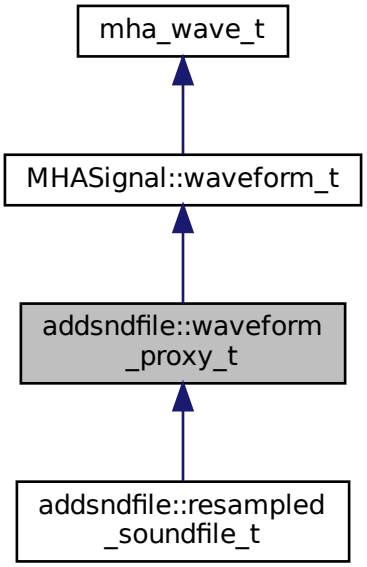
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

5.37 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. 286).

Inheritance diagram for addsndfile::waveform_proxy_t:



Public Member Functions

- **waveform_proxy_t** (unsigned frames, unsigned **channels**)

Additional Inherited Members

5.37.1 Detailed Description

Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. 286).

5.37.2 Constructor & Destructor Documentation

5.37.2.1 waveform_proxy_t()

```
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**

5.38 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), bool update↩ _beta=true)

  ***ADM** (p. 290) 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**

5.38.1 Detailed Description

template<**class F**>
class ADM::ADM< **F** >

Adaptive differential microphone, working for speech frequency range.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 ADM() 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 adaptatio

Parameters

lp_alphas	Pointer to array of alpha coefficients for the lowpas filter used for adaptation. Since this class uses line phase FIR filters only, only the first half (order/2 + the coefficients will be read (coefficients for linear-phase FIR filters are symmetric).
decomb_order	Filter order of FIR compensation filter (compensat for comb filter characteristic). decomb_order <= 1 deactivates filter.
decomb_alphas	Pointer to array of alpha coefficients for the compensation filter used to compensate for the co filter characteristic. Since this class uses linear ph FIR filters only, only the first half (order/2 + 1)of the coefficients will be read (coefficients for linear-pha FIR filters are symmetric).
tau_beta	Time constant of the lowpass filter used for averag the power of the output signal
mu_beta	adaptation speed

5.38.3 Member Function Documentation

5.38.3.1 process() template<class F >

```
F  ADM::ADM< F >::process (
            const F & front,
            const F & back,
            const F & external_beta = F(-1),
            bool update_beta = true )  [inline]
```

ADM (p. 290) 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.
update_beta	Perform the beta adaptation step?

Returns

The computed output sample

5.38.3.2 beta() template<class F >

F **ADM::ADM**< F >::beta ( ) const [inline]

5.38.4 Member Data Documentation

5.38.4.1 m_delay_front template<class F >

Delay<F> **ADM::ADM**< F >::m_delay_front [private]

5.38.4.2 m_delay_back template<class F >

Delay<F> **ADM::ADM**< F >::m_delay_back [private]

5.38.4.3 m_lp_bf template<class F >

Linearphase_FIR<F> **ADM::ADM**< F >::m_lp_bf [private]

5.38.4.4 m_lp_result template<class F >

Linearphase_FIR<F> **ADM::ADM**< F >::m_lp_result [private]

5.38.4.5 m_decomb template<class F >

Linearphase_FIR<F> **ADM::ADM**< F >::m_decomb [private]

5.38.4.6 m_beta template<class F >

F **ADM::ADM**< F >::m_beta [private]

5.38.4.7 m_mu_beta template<class F >

F **ADM::ADM**< F >::m_mu_beta [private]

5.38.4.8 m_powerfilter_coeff template<class F >

F **ADM::ADM**< F >::m_powerfilter_coeff [private]

5.38.4.9 m_powerfilter_norm template<class F >

F **ADM::ADM**< F >::m_powerfilter_norm [private]

5.38.4.10 m_powerfilter_state template<class F >

F **ADM::ADM**< F >::m_powerfilter_state [private]

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

- **adm.hh**

5.39 ADM::Delay< F > Class Template Reference

A delay-line class.

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

5.39.1 Detailed Description

template<class F>
class ADM::Delay< F >

A delay-line class.

It can delay samples in a single audio channel. It stores samples while they are delayed until they have reached their target delay. It can also do subsample-delays for a limited frequency range below fs/4.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 Delay() 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)

Parameters

f_design	design frequency (in Hz). Subsampledelay is exact for this frequency and approximate for different frequencies
fs	sampling frequency (in Hz).

5.39.2.2 ~Delay() template<class F >

ADM::Delay< F >::~ **Delay**

5.39.3 Member Function Documentation

5.39.3.1 process() template<class F >

```
F  ADM::Delay< F >::process (
            const F & in_sample )  [inline]
```

Apply delay to signal.

Whenever a new audio sample enters the delay line, a previous audio sample, now delayed, is returned by this method. Sub-sample-delays are implemented by applying a first-order recursive lowpass filter. This method needs to be called repeatedly, once for each incoming audio sample in correct order for a block of audio with multiple samples (oldest first, newest last).

Parameters

in_sample	The current input signal sample

Returns

The output sample, which is one of the previously received input samples except for the sub-sample delay.

5.39.4 Member Data Documentation

5.39.4.1 m_fullsamples template<class F >

unsigned **ADM::Delay**< F >::m_fullsamples [private]

Integer part of delay.

5.39.4.2 m_coeff template<class F >

F **ADM::Delay**< F >::m_coeff [private]

coefficient for 1st order IIR lowpass filter which does the subsample delay

5.39.4.3 m_norm template<class F >

F **ADM::Delay**< F >::m_norm [private]

normalization for the IIR subsample delay filter

5.39.4.4 m_state template<class F >

F* **ADM::Delay**< F >::m_state [private]

Ringbuffer: Delayline.

5.39.4.5 m_now_in 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**

5.40 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

5.40.1 Detailed Description

template<class F>
class ADM::Linearphase_FIR< F >

An efficient linear-phase fir filter implementation.

5.40.2 Constructor & Destructor Documentation

5.40.2.1 Linearphase_FIR() `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.)

5.40.2.2 ~Linearphase_FIR() template<class F >

ADM::Linearphase_FIR< F >::~ **Linearphase_FIR**

5.40.3 Member Function Documentation

5.40.3.1 process() 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

5.40.4 Member Data Documentation

5.40.4.1 m_order template<class F >

unsigned **ADM::Linearphase_FIR**< F >::m_order [private]

The filter order of this linear-phase FIR filter.

5.40.4.2 m_alphas 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

5.40.4.3 m_output template<class F >

F* ADM::Linearphase_FIR< F >::m_output [private]

Ringbuffer for building future output.

5.40.4.4 m_now template<class F >

unsigned ADM::Linearphase_FIR< F >::m_now [private]

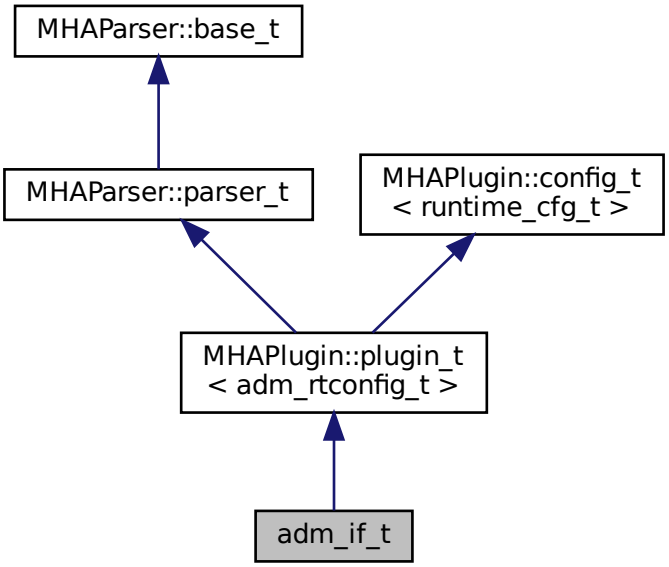
current start of ringbuffer

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

- **adm.hh**

5.41 adm_if_t Class Reference

Inheritance diagram for adm_if_t:



Public Member Functions

- **adm_if_t** (**MHA_AC::algo_comm_t** & **ac**, const std::string &configured_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**
- **MHAParser::int_t adaptation_ratio**
- unsigned **input_channels**
- int **framecnt**
- **mha_real_t srate**
- **MHAEvents::patchbay_t**< **adm_if_t** > **patchbay**

Additional Inherited Members

5.41.1 Constructor & Destructor Documentation

5.41.1.1 adm_if_t() `adm_if_t::adm_if_t (`

```
    MHA_AC::algo_comm_t & ac,
    const std::string & configured_name )
```

5.41.2 Member Function Documentation

5.41.2.1 process() `mha_wave_t * adm_if_t::process ( mha_wave_t * in )`

5.41.2.2 prepare() `void adm_if_t::prepare ( mhaconfig_t & cfg ) [virtual]`

Implements **MHAPlugin::plugin_t< adm_rtconfig_t >** (p. 1301).

5.41.2.3 release() `void adm_if_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< adm_rtconfig_t >** (p. 1302).

5.41.2.4 update() `void adm_if_t::update ( ) [private]`

5.41.2.5 is_prepared() `bool adm_if_t::is_prepared ( ) [inline], [private]`

5.41.3 Member Data Documentation

5.41.3.1 out `MHASignal::waveform_t* adm_if_t::out [private]`

5.41.3.2 front_channels `MHAParser::vint_t adm_if_t::front_channels [private]`

5.41.3.3 rear_channels `MHAParser::vint_t adm_if_t::rear_channels [private]`

5.41.3.4 distances `MHAParser::vfloat_t adm_if_t::distances [private]`

5.41.3.5 lp_order `MHAParser::int_t adm_if_t::lp_order [private]`

5.41.3.6 decomb_order `MHAParser::int_t adm_if_t::decomb_order [private]`

5.41.3.7 bypass `MHAParser::int_t adm_if_t::bypass [private]`

5.41.3.8 beta `MHAParser::float_t adm_if_t::beta [private]`

5.41.3.9 mu_beta `MHAParser::vfloat_t adm_if_t::mu_beta [private]`

5.41.3.10 tau_beta `MHAParser::vfloat_t adm_if_t::tau_beta [private]`

5.41.3.11 coeff_lp `MHAParser::vfloat_mon_t adm_if_t::coeff_lp [private]`

5.41.3.12 coeff_decomb `MHAParser::vfloat_mon_t adm_if_t::coeff_decomb [private]`

5.41.3.13 adaptation_ratio `MHAParser::int_t adm_if_t::adaptation_ratio [private]`

5.41.3.14 input_channels `unsigned adm_if_t::input_channels [private]`

5.41.3.15 framecnt `int adm_if_t::framecnt [private]`

5.41.3.16 srate `mha_real_t adm_if_t::srate [private]`

5.41.3.17 patchbay `MHAEvents::patchbay_t< adm_if_t> adm_if_t::patchbay [private]`

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

- **adm.cpp**

5.42 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, int adaptation_↩ ratio_, 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.
- int **get_adaptation_ratio** () const

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.
- int **adaptation_ratio**
  Prescale.
- **MHASignal::waveform_t** * **lp_coeffs**
  Lowpass filter coefficients.
- std::vector< **MHASignal::waveform_t** * > **decomb_coeffs**
  Decomb-Filter coefficients.
- std::vector< **adm_t** * > **adms**
  ADMs.

5.42.1 Member Typedef Documentation

5.42.1.1 adm_t typedef ADM::ADM< mha_real_t> adm_rtconfig_t::adm_t

5.42.2 Constructor & Destructor Documentation

5.42.2.1 adm_rtconfig_t() adm_rtconfig_t::adm_rtconfig_t (

```
unsigned nchannels_in,
unsigned nchannels_out,
int adaptation_ratio_,
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
*adaptation_ratio_*	Update beta every adaptation_ratio frames
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 (compensat comb filter characteristic)
tau_beta	Time constants of the lowpass filter used for avera the power of the output signal used for adaptation
mu_beta	Adaptation step sizes

5.42.2.2 ~adm_rtconfig_t() adm_rtconfig_t::~adm_rtconfig_t ( ) [virtual]

5.42.3 Member Function Documentation

5.42.3.1 check_index() `void adm_rtconfig_t::check_index ( unsigned index ) const [inline], [private]`

Index checking for all internal arrays.

Exceptions

MHA_Error *(p. 906)*	if index out of range.

5.42.3.2 num_adms() `size_t adm_rtconfig_t::num_adms ( ) const [inline]`

5.42.3.3 adm() `adm_t& adm_rtconfig_t::adm ( unsigned index ) [inline]`

Returns adm object number index.

5.42.3.4 front_channel() `int adm_rtconfig_t::front_channel ( unsigned index ) const [inline]`

Returns index of front channel for adm number index.

5.42.3.5 rear_channel() `int adm_rtconfig_t::rear_channel ( unsigned index ) const [inline]`

Returns index of rear channel for adm number index.

5.42.3.6 get_adaptation_ratio() int adm_rtconfig_t::get_adaptation_ratio ( ) const [inline]

5.42.4 Member Data Documentation

5.42.4.1 front_channels std::vector<int> adm_rtconfig_t::front_channels [private]

Indices of channels containing the signals from the front microphones.

5.42.4.2 rear_channels std::vector<int> adm_rtconfig_t::rear_channels [private]

Indices of channels containing the signals from the rear microphones.

5.42.4.3 adaptation_ratio int adm_rtconfig_t::adaptation_ratio [private]

Prescale.

5.42.4.4 lp_coeffs **MHASignal::waveform_t*** adm_rtconfig_t::lp_coeffs [private]

Lowpass filter coefficients.

5.42.4.5 decomb_coeffs std::vector< **MHASignal::waveform_t***> adm_rtconfig_t↩ ::decomb_coeffs [private]

Decomb-Filter coefficients.

5.42.4.6 adms `std::vector< adm_t *> adm_rtconfig_t::adms [private]`

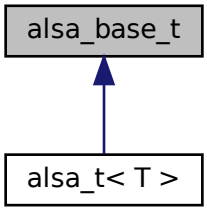
ADMs.

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

- **adm.cpp**

5.43 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 bool **read** (**mha_wave_t** **)=0

  *read audio samples from the device into an internal **mha_wave_t** (p. 985) buffer, then update the pointer given as parameter to point to the internal structure.*
- virtual bool **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.

5.43.1 Constructor & Destructor Documentation

5.43.1.1 alsa_base_t() `alsa_base_t::alsa_base_t ( ) [inline]`

5.43.1.2 ~alsa_base_t() `virtual alsa_base_t::~alsa_base_t ( ) [virtual], [default]`

5.43.2 Member Function Documentation

5.43.2.1 start() `virtual void alsa_base_t::start ( ) [pure virtual]`

start puts alsa device in usable state

Implemented in **alsa_t**< **T** > (p. 315).

5.43.2.2 stop() `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. 315).

5.43.2.3 read() `virtual bool alsa_base_t::read ( mha_wave_t ** ) [pure virtual]`

read audio samples from the device into an internal **mha_wave_t** (p. 985) 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. 316).

5.43.2.4 write() virtual bool 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. 316).

5.43.3 Member Data Documentation

5.43.3.1 pcm snd_pcm_t* alsa_base_t::pcm

The underlying alsa handle to this sound card.

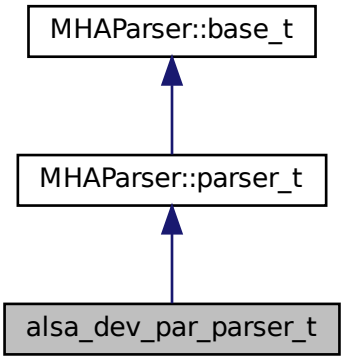
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.44 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

5.44.1 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.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 alsa_dev_par_parser_t() `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

5.44.3 Member Data Documentation

5.44.3.1 device `MHAParser::string_t alsa_dev_par_parser_t::device`

Name of the device in the alsa world, like "hw:0.0", "default", etc.

5.44.3.2 nperiods `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.

5.44.3.3 stream_dir `snd_pcm_stream_t alsa_dev_par_parser_t::stream_dir`

Remember the direction (capture/playback) of this device.

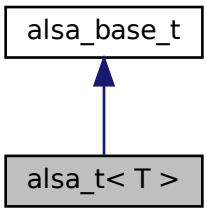
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.45 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
- bool **read** (**mha_wave_t** ∗∗) override

  *read audio samples from the device into an internal **mha_wave_t** (p. 985) buffer, then update the pointer given as parameter to point to the internal structure.*
- bool **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**
- snd_pcm_format_t **pcm_format**

Additional Inherited Members

5.45.1 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.

5.45.2 Constructor & Destructor Documentation

5.45.2.1 alsa_t() 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

5.45.2.2 ~alsa_t() template<typename T >

```
alsa_t< T >::~ alsa_t
```

Destructor closes the sound device.

5.45.3 Member Function Documentation

5.45.3.1 start() template<typename T >

```
void  alsa_t< T >::start  [override], [virtual]
```

start puts alsa device in usable state

Implements **alsa_base_t** (p. 310).

5.45.3.2 stop() 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. 310).

5.45.3.3 read() template<typename T >

```
bool  alsa_t< T >::read (
            mha_wave_t ** s )  [override], [virtual]
```

read audio samples from the device into an internal **mha_wave_t** (p. 985) 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. 310).

5.45.3.4 write() template<typename T >

```
bool  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. 310).

5.45.4 Member Data Documentation

5.45.4.1 channels template<typename T >

```
unsigned int  alsa_t< T >::channels  [private]
```

5.45.4.2 fragsize template<typename T >

unsigned int **alsa_t**< T >::fragsize [private]

5.45.4.3 buffer template<typename T >

T* **alsa_t**< T >::buffer [private]

5.45.4.4 frame_data template<typename T >

std::vector< **mha_real_t**> **alsa_t**< T >::frame_data [private]

5.45.4.5 wave template<typename T >

MHASignal::waveform_t **alsa_t**< T >::wave [private]

internal buffer to store sound samples coming from the sound card.

5.45.4.6 gain template<typename T >

const **mha_real_t** **alsa_t**< T >::gain [private]

5.45.4.7 invgain template<typename T >

const **mha_real_t** **alsa_t**< T >::invgain [private]

5.45.4.8 pcm_format template<typename T >

snd_pcm_format_t **alsa_t**< T >::pcm_format [private]

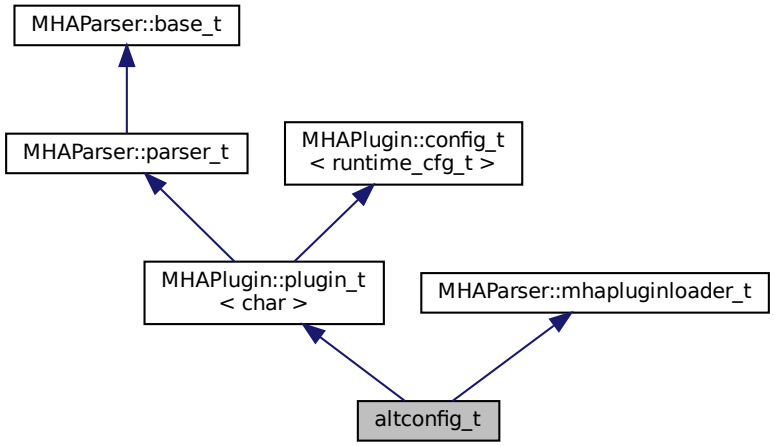
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

5.46 altconfig_t Class Reference

Single class implementing plugin altconfig.

Inheritance diagram for altconfig_t:



Public Member Functions

- **altconfig_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor initializes an instance of the altconfig plugin.
- void **prepare** (**mhaconfig_t** &cf)
  Invoked by MHA when this plugin should prepare for signal processing.
- void **release** ()
  Invoked by MHA when this plugin should call its release function.

Private Member Functions

- void **on_set_algos** ()
  Callback executed when the configuration variable "algos" is written to at run time.
- void **on_set_select** ()
  Callback executed when the configuration variable "select" is successfully written to at run time.
- void **event_select_all** ()
  Callback executed when the configuration variable "selectall" is written to at run time.
- std::map< std::string, **MHAParser::string_t** > **save_state** ()
  Save the old state of the user-defined sub-parsers for eventual restoration later.
- void **restore_state** (std::map< std::string, **MHAParser::string_t** > &state, std::map< std::string, **MHAParser::string_t** > &failed_state)
  Restore the old parser state from a saved state.

Private Attributes

- **MHAParser::vstring_t parser_algos**
- **MHAParser::kw_t select_plug**
- **MHAParser::bool_t selectall**
- std::map< std::string, **MHAParser::string_t** > **configs**
  Storage for alternative configuration commands.
- **MHAEvents::patchbay_t**< **altconfig_t** > **patchbay**
  Configuration event broker.

Additional Inherited Members

5.46.1 Detailed Description

Single class implementing plugin altconfig.

altconfig loads another plugin and can send configuration commands to that plugin. altconfig does not need a separate runtime configuration class, template parameter char is used as a placeholder. Uses parser variable names "algos" and "select" even though the alternatives that can be selected in this plugin are not algorithms or plugins but configuration commands in order to be interface-compatible with plugin altplugs. mhacontrol has a UI area that automatically fills with the alternatives found in either altplugs or altconfig if the complete MHA loads exactly one plugin with this interface.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 altconfig_t() `altconfig_t::altconfig_t ( MHA_AC::algo_comm_t & iac, const std::string & configured_name )`

Constructor initializes an instance of the altconfig plugin.

Parameters

iac	Algorithm communication variable space, not use this plugin except to initialize base class.
configured_name	Configured name of this plugin.

5.46.3 Member Function Documentation

5.46.3.1 prepare() `void altconfig_t::prepare ( mhaconfig_t & cf ) [inline], [virtual]`

Invoked by MHA when this plugin should prepare for signal processing.

altconfig delegates to the loaded plugin and does not need to do more work to prepare.

Parameters

cf	signal dimensions, forwarded to loaded plugin which may change the signal dimensions.

Implements **MHAPlugin::plugin_t< char >** (p. 1301).

5.46.3.2 release() `void altconfig_t::release ( void ) [inline], [virtual]`

Invoked by MHA when this plugin should call its release function.

altconfig delegates to the loaded plugin.

Reimplemented from **MHAPlugin::plugin_t< char >** (p. 1302).

5.46.3.3 on_set_algos() `void altconfig_t::on_set_algos ( ) [private]`

Callback executed when the configuration variable "algos" is written to at run time.

Adds the configuration variables that store the alternative configuration commands based on the new names and sets the allowed values of configuration variable "select"

5.46.3.4 on_set_select() `void altconfig_t::on_set_select ( ) [private]`

Callback executed when the configuration variable "select" is successfully written to at run time.

Causes the execution of the stored command for the selected condition in the context of the loaded plugin.

5.46.3.5 event_select_all() void altconfig_t::event_select_all ( ) [private]

Callback executed when the configuration variable "selectall" is written to at run time.

When set to yes, iterates once through all stored configurations in the order they appear in configuration variable algos.

5.46.3.6 save_state() std::map< std::string, **MHAParser::string_t** > altconfig_t↩ ::save_state ( ) [private]

Save the old state of the user-defined sub-parsers for eventual restoration later.

operator= does not suffice to store/restore the old state because we need to re-insert the string_t's that were removed, but a copy of the string_t keeps a pointer to its parent and complains if we try to insert_item it again. So we just copy the relevant data into a new string_t.

Parameters

old_state	old parser state

Returns

Copy of the old state

5.46.3.7 restore_state() void altconfig_t::restore_state (std::map< std::string, **MHAParser::string_t** > & *state*, std::map< std::string, **MHAParser::string_t** > & *failed_state*) [private]

Restore the old parser state from a saved state.

Parameters

state	old parser state

5.46.4 Member Data Documentation

5.46.4.1 parser_algos `MHAParser::vstring_t altconfig_t::parser_algos [private]`

5.46.4.2 select_plug `MHAParser::kw_t altconfig_t::select_plug [private]`

5.46.4.3 selectall `MHAParser::bool_t altconfig_t::selectall [private]`

5.46.4.4 configs `std::map<std::string, MHAParser::string_t> altconfig_t::configs [private]`

Storage for alternative configuration commands.

New entries are registered with the plugin's parser when parser_algos is updated.

5.46.4.5 patchbay `MHAEvents::patchbay_t< altconfig_t> altconfig_t::patchbay [private]`

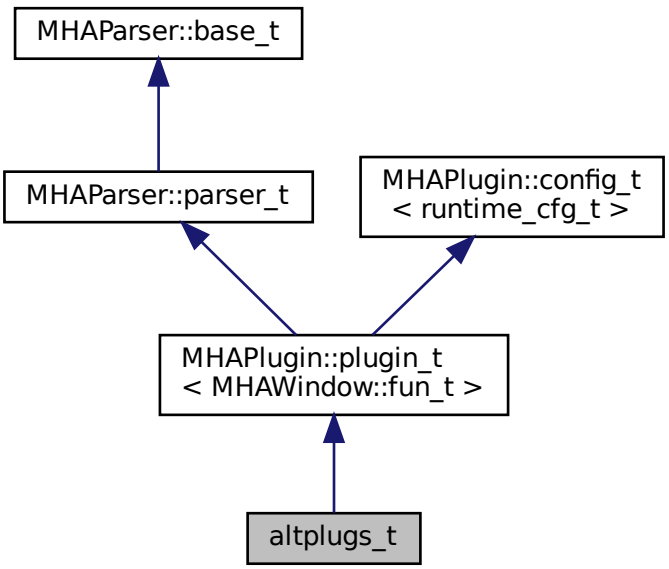
Configuration event broker.

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

- **altconfig.hh**
- **altconfig.cpp**

5.47 altplugs_t Class Reference

Inheritance diagram for altplugs_t:



Public Member Functions

- **altplugs_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.47.1 Constructor & Destructor Documentation

5.47.1.1 altplugs_t() `altplugs_t::altplugs_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.47.2 Member Function Documentation

5.47.2.1 prepare() `void altplugs_t::prepare (`

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t**< **MHAWindow::fun_t** > (p. 1301).

5.47.2.2 release() void altplugs_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< MHAWindow::fun_t >** (p. 1302).

5.47.2.3 process() [1/4] void altplugs_t::process (mha_wave_t * sIn, mha_wave_t ** sOut)

5.47.2.4 process() [2/4] void altplugs_t::process (mha_spec_t * sIn, mha_wave_t ** sOut)

5.47.2.5 process() [3/4] void altplugs_t::process (mha_wave_t * sIn, mha_spec_t ** sOut)

5.47.2.6 process() [4/4] void altplugs_t::process (mha_spec_t * sIn, mha_spec_t ** sOut)

5.47.2.7 parse() [1/2] std::string altplugs_t::parse (const std::string & arg) [virtual]

Reimplemented from **MHAParser::base_t** (p. 1177).

5.47.2.8 parse() [2/2]

```
virtual void altplugs_t::parse (
    const char * a1,
    char * a2,
    unsigned int a3 ) [inline], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1178).

5.47.2.9 event_set_plugs()

```
void altplugs_t::event_set_plugs ( ) [private]
```

5.47.2.10 event_add_plug()

```
void altplugs_t::event_add_plug ( ) [private]
```

5.47.2.11 event_delete_plug()

```
void altplugs_t::event_delete_plug ( ) [private]
```

5.47.2.12 event_select_plug()

```
void altplugs_t::event_select_plug ( ) [private]
```

5.47.2.13 update_selector_list()

```
void altplugs_t::update_selector_list ( ) [private]
```

5.47.2.14 update_ramplen()

```
void altplugs_t::update_ramplen ( ) [private]
```

5.47.2.15 proc_ramp()

```
void altplugs_t::proc_ramp (
    mha_wave_t * s ) [private]
```

5.47.3 Member Data Documentation

5.47.3.1 use_own_ac

`MHAParser::bool_t altplugs_t::use_own_ac [private]`

5.47.3.2 parser_plugs

`MHAParser::vstring_t altplugs_t::parser_plugs [private]`

5.47.3.3 add_plug

`MHAParser::string_t altplugs_t::add_plug [private]`

5.47.3.4 delete_plug

`MHAParser::string_t altplugs_t::delete_plug [private]`

5.47.3.5 ramplen

`MHAParser::float_t altplugs_t::ramplen [private]`

5.47.3.6 select_plug

`MHAParser::kw_t altplugs_t::select_plug [private]`

5.47.3.7 current

`MHAParser::parser_t altplugs_t::current [private]`

5.47.3.8 nondefault_labels

`MHAParser::vstring_mon_t altplugs_t::nondefault_labels [private]`

5.47.3.9 plugs

`std::vector< mhaplug_cfg_t* > altplugs_t::plugs [private]`

5.47.3.10 selected_plug

`mhaplug_cfg_t* altplugs_t::selected_plug [private]`

5.47.3.11 patchbay

`MHAEvents::patchbay_t< altplugs_t> altplugs_t::patchbay [private]`

5.47.3.12 fallback_wave

`MHASignal::waveform_t* altplugs_t::fallback_wave [private]`

5.47.3.13 fallback_spec

`MHASignal::spectrum_t* altplugs_t::fallback_spec [private]`

5.47.3.14 cfin

`mhaconfig_t altplugs_t::cfin [private]`

5.47.3.15 cfout

`mhaconfig_t altplugs_t::cfout [private]`

5.47.3.16 prepared

`bool altplugs_t::prepared [private]`

5.47.3.17 added_via_plugs

`bool altplugs_t::added_via_plugs [private]`

5.47.3.18 ramp_counter

`unsigned int altplugs_t::ramp_counter [private]`

5.47.3.19 ramp_len `unsigned int altplugs_t::ramp_len [private]`

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

- **altplugs.cpp**

5.48 analysepath_t Class Reference

Public Member Functions

- **analysepath_t** (unsigned int nchannels_in, unsigned int inner_fragsize, int **priority**, **MHAProc_wave2wave_t** inner_proc_wave2wave, **MHAProc_wave2spec_t** inner_↩ proc_wave2spec, void ∗ilibdata, **MHA_AC::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_lf_t**< **mha_real_t** > **wave_fifo**
- **mha_fifo_lf_t**< **MHA_AC::acspace2matrix_t** > **ac_fifo**
- **MHA_AC::acspace2matrix_t inner_ac_copy**
- **MHA_AC::acspace2matrix_t outer_ac_copy**
- **MHA_AC::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**

5.48.1 Constructor & Destructor Documentation

5.48.1.1 analysepath_t() analysepath_t::analysepath_t (

```
unsigned int nchannels_in,
unsigned int inner_fragsize,
int priority,
 MHAProc_wave2wave_t inner_proc_wave2wave,
 MHAProc_wave2spec_t inner_proc_wave2spec,
void * ilibdata,
 MHA_AC::algo_comm_t & outer_ac,
const  MHA_AC::acspace2matrix_t & acspace_template,
 mha_domain_t inner_out_domain,
unsigned int fifo_len_blocks )
```

5.48.1.2 ∼analysepath_t() analysepath_t::∼analysepath_t ( ) [virtual]

5.48.2 Member Function Documentation

5.48.2.1 rt_process() void analysepath_t::rt_process (

```
mha_wave_t * outer_input )
```

5.48.2.2 svc() int analysepath_t::svc ( ) [virtual]

5.48.3 Member Data Documentation

5.48.3.1 inner_process_wave2wave MHAProc_wave2wave_t analysepath_t::inner_↩process_wave2wave [private]

5.48.3.2 inner_process_wave2spec

MHAProc_wave2spec_t analysepath_t::inner_process_wave2spec [private]

5.48.3.3 inner_input

MHASignal::waveform_t analysepath_t::inner_input [private]

5.48.3.4 libdata

void* analysepath_t::libdata [private]

5.48.3.5 wave_fifo

mha_fifo_lf_t< **mha_real_t**> analysepath_t::wave_fifo [private]

5.48.3.6 ac_fifo

mha_fifo_lf_t< **MHA_AC::acspace2matrix_t**> analysepath_t::ac_fifo [private]

5.48.3.7 inner_ac_copy

MHA_AC::acspace2matrix_t analysepath_t::inner_ac_copy [private]

5.48.3.8 outer_ac_copy

MHA_AC::acspace2matrix_t analysepath_t::outer_ac_copy [private]

5.48.3.9 outer_ac

MHA_AC::algo_comm_t& analysepath_t::outer_ac [private]

5.48.3.10 inner_out_domain

mha_domain_t analysepath_t::inner_out_domain [private]

5.48.3.11 inner_error `MHA_Error` analysepath_t::inner_error [private]

5.48.3.12 has_inner_error bool analysepath_t::has_inner_error [private]

5.48.3.13 flag_terminate_inner_thread bool analysepath_t::flag_terminate_inner_↩ thread [private]

5.48.3.14 input_to_process int analysepath_t::input_to_process [private]

5.48.3.15 ProcessMutex pthread_mutex_t analysepath_t::ProcessMutex [private]

5.48.3.16 attr pthread_attr_t analysepath_t::attr [private]

5.48.3.17 priority struct sched_param analysepath_t::priority [private]

5.48.3.18 scheduler int analysepath_t::scheduler [private]

5.48.3.19 thread pthread_t analysepath_t::thread [private]

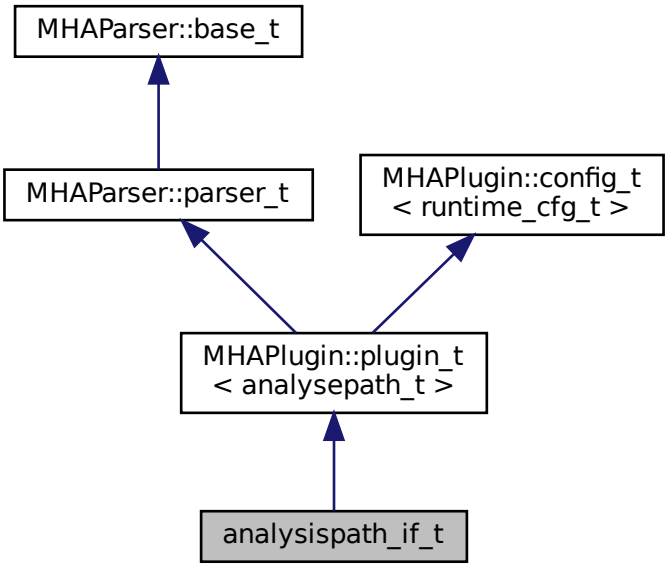
5.48.3.20 cond_to_process `pthread_cond_t analysepath_t::cond_to_process [private]`

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

- **analysispath.cpp**

5.49 analysispath_if_t Class Reference

Inheritance diagram for analysispath_if_t:



Public Member Functions

- **analysispath_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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 **algo**
- **MHA_AC::acspace2matrix_t * acspace_template**

Additional Inherited Members

5.49.1 Constructor & Destructor Documentation

5.49.1.1 analysispath_if_t()

```
analysispath_if_t::analysispath_if_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.49.1.2 ~analysispath_if_t()

```
analysispath_if_t::~analysispath_if_t ( )
```

5.49.2 Member Function Documentation

5.49.2.1 process()

```
mha_wave_t * analysispath_if_t::process (
    mha_wave_t * s )
```

5.49.2.2 prepare()

```
void analysispath_if_t::prepare (
    mhaconfig_t & conf ) [virtual]
```

Implements **MHAPlugin::plugin_t< analysepath_t >** (p. 1301).

5.49.2.3 release() void analysispath_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **analysepath_t** > (p. 1302).

5.49.2.4 loadlib() void analysispath_if_t::loadlib ( ) [private]

5.49.3 Member Data Documentation

5.49.3.1 patchbay **MHAEvents::patchbay_t**< **analysispath_if_t** > analysispath_if_t::patchbay [private]

5.49.3.2 libname **MHAParser::string_t** analysispath_if_t::libname [private]

5.49.3.3 fragsize **MHAParser::int_t** analysispath_if_t::fragsize [private]

5.49.3.4 fifolen **MHAParser::int_t** analysispath_if_t::fifolen [private]

5.49.3.5 priority **MHAParser::int_t** analysispath_if_t::priority [private]

5.49.3.6 vars **MHAParser::vstring_t** analysispath_if_t::vars [private]

5.49.3.7 plug

`plug_t* analysispath_if_t::plug [private]`

5.49.3.8 algo

`std::string analysispath_if_t::algo [private]`

5.49.3.9 acspace_template

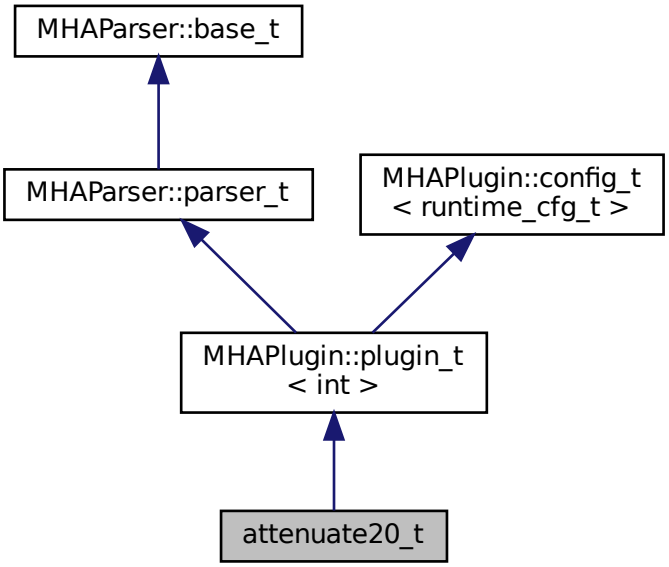
`MHA_AC::acspace2matrix_t* analysispath_if_t::acspace_template [private]`

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

- **analysispath.cpp**

5.50 attenuate20_t Class Reference

Inheritance diagram for attenuate20_t:



Public Member Functions

- **attenuate20_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **release** (void) override
- void **prepare** (**mhaconfig_t** &signal_info) override
- **mha_wave_t** * **process** (**mha_wave_t** *signal)

Additional Inherited Members

5.50.1 Constructor & Destructor Documentation

5.50.1.1 attenuate20_t() `attenuate20_t::attenuate20_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name ) [inline]
```

5.50.2 Member Function Documentation

5.50.2.1 release() `void attenuate20_t::release (`

```
void ) [inline], [override], [virtual]
```

Reimplemented from **MHAPlugin::plugin_t**< **int** > (p. 1302).

5.50.2.2 prepare() `void attenuate20_t::prepare (`

```
mhaconfig_t & signal_info ) [inline], [override], [virtual]
```

Implements **MHAPlugin::plugin_t**< **int** > (p. 1301).

5.50.2.3 process() `mha_wave_t* attenuate20_t::process (`

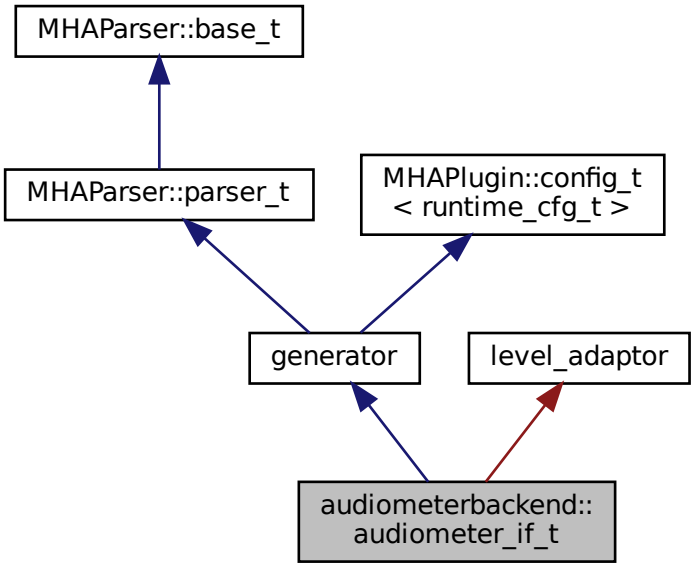
```
mha_wave_t * signal ) [inline]
```

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

- **attenuate20.cpp**

5.51 audiometerbackend::audiometer_if_t Class Reference

Inheritance diagram for audiometerbackend::audiometer_if_t:



Public Member Functions

- **audiometer_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update** ()
- void **change_mode** ()
- void **set_level** ()

Private Attributes

- **MHAParser::int_t freq**
- **MHAParser::kw_t sigtype**
- **MHAParser::float_t level**
- **MHAParser::kw_t mode**
- **MHAParser::float_t ramplen**
- **MHASignal::loop_wavefragment_t::playback_mode_t pmode**
- std::vector< int > **outchannel**
- **MHAEvents::patchbay_t**< **audiometer_if_t** > **patchbay**

Additional Inherited Members

5.51.1 Constructor & Destructor Documentation

5.51.1.1 audiometer_if_t() audiometerbackend::audiometer_if_t::audiometer_if_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.51.2 Member Function Documentation

5.51.2.1 process() mha_wave_t * audiometerbackend::audiometer_if_t::process (mha_wave_t * s)

5.51.2.2 prepare() void audiometerbackend::audiometer_if_t::prepare (mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1301).

5.51.2.3 update() void audiometerbackend::audiometer_if_t::update ( ) [private]

5.51.2.4 change_mode() void audiometerbackend::audiometer_if_t::change_mode ( ) [private]

5.51.2.5 set_level() void audiometerbackend::audiometer_if_t::set_level ( ) [private]

5.51.3 Member Data Documentation

5.51.3.1 freq

MHAParser::int_t audiometerbackend::audiometer_if_t::freq [private]

5.51.3.2 sigtype

MHAParser::kw_t audiometerbackend::audiometer_if_t::sigtype [private]

5.51.3.3 level

MHAParser::float_t audiometerbackend::audiometer_if_t::level [private]

5.51.3.4 mode

MHAParser::kw_t audiometerbackend::audiometer_if_t::mode [private]

5.51.3.5 ramplen

MHAParser::float_t audiometerbackend::audiometer_if_t::ramplen [private]

5.51.3.6 pmode

MHASignal::loop_wavefragment_t::playback_mode_t audiometerbackend↩ ::audiometer_if_t::pmode [private]

5.51.3.7 outchannel

std::vector<int> audiometerbackend::audiometer_if_t::outchannel [private]

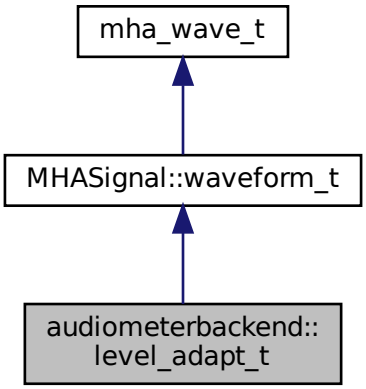
5.51.3.8 patchbay `MHAEvents::patchbay_t< audiometer_if_t> audiometerbackend↩ ::audiometer_if_t::patchbay [private]`

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

- **audiometerbackend.cpp**

5.52 audiometerbackend::level_adapt_t Class Reference

Inheritance diagram for audiometerbackend::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

5.52.1 Constructor & Destructor Documentation

5.52.1.1 level_adapt_t() audiometerbackend::level_adapt_t::level_adapt_t (

```
mhaconfig_t cf,
mha_real_t adapt_len,
mha_real_t l_new_,
mha_real_t l_old_ )
```

5.52.2 Member Function Documentation

5.52.2.1 update_frame() void audiometerbackend::level_adapt_t::update_frame ( )

5.52.2.2 get_level() mha_real_t audiometerbackend::level_adapt_t::get_level ( ) const [inline]

5.52.2.3 can_update() bool audiometerbackend::level_adapt_t::can_update ( ) const [inline]

5.52.3 Member Data Documentation

5.52.3.1 ilen unsigned int audiometerbackend::level_adapt_t::ilen [private]

5.52.3.2 pos `unsigned int audiometerbackend::level_adapt_t::pos [private]`

5.52.3.3 wnd `MHAWindow::fun_t audiometerbackend::level_adapt_t::wnd [private]`

5.52.3.4 l_new `mha_real_t audiometerbackend::level_adapt_t::l_new [private]`

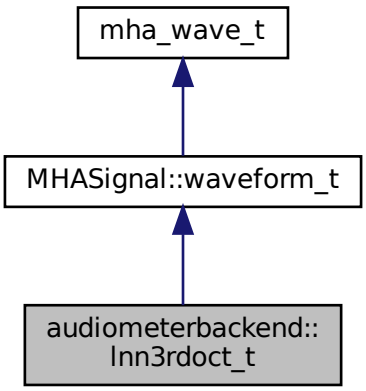
5.52.3.5 l_old `mha_real_t audiometerbackend::level_adapt_t::l_old [private]`

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

- **audiometerbackend.cpp**

5.53 audiometerbackend::lnn3rdoct_t Class Reference

Inheritance diagram for audiometerbackend::lnn3rdoct_t:



Public Member Functions

- **lnn3rdoct_t** (unsigned int fs, unsigned int f, float bw, unsigned int niter)

Private Member Functions

- void **iterate_lnn** ()
- void **bandpass** ()

Private Attributes

- unsigned int **_fmin**
- unsigned int **_fmax**
- std::random_device **dev**
- std::default_random_engine **rng**
- std::uniform_real_distribution< float > **random**

Additional Inherited Members

5.53.1 Constructor & Destructor Documentation

5.53.1.1 lnn3rdoct_t() `audiometerbackend::lnn3rdoct_t::lnn3rdoct_t (`

```
unsigned int fs,
unsigned int f,
float bw,
unsigned int niter )
```

5.53.2 Member Function Documentation

5.53.2.1 iterate_lnn() `void audiometerbackend::lnn3rdoct_t::iterate_lnn ( ) [private]`

5.53.2.2 bandpass() `void audiometerbackend::lnn3rdoct_t::bandpass ( ) [private]`

5.53.3 Member Data Documentation

5.53.3.1 _fmin

`unsigned int audiometerbackend::lnn3rdoct_t::_fmin [private]`

5.53.3.2 _fmax

`unsigned int audiometerbackend::lnn3rdoct_t::_fmax [private]`

5.53.3.3 dev

`std::random_device audiometerbackend::lnn3rdoct_t::dev [private]`

5.53.3.4 rng

`std::default_random_engine audiometerbackend::lnn3rdoct_t::rng [private]`

5.53.3.5 random

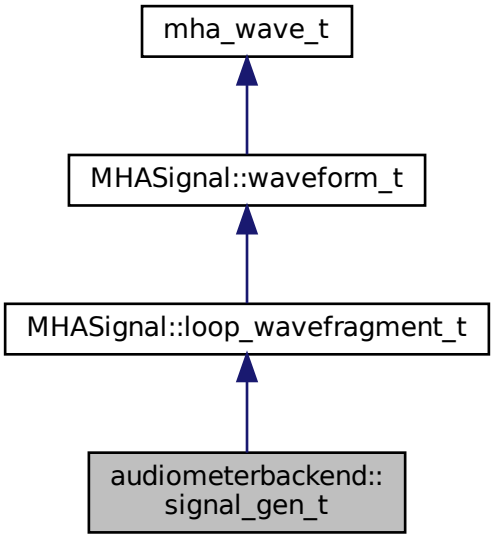
`std::uniform_real_distribution<float> audiometerbackend::lnn3rdoct_t::random [private]`

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

- **audiometerbackend.cpp**

5.54 audiometerbackend::signal_gen_t Class Reference

Inheritance diagram for audiometerbackend::signal_gen_t:



Public Member Functions

- **signal_gen_t** (int f, int fs, unsigned int sigtype)

Additional Inherited Members

5.54.1 Constructor & Destructor Documentation

5.54.1.1 signal_gen_t()

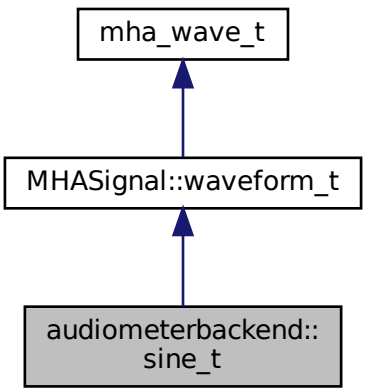
```
audiometerbackend::signal_gen_t::signal_gen_t (
    int f,
    int fs,
    unsigned int sigtype )
```

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

- **audiometerbackend.cpp**

5.55 audiometerbackend::sine_t Class Reference

Inheritance diagram for audiometerbackend::sine_t:



Public Member Functions

- **sine_t** (unsigned int fs, unsigned int f)

Additional Inherited Members

5.55.1 Constructor & Destructor Documentation

5.55.1.1 sine_t() `sine_t::sine_t (`

```
unsigned int fs,
unsigned int f )
```

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

- **audiometerbackend.cpp**

5.56 AuditoryProfile::fmap_t Class Reference

A class to store frequency dependent data (e.g., HTL and UCL).

Inherits std::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

5.56.1 Detailed Description

A class to store frequency dependent data (e.g., HTL and UCL).

5.56.2 Member Function Documentation

5.56.2.1 get_frequencies() `std::vector< mha_real_t > AuditoryProfile::fmap_t::get_frequencies ( ) const`

Return configured frequencies.

5.56.2.2 get_values() `std::vector< mha_real_t > AuditoryProfile::fmap_t::get_values ( ) const`

Return stored values corresponding to the frequencies.

5.56.2.3 isempty() `bool AuditoryProfile::fmap_t::isempty ( ) const [inline]`

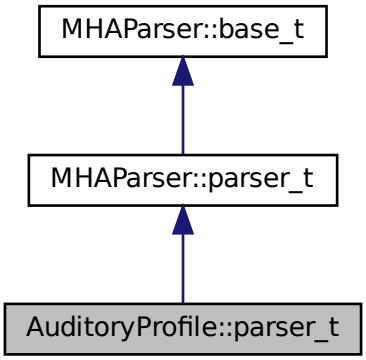
The documentation for this class was generated from the following files:

- **auditory_profile.h**
- **auditory_profile.cpp**

5.57 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

5.57.1 Detailed Description

Class to make the auditory profile accessible through the parser interface.

5.57.2 Constructor & Destructor Documentation

5.57.2.1 parser_t() `AuditoryProfile::parser_t::parser_t ( )`

5.57.3 Member Function Documentation

5.57.3.1 get_current_profile() **`AuditoryProfile::profile_t`** `AuditoryProfile::parser_t::get_current_profile ( )`

5.57.4 Member Data Documentation

5.57.4.1 L `AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::L [private]`

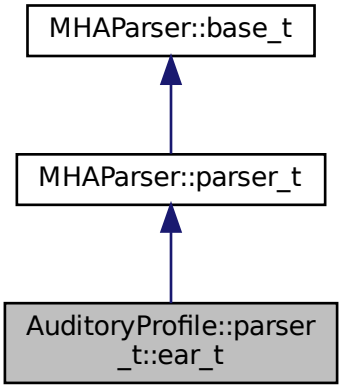
5.57.4.2 R `AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::R [private]`

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.58 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

5.58.1 Constructor & Destructor Documentation

5.58.1.1 ear_t() `AuditoryProfile::parser_t::ear_t::ear_t ( )`

5.58.2 Member Function Documentation

5.58.2.1 get_ear() **AuditoryProfile::profile_t::ear_t** `AuditoryProfile::parser_t↩ ::ear_t::get_ear ( ) const`

5.58.3 Member Data Documentation

5.58.3.1 HTL **AuditoryProfile::parser_t::fmap_t** `AuditoryProfile::parser_t::ear_t↩ ::HTL [private]`

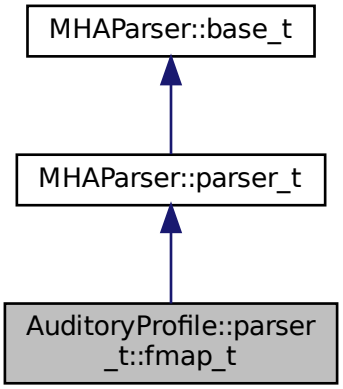
5.58.3.2 UCL **AuditoryProfile::parser_t::fmap_t** `AuditoryProfile::parser_t::ear_t↩ ::UCL [private]`

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.59 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

5.59.1 Constructor & Destructor Documentation

5.59.1.1 fmap_t()

```
AuditoryProfile::parser_t::fmap_t::fmap_t (
    const std::string & name,
    const std::string & help )
```

5.59.2 Member Function Documentation

5.59.2.1 get_fmap()

AuditoryProfile::fmap_t AuditoryProfile::parser_t::fmap_t::get_fmap ( ) const

5.59.2.2 validate()

void AuditoryProfile::parser_t::fmap_t::validate ( ) [private]

5.59.3 Member Data Documentation

5.59.3.1 patchbay

MHAEvents::patchbay_t< **AuditoryProfile::parser_t::fmap_t**> AuditoryProfile::parser_t::fmap_t::patchbay [private]

5.59.3.2 f

MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::f [private]

5.59.3.3 value

MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::value [private]

5.59.3.4 name_

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**

5.60 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.

5.60.1 Detailed Description

The Auditory Profile class.

See definition of auditory profile

Currently only the audiogram data is stored.

5.60.2 Member Function Documentation

5.60.2.1 get_ear() **AuditoryProfile::profile_t::ear_t** AuditoryProfile::profile_t↩ ::get_ear (

```
            unsigned int channel ) const  [inline]
```

Return ear information of channel number.

5.60.3 Member Data Documentation

5.60.3.1 L `AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::L`

Left ear data.

5.60.3.2 R `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**

5.61 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**

5.61.1 Detailed Description

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

5.61.2 Member Function Documentation

5.61.2.1 convert_empty2normal() `void AuditoryProfile::profile_t::ear_t::convert_empty2normal ( )`

5.61.3 Member Data Documentation

5.61.3.1 HTL `AuditoryProfile::fmap_t AuditoryProfile::profile_t::ear_t::HTL`

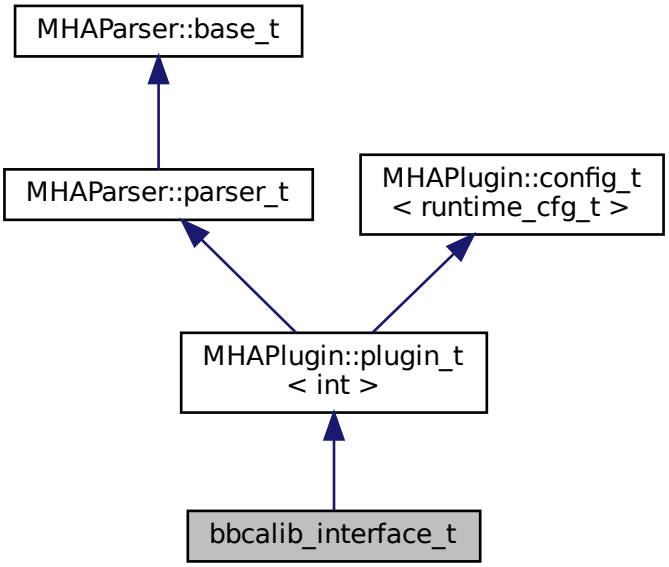
5.61.3.2 UCL `AuditoryProfile::fmap_t AuditoryProfile::profile_t::ear_t::UCL`

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.62 bbcalib_interface_t Class Reference

Inheritance diagram for bbcalib_interface_t:



Public Member Functions

- **bbcalib_interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
- ∼**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

5.62.1 Constructor & Destructor Documentation

5.62.1.1 bbcalib_interface_t() `bbcalib_interface_t::bbcalib_interface_t (`

```
    MHA_AC::algo_comm_t & iac,
  const std::string & configured_name )
```

5.62.1.2 ∼bbcalib_interface_t() `bbcalib_interface_t::∼bbcalib_interface_t ( )`

5.62.2 Member Function Documentation

5.62.2.1 process() `mha_wave_t * bbcalib_interface_t::process (`

```
    mha_wave_t * s )
```

5.62.2.2 prepare() `void bbcalib_interface_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.62.2.3 release() `void bbcalib_interface_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.62.3 Member Data Documentation

5.62.3.1 calib_in `calibrator_t bbcalib_interface_t::calib_in [private]`

5.62.3.2 calib_out `calibrator_t bbcalib_interface_t::calib_out [private]`

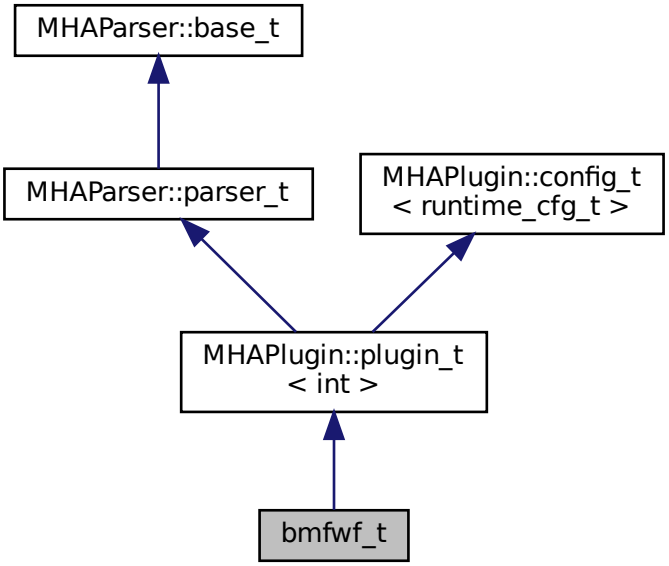
5.62.3.3 plugloader `MHAParser::mhapluginloader_t bbcalib_interface_t::plugloader [private]`

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

- **transducers.cpp**

5.63 bmfwf_t Class Reference

Inheritance diagram for bmfwf_t:



Public Member Functions

- **bmfwf_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &signal_info)

  Plugin preparation.
- void **release** (void)
- **mha_spec_t** * **process** (**mha_spec_t** *signal)

  Signal processing performed by the plugin.

Private Member Functions

- void **load_model** ()

Private Attributes

- **MHAParser::string_t model_file**
- **MHAParser::int_mon_t prepared**
  Keep Track of the prepare/release calls.
- **MHAParser::float_t scaling_factor**
- **MHAParser::float_t unscaling_ratio**
- **MHAParser::float_t mix_back**
- **MHAParser::float_t calib_factor**
- int64_t **frame_index**
- float_t **unscaling_factor**
- float_t **calib_factor_lin**
- torch::Tensor **noisy_frame**
- torch::Tensor **noisy_frame_real**
- torch::Tensor **noisy_frame_imag**
- torch::Tensor **output_tensor**
- torch::jit::script::Module **model**
- **MHASignal::spectrum_t** $*$ **out**

Additional Inherited Members

5.63.1 Constructor & Destructor Documentation

5.63.1.1 bmfwf_t()

```
bmfwf_t::bmfwf_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.63.2 Member Function Documentation

5.63.2.1 prepare()

```
void bmfwf_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

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. 1301).

5.63.2.2 release() `void bmfwf_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.63.2.3 process() `mha_spec_t * bmfwf_t::process ( mha_spec_t * signal )`

Signal processing performed by the plugin.

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)

5.63.2.4 load_model() `void bmfwf_t::load_model ( ) [private]`

5.63.3 Member Data Documentation

5.63.3.1 model_file `MHAParser::string_t bmfwf_t::model_file [private]`

5.63.3.2 prepared `MHAParser::int_mon_t bmfwf_t::prepared [private]`

Keep Track of the prepare/release calls.

5.63.3.3 scaling_factor `MHAParser::float_t bmfwf_t::scaling_factor [private]`

5.63.3.4 unscaling_ratio `MHAParser::float_t bmfwf_t::unscaling_ratio [private]`

5.63.3.5 mix_back `MHAParser::float_t bmfwf_t::mix_back [private]`

5.63.3.6 calib_factor `MHAParser::float_t bmfwf_t::calib_factor [private]`

5.63.3.7 frame_index `int64_t bmfwf_t::frame_index [private]`

5.63.3.8 unscaling_factor `float_t bmfwf_t::unscaling_factor [private]`

5.63.3.9 calib_factor_lin `float_t bmfwf_t::calib_factor_lin [private]`

5.63.3.10 noisy_frame `torch::Tensor bmfwf_t::noisy_frame [private]`

5.63.3.11 noisy_frame_real `torch::Tensor bmfwf_t::noisy_frame_real [private]`

5.63.3.12 noisy_frame_imag `torch::Tensor bmfwf_t::noisy_frame_imag [private]`

5.63.3.13 output_tensor `torch::Tensor bmfwf_t::output_tensor [private]`

5.63.3.14 model `torch::jit::script::Module bmfwf_t::model [private]`

5.63.3.15 out `MHASignal::spectrum_t* bmfwf_t::out [private]`

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

- **bmfwf.cpp**

5.64 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**

  The softclipper, only used when b_is_input == false.
- 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**

5.64.1 Constructor & Destructor Documentation

5.64.1.1 calibrator_runtime_layer_t()

```
calibrator_runtime_layer_t::calibrator_runtime_layer_t (
            bool is_input,
            const  mhaconfig_t & tf,
             calibrator_variables_t & vars )
```

5.64.2 Member Function Documentation

5.64.2.1 process()

```
mha_real_t calibrator_runtime_layer_t::process (
             mha_wave_t ** s )
```

5.64.2.2 firfirlen()

```
unsigned int calibrator_runtime_layer_t::firfirlen (
            const std::vector< std::vector< float > > & fir ) [static], [private]
```

5.64.2.3 firfir2fftlen()

```
unsigned int calibrator_runtime_layer_t::firfir2fftlen (
            unsigned int fragsize,
            const std::vector< std::vector< float > > & fir ) [static], [private]
```

5.64.3 Member Data Documentation

5.64.3.1 fir

`MHAFilter::fftfilter_t calibrator_runtime_layer_t::fir [private]`

5.64.3.2 quant

`MHASignal::quantizer_t calibrator_runtime_layer_t::quant [private]`

5.64.3.3 gain

`MHASignal::waveform_t calibrator_runtime_layer_t::gain [private]`

5.64.3.4 softclip

`softclipper_t calibrator_runtime_layer_t::softclip [private]`

The softclipper, only used when b_is_input == false.

5.64.3.5 b_is_input

`bool calibrator_runtime_layer_t::b_is_input [private]`

5.64.3.6 b_use_fir

`bool calibrator_runtime_layer_t::b_use_fir [private]`

5.64.3.7 b_use_clipping

`bool calibrator_runtime_layer_t::b_use_clipping [private]`

5.64.3.8 speechnoise

`MHASignal::loop_wavefragment_t calibrator_runtime_layer_t::speechnoise [private]`

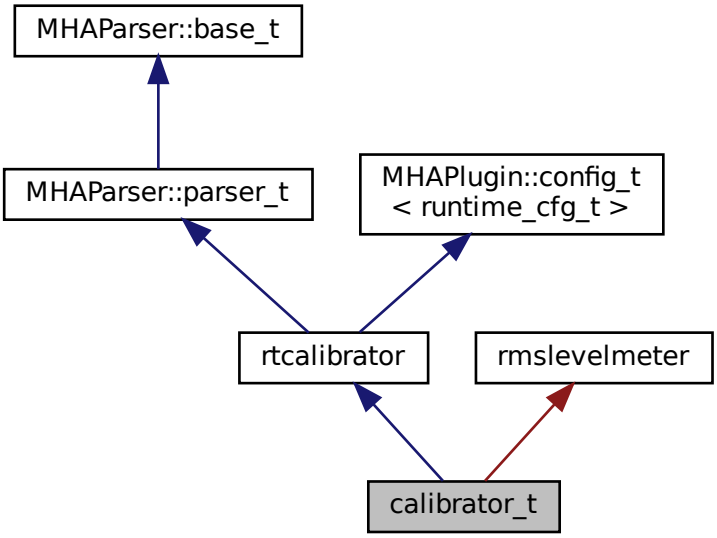
5.64.3.9 pmode `MHASignal::loop_wavefragment_t::playback_mode_t calibrator_↩ runtime_layer_t::pmode [private]`

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

- **transducers.cpp**

5.65 calibrator_t Class Reference

Inheritance diagram for calibrator_t:



Public Member Functions

- **calibrator_t** (**MHA_AC::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

5.65.1 Constructor & Destructor Documentation

5.65.1.1 calibrator_t() `calibrator_t::calibrator_t (`

```
MHA_AC::algo_comm_t & iac,
bool is_input )
```

5.65.2 Member Function Documentation

5.65.2.1 prepare() `void calibrator_t::prepare (`

```
mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 1301).

5.65.2.2 release() `void calibrator_t::release (`

```
void ) [inline], [virtual]
```

Reimplemented from **MHAPlugin::plugin_t**< **runtime_cfg_t** > (p. 1302).

5.65.2.3 process() `mha_wave_t * calibrator_t::process (`

```
mha_wave_t * s )
```

5.65.2.4 update() void calibrator_t::update ( ) [private]

5.65.2.5 update_tau_level() void calibrator_t::update_tau_level ( ) [private]

5.65.2.6 read_levels() void calibrator_t::read_levels ( ) [private]

5.65.3 Member Data Documentation

5.65.3.1 b_is_input bool calibrator_t::b_is_input [private]

5.65.3.2 patchbay **MHAEvents::patchbay_t**< **calibrator_t**> calibrator_t::patchbay [private]

5.65.3.3 vars **calibrator_variables_t** calibrator_t::vars [private]

5.65.3.4 prepared bool calibrator_t::prepared [private]

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

- **transducers.cpp**

5.66 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**

5.66.1 Constructor & Destructor Documentation

5.66.1.1 calibrator_variables_t()

```
calibrator_variables_t::calibrator_variables_t (
    bool is_input,
    MHAParser::parser_t & parent )
```

5.66.2 Member Data Documentation

5.66.2.1 peaklevel

`MHAParser::vfloat_t calibrator_variables_t::peaklevel`

5.66.2.2 fir

`MHAParser::mfloat_t calibrator_variables_t::fir`

5.66.2.3 nbits **MHAParser::int_t** calibrator_variables_t::nbits

5.66.2.4 tau_level **MHAParser::float_t** calibrator_variables_t::tau_level

5.66.2.5 spnoise_mode **MHAParser::kw_t** calibrator_variables_t::spnoise_mode

5.66.2.6 spnoise_channels **MHAParser::vint_t** calibrator_variables_t::spnoise_↩ channels

5.66.2.7 spnoise_level **MHAParser::float_t** calibrator_variables_t::spnoise_level

5.66.2.8 rmslevel **MHAParser::vfloat_mon_t** calibrator_variables_t::rmslevel

5.66.2.9 spnoise_parser **MHAParser::parser_t** calibrator_variables_t::spnoise_↩ parser

5.66.2.10 srate **MHAParser::float_mon_t** calibrator_variables_t::srate

5.66.2.11 fragsize **MHAParser::int_mon_t** calibrator_variables_t::fragsize

5.66.2.12 num_channels `MHAParser::int_mon_t calibrator_variables_t::num_channels`

5.66.2.13 config_parser `MHAParser::parser_t calibrator_variables_t::config_parser`

5.66.2.14 softclip `softclipper_variables_t calibrator_variables_t::softclip`

5.66.2.15 do_clipping `MHAParser::bool_t calibrator_variables_t::do_clipping`

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

- **transducers.cpp**

5.67 cfg_t Class Reference

Public Member Functions

- **cfg_t** (unsigned int ichannel, unsigned int numchannels, const **mha_complex_t** &iscale)
- **cfg_t** (unsigned int, unsigned int)
- **cfg_t** (**mhaconfig_t** chcfg, **mha_real_t** newlev, bool replace, **mha_real_t** len, int seed)
- 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_complex_t scale**
- **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**
- std::default_random_engine **rng**
- std::uniform_real_distribution< **mha_real_t** > **rand_dist**

5.67.1 Constructor & Destructor Documentation

5.67.1.1 cfg_t() [1/4]

```
cfg_t::cfg_t (
    unsigned int ichannel,
    unsigned int numchannels,
    const  mha_complex_t & iscale )
```

5.67.1.2 cfg_t() [2/4]

```
cfg_t::cfg_t (
    unsigned int ichannel,
    unsigned int numchannels )
```

5.67.1.3 cfg_t() [3/4]

```
cfg_t::cfg_t (
    mhaconfig_t chcfg,
    mha_real_t newlev,
    bool replace,
    mha_real_t len,
    int seed )
```

5.67.1.4 cfg_t() [4/4]

```
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 )
```

5.67.2 Member Function Documentation

5.67.2.1 process() [1/2] void cfg_t::process (mha_wave_t * s) [inline]

5.67.2.2 process() [2/2] void cfg_t::process (mha_spec_t * s) [inline]

5.67.3 Member Data Documentation

5.67.3.1 channel unsigned int cfg_t::channel

5.67.3.2 scale mha_complex_t cfg_t::scale

5.67.3.3 gain_wave_ mha_real_t cfg_t::gain_wave_ [private]

5.67.3.4 gain_spec_ mha_real_t cfg_t::gain_spec_ [private]

5.67.3.5 replace_ bool cfg_t::replace_ [private]

5.67.3.6 use_frozen_ bool cfg_t::use_frozen_ [private]

5.67.3.7 frozen_noise_ MHASignal::waveform_t cfg_t::frozen_noise_ [private]

5.67.3.8 pos unsigned int cfg_t::pos [private]

5.67.3.9 rng std::default_random_engine cfg_t::rng [private]

5.67.3.10 rand_dist std::uniform_real_distribution< mha_real_t> cfg_t::rand_dist [private]

5.67.3.11 start_lin mha_real_t cfg_t::start_lin

5.67.3.12 alpha mha_real_t cfg_t::alpha

5.67.3.13 attack MHAFilter::o1flt_lowpass_t cfg_t::attack

5.67.3.14 decay MHAFilter::o1flt_maxtrack_t cfg_t::decay

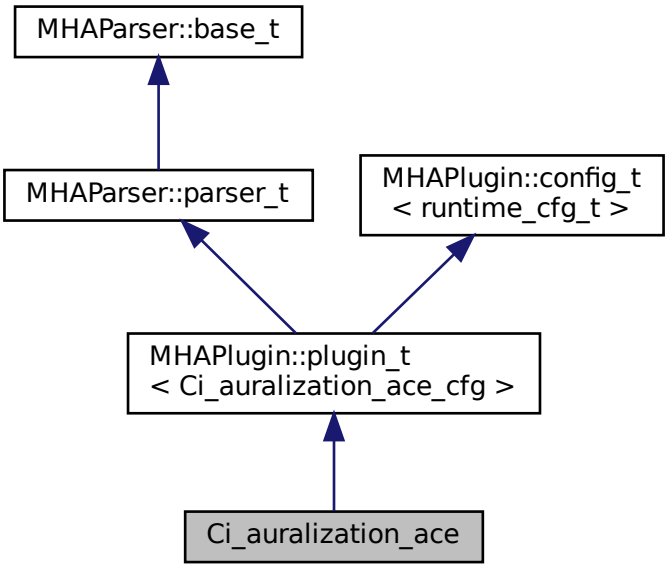
The documentation for this class was generated from the following files:

- **complex_scale_channel.cpp**
- **example6.cpp**
- **noise.cpp**
- **softclip.cpp**

5.68 Ci_auralization_ace Class Reference

Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

Inheritance diagram for Ci_auralization_ace:



Public Member Functions

- **Ci_auralization_ace** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**

  Name of the algorithm within the plugin chain.
- **MHAParser::string_t ac_name**

  Name of the AC variable containing the electrodogram (cannot be changed at runtime)
- **MHAParser::float_t compression_coefficient**

  Compression coefficient of the loudness growth function.
- **MHAParser::float_t base_level**

  Base level of the input (acoustic) dynamic range / Pa.
- **MHAParser::float_t saturation_level**

  Saturation level of the input (acoustic) dynamic range / Pa.
- **MHAParser::vfloat_t threshold_level**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode.
- **MHAParser::vfloat_t comfort_level**

  Vector containing the comfort level of the output (electric) dynamic range for each electrode.
- **MHAParser::float_t electrode_distance**

  Distance of the electrodes / m.
- **MHAParser::float_t lambda**

  Length constant of exponential spread of excitation / m.
- **MHAParser::float_t phase_duration**

  Duration of one phase of a biphasic pulse / s.
- **MHAParser::float_t interphase_gap**

  Duration of the gap between the phases of a biphasic pulse / s.
- **MHAParser::kw_t phase_order**

  Order of the phases of a biphasic pulse.
- **MHAEvents::patchbay_t**< **Ci_auralization_ace** > **patchbay**

  Data member connecting an event emitter (i.e.
- **mha_wave_t electrodogram**

  Electrodogram (contains a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrrespondes to the temporal sequence of their activation)
- unsigned int **n_electrodes**

  Number of active electrodes.
- unsigned int **m_electrodes**

  Total number of electrodes per side.
- **MHA_AC::waveform_t** * **stimulation_signal_ac**

  AC variable containing a pointer to the stimulation signal (containing a matrix with dimensions m x fragsize, where m = total number of electrodes and fragsize = fragment size / samples, representing the actual electrical stimulation per electrode over time for the current fragment)

Additional Inherited Members

5.68.1 Detailed Description

Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

5.68.2 Constructor & Destructor Documentation

5.68.2.1 Ci_auralization_ace() `Ci_auralization_ace::Ci_auralization_ace (`

```
algo_comm_t & ac,
const std::string & algo_name )
```

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.68.3 Member Function Documentation

5.68.3.1 prepare() `void Ci_auralization_ace::prepare (`

```
mhaconfig_t & signal_info ) [virtual]
```

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Ci_auralization_ace_cfg >** (p. 1301).

5.68.3.2 process() `mha_wave_t * Ci_auralization_ace::process ( mha_wave_t * signal )`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.68.3.3 release() `void Ci_auralization_ace::release ( void ) [virtual]`

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Ci_auralization_ace_cfg >** (p. 1302).

5.68.3.4 update_cfg() `void Ci_auralization_ace::update_cfg ( ) [private]`

Runtime configuration update function of the plugin interface class.

5.68.4 Member Data Documentation

5.68.4.1 algo_name `std::string Ci_auralization_ace::algo_name [private]`

Name of the algorithm within the plugin chain.

5.68.4.2 ac_name

`MHAParser::string_t Ci_auralization_ace::ac_name [private]`

Name of the AC variable containing the electrodogram (cannot be changed at runtime)

5.68.4.3 compression_coefficient

`MHAParser::float_t Ci_auralization_ace::compression_coefficient [private]`

Compression coefficient of the loudness growth function.

5.68.4.4 base_level

`MHAParser::float_t Ci_auralization_ace::base_level [private]`

Base level of the input (acoustic) dynamic range / Pa.

5.68.4.5 saturation_level

`MHAParser::float_t Ci_auralization_ace::saturation_level [private]`

Saturation level of the input (acoustic) dynamic range / Pa.

5.68.4.6 threshold_level

`MHAParser::vfloat_t Ci_auralization_ace::threshold_level [private]`

Vector containing the threshold level of the output (electric) dynamic range for each electrode.

5.68.4.7 comfort_level

`MHAParser::vfloat_t Ci_auralization_ace::comfort_level [private]`

Vector containing the comfort level of the output (electric) dynamic range for each electrode.

5.68.4.8 electrode_distance

`MHAParser::float_t Ci_auralization_ace::electrode_↩ distance [private]`

Distance of the electrodes / m.

5.68.4.9 lambda

`MHAParser::float_t Ci_auralization_ace::lambda [private]`

Length constant of exponential spread of excitation / m.

5.68.4.10 phase_duration

`MHAParser::float_t Ci_auralization_ace::phase_duration [private]`

Duration of one phase of a biphasic pulse / s.

5.68.4.11 interphase_gap

`MHAParser::float_t Ci_auralization_ace::interphase_gap [private]`

Duration of the gap between the phases of a biphasic pulse / s.

5.68.4.12 phase_order

`MHAParser::kw_t Ci_auralization_ace::phase_order [private]`

Order of the phases of a biphasic pulse.

5.68.4.13 patchbay

`MHAEvents::patchbay_t< Ci_auralization_ace> Ci_auralization↩ _ace::patchbay [private]`

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

5.68.4.14 electrodogram `mha_wave_t Ci_auralization_ace::electrodogram [private]`

Electrodogram (contains a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrresponddes to the temporal sequence of their activation)

5.68.4.15 n_electrodes `unsigned int Ci_auralization_ace::n_electrodes [private]`

Number of active electrodes.

5.68.4.16 m_electrodes `unsigned int Ci_auralization_ace::m_electrodes [private]`

Total number of electrodes per side.

5.68.4.17 stimulation_signal_ac `MHA_AC::waveform_t* Ci_auralization_ace::stimulation_signal_ac [private]`

AC variable containing a pointer to the stimulation signal (containing a matrix with dimensions m x fragsize, where m = total number of electrodes and fragsize = fragment size / samples, representing the actual electrical stimulation per electrode over time for the current fragment)

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

- **ci_auralization_ace.hh**
- **ci_auralization_ace.cpp**

5.69 Ci_auralization_ace_cfg Class Reference

Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

Public Member Functions

- **Ci_auralization_ace_cfg** (std::string ac_name, **mha_real_t** compression_coefficient, **mha_real_t** base_level, **mha_real_t** saturation_level, std::vector< **mha_real_t** > threshold_level, std::vector< **mha_real_t** > comfort_level, **mha_real_t** electrode_↩ distance, **mha_real_t** lambda, **mha_real_t** phase_duration, **mha_real_t** interphase_↩ gap, unsigned int phase_order, **mha_wave_t** electrodogram, unsigned int n_electrodes, unsigned int m_electrodes, **Ci_auralization_ace** ∗ **Ci_auralization_ace**)

  Constructor of the runtime configuration class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal, **Ci_auralization_ace** ∗ **Ci_↩ auralization_ace**, **MHA_AC::waveform_t** ∗stimulation_signal_ac)

  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- std::string **ac_name_cfg**

  Name of the AC variable containing the electrodogram (cannot be changed at runtime)
- **mha_real_t compression_coefficient_cfg**

  Compression coefficient of the loudness growth function.
- **mha_real_t base_level_cfg**

  Base level of the input (acoustic) dynamic range / Pa.
- **mha_real_t saturation_level_cfg**

  Saturation level of the input (acoustic) dynamic range / Pa.
- std::vector< **mha_real_t** > **threshold_level_cfg**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode.
- std::vector< **mha_real_t** > **comfort_level_cfg**

  Vector containing the comfort level of the output (electric) dynamic range for each electrode.
- **mha_real_t electrode_distance_cfg**

  Distance of the electrodes / m.
- **mha_real_t lambda_cfg**

  Length constant of exponential spread of excitation / m.
- **mha_real_t phase_duration_cfg**

  Duration of one phase of a biphasic pulse / s.
- **mha_real_t interphase_gap_cfg**

  Duration of the gap between the phases of a biphasic pulse / s.
- unsigned int **phase_order_cfg**

  Order of the phases of a biphasic pulse.
- **mha_wave_t electrodogram_cfg**

  Electrodogram (contains a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrrespondes to the temporal sequence of their activation)
- unsigned int **n_electrodes_cfg**

  Number of active electrodes.
- unsigned int **m_electrodes_cfg**

  Total number of electrodes per side.

5.69.1 Detailed Description

Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

5.69.2 Constructor & Destructor Documentation

5.69.2.1 Ci_auralization_ace_cfg() `Ci_auralization_ace_cfg::Ci_auralization_ace_cfg`

```
(
            std::string ac_name,
             mha_real_t compression_coefficient,
             mha_real_t base_level,
             mha_real_t saturation_level,
            std::vector<  mha_real_t > threshold_level,
            std::vector<  mha_real_t > comfort_level,
             mha_real_t electrode_distance,
             mha_real_t lambda,
             mha_real_t phase_duration,
             mha_real_t interphase_gap,
            unsigned int phase_order,
             mha_wave_t electrodogram,
            unsigned int n_electrodes,
            unsigned int m_electrodes,
             Ci_auralization_ace * Ci_auralization_ace )
```

Constructor of the runtime configuration class.

Parameters

ac_name	Name of the AC variable containing the ele (cannot be changed at runtime)
compression_coefficient	Compression coefficient of the loudness gr
base_level	Base level of the input (acoustic) dynamic r
saturation_level	Saturation level of the input (acoustic) dyna
threshold_level	Vector containing the threshold level of the dynamic range for each electrode
comfort_level	Vector containing the comfort level of the ou dynamic range for each electrode
electrode_distance	Distance of the electrodes / m
lambda	Length constant of exponential spread of ex
phase_duration	Duration of one phase of a biphasic pulse /

Parameters

interphase_gap	Duration of the gap between the phases of s
phase_order	Order of the phases of a biphasic pulse
electrodogram	Electrodogram (contains a matrix with dime where m = total number of electrodes and r active electrodes, and where the order of th electrodes corrrespondes to the temporal s activation)
n_electrodes	Number of active electrodes
m_electrodes	Total number of electrodes per side
Ci_auralization_ace *(p. 375)*	Pointer to the current instance of the plugin

5.69.3 Member Function Documentation

5.69.3.1 process()

```
mha_wave_t * Ci_auralization_ace_cfg::process (
    mha_wave_t * signal,
    Ci_auralization_ace * Ci_auralization_ace,
    MHA_AC::waveform_t * stimulation_signal_ac )
```

Process function of the runtime configuration class (main signal processing function).

It leaves the input signal fragment unchanged; its purpose is to compute the AC variable stimulation_signal_ac, which contains all the electrode-specific information (including concrete time specifications) necessary for auralization, for further processing in the plugin chain

Parameters

signal	Pointer to the current input signal fragment
Ci_auralization_ace *(p. 375)*	Pointer to the current instance of the plugin interface class
stimulation_signal_ac	AC variable containing a pointer to the stimulation signal

Returns

Pointer to the (unchanged) output signal fragment

5.69.4 Member Data Documentation

5.69.4.1 ac_name_cfg

`std::string Ci_auralization_ace_cfg::ac_name_cfg [private]`

Name of the AC variable containing the electrodogram (cannot be changed at runtime)

5.69.4.2 compression_coefficient_cfg

`mha_real_t Ci_auralization_ace_cfg::compression_coefficient_cfg [private]`

Compression coefficient of the loudness growth function.

5.69.4.3 base_level_cfg

`mha_real_t Ci_auralization_ace_cfg::base_level_cfg [private]`

Base level of the input (acoustic) dynamic range / Pa.

5.69.4.4 saturation_level_cfg

`mha_real_t Ci_auralization_ace_cfg::saturation_level_cfg [private]`

Saturation level of the input (acoustic) dynamic range / Pa.

5.69.4.5 threshold_level_cfg

`std::vector< mha_real_t> Ci_auralization_ace_cfg::threshold_level_cfg [private]`

Vector containing the threshold level of the output (electric) dynamic range for each electrode.

5.69.4.6 comfort_level_cfg

`std::vector< mha_real_t> Ci_auralization_ace_cfg::comfort_level_cfg [private]`

Vector containing the comfort level of the output (electric) dynamic range for each electrode.

5.69.4.7 electrode_distance_cfg

mha_real_t Ci_auralization_ace_cfg::electrode_↩distance_cfg [private]

Distance of the electrodes / m.

5.69.4.8 lambda_cfg

mha_real_t Ci_auralization_ace_cfg::lambda_cfg [private]

Length constant of exponential spread of excitation / m.

5.69.4.9 phase_duration_cfg

mha_real_t Ci_auralization_ace_cfg::phase_duration_↩cfg [private]

Duration of one phase of a biphasic pulse / s.

5.69.4.10 interphase_gap_cfg

mha_real_t Ci_auralization_ace_cfg::interphase_gap↩_cfg [private]

Duration of the gap between the phases of a biphasic pulse / s.

5.69.4.11 phase_order_cfg

unsigned int Ci_auralization_ace_cfg::phase_order_cfg [private]

Order of the phases of a biphasic pulse.

5.69.4.12 electrodogram_cfg

mha_wave_t Ci_auralization_ace_cfg::electrodogram_↩cfg [private]

Electrodogram (contains a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrresponndes to the temporal sequence of their activation)

5.69.4.13 n_electrodes_cfg `unsigned int Ci_auralization_ace_cfg::n_electrodes_cfg [private]`

Number of active electrodes.

5.69.4.14 m_electrodes_cfg `unsigned int Ci_auralization_ace_cfg::m_electrodes_cfg [private]`

Total number of electrodes per side.

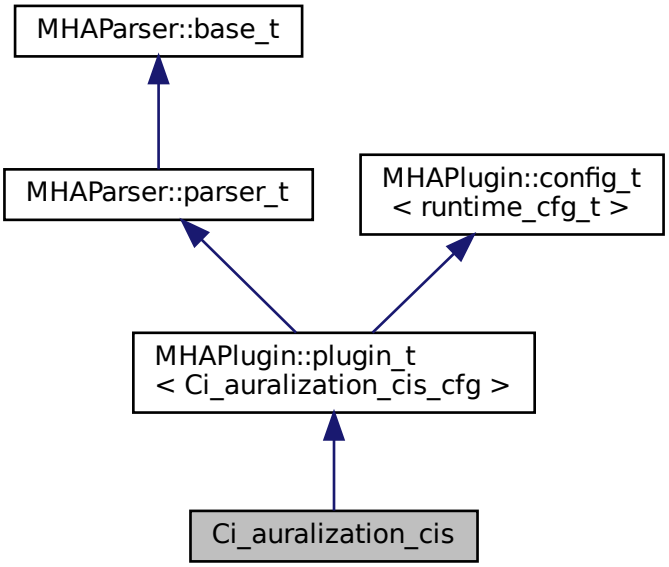
The documentation for this class was generated from the following files:

- **ci_auralization_ace.hh**
- **ci_auralization_ace.cpp**

5.70 Ci_auralization_cis Class Reference

Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

Inheritance diagram for Ci_auralization_cis:



Public Member Functions

- **Ci_auralization_cis** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**
  Name of the algorithm within the plugin chain.
- **MHAParser::string_t ac_name**
  Name of the AC variable containing the electrodogram (cannot be changed at runtime)
- **MHAParser::float_t compression_coefficient**
  Compression coefficient of the loudness growth function.
- **MHAParser::float_t base_level**
  Base level of the input (acoustic) dynamic range / Pa.
- **MHAParser::float_t saturation_level**
  Saturation level of the input (acoustic) dynamic range / Pa.
- **MHAParser::vfloat_t threshold_level**
  Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.
- **MHAParser::vfloat_t maximum_comfortable_level**
  Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.
- **MHAParser::float_t electrode_distance**
  Distance of the electrodes / m.
- **MHAParser::float_t lambda**
  Length constant of exponential spread of excitation / m.
- **MHAParser::float_t phase_duration**
  Duration of one phase of a biphasic pulse / s.
- **MHAParser::float_t interphase_gap**
  Duration of the gap between the phases of a biphasic pulse / s.
- **MHAParser::kw_t phase_order**
  Order of the phases of a biphasic pulse.

- **MHAEvents::patchbay_t< Ci_auralization_cis > patchbay**

  Data member connecting an event emitter (i.e.
- **mha_wave_t electrodogram**

  Electrodogram (contains a matrix with dimensions m x m, where m = total number of electrodes n = number of active electrodes, and where the order of the n active electrodes corrrespondes to the temporal sequence of their activation)
- unsigned int **n_electrodes**

  Number of active electrodes.
- unsigned int **m_electrodes**

  Total number of electrodes per side.
- **MHA_AC::waveform_t * stimulation_signal_ac**

  AC variable containing a pointer to the stimulation signal (containing a matrix with dimensions m x fragsize, where m = total number of electrodes and fragsize = fragment size / samples, representing the actual electrical stimulation per electrode over time for the current fragment)

Additional Inherited Members

5.70.1 Detailed Description

Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

5.70.2 Constructor & Destructor Documentation

5.70.2.1 Ci_auralization_cis() `Ci_auralization_cis::Ci_auralization_cis (`

```
    algo_comm_t & ac,
  const std::string & algo_name )
```

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.70.3 Member Function Documentation

5.70.3.1 prepare() `void Ci_auralization_cis::prepare ( mhaconfig_t & signal_info ) [virtual]`

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Ci_auralization_cis_cfg >** (p. 1301).

5.70.3.2 process() `mha_wave_t * Ci_auralization_cis::process ( mha_wave_t * signal )`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.70.3.3 release() `void Ci_auralization_cis::release ( void ) [virtual]`

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Ci_auralization_cis_cfg >** (p. 1302).

5.70.3.4 update_cfg() void Ci_auralization_cis::update_cfg ( ) [private]

Runtime configuration update function of the plugin interface class.

5.70.4 Member Data Documentation

5.70.4.1 algo_name std::string Ci_auralization_cis::algo_name [private]

Name of the algorithm within the plugin chain.

5.70.4.2 ac_name MHAParser::string_t Ci_auralization_cis::ac_name [private]

Name of the AC variable containing the electrodogram (cannot be changed at runtime)

5.70.4.3 compression_coefficient MHAParser::float_t Ci_auralization_cis::compression↩ _coefficient [private]

Compression coefficient of the loudness growth function.

5.70.4.4 base_level MHAParser::float_t Ci_auralization_cis::base_level [private]

Base level of the input (acoustic) dynamic range / Pa.

5.70.4.5 saturation_level MHAParser::float_t Ci_auralization_cis::saturation_level [private]

Saturation level of the input (acoustic) dynamic range / Pa.

5.70.4.6 threshold_level

MHAParser::vfloat_t Ci_auralization_cis::threshold_level [private]

Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.

5.70.4.7 maximum_comfortable_level

MHAParser::vfloat_t Ci_auralization_cis↩ ::maximum_comfortable_level [private]

Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.

5.70.4.8 electrode_distance

MHAParser::float_t Ci_auralization_cis::electrode_↩ distance [private]

Distance of the electrodes / m.

5.70.4.9 lambda

MHAParser::float_t Ci_auralization_cis::lambda [private]

Length constant of exponential spread of excitation / m.

5.70.4.10 phase_duration

MHAParser::float_t Ci_auralization_cis::phase_duration [private]

Duration of one phase of a biphasic pulse / s.

5.70.4.11 interphase_gap

MHAParser::float_t Ci_auralization_cis::interphase_gap [private]

Duration of the gap between the phases of a biphasic pulse / s.

5.70.4.12 phase_order

MHAParser::kw_t Ci_auralization_cis::phase_order [private]

Order of the phases of a biphasic pulse.

5.70.4.13 patchbay

MHAEvents::patchbay_t< **Ci_auralization_cis**> Ci_auralization_cis::patchbay [private]

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

5.70.4.14 electrodogram

mha_wave_t Ci_auralization_cis::electrodogram [private]

Electrodogram (contains a matrix with dimensions m x m, where m = total number of electrodes n = number of active electrodes, and where the order of the n active electrodes corrrespondes to the temporal sequence of their activation)

5.70.4.15 n_electrodes

unsigned int Ci_auralization_cis::n_electrodes [private]

Number of active electrodes.

5.70.4.16 m_electrodes

unsigned int Ci_auralization_cis::m_electrodes [private]

Total number of electrodes per side.

5.70.4.17 stimulation_signal_ac

MHA_AC::waveform_t∗ Ci_auralization_cis::stimulation_signal_ac [private]

AC variable containing a pointer to the stimulation signal (containing a matrix with dimensions m x fragsize, where m = total number of electrodes and fragsize = fragment size / samples, representing the actual electrical stimulation per electrode over time for the current fragment)

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

- **ci_auralization_cis.hh**
- **ci_auralization_cis.cpp**

5.71 Ci_auralization_cis_cfg Class Reference

Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

Public Member Functions

- **Ci_auralization_cis_cfg** (std::string ac_name, **mha_real_t** steepness, **mha_real_t** base_level, **mha_real_t** saturation_level, std::vector< **mha_real_t** > threshold_level, std::vector< **mha_real_t** > comfort_level, **mha_real_t** electrode_distance, **mha_real_t** lambda, **mha_real_t** phase_duration, **mha_real_t** interphase_gap, unsigned int phase_order, **mha_wave_t** electrodogram, unsigned int n_electrodes, unsigned int m_electrodes, **Ci_auralization_cis** * **Ci_auralization_cis**)

  Constructor of the runtime configuration class.
- **mha_wave_t** * **process** (**mha_wave_t** *signal, **Ci_auralization_cis** * **Ci_auralization_cis**, **MHA_AC::waveform_t** *stimulation_signal_ac)

  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- std::string **ac_name_cfg**

  Name of the AC variable containing the electrodogram (cannot be changed at runtime)
- **mha_real_t compression_coefficient_cfg**

  Compression coefficient of the loudness growth function.
- **mha_real_t base_level_cfg**

  Base level of the input (acoustic) dynamic range / Pa.
- **mha_real_t saturation_level_cfg**

  Saturation level of the input (acoustic) dynamic range / Pa.
- std::vector< **mha_real_t** > **threshold_level_cfg**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.
- std::vector< **mha_real_t** > **maximum_comfortable_level_cfg**

  Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.
- **mha_real_t electrode_distance_cfg**

  Distance of the electrodes / m.
- **mha_real_t lambda_cfg**

  Length constant of exponential spread of excitation / m.
- **mha_real_t phase_duration_cfg**

  Duration of one phase of a biphasic pulse / s.
- **mha_real_t interphase_gap_cfg**

  Duration of the gap between the phases of a biphasic pulse / s.
- unsigned int **phase_order_cfg**

Order of the phases of a biphasic pulse.

- **mha_wave_t electrodogram_cfg**

  Electrodogram (contains a matrix with dimensions m x m, where m = total number of electrodes = number of active electrodes, and where the order of the active electrodes corrresponds to the temporal sequence of their activation)

- unsigned int **n_electrodes_cfg**

  Number of active electrodes.

- unsigned int **m_electrodes_cfg**

  Total number of electrodes per side.

5.71.1 Detailed Description

Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

5.71.2 Constructor & Destructor Documentation

5.71.2.1 Ci_auralization_cis_cfg() `Ci_auralization_cis_cfg::Ci_auralization_cis_cfg`

```
(
            std::string ac_name,
             mha_real_t steepness,
             mha_real_t base_level,
             mha_real_t saturation_level,
            std::vector<  mha_real_t > threshold_level,
            std::vector<  mha_real_t > comfort_level,
             mha_real_t electrode_distance,
             mha_real_t lambda,
             mha_real_t phase_duration,
             mha_real_t interphase_gap,
            unsigned int phase_order,
             mha_wave_t electrodogram,
            unsigned int n_electrodes,
            unsigned int m_electrodes,
             Ci_auralization_cis * Ci_auralization_cis )
```

Constructor of the runtime configuration class.

Parameters

ac_name	Name of the AC variable containing the elec (cannot be changed at runtime)

Parameters

compression_coefficient	Compression coefficient of the loudness gro
base_level	Base level of the input (acoustic) dynamic ra
saturation_level	Saturation level of the input (acoustic) dyna
threshold_level	Vector containing the threshold level of the dynamic range for each electrode / cu
maximum_comfortable_level	Vector containing the maximum comfortable output (electric) dynamic range for each ele
electrode_distance	Distance of the electrodes / m
lambda	Length constant of exponential spread of ex
phase_duration	Duration of one phase of a biphasic pulse /
interphase_gap	Duration of the gap between the phases of s
phase_order	Order of the phases of a biphasic pulse
electrodogram	Electrodogram (contains a matrix with dime where m = total number of electrodes = num electrodes, and where the order of the activ corrresponds to the temporal sequence of t
n_electrodes	Number of active electrodes
m_electrodes	Total number of electrodes per side
Ci_auralization_cis *(p. 387)*	Pointer to the current instance of the plugin

5.71.3 Member Function Documentation

5.71.3.1 process()

```
mha_wave_t * Ci_auralization_cis_cfg::process (
    mha_wave_t * signal,
    Ci_auralization_cis * Ci_auralization_cis,
    MHA_AC::waveform_t * stimulation_signal_ac )
```

Process function of the runtime configuration class (main signal processing function).

It leaves the input signal fragment unchanged; its purpose is to compute the AC variable stimulation_signal_ac, which contains all the electrode-specific information (including concrete time specifications) necessary for auralization, for further processing in the plugin chain

Parameters

signal	Pointer to the current input signal fragment
Ci_auralization_cis *(p. 387)*	Pointer to the current instance of the plugin interface class

Parameters

stimulation_signal_ac	AC variable containing a pointer to the stimulation signal

Returns

Pointer to the (unchanged) output signal fragment

5.71.4 Member Data Documentation

5.71.4.1 ac_name_cfg `std::string Ci_auralization_cis_cfg::ac_name_cfg [private]`

Name of the AC variable containing the electrodogram (cannot be changed at runtime)

5.71.4.2 compression_coefficient_cfg `mha_real_t Ci_auralization_cis_cfg::compression_coefficient_cfg [private]`

Compression coefficient of the loudness growth function.

5.71.4.3 base_level_cfg `mha_real_t Ci_auralization_cis_cfg::base_level_cfg [private]`

Base level of the input (acoustic) dynamic range / Pa.

5.71.4.4 saturation_level_cfg `mha_real_t Ci_auralization_cis_cfg::saturation_level_cfg [private]`

Saturation level of the input (acoustic) dynamic range / Pa.

5.71.4.5 threshold_level_cfg

`std::vector< mha_real_t> Ci_auralization_cis_cfg↩ ::threshold_level_cfg [private]`

Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.

5.71.4.6 maximum_comfortable_level_cfg

`std::vector< mha_real_t> Ci_auralization↩ _cis_cfg::maximum_comfortable_level_cfg [private]`

Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.

5.71.4.7 electrode_distance_cfg

`mha_real_t Ci_auralization_cis_cfg::electrode_↩ distance_cfg [private]`

Distance of the electrodes / m.

5.71.4.8 lambda_cfg

`mha_real_t Ci_auralization_cis_cfg::lambda_cfg [private]`

Length constant of exponential spread of excitation / m.

5.71.4.9 phase_duration_cfg

`mha_real_t Ci_auralization_cis_cfg::phase_duration_↩ cfg [private]`

Duration of one phase of a biphasic pulse / s.

5.71.4.10 interphase_gap_cfg

`mha_real_t Ci_auralization_cis_cfg::interphase_gap↩ _cfg [private]`

Duration of the gap between the phases of a biphasic pulse / s.

5.71.4.11 phase_order_cfg

`unsigned int Ci_auralization_cis_cfg::phase_order_cfg [private]`

Order of the phases of a biphasic pulse.

5.71.4.12 electrodogram_cfg

`mha_wave_t Ci_auralization_cis_cfg::electrodogram_cfg [private]`

Electrodogram (contains a matrix with dimensions m x m, where m = total number of electrodes = number of active electrodes, and where the order of the active electrodes corrresponds to the temporal sequence of their activation)

5.71.4.13 n_electrodes_cfg

`unsigned int Ci_auralization_cis_cfg::n_electrodes_cfg [private]`

Number of active electrodes.

5.71.4.14 m_electrodes_cfg

`unsigned int Ci_auralization_cis_cfg::m_electrodes_cfg [private]`

Total number of electrodes per side.

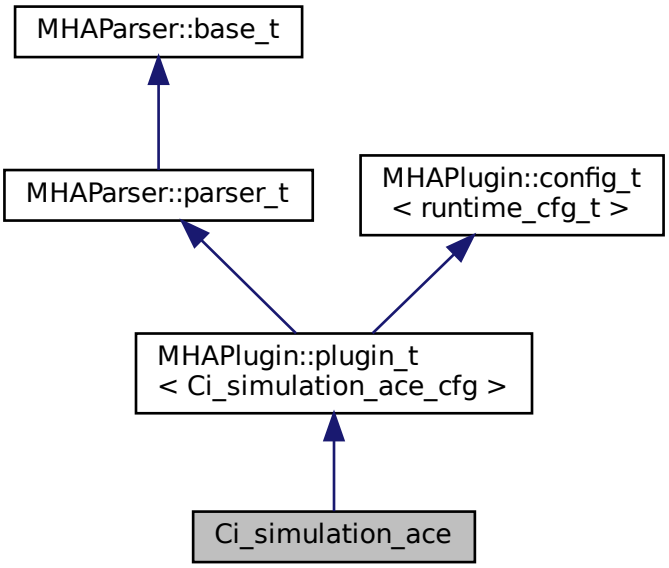
The documentation for this class was generated from the following files:

- **ci_auralization_cis.hh**
- **ci_auralization_cis.cpp**

5.72 Ci_simulation_ace Class Reference

Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

Inheritance diagram for Ci_simulation_ace:



Public Member Functions

- **Ci_simulation_ace** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_spec_t** * **process** (**mha_spec_t** *signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**

  Name of the algorithm within the plugin chain.
- **MHAParser::int_t n_electrodes**

  Number of active electrodes (cannot be changed at runtime)
- **MHAParser::float_t compression_coefficient**

  Compression coefficient of the loudness growth function.
- **MHAParser::float_t base_level**

  Base level of the input (acoustic) dynamic range / Pa.
- **MHAParser::float_t saturation_level**

  Saturation level of the input (acoustic) dynamic range / Pa.
- **MHAParser::vfloat_t threshold_level**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode / CU.
- **MHAParser::vfloat_t comfort_level**

  Vector containing the comfort level of the output (electric) dynamic range for each electrode / CU.
- **MHAParser::vint_t disabled_electrodes**

  Vector containig the indices of any disabled electrodes.
- **MHAParser::kw_t stimulation_order**

  Electrode stimulation order.
- **MHAParser::int_t randomization_seed**

  Seed for randomization of stimulation order.
- **MHAEvents::patchbay_t**< **Ci_simulation_ace** > **patchbay**

  Data member connecting an event emitter (i.e.
- std::default_random_engine **random_number_generator**

  Random number generator for randomization of stimulation order.
- **MHA_AC::waveform_t** * **electrodogram_ac**

  AC variable containing a pointer to the electrodogram (containing a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrresponds to the temporal sequence of their activation)

Additional Inherited Members

5.72.1 Detailed Description

Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

5.72.2 Constructor & Destructor Documentation

5.72.2.1 Ci_simulation_ace() `Ci_simulation_ace::Ci_simulation_ace (`

`algo_comm_t` `&` *ac,*
`const std::string &` *algo_name* `)`

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.72.3 Member Function Documentation

5.72.3.1 prepare() `void Ci_simulation_ace::prepare (`

`mhaconfig_t` `&` *signal_info* `) [virtual]`

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Ci_simulation_ace_cfg >** (p. 1301).

5.72.3.2 process() **`mha_spec_t`** `* Ci_simulation_ace::process (`

`mha_spec_t` `*` *signal* `)`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.72.3.3 release() void Ci_simulation_ace::release (void) [virtual]

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Ci_simulation_ace_cfg >** (p. 1302).

5.72.3.4 update_cfg() void Ci_simulation_ace::update_cfg ( ) [private]

Runtime configuration update function of the plugin interface class.

5.72.4 Member Data Documentation

5.72.4.1 algo_name std::string Ci_simulation_ace::algo_name [private]

Name of the algorithm within the plugin chain.

5.72.4.2 n_electrodes MHAParser::int_t Ci_simulation_ace::n_electrodes [private]

Number of active electrodes (cannot be changed at runtime)

5.72.4.3 compression_coefficient MHAParser::float_t Ci_simulation_ace::compression↩ _coefficient [private]

Compression coefficient of the loudness growth function.

5.72.4.4 base_level MHAParser::float_t Ci_simulation_ace::base_level [private]

Base level of the input (acoustic) dynamic range / Pa.

5.72.4.5 saturation_level

MHAParser::float_t Ci_simulation_ace::saturation_level [private]

Saturation level of the input (acoustic) dynamic range / Pa.

5.72.4.6 threshold_level

MHAParser::vfloat_t Ci_simulation_ace::threshold_level [private]

Vector containing the threshold level of the output (electric) dynamic range for each electrode / CU.

5.72.4.7 comfort_level

MHAParser::vfloat_t Ci_simulation_ace::comfort_level [private]

Vector containing the comfort level of the output (electric) dynamic range for each electrode / CU.

5.72.4.8 disabled_electrodes

MHAParser::vint_t Ci_simulation_ace::disabled_↩ electrodes [private]

Vector containig the indices of any disabled electrodes.

5.72.4.9 stimulation_order

MHAParser::kw_t Ci_simulation_ace::stimulation_order [private]

Electrode stimulation order.

5.72.4.10 randomization_seed

MHAParser::int_t Ci_simulation_ace::randomization↩ _seed [private]

Seed for randomization of stimulation order.

5.72.4.11 patchbay `MHAEvents::patchbay_t< Ci_simulation_ace> Ci_simulation_ace::patchbay [private]`

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

5.72.4.12 random_number_generator `std::default_random_engine Ci_simulation_ace::random_number_generator [private]`

Random number generator for randomization of stimulation order.

5.72.4.13 electrodogram_ac `MHA_AC::waveform_t* Ci_simulation_ace::electrodogram_ac [private]`

AC variable containing a pointer to the electrodogram (containing a matrix with dimensions m x n, where m = total number of electrodes and n = number of active electrodes, and where the order of the n active electrodes corrresponds to the temporal sequence of their activation)

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

- **ci_simulation_ace.hh**
- **ci_simulation_ace.cpp**

5.73 Ci_simulation_ace_cfg Class Reference

Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

Public Member Functions

- **Ci_simulation_ace_cfg** (unsigned int n_electrodes, **mha_real_t** compression_coefficient, **mha_real_t** base_level, **mha_real_t** saturation_level, std::vector< **mha_real_t** > threshold_level, std::vector< **mha_real_t** > comfort_level, std::vector< int > disabled_electrodes, unsigned int stimulation_order, std::default_random_engine random_number_generator)

  Constructor of the runtime configuration class.
- **mha_spec_t** * **process** (**mha_spec_t** *signal, **MHA_AC::waveform_t** *electrodogram_ac)

  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- unsigned int **n_electrodes_cfg**

  Number of active electrodes (cannot be changed at runtime)
- **mha_real_t compression_coefficient_cfg**

  Compression coefficient of the loudness growth function.
- **mha_real_t base_level_cfg**

  Base level of the input (acoustic) dynamic range / Pa.
- **mha_real_t saturation_level_cfg**

  Saturation level of the input (acoustic) dynamic range / Pa.
- std::vector< **mha_real_t** > **threshold_level_cfg**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode / CU.
- std::vector< **mha_real_t** > **comfort_level_cfg**

  Vector containing the comfort level of the output (electric) dynamic range for each electrode / CU.
- std::vector< int > **disabled_electrodes_cfg**

  Vector containing the indices of any disabled electrodes.
- unsigned int **stimulation_order_cfg**

  Electrode stimulation order.
- std::default_random_engine **random_number_generator_cfg**

  Random number generator for randomization of stimulation order.

5.73.1 Detailed Description

Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

5.73.2 Constructor & Destructor Documentation

5.73.2.1 Ci_simulation_ace_cfg()

```
Ci_simulation_ace_cfg::Ci_simulation_ace_cfg (
    unsigned int n_electrodes,
     mha_real_t compression_coefficient,
     mha_real_t base_level,
     mha_real_t saturation_level,
    std::vector< mha_real_t > threshold_level,
    std::vector< mha_real_t > comfort_level,
    std::vector< int > disabled_electrodes,
    unsigned int stimulation_order,
    std::default_random_engine random_number_generator )
```

Constructor of the runtime configuration class.

Parameters

n_electrodes	Number of active electrodes (cannot be chan
compression_coefficient	Compression coefficient of the loudness grow
base_level	Base level of the input (acoustic) dynamic ra
saturation_level	Saturation level of the input (acoustic) dynan
threshold_level	Vector containing the threshold level of the o dynamic range for each electrode / CU
comfort_level	Vector containing the comfort level of the out dynamic range for each electrode / CU
disabled_electrodes	Vector containing the indices of any disablec
stimulation_order	Electrode stimulation order
random_number_generator	Random number generator for randomizatio order

5.73.3 Member Function Documentation

5.73.3.1 process()

```
mha_spec_t * Ci_simulation_ace_cfg::process (
    mha_spec_t * signal,
    MHA_AC::waveform_t * electrodogram_ac )
```

Process function of the runtime configuration class (main signal processing function).

It leaves the input signal fragment unchanged; its purpose is to compute the AC variable electrodogram_ac for further processing in the plugin chain

Parameters

signal	Pointer to the current input signal fragment
electrodogram_ac	AC variable containing a pointer to the electrodogram

Returns

Pointer to the (unchanged) output signal fragment

5.73.4 Member Data Documentation

5.73.4.1 n_electrodes_cfg `unsigned int Ci_simulation_ace_cfg::n_electrodes_cfg [private]`

Number of active electrodes (cannot be changed at runtime)

5.73.4.2 compression_coefficient_cfg `mha_real_t Ci_simulation_ace_cfg::compression_coefficient_cfg [private]`

Compression coefficient of the loudness growth function.

5.73.4.3 base_level_cfg `mha_real_t Ci_simulation_ace_cfg::base_level_cfg [private]`

Base level of the input (acoustic) dynamic range / Pa.

5.73.4.4 saturation_level_cfg `mha_real_t Ci_simulation_ace_cfg::saturation_level_cfg [private]`

Saturation level of the input (acoustic) dynamic range / Pa.

5.73.4.5 threshold_level_cfg `std::vector< mha_real_t> Ci_simulation_ace_cfg::threshold_level_cfg [private]`

Vector containing the threshold level of the output (electric) dynamic range for each electrode / CU.

5.73.4.6 comfort_level_cfg

`std::vector< mha_real_t> Ci_simulation_ace_cfg::comfort_level_cfg [private]`

Vector containing the comfort level of the output (electric) dynamic range for each electrode / CU.

5.73.4.7 disabled_electrodes_cfg

`std::vector<int> Ci_simulation_ace_cfg::disabled_electrodes_cfg [private]`

Vector containing the indices of any disabled electrodes.

5.73.4.8 stimulation_order_cfg

`unsigned int Ci_simulation_ace_cfg::stimulation_order_cfg [private]`

Electrode stimulation order.

5.73.4.9 random_number_generator_cfg

`std::default_random_engine Ci_simulation_ace_cfg::random_number_generator_cfg [private]`

Random number generator for randomization of stimulation order.

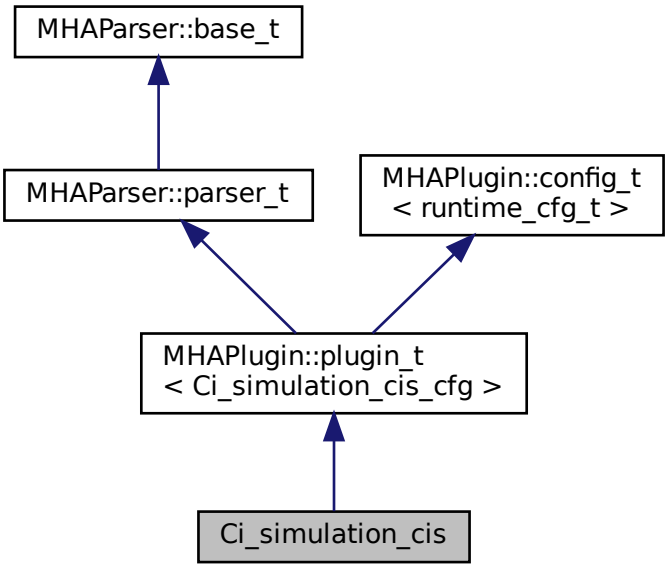
The documentation for this class was generated from the following files:

- **ci_simulation_ace.hh**
- **ci_simulation_ace.cpp**

5.74 Ci_simulation_cis Class Reference

Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

Inheritance diagram for Ci_simulation_cis:



Public Member Functions

- **Ci_simulation_cis** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**

  Name of the algorithm within the plugin chain.
- **MHAParser::vfloat_t weights**

  Vector containing the weights for the analysis filterbank bands.
- **MHAParser::float_t compression_coefficient**

  Compression coefficient of the loudness growth function.
- **MHAParser::float_t base_level**

  Base level of the input (acoustic) dynamic range / Pa.
- **MHAParser::float_t saturation_level**

  Saturation level of the input (acoustic) dynamic range / Pa.
- **MHAParser::vfloat_t threshold_level**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.
- **MHAParser::vfloat_t maximum_comfortable_level**

  Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.
- **MHAParser::vint_t disabled_electrodes**

  Vector containig the indices of any disabled electrodes.
- **MHAParser::kw_t stimulation_order**

  Electrode stimulation order.
- **MHAParser::int_t randomization_seed**

  Seed for randomization of stimulation order.
- **MHAEvents::patchbay_t**$<$ **Ci_simulation_cis** $>$ **patchbay**

  Data member connecting an event emitter (i.e.
- std::default_random_engine **random_number_generator**

  Random number generator for randomization of stimulation order.
- **MHA_AC::waveform_t** $*$ **electrodogram_ac**

  AC variable containing a pointer to the electrodogram (containing a matrix with dimensions m x m, where m = total number of electrodes = number of active electrodes, and where the order of the active electrodes corrresponds to the temporal sequence of their activation)

Additional Inherited Members

5.74.1 Detailed Description

Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

5.74.2 Constructor & Destructor Documentation

5.74.2.1 Ci_simulation_cis() `Ci_simulation_cis::Ci_simulation_cis (`

`algo_comm_t` `&` *`ac,`*
`const std::string &` *`algo_name`* `)`

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.74.3 Member Function Documentation

5.74.3.1 prepare() `void Ci_simulation_cis::prepare (`

`mhaconfig_t` `&` *`signal_info`* `) [virtual]`

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Ci_simulation_cis_cfg >** (p. 1301).

5.74.3.2 process() **`mha_wave_t`** `* Ci_simulation_cis::process (`

`mha_wave_t` `*` *`signal`* `)`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.74.3.3 release() `void Ci_simulation_cis::release ( void ) [virtual]`

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Ci_simulation_cis_cfg >** (p. 1302).

5.74.3.4 update_cfg() `void Ci_simulation_cis::update_cfg ( ) [private]`

Runtime configuration update function of the plugin interface class.

5.74.4 Member Data Documentation

5.74.4.1 algo_name `std::string Ci_simulation_cis::algo_name [private]`

Name of the algorithm within the plugin chain.

5.74.4.2 weights `MHAParser::vfloat_t Ci_simulation_cis::weights [private]`

Vector containing the weights for the analysis filterbank bands.

5.74.4.3 compression_coefficient `MHAParser::float_t Ci_simulation_cis::compression_coefficient [private]`

Compression coefficient of the loudness growth function.

5.74.4.4 base_level `MHAParser::float_t Ci_simulation_cis::base_level [private]`

Base level of the input (acoustic) dynamic range / Pa.

5.74.4.5 saturation_level

MHAParser::float_t Ci_simulation_cis::saturation_level [private]

Saturation level of the input (acoustic) dynamic range / Pa.

5.74.4.6 threshold_level

MHAParser::vfloat_t Ci_simulation_cis::threshold_level [private]

Vector containing the threshold level of the output (electric) dynamic range for each electrode / cu.

5.74.4.7 maximum_comfortable_level

MHAParser::vfloat_t Ci_simulation_cis::maximum_comfortable_level [private]

Vector containing the maximum comfortable level of the output (electric) dynamic range for each electrode / cu.

5.74.4.8 disabled_electrodes

MHAParser::vint_t Ci_simulation_cis::disabled_electrodes [private]

Vector containig the indices of any disabled electrodes.

5.74.4.9 stimulation_order

MHAParser::kw_t Ci_simulation_cis::stimulation_order [private]

Electrode stimulation order.

5.74.4.10 randomization_seed

MHAParser::int_t Ci_simulation_cis::randomization_seed [private]

Seed for randomization of stimulation order.

5.74.4.11 patchbay

`MHAEvents::patchbay_t< Ci_simulation_cis> Ci_simulation_cis::patchbay [private]`

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

5.74.4.12 random_number_generator

`std::default_random_engine Ci_simulation_cis::random_number_generator [private]`

Random number generator for randomization of stimulation order.

5.74.4.13 electrodogram_ac

`MHA_AC::waveform_t* Ci_simulation_cis::electrodogram_ac [private]`

AC variable containing a pointer to the electrodogram (containing a matrix with dimensions m x m, where m = total number of electrodes = number of active electrodes, and where the order of the active electrodes corrresponds to the temporal sequence of their activation)

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

- **ci_simulation_cis.hh**
- **ci_simulation_cis.cpp**

5.75 Ci_simulation_cis_cfg Class Reference

Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

Public Member Functions

- **Ci_simulation_cis_cfg** (std::vector< **mha_real_t** > weights, **mha_real_t** compression_coefficient, **mha_real_t** base_level, **mha_real_t** saturation_level, std::vector< **mha_real_t** > threshold_level, std::vector< **mha_real_t** > maximum_comfortable_level, std::vector< int > disabled_electrodes, unsigned int stimulation_order, std::default_random_engine random_number_generator, **Ci_simulation_cis** * **Ci_simulation_cis**)

  Constructor of the runtime configuration class.
- ~**Ci_simulation_cis_cfg** ()

  Destructor of the runtime configuration class.
- **mha_wave_t** * **process** (**mha_wave_t** *signal, **Ci_simulation_cis** * **Ci_simulation_cis**, **MHA_AC::waveform_t** *electrodogram_ac)

  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- std::vector< **mha_real_t** > **weights_cfg**

  Vector containing the weights for the analysis filterbank bands.
- **mha_real_t compression_coefficient_cfg**

  Compression coefficient of the loudness growth function.
- **mha_real_t base_level_cfg**

  Base level of the input (acoustic) dynamic range.
- **mha_real_t saturation_level_cfg**

  Saturation level of the input (acoustic) dynamic range.
- std::vector< **mha_real_t** > **threshold_level_cfg**

  Vector containing the threshold level of the output (electric) dynamic range for each electrode.
- std::vector< **mha_real_t** > **maximum_comfortable_level_cfg**

  Vector containing the maximum comfortable level of the output (electric) dynamicrange for each electrode.
- std::vector< int > **disabled_electrodes_cfg**

  Vector containing the indices of any disabled electrodes.
- unsigned int **stimulation_order_cfg**

  Electrode stimulation order.
- std::default_random_engine **random_number_generator_cfg**

  Random number generator for randomization of stimulation order.
- **MHASignal::waveform_t** * **signal_out**

  Pointer to the current output signal fragment.

5.75.1 Detailed Description

Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

5.75.2 Constructor & Destructor Documentation

5.75.2.1 Ci_simulation_cis_cfg()

```
Ci_simulation_cis_cfg::Ci_simulation_cis_cfg (
    std::vector< mha_real_t > weights,
     mha_real_t compression_coefficient,
     mha_real_t base_level,
     mha_real_t saturation_level,
    std::vector< mha_real_t > threshold_level,
    std::vector< mha_real_t > maximum_comfortable_level,
    std::vector< int > disabled_electrodes,
    unsigned int stimulation_order,
    std::default_random_engine random_number_generator,
     Ci_simulation_cis * Ci_simulation_cis )
```

Constructor of the runtime configuration class.

Parameters

weights	Vector containing the weights for the analysi
compression_coefficient	Compression coefficient of the loudness gro
base_level	Base level of the input (acoustic) dynamic ra
saturation_level	Saturation level of the input (acoustic) dynar
threshold_level	Vector containing the threshold level of the c dynamic range for each electrode / cu
maximum_comfortable_level	Vector containing the maximum comfortable output (electric) dynamicrange for each elec
disabled_electrodes	Vector containing the indices of any disable
stimulation_order	Electrode stimulation order
random_number_generator	Random number generator for randomizatio order
Ci_simulation_cis *(p. 410)*	Pointer to the current instance of the plugin

5.75.2.2 ~Ci_simulation_cis_cfg()

```
Ci_simulation_cis_cfg::~Ci_simulation_cis_cfg (
)
```

Destructor of the runtime configuration class.

5.75.3 Member Function Documentation

5.75.3.1 process()

```
mha_wave_t * Ci_simulation_cis_cfg::process (
    mha_wave_t * signal,
    Ci_simulation_cis * Ci_simulation_cis,
    MHA_AC::waveform_t * electrodogram_ac )
```

Process function of the runtime configuration class (main signal processing function).

It takes an input signal fragment as well as the real and imaginary outputs of a gammatone filterbank with m bands as a parameter, resulting in a total of 1 + m ∗ 2 input signal channels. It returns the input signal fragment unchanged; the purpose of the function is to compute the AC variable electrodogram_ac from the gammatone filterbank output for further processing in the plugin chain

Parameters

signal	Pointer to the current input signal fragment + gammatone filterbank output
Ci_simulation_cis *(p. 410)*	Pointer to the current instance of the plugin interface class
electrodogram_ac	AC variable containing a pointer to the electrodogram

Returns

Pointer to the (unchanged) output signal fragment

5.75.4 Member Data Documentation

5.75.4.1 weights_cfg

```
std::vector< mha_real_t> Ci_simulation_cis_cfg::weights_cfg
[private]
```

Vector containing the weights for the analysis filterbank bands.

5.75.4.2 compression_coefficient_cfg

```
mha_real_t Ci_simulation_cis_cfg::compression↩
_coefficient_cfg [private]
```

Compression coefficient of the loudness growth function.

5.75.4.3 base_level_cfg

`mha_real_t Ci_simulation_cis_cfg::base_level_cfg [private]`

Base level of the input (acoustic) dynamic range.

5.75.4.4 saturation_level_cfg

`mha_real_t Ci_simulation_cis_cfg::saturation_level_cfg [private]`

Saturation level of the input (acoustic) dynamic range.

5.75.4.5 threshold_level_cfg

`std::vector< mha_real_t> Ci_simulation_cis_cfg::threshold_level_cfg [private]`

Vector containing the threshold level of the output (electric) dynamic range for each electrode.

5.75.4.6 maximum_comfortable_level_cfg

`std::vector< mha_real_t> Ci_simulation_cis_cfg::maximum_comfortable_level_cfg [private]`

Vector containing the maximum comfortable level of the output (electric) dynamicrange for each electrode.

5.75.4.7 disabled_electrodes_cfg

`std::vector<int> Ci_simulation_cis_cfg::disabled_electrodes_cfg [private]`

Vector containing the indices of any disabled electrodes.

5.75.4.8 stimulation_order_cfg

`unsigned int Ci_simulation_cis_cfg::stimulation_order_cfg [private]`

Electrode stimulation order.

5.75.4.9 random_number_generator_cfg `std::default_random_engine Ci_simulation_cis_cfg::random_number_generator_cfg [private]`

Random number generator for randomization of stimulation order.

5.75.4.10 signal_out `MHASignal::waveform_t* Ci_simulation_cis_cfg::signal_out [private]`

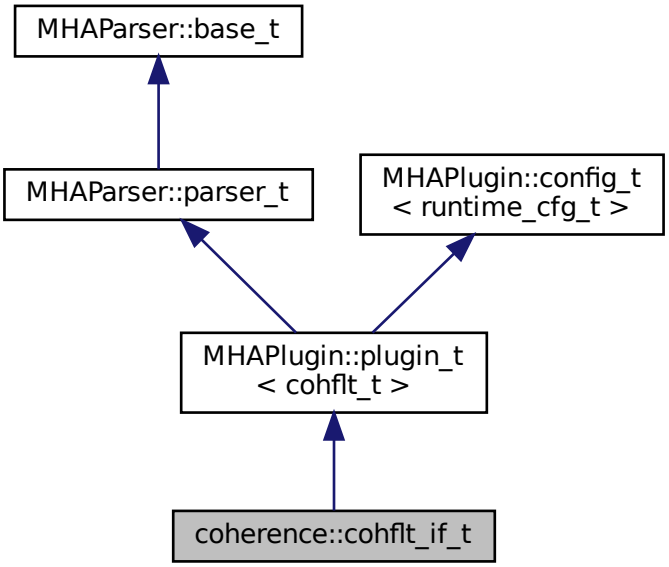
Pointer to the current output signal fragment.

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

- **ci_simulation_cis.hh**
- **ci_simulation_cis.cpp**

5.76 coherence::cohflt_if_t Class Reference

Inheritance diagram for coherence::cohflt_if_t:



Public Member Functions

- **cohflt_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.76.1 Constructor & Destructor Documentation

5.76.1.1 cohflt_if_t() `coherence::cohflt_if_t::cohflt_if_t (`

```
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.76.2 Member Function Documentation

5.76.2.1 prepare() `void coherence::cohflt_if_t::prepare (`

```
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t**< **cohflt_t** > (p. 1301).

5.76.2.2 release() void coherence::cohflt_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< cohflt_t >** (p. 1302).

5.76.2.3 process() **mha_spec_t** * coherence::cohflt_if_t::process (**mha_spec_t** * *s*)

5.76.2.4 update() void coherence::cohflt_if_t::update ( ) [private]

5.76.3 Member Data Documentation

5.76.3.1 patchbay **MHAEvents::patchbay_t< cohflt_if_t>** coherence::cohflt_if_t↩ ::patchbay [private]

5.76.3.2 vars **vars_t** coherence::cohflt_if_t::vars [private]

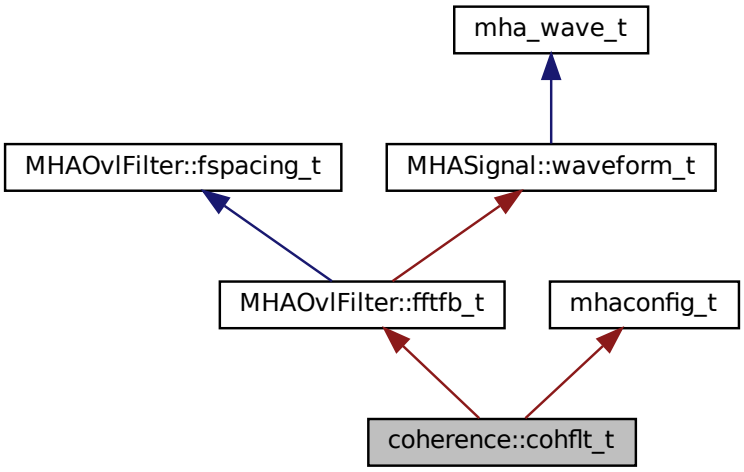
5.76.3.3 algo const std::string coherence::cohflt_if_t::algo [private]

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

- **coherence.cpp**

5.77 coherence::cohflt_t Class Reference

Inheritance diagram for coherence::cohflt_t:



Public Member Functions

- **cohflt_t** (**vars_t** &v, const **mhaconfig_t** &icf, **MHA_AC::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

5.77.1 Constructor & Destructor Documentation

5.77.1.1 cohflt_t()

```
coherence::cohflt_t::cohflt_t (
    vars_t & v,
    const mhaconfig_t & icf,
    MHA_AC::algo_comm_t & iac,
    const std::string & name )
```

5.77.2 Member Function Documentation

5.77.2.1 process()

```
mha_spec_t * coherence::cohflt_t::process (
    mha_spec_t * s )
```

5.77.2.2 insert()

```
void coherence::cohflt_t::insert ( )
```

5.77.3 Member Data Documentation

5.77.3.1 nbands

```
unsigned int coherence::cohflt_t::nbands [private]
```

5.77.3.2 avg_ipd bool coherence::cohflt_t::avg_ipd [private]

5.77.3.3 cg mha_complex_t coherence::cohflt_t::cg [private]

5.77.3.4 g float coherence::cohflt_t::g [private]

5.77.3.5 c_scale float coherence::cohflt_t::c_scale [private]

5.77.3.6 c_min float coherence::cohflt_t::c_min [private]

5.77.3.7 alpha MHASignal::waveform_t coherence::cohflt_t::alpha [private]

5.77.3.8 limit float coherence::cohflt_t::limit [private]

5.77.3.9 lp1r MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1r [private]

5.77.3.10 lp1i MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1i [private]

5.77.3.11 coh_c

`MHA_AC::spectrum_t coherence::cohflt_t::coh_c [private]`

5.77.3.12 coh_rlp

`MHA_AC::waveform_t coherence::cohflt_t::coh_rlp [private]`

5.77.3.13 gain

`MHASignal::waveform_t coherence::cohflt_t::gain [private]`

5.77.3.14 gain_delay

`MHASignal::delay_wave_t coherence::cohflt_t::gain_delay [private]`

5.77.3.15 s_out

`MHASignal::spectrum_t coherence::cohflt_t::s_out [private]`

5.77.3.16 bInvert

`bool coherence::cohflt_t::bInvert [private]`

5.77.3.17 lp1ltg

`MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1ltg [private]`

5.77.3.18 b_ltg

`bool coherence::cohflt_t::b_ltg [private]`

5.77.3.19 staticgain

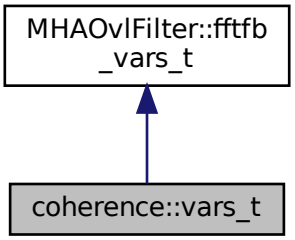
`std::vector<float> coherence::cohflt_t::staticgain [private]`

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

- **coherence.cpp**

5.78 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**

5.78.1 Constructor & Destructor Documentation

5.78.1.1 vars_t() `coherence::vars_t::vars_t (`

MHAParser::parser_t $*$ *p*)

5.78.2 Member Data Documentation

5.78.2.1 tau_unit **MHAParser::kw_t** coherence::vars_t::tau_unit

5.78.2.2 tau **MHAParser::vfloat_t** coherence::vars_t::tau

5.78.2.3 alpha **MHAParser::vfloat_t** coherence::vars_t::alpha

5.78.2.4 limit **MHAParser::float_t** coherence::vars_t::limit

5.78.2.5 mapping **MHAParser::vfloat_t** coherence::vars_t::mapping

5.78.2.6 average **MHAParser::kw_t** coherence::vars_t::average

5.78.2.7 invert **MHAParser::bool_t** coherence::vars_t::invert

5.78.2.8 ltgcomp **MHAParser::bool_t** coherence::vars_t::ltgcomp

5.78.2.9 ltgtau `MHAParser::vfloat_t coherence::vars_t::ltgtau`

5.78.2.10 staticgain `MHAParser::vfloat_t coherence::vars_t::staticgain`

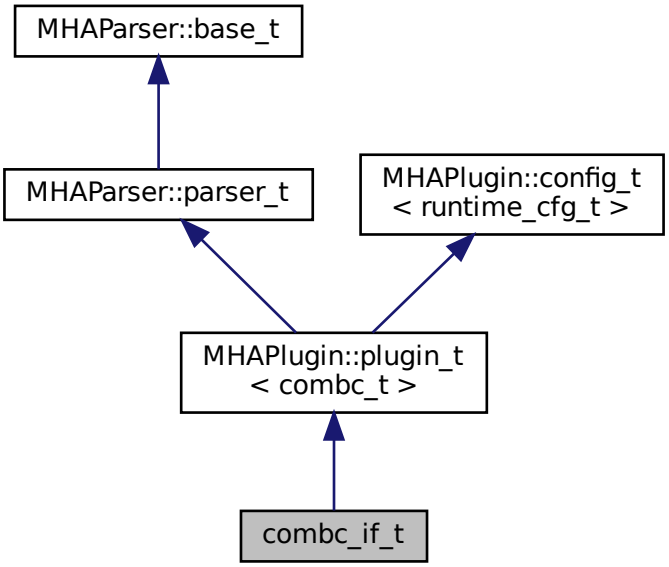
5.78.2.11 delay `MHAParser::int_t coherence::vars_t::delay`

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

- **coherence.cpp**

5.79 combc_if_t Class Reference

Inheritance diagram for combc_if_t:



Public Member Functions

- **combc_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.79.1 Constructor & Destructor Documentation

5.79.1.1 combc_if_t() `combc_if_t::combc_if_t (`

```
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.79.2 Member Function Documentation

5.79.2.1 prepare() `void combc_if_t::prepare (`

```
    mhaconfig_t & chcfg ) [virtual]
```

Implements **MHAPlugin::plugin_t**< **combc_t** > (p. 1301).

5.79.2.2 process() [1/2] **mha_wave_t** ∗ `combc_if_t::process (`

```
    mha_wave_t * s )
```

5.79.2.3 process() [2/2] mha_spec_t * combc_if_t::process (mha_spec_t * s)

5.79.3 Member Data Documentation

5.79.3.1 outchannels MHAParser::int_t combc_if_t::outchannels [private]

5.79.3.2 interleaved MHAParser::bool_t combc_if_t::interleaved [private]

5.79.3.3 channel_gain_name MHAParser::string_t combc_if_t::channel_gain_name [private]

5.79.3.4 element_gain_name MHAParser::string_t combc_if_t::element_gain_name [private]

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

- **combinechannels.cpp**

5.80 combc_t Class Reference

Public Member Functions

- **combc_t** (**MHA_AC::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

- **MHA_AC::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_**

5.80.1 Constructor & Destructor Documentation

5.80.1.1 combc_t()

```
combc_t::combc_t (
    MHA_AC::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 )
```

5.80.2 Member Function Documentation

5.80.2.1 process() [1/2]

```
mha_wave_t * combc_t::process (
    mha_wave_t * s )
```

5.80.2.2 process() [2/2]

```
mha_spec_t * combc_t::process (
    mha_spec_t * s )
```

5.80.3 Member Data Documentation

5.80.3.1 ac_ MHA_AC::algo_comm_t& combc_t::ac_ [private]

5.80.3.2 interleaved_ bool combc_t::interleaved_ [private]

5.80.3.3 nbands unsigned int combc_t::nbands [private]

5.80.3.4 w_out MHASignal::waveform_t combc_t::w_out [private]

5.80.3.5 s_out MHASignal::spectrum_t combc_t::s_out [private]

5.80.3.6 channel_gains_ std::vector< mha_real_t> combc_t::channel_gains_ [private]

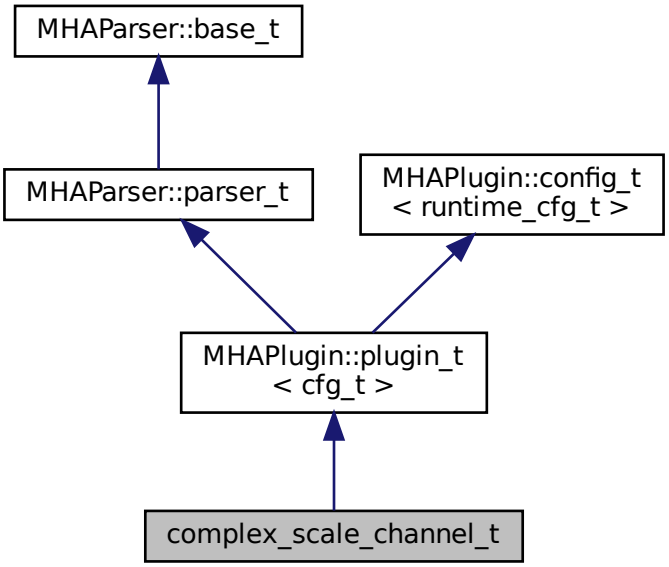
5.80.3.7 element_gain_name_ std::string combc_t::element_gain_name_ [private]

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

- **combinechannels.cpp**

5.81 complex_scale_channel_t Class Reference

Inheritance diagram for complex_scale_channel_t:



Public Member Functions

- **complex_scale_channel_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **complex_scale_channel_t** > **patchbay**
- **MHAParser::int_t** **scale_ch**
- **MHAParser::complex_t** **factor**

Additional Inherited Members

5.81.1 Constructor & Destructor Documentation

5.81.1.1 complex_scale_channel_t() complex_scale_channel_t::complex_scale_channel↩ _t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.81.2 Member Function Documentation

5.81.2.1 process() mha_spec_t * complex_scale_channel_t::process (

```
mha_spec_t * spec )
```

5.81.2.2 prepare() void complex_scale_channel_t::prepare (

```
mhaconfig_t & cfg ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.81.2.3 update_cfg() void complex_scale_channel_t::update_cfg ( ) [private]

5.81.3 Member Data Documentation

5.81.3.1 patchbay MHAEvents::patchbay_t< complex_scale_channel_t> complex_scale↩ _channel_t::patchbay [private]

5.81.3.2 scale_ch `MHAParser::int_t complex_scale_channel_t::scale_ch [private]`

5.81.3.3 factor `MHAParser::complex_t complex_scale_channel_t::factor [private]`

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

- **complex_scale_channel.cpp**

5.82 cpuload::cpuload_cfg_t Class Reference

Public Member Functions

- **cpuload_cfg_t** (**mha_real_t** factor_, size_t table_size_, bool use_sine_)
  Ctor of the runtime configuration class.
- void **process** (unsigned fac_)
  Process callback. Does not actually change signal.

Private Member Functions

- void **calc_sine** ()
- void **write_to_table** ()

Private Attributes

- float **phase** =0
  Phase of the sine.
- volatile float **result** =0
  Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.
- bool **use_sine** =false
  Use sine or do table operation.
- float **factor** =0
  cpu load factor. Values $>$ 1 increase cpu load, values $<$ 1 decrease it
- std::vector$<$ float $>$ **table**
  Table with arbitrary values to operate on. Unused if use_sine=true.

5.82.1 Constructor & Destructor Documentation

5.82.1.1 cpuload_cfg_t() `cpuload::cpuload_cfg_t::cpuload_cfg_t (`

```
    mha_real_t factor_,
  size_t table_size_,
  bool use_sine_ )
```

Ctor of the runtime configuration class.

Parameters

*factor_*	cpu load factor. Values $>$ 1 increase cpu load, values $<$ 1 decrease it
*table_size_*	
*use_sine_*	

5.82.2 Member Function Documentation

5.82.2.1 process() `void cpuload::cpuload_cfg_t::process (`

```
  unsigned fac_ )
```

Process callback. Does not actually change signal.

5.82.2.2 calc_sine() `void cpuload::cpuload_cfg_t::calc_sine ( ) [private]`

5.82.2.3 write_to_table() `void cpuload::cpuload_cfg_t::write_to_table ( ) [private]`

5.82.3 Member Data Documentation

5.82.3.1 phase `float cpuload::cpuload_cfg_t::phase =0 [private]`

Phase of the sine.

5.82.3.2 result `volatile float cpuload::cpuload_cfg_t::result =0 [private]`

Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.

5.82.3.3 use_sine `bool cpuload::cpuload_cfg_t::use_sine =false [private]`

Use sine or do table operation.

5.82.3.4 factor `float cpuload::cpuload_cfg_t::factor =0 [private]`

cpu load factor. Values > 1 increase cpu load, values < 1 decrease it

5.82.3.5 table `std::vector<float> cpuload::cpuload_cfg_t::table [private]`

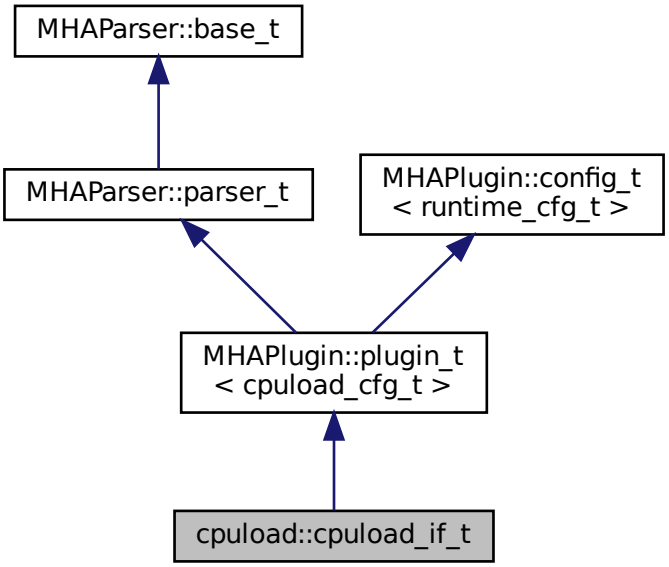
Table with arbitrary values to operate on. Unused if use_sine=true.

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

- **cpuload.cpp**

5.83 cpuload::cpuload_if_t Class Reference

Inheritance diagram for cpuload::cpuload_if_t:



Public Member Functions

- **cpuload_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::float_t factor**
- **MHAParser::int_t table_size**
- **MHAParser::bool_t use_sine**
- **MHAEvents::patchbay_t**< **cpuload_if_t** > **patchbay**

Additional Inherited Members

5.83.1 Constructor & Destructor Documentation

5.83.1.1 cpuload_if_t() cpuload::cpuload_if_t::cpuload_if_t (**MHA_AC::algo_comm_t** & *iac*, const std::string & *configured_name*)

5.83.2 Member Function Documentation

5.83.2.1 process() [1/2] **mha_spec_t** * cpuload::cpuload_if_t::process (**mha_spec_t** * *s*)

5.83.2.2 process() [2/2] mha_wave_t * cpuload::cpuload_if_t::process (mha_wave_t * s)

5.83.2.3 prepare() void cpuload::cpuload_if_t::prepare (mhaconfig_t &) [virtual]

Implements **MHAPlugin::plugin_t< cpuload_cfg_t >** (p. 1301).

5.83.2.4 update() void cpuload::cpuload_if_t::update ( ) [private]

5.83.3 Member Data Documentation

5.83.3.1 factor MHAParser::float_t cpuload::cpuload_if_t::factor [private]

5.83.3.2 table_size MHAParser::int_t cpuload::cpuload_if_t::table_size [private]

5.83.3.3 use_sine MHAParser::bool_t cpuload::cpuload_if_t::use_sine [private]

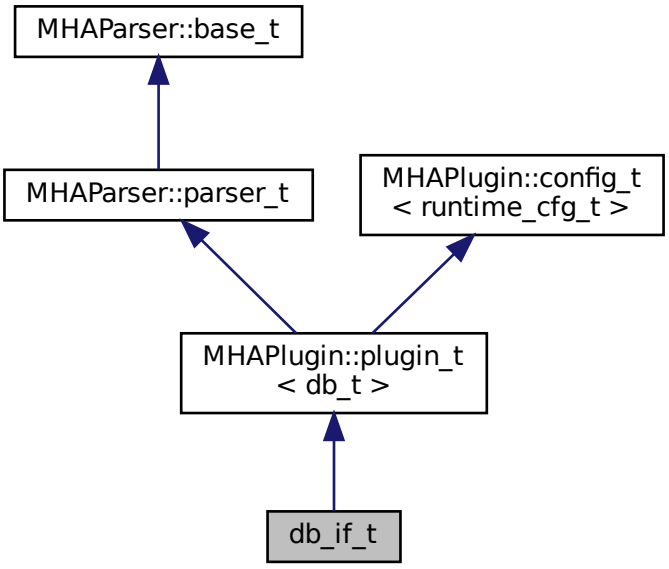
5.83.3.4 patchbay MHAEvents::patchbay_t< cpuload_if_t> cpuload::cpuload_if_t↩ ::patchbay [private]

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

- **cpuload.cpp**

5.84 db_if_t Class Reference

Inheritance diagram for db_if_t:



Public Member Functions

- **db_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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 **algo**
- bool **bypass**

Additional Inherited Members

5.84.1 Constructor & Destructor Documentation

5.84.1.1 db_if_t() `db_if_t::db_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.84.1.2 ~db_if_t() `db_if_t::~db_if_t ( )`

5.84.2 Member Function Documentation

5.84.2.1 process() `mha_wave_t * db_if_t::process (`

```
mha_wave_t * s )
```

5.84.2.2 prepare() `void db_if_t::prepare (`

```
mhaconfig_t & conf ) [virtual]
```

Implements **MHAPlugin::plugin_t< db_t >** (p. 1301).

5.84.2.3 release() `void db_if_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< db_t >** (p. 1302).

5.84.3 Member Data Documentation

5.84.3.1 patchbay `MHAEvents::patchbay_t< db_if_t > db_if_t::patchbay [private]`

5.84.3.2 fragsize `MHAParser::int_t db_if_t::fragsize [private]`

5.84.3.3 plugloader `MHAParser::mhapluginloader_t db_if_t::plugloader [private]`

5.84.3.4 algo `std::string db_if_t::algo [private]`

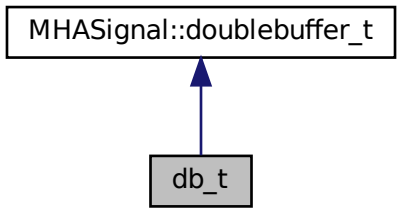
5.84.3.5 bypass `bool db_if_t::bypass [private]`

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

- **db.cpp**

5.85 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

5.85.1 Constructor & Destructor Documentation

5.85.1.1 db_t()

```
db_t::db_t (
    unsigned int outer_fragsize,
    unsigned int inner_fragsize,
    unsigned int nch_in,
    unsigned int nch_out,
    MHAParser::mhapluginloader_t & plug )
```

5.85.2 Member Function Documentation

5.85.2.1 inner_process()

```
mha_wave_t * db_t::inner_process (
    mha_wave_t * s ) [virtual]
```

Implements **MHASignal::doublebuffer_t** (p. 1358).

5.85.3 Member Data Documentation

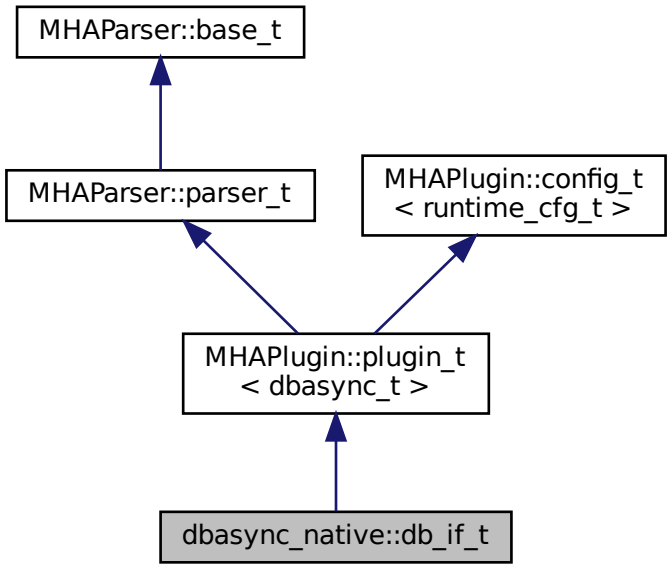
5.85.3.1 plugloader `MHAParser::mhapluginloader_t& db_t::plugloader [private]`

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

- **db.cpp**

5.86 dbasync_native::db_if_t Class Reference

Inheritance diagram for dbasync_native::db_if_t:



Public Member Functions

- **db_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- ∼**db_if_t** ()=default

Private Attributes

- **MHA_AC::algo_comm_class_t sub_ac**
- **MHAParser::mhapluginloader_t plugloader**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t delay**
- **MHAParser::kw_t worker_thread_scheduler**
  Scheduler used for worker thread.
- **MHAParser::int_t worker_thread_priority**
  Priority of worker thread.
- **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.
- std::string **algo**

Additional Inherited Members

5.86.1 Constructor & Destructor Documentation

5.86.1.1 db_if_t() `db_if_t::db_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.86.1.2 ~db_if_t() `dbasync_native::db_if_t::~db_if_t ( ) [default]`

5.86.2 Member Function Documentation

5.86.2.1 process() `mha_wave_t * db_if_t::process (`

```
mha_wave_t * s )
```

5.86.2.2 prepare() void db_if_t::prepare (mhaconfig_t & conf) [virtual]

Implements **MHAPlugin::plugin_t< dbasync_t >** (p. 1301).

5.86.2.3 release() void db_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< dbasync_t >** (p. 1302).

5.86.3 Member Data Documentation

5.86.3.1 sub_ac MHA_AC::algo_comm_class_t dbasync_native::db_if_t::sub_ac [private]

5.86.3.2 plugloader MHAParser::mhapluginloader_t dbasync_native::db_if_t::plugloader [private]

5.86.3.3 fragsize MHAParser::int_t dbasync_native::db_if_t::fragsize [private]

5.86.3.4 delay MHAParser::int_t dbasync_native::db_if_t::delay [private]

5.86.3.5 worker_thread_scheduler MHAParser::kw_t dbasync_native::db_if_t::worker_thread_scheduler [private]

Scheduler used for worker thread.

5.86.3.6 worker_thread_priority

MHAParser::int_t dbasync_native::db_if_t::worker_thread_priority [private]

Priority of worker thread.

5.86.3.7 framework_thread_scheduler

MHAParser::string_mon_t dbasync_native::db_if_t::framework_thread_scheduler [private]

Scheduler of the signal processing thread.

5.86.3.8 framework_thread_priority

MHAParser::int_mon_t dbasync_native::db_if_t::framework_thread_priority [private]

Priority of signal processing thread.

5.86.3.9 algo

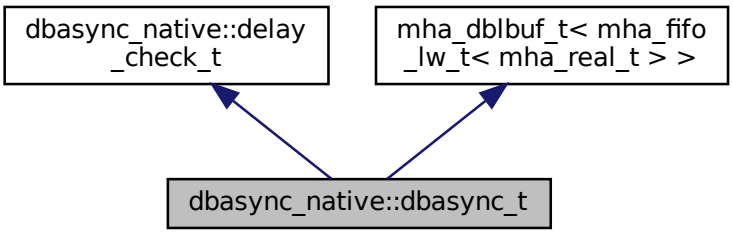
std::string dbasync_native::db_if_t::algo [private]

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

- **dbasync.cpp**

5.87 dbasync_native::dbasync_t Class Reference

Inheritance diagram for dbasync_native::dbasync_t:



Public Member Functions

- **dbasync_t** (unsigned int nchannels_in, unsigned int nchannels_out, unsigned int outer↩ _fragsize, unsigned int inner_fragsize, int **delay**, const std::string &thread_scheduler, int thread_priority, **MHAParser::mhapluginloader_t** &plug)
- ∼**dbasync_t** ()
- **mha_wave_t** ∗ **outer_process** (**mha_wave_t** ∗)
- int **svc** ()

Private Attributes

- **MHAParser::mhapluginloader_t** & **plugloader**
- **MHASignal::waveform_t** **inner_input**
- **MHASignal::waveform_t** **outer_output**
- pthread_attr_t **attr**
- struct sched_param **priority**
- int **scheduler**
- pthread_t **thread**

Additional Inherited Members

5.87.1 Constructor & Destructor Documentation

5.87.1.1 dbasync_t() dbasync_native::dbasync_t::dbasync_t (

```
unsigned int nchannels_in,
unsigned int nchannels_out,
unsigned int outer_fragsize,
unsigned int inner_fragsize,
int delay,
const std::string & thread_scheduler,
int thread_priority,
 MHAParser::mhapluginloader_t & plug )
```

5.87.1.2 ∼dbasync_t() dbasync_native::dbasync_t::∼dbasync_t ( )

5.87.2 Member Function Documentation

5.87.2.1 outer_process()

mha_wave_t * dbasync_native::dbasync_t::outer_process (**mha_wave_t** * *outer_input*)

5.87.2.2 svc()

int dbasync_native::dbasync_t::svc ( )

5.87.3 Member Data Documentation

5.87.3.1 plugloader

MHAParser::**mhapluginloader_t**& dbasync_native::dbasync_t↩
::plugloader [private]

5.87.3.2 inner_input

MHASignal::**waveform_t** dbasync_native::dbasync_t::inner_input [private]

5.87.3.3 outer_output

MHASignal::**waveform_t** dbasync_native::dbasync_t::outer_↩
output [private]

5.87.3.4 attr

pthread_attr_t dbasync_native::dbasync_t::attr [private]

5.87.3.5 priority

struct sched_param dbasync_native::dbasync_t::priority [private]

5.87.3.6 scheduler

int dbasync_native::dbasync_t::scheduler [private]

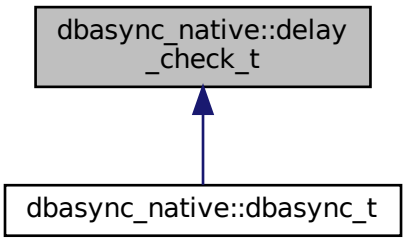
5.87.3.7 thread `pthread_t dbasync_native::dbasync_t::thread [private]`

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

- **dbasync.cpp**

5.88 dbasync_native::delay_check_t Class Reference

Inheritance diagram for dbasync_native::delay_check_t:



Protected Member Functions

- **delay_check_t** (int delay, unsigned inner_fragsize, unsigned outer_fragsize)

5.88.1 Constructor & Destructor Documentation

5.88.1.1 delay_check_t()

```
dbasync_native::delay_check_t::delay_check_t (
    int delay,
    unsigned inner_fragsize,
    unsigned outer_fragsize ) [protected]
```

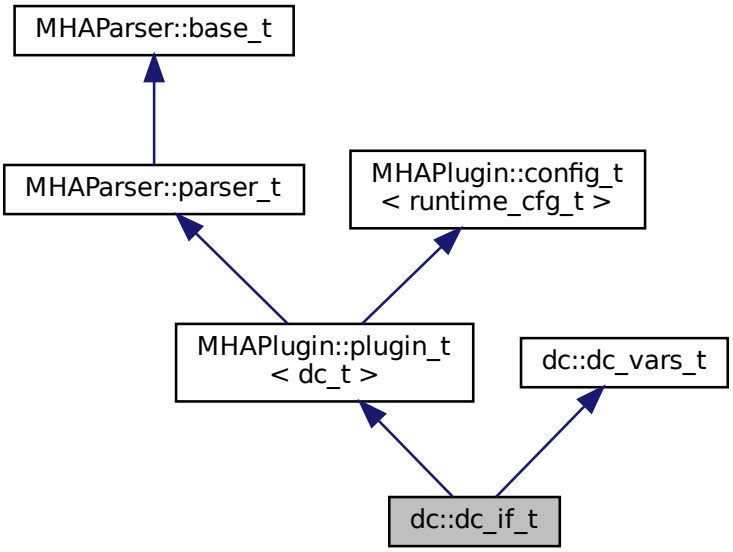
The documentation for this class was generated from the following file:

- **dbasync.cpp**

5.89 dc::dc_if_t Class Reference

Plugin interface class of the dynamic compression plugin `dc`.

Inheritance diagram for dc::dc_if_t:



Public Member Functions

- **dc_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)

  Standard MHA plugin constructor.
- void **prepare** (**mhaconfig_t** &tf)

  Prepare plugin for signal processing.
- void **release** ()

  Release plugin from signal processing.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s_in)

  Process method extracts band-specific input levels using the rmslevel, attack and decay filters on each input sample, looks up the gains and applies them to each sample of the signal in place.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s_in)

  Process method extracts band-specific input levels using the attack and decay filters on the latest STFT spectrum, looks up the gains and applies them in place.

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**

  Configured name of this plugin, used as prefix for names of published AC variables.
- **MHAEvents::patchbay_t**< **dc_if_t** > **patchbay**

  Connects configuration events to callbacks.
- unsigned **broadband_audiochannels** = {0U}

  Number of broadband audio channels (before the filterbank).
- unsigned **bands_per_channel** = {0U}

  Number of frequency bands per broadband audio channel.
- std::string **cf_name**

  Name of AC variable containing the filterbank centre frequencies in Hz.
- std::string **ef_name**

  Name of AC variable containing the filterbank edge frequencies in Hz.
- std::string **bw_name**

  Name of the AC variable containing the filterbank band weights.

Additional Inherited Members

5.89.1 Detailed Description

Plugin interface class of the dynamic compression plugin `dc`.

5.89.2 Constructor & Destructor Documentation

5.89.2.1 dc_if_t() `dc_if_t::dc_if_t (`
`MHA_AC::algo_comm_t & iac,`
`const std::string & configured_name )`

Standard MHA plugin constructor.

Parameters

iac	Algorithm communication variable space.
configured_name	The name given to this plugin by the configuration.

5.89.3 Member Function Documentation

5.89.3.1 prepare() `void dc_if_t::prepare (`
`mhaconfig_t & tf ) [virtual]`

Prepare plugin for signal processing.

Parameters

tf	Input signal dimensions. They are not modified.

Implements **MHAPlugin::plugin_t< dc_t >** (p. 1301).

5.89.3.2 release() `void dc_if_t::release (`
`void ) [virtual]`

Release plugin from signal processing.

Reimplemented from **MHAPlugin::plugin_t< dc_t >** (p. 1302).

5.89.3.3 process() [1/2] mha_wave_t * dc_if_t::process (mha_wave_t * s_in)

Process method extracts band-specific input levels using the rmslevel, attack and decay filters on each input sample, looks up the gains and applies them to each sample of the signal in place.

Parameters

s_in	Latest block of time-domain input signal.

Returns

s_in after modifying the signal in place.

5.89.3.4 process() [2/2] mha_spec_t * dc_if_t::process (mha_spec_t * s_in)

Process method extracts band-specific input levels using the attack and decay filters on the latest STFT spectrum, looks up the gains and applies them in place.

Parameters

s_in	Latest spectrum of the STFT input signal.

Returns

s_in after modifying the signal in place.

5.89.3.5 update_monitors() void dc_if_t::update_monitors ( ) [private]

Called from within the processing routines: updates the monitor variables.

5.89.3.6 update() void dc_if_t::update ( ) [private]

Called by MHA configuration change event mechanism: creates new runtime configuration.

5.89.4 Member Data Documentation

5.89.4.1 algo std::string dc::dc_if_t::algo [private]

Configured name of this plugin, used as prefix for names of published AC variables.

5.89.4.2 patchbay MHAEvents::patchbay_t< dc_if_t> dc::dc_if_t::patchbay [private]

Connects configuration events to callbacks.

5.89.4.3 broadband_audiochannels unsigned dc::dc_if_t::broadband_audiochannels = {0U} [private]

Number of broadband audio channels (before the filterbank).

This value is filled in during **prepare()** (p. 454).

5.89.4.4 bands_per_channel unsigned dc::dc_if_t::bands_per_channel = {0U} [private]

Number of frequency bands per broadband audio channel.

Initialized during **prepare()** (p. 454).

5.89.4.5 cf_name std::string dc::dc_if_t::cf_name [private]

Name of AC variable containing the filterbank centre frequencies in Hz.

Initialized during **prepare()** (p. 454).

5.89.4.6 ef_name `std::string dc::dc_if_t::ef_name [private]`

Name of AC variable containing the filterbank edge frequencies in Hz.

Initialized during **prepare()** (p. 454).

5.89.4.7 bw_name `std::string dc::dc_if_t::bw_name [private]`

Name of the AC variable containing the filterbank band weights.

Initialized during **prepare()** (p. 454).

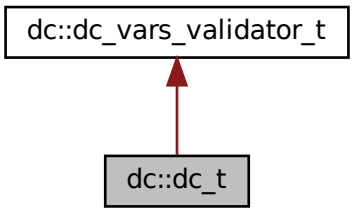
The documentation for this class was generated from the following files:

- **dc.hh**
- **dc.cpp**

5.90 dc::dc_t Class Reference

Runtime configuration class of dynamic compression plugin `dc`.

Inheritance diagram for dc::dc_t:



Public Member Functions

- **dc_t** (**dc_vars_t** vars, **mha_real_t** filter_rate, unsigned int nch_, **MHA_AC↩ ::algo_comm_t** &ac, **mha_domain_t** domain, unsigned int **fftlen**, unsigned int naudiochannels_, const std::string &configured_name, const std::vector< **mha_real_t** > &rmslevel_state={}, const std::vector< **mha_real_t** > &attack_state={}, const std↩ ::vector< **mha_real_t** > &decay_state={})

  Constructor.
- **mha_wave_t** * **process** (**mha_wave_t** *s_in)

  Process method extracts band-specific input levels using the rmslevel, attack and decay filters on each input sample, looks up the gains and applies them to each sample of the signal in place.
- **mha_spec_t** * **process** (**mha_spec_t** *s_in)

  Process method extracts band-specific input levels using the attack and decay filters on the latest STFT spectrum, looks up the gains and applies them in place.
- void **explicit_insert** ()
- unsigned **get_nbands** () const
- unsigned **get_nch** () const
- const **MHASignal::waveform_t** & **get_level_in_db** () const
- const **MHASignal::waveform_t** & **get_level_in_db_adjusted** () const
- std::vector< **mha_real_t** > **get_rmslevel_filter_state** () const
- std::vector< **mha_real_t** > **get_attack_filter_state** () const
- std::vector< **mha_real_t** > **get_decay_filter_state** () const

Private Attributes

- std::vector< **MHATableLookup::linear_table_t** > **gt**

  Dynamic compression gains.
- std::vector< **mha_real_t** > **offset**

  band-specific dB offsets added to measured input levels before gain lookup is performed.
- **MHAFilter::o1flt_lowpass_t rmslevel**

  Envelope extraction filters used in waveform processing.
- **MHAFilter::o1flt_lowpass_t attack**

  Attack filters used in input level estimation.
- **MHAFilter::o1flt_maxtrack_t decay**

  Maximum-tracking decay filters used in input level estimation.
- bool **bypass**

  Dynamic compression is not applied if bypass == true.
- bool **log_interp**

  Flag whether gain table interpolation should be done in dB domain.
- unsigned int **naudiochannels**

  Number of broadband audio channels (before the upstream filterbank)
- unsigned int **nbands**

  Number of bands per broadband audio channel.
- unsigned int **nch**

  `nbands * naudiochannels`

- **MHA_AC::waveform_t level_in_db**
  Matrix of latest input levels before attack/decay filter.
- **MHA_AC::waveform_t level_in_db_adjusted**
  Matrix of latest input levels after attack/decay filter.
- unsigned int **fftlen**
  FFT length in samples, required for computing levels correctly.

Additional Inherited Members

5.90.1 Detailed Description

Runtime configuration class of dynamic compression plugin `dc`.

5.90.2 Constructor & Destructor Documentation

5.90.2.1 dc_t()

```
dc::dc_t::dc_t (
    dc_vars_t vars,
    mha_real_t filter_rate,
    unsigned int nch_,
    MHA_AC::algo_comm_t & ac,
    mha_domain_t domain,
    unsigned int fftlen,
    unsigned int naudiochannels_,
    const std::string & configured_name,
    const std::vector< mha_real_t > & rmslevel_state = {},
    const std::vector< mha_real_t > & attack_state = {},
    const std::vector< mha_real_t > & decay_state = {} )
```

Constructor.

Parameters

vars	A copy of all configuration language variables of d
filter_rate	The rate in Hz with which the level filters of plugin are called. For waveform processing this is equal audio sampling rate. For spectral processing, this equal to the audio block rate.
*nch_*	Total number of compression bands: bands x cha
ac	Algorithm communication variable space. The constructor will not interact with it.

Parameters

domain	`MHA_WAVEFORM` or `MHA_SPECTRUM`.
fftlen	FFT length used for STFT processing, in samples
*naudiochannels_*	Number of broadband audio channels (before the upstream filterbank).
configured_name	The configured name of this plugin in the MHA configuration. Used to derive the names of the AC variables published by this plugin.
rmslevel_state	Start state of rmslevel filters.
attack_state	Start state of attack level filters.
decay_state	Start state of decay level filters.

5.90.3 Member Function Documentation

5.90.3.1 process() [1/2] `mha_wave_t * dc_t::process ( mha_wave_t * s_in )`

Process method extracts band-specific input levels using the rmslevel, attack and decay filters on each input sample, looks up the gains and applies them to each sample of the signal in place.

Parameters

s_in	Latest block of time-domain input signal.

Returns

s_in after modifying the signal in place.

5.90.3.2 process() [2/2] mha_spec_t * dc_t::process (mha_spec_t * s_in)

Process method extracts band-specific input levels using the attack and decay filters on the latest STFT spectrum, looks up the gains and applies them in place.

Parameters

s_in	Latest spectrum of the STFT input signal.

Returns

s_in after modifying the signal in place.

5.90.3.3 explicit_insert() void dc_t::explicit_insert ( )

5.90.3.4 get_nbands() unsigned dc::dc_t::get_nbands ( ) const [inline]

Returns

Number of frequency bands per broadband input channels.

5.90.3.5 get_nch() unsigned dc::dc_t::get_nch ( ) const [inline]

Returns

Number of frequency bands times broadband input channels.

5.90.3.6 get_level_in_db() const MHASignal::waveform_t& dc::dc_t::get_level_in_db ( ) const [inline]

Returns

Const reference to the Matrix of input levels as computed before processed by the attack/decay filter.

5.90.3.7 get_level_in_db_adjusted() const MHASignal::waveform_t& dc::dc_t::get_↩ level_in_db_adjusted ( ) const [inline]

Returns

Const reference to the Matrix of input levels after being filtered by the attack/decay filter.

5.90.3.8 get_rmslevel_filter_state() std::vector< mha_real_t> dc::dc_t::get_rmslevel↩ _filter_state ( ) const [inline]

Returns

Filter states of first-order rmslevel low-pass filters.

5.90.3.9 get_attack_filter_state() std::vector< mha_real_t> dc::dc_t::get_attack_↩ filter_state ( ) const [inline]

Returns

Filter states of first-order attack low-pass filters.

5.90.3.10 get_decay_filter_state() std::vector< mha_real_t> dc::dc_t::get_decay_↩ filter_state ( ) const [inline]

Returns

Filter states of max-tracking decay low-pass filters.

5.90.4 Member Data Documentation

5.90.4.1 gt `std::vector< MHATableLookup::linear_table_t> dc::dc_t::gt [private]`

Dynamic compression gains.

If `log_interp` is true, then they are stored as dB gains, otherwise they are stored as linear gains.

5.90.4.2 offset `std::vector< mha_real_t> dc::dc_t::offset [private]`

band-specific dB offsets added to measured input levels before gain lookup is performed.

5.90.4.3 rmslevel `MHAFilter::o1flt_lowpass_t dc::dc_t::rmslevel [private]`

Envelope extraction filters used in waveform processing.

5.90.4.4 attack `MHAFilter::o1flt_lowpass_t dc::dc_t::attack [private]`

Attack filters used in input level estimation.

5.90.4.5 decay `MHAFilter::o1flt_maxtrack_t dc::dc_t::decay [private]`

Maximum-tracking decay filters used in input level estimation.

5.90.4.6 bypass `bool dc::dc_t::bypass [private]`

Dynamic compression is not applied if bypass == true.

5.90.4.7 log_interp `bool dc::dc_t::log_interp [private]`

Flag whether gain table interpolation should be done in dB domain.

5.90.4.8 naudiochannels `unsigned int dc::dc_t::naudiochannels [private]`

Number of broadband audio channels (before the upstream filterbank)

5.90.4.9 nbands `unsigned int dc::dc_t::nbands [private]`

Number of bands per broadband audio channel.

5.90.4.10 nch `unsigned int dc::dc_t::nch [private]`

`nbands * naudiochannels`

5.90.4.11 level_in_db **MHA_AC::waveform_t** `dc::dc_t::level_in_db [private]`

Matrix of latest input levels before attack/decay filter.

5.90.4.12 level_in_db_adjusted **MHA_AC::waveform_t** `dc::dc_t::level_in_db_adjusted [private]`

Matrix of latest input levels after attack/decay filter.

5.90.4.13 fftlen `unsigned int dc::dc_t::fftlen [private]`

FFT length in samples, required for computing levels correctly.

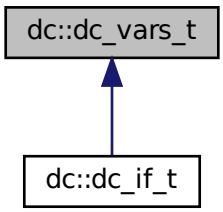
The documentation for this class was generated from the following files:

- **dc.hh**
- **dc.cpp**

5.91 dc::dc_vars_t Class Reference

Collection of configuration variables of the `dc` plugin.

Inheritance diagram for dc::dc_vars_t:



Public Member Functions

- **dc_vars_t** (**MHAParser::parser_t** &p)

  Constructor initializes the configuration language variables' data members and inserts them into the parser p.
- **dc_vars_t** (const **dc_vars_t** &)=default

  Default copy constructor is used to pass a copy of all configuration variables to the runtime configuration constructor ***dc_t::dc_t*** (p. 459).

Public Attributes

- **MHAParser::mfloat_t gtdata**

  Gain table with gains in dB.
- **MHAParser::vfloat_t gtmin**

  Narrow-band input levels in dB SPL specifying for each row of `gtdata` *to which input level the first gain in this row applies.*
- **MHAParser::vfloat_t gtstep**

  Input level increment between elements for each row of `gtdata`*.*
- **MHAParser::vfloat_t taurmslevel**

  Low-pass filter time constants for time-domain envelope extraction.
- **MHAParser::vfloat_t tauattack**

  Low-pass filter time constants for level extraction when level rises.
- **MHAParser::vfloat_t taudecay**

  Low-pass filter time constants for level extraction when level falls.
- **MHAParser::vfloat_t offset**

  Row-specific input level corrections in dB to apply before gain lookup in `gtdata`*.*
- **MHAParser::string_t filterbank**

  Configured name of filterbank plugin placed upstream of this `dc` *plugin.*
- **MHAParser::string_t chname**

  Name of the AC variable containing the number of audiochannels before the filterbank.
- **MHAParser::bool_t bypass**

  Switch for bypassing the dynamic compression.
- **MHAParser::bool_t log_interp**

  Interpolate gain table in dBs (vs.
- **MHAParser::string_t clientid**

  Metadata: Some ID of the hearing impaired subject.
- **MHAParser::string_t gainrule**

  Metadata: Some name of the gain rule that was used to compute gtdata.
- **MHAParser::string_t preset**

  Metadata: Some name given to the current setting.
- **MHAParser::vfloat_mon_t input_level**

  Narrow-band input levels of current block before attack/decay filter.
- **MHAParser::vfloat_mon_t filtered_level**

  Narrow-band input levels of current block after attack/decay filter.
- **MHAParser::vfloat_mon_t center_frequencies**

  Center frequencies of upstream filterbank.
- **MHAParser::vfloat_mon_t edge_frequencies**

  Edge frequencies of upstream filterbank.
- **MHAParser::vfloat_mon_t band_weights**

  Center frequencies of upstream filterbank.

5.91.1 Detailed Description

Collection of configuration variables of the dc plugin.

The dc plugin interface class **dc_if_t** (p. 452) inherits from **dc_vars_t** (p. 465). This class is also used to pass a copy of all configuration variables to the constructor of the runtime configuration class **dc_t** (p. 457).

5.91.2 Constructor & Destructor Documentation

5.91.2.1 dc_vars_t() [1/2] dc_vars_t::dc_vars_t (MHAParser::parser_t & p) [explicit]

Constructor initializes the configuration language variables' data members and inserts them into the parser p.

Parameters

p	The dc plugin interface instance into which to insert the configuration variables.

5.91.2.2 dc_vars_t() [2/2] dc::dc_vars_t::dc_vars_t (const dc_vars_t &) [default]

Default copy constructor is used to pass a copy of all configuration variables to the runtime configuration constructor **dc_t::dc_t** (p. 459).

5.91.3 Member Data Documentation

5.91.3.1 gtdata MHAParser::mfloat_t dc::dc_vars_t::gtdata

Gain table with gains in dB.

5.91.3.2 gtmin **MHAParser::vfloat_t** dc::dc_vars_t::gtmin

Narrow-band input levels in dB SPL specifying for each row of `gtdata` to which input level the first gain in this row applies.

5.91.3.3 gtstep **MHAParser::vfloat_t** dc::dc_vars_t::gtstep

Input level increment between elements for each row of `gtdata`.

5.91.3.4 taurmslevel **MHAParser::vfloat_t** dc::dc_vars_t::taurmslevel

Low-pass filter time constants for time-domain envelope extraction.

5.91.3.5 tauattack **MHAParser::vfloat_t** dc::dc_vars_t::tauattack

Low-pass filter time constants for level extraction when level rises.

5.91.3.6 taudecay **MHAParser::vfloat_t** dc::dc_vars_t::taudecay

Low-pass filter time constants for level extraction when level falls.

5.91.3.7 offset **MHAParser::vfloat_t** dc::dc_vars_t::offset

Row-specific input level corrections in dB to apply before gain lookup in `gtdata`.

5.91.3.8 filterbank **MHAParser::string_t** dc::dc_vars_t::filterbank

Configured name of filterbank plugin placed upstream of this `dc` plugin.

Used to lookup the AC variables published by the filterbank.

5.91.3.9 chname **MHAParser::string_t** dc::dc_vars_t::chname

Name of the AC variable containing the number of audiochannels before the filterbank.

5.91.3.10 bypass **MHAParser::bool_t** dc::dc_vars_t::bypass

Switch for bypassing the dynamic compression.

5.91.3.11 log_interp **MHAParser::bool_t** dc::dc_vars_t::log_interp

Interpolate gain table in dBs (vs.

interpolating linear factors).

5.91.3.12 clientid **MHAParser::string_t** dc::dc_vars_t::clientid

Metadata: Some ID of the hearing impaired subject.

5.91.3.13 gainrule **MHAParser::string_t** dc::dc_vars_t::gainrule

Metadata: Some name of the gain rule that was used to compute gtdata.

5.91.3.14 preset **MHAParser::string_t** dc::dc_vars_t::preset

Metadata: Some name given to the current setting.

5.91.3.15 input_level **MHAParser::vfloat_mon_t** dc::dc_vars_t::input_level

Narrow-band input levels of current block before attack/decay filter.

5.91.3.16 filtered_level `MHAParser::vfloat_mon_t dc::dc_vars_t::filtered_level`

Narrow-band input levels of current block after attack/decay filter.

5.91.3.17 center_frequencies `MHAParser::vfloat_mon_t dc::dc_vars_t::center_frequencies`

Center frequencies of upstream filterbank.

5.91.3.18 edge_frequencies `MHAParser::vfloat_mon_t dc::dc_vars_t::edge_frequencies`

Edge frequencies of upstream filterbank.

5.91.3.19 band_weights `MHAParser::vfloat_mon_t dc::dc_vars_t::band_weights`

Center frequencies of upstream filterbank.

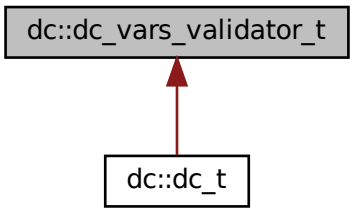
The documentation for this class was generated from the following files:

- **dc.hh**
- **dc.cpp**

5.92 dc::dc_vars_validator_t Class Reference

Consistency checker.

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)
  Expands vectors in v, checks for consistency.

5.92.1 Detailed Description

Consistency checker.

The runtime configuration class **dc_t** (p. 457) inherits from this class.

5.92.2 Constructor & Destructor Documentation

5.92.2.1 dc_vars_validator_t() dc_vars_validator_t::dc_vars_validator_t (

```
dc_vars_t & v,
unsigned int s,
mha_domain_t domain )
```

Expands vectors in v, checks for consistency.

Parameters

v	Reference to **dc_t** (p. 457)'s copy of the configuration variables.
s	Total number of compression bands: bands x channels.
domain	MHA_WAVEFORM or MHA_SPECTRUM.

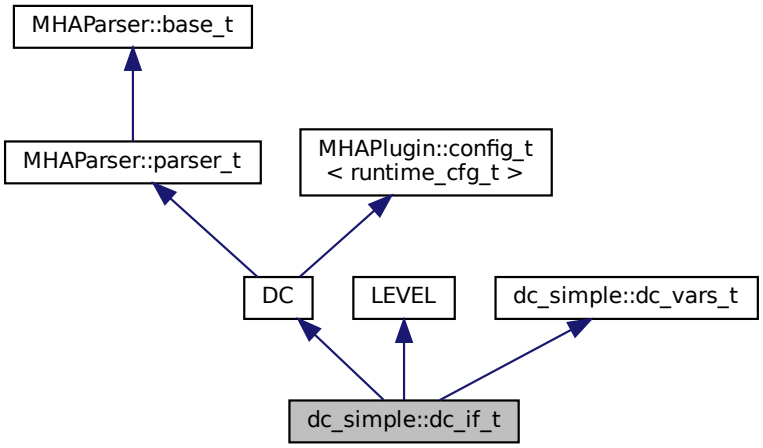
The documentation for this class was generated from the following files:

- **dc.hh**
- **dc.cpp**

5.93 dc_simple::dc_if_t Class Reference

interface class for **dc_simple** (p. 89)

Inheritance diagram for dc_simple::dc_if_t:



Public Member Functions

- **dc_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  *Constructor instantiates one **dc_simple** (p. 89) plugin.*
- void **prepare** (**mhaconfig_t** &tf)
  *Prepare **dc_simple** (p. 89) plugin for signal processing.*
- void **release** ()
  Sets prepared back to False.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)
  Main process callback.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
  Main process callback.

Private Member Functions

- void **update_dc** ()
  *Update **dc_t** (p. 477) runtime config when configuration parameters have changed.*
- void **update_level** ()
  *Update **level_smoother_t** (p. 487) runtime config when configuration parameters have changed.*

- void **has_been_modified** ()
- void **read_modified** ()
- void **update_level_mon** ()

  *Updates the data of variable mon_l in **dc_t** (p. 477).*
- void **update_gain_mon** ()

  *Updates the data of variable mon_g in **dc_t** (p. 477).*

Private Attributes

- **MHAParser::string_t clientid**

  MHA Parser variables.
- **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

5.93.1 Detailed Description

interface class for **dc_simple** (p. 89)

5.93.2 Constructor & Destructor Documentation

5.93.2.1 dc_if_t() `dc_simple::dc_if_t::dc_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor instantiates one **dc_simple** (p. 89) plugin.

5.93.3 Member Function Documentation

5.93.3.1 prepare() `void dc_simple::dc_if_t::prepare ( mhaconfig_t & tf ) [virtual]`

Prepare **dc_simple** (p. 89) plugin for signal processing.

Parameters

tf	signal_↩ dimensions

Implements **MHAPlugin::plugin_t< dc_t >** (p. 1301).

5.93.3.2 release() `void dc_simple::dc_if_t::release ( void ) [virtual]`

Sets prepared back to False.

Reimplemented from **MHAPlugin::plugin_t< dc_t >** (p. 1302).

5.93.3.3 process() [1/2] `mha_spec_t * dc_simple::dc_if_t::process ( mha_spec_t * s )`

Main process callback.

Takes mhatype spectrum input and calls type DC and LEVEL process methods, returning mhatype spectrum.

Parameters

s	input/output signal

5.93.3.4 process() [2/2] mha_wave_t * dc_simple::dc_if_t::process (mha_wave_t * s)

Main process callback.

Takes mhatype wave input and calls type DC and LEVEL process methods, returning mhatype wave.

Parameters

s	input/output signal

5.93.3.5 update_dc() void dc_simple::dc_if_t::update_dc ( ) [private]

Update **dc_t** (p. 477) runtime config when configuration parameters have changed.

5.93.3.6 update_level() void dc_simple::dc_if_t::update_level ( ) [private]

Update **level_smoother_t** (p. 487) runtime config when configuration parameters have changed.

5.93.3.7 has_been_modified() void dc_simple::dc_if_t::has_been_modified ( ) [inline], [private]

5.93.3.8 read_modified() void dc_simple::dc_if_t::read_modified ( ) [inline], [private]

5.93.3.9 update_level_mon() void dc_simple::dc_if_t::update_level_mon ( ) [private]

Updates the data of variable mon_l in **dc_t** (p. 477).

5.93.3.10 update_gain_mon() void dc_simple::dc_if_t::update_gain_mon ( ) [private]

Updates the data of variable mon_g in **dc_t** (p. 477).

5.93.4 Member Data Documentation

5.93.4.1 clientid MHAParser::string_t dc_simple::dc_if_t::clientid [private]

MHA Parser variables.

5.93.4.2 gainrule MHAParser::string_t dc_simple::dc_if_t::gainrule [private]

5.93.4.3 preset MHAParser::string_t dc_simple::dc_if_t::preset [private]

5.93.4.4 modified MHAParser::int_mon_t dc_simple::dc_if_t::modified [private]

5.93.4.5 mon_l MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_l [private]

5.93.4.6 mon_g MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_g [private]

5.93.4.7 filterbank MHAParser::string_t dc_simple::dc_if_t::filterbank [private]

5.93.4.8 center_frequencies

`MHAParser::vfloat_mon_t dc_simple::dc_if_t::center_frequencies [private]`

5.93.4.9 edge_frequencies

`MHAParser::vfloat_mon_t dc_simple::dc_if_t::edge_frequencies [private]`

5.93.4.10 patchbay

`MHAEvents::patchbay_t< dc_if_t> dc_simple::dc_if_t::patchbay [private]`

5.93.4.11 prepared

`bool dc_simple::dc_if_t::prepared [private]`

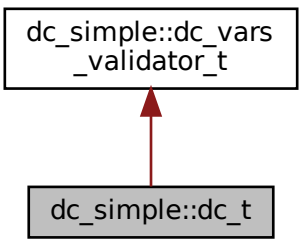
The documentation for this class was generated from the following files:

- **dc_simple.hh**
- **dc_simple.cpp**

5.94 dc_simple::dc_t Class Reference

Runtime config class for **dc_simple** (p. 89) plugin.

Inheritance diagram for dc_simple::dc_t:



Classes

- class **line_t**
  Helper class for usage in computing compression, expansion and limiting.

Public Member Functions

- **dc_t** (const **dc_vars_t** &vars, unsigned int nch)
  Constructor.
- **mha_spec_t** * **process** (**mha_spec_t** *s, **mha_wave_t** *level_db)
  Process callback.
- **mha_wave_t** * **process** (**mha_wave_t** *s, **mha_wave_t** *level_db)
  Process callback.

Public Attributes

- std::vector< float > **mon_l**
- std::vector< float > **mon_g**

Private Attributes

- std::vector< **mha_real_t** > **expansion_threshold**
  Threshold below which to apply expansion.
- std::vector< **mha_real_t** > **limiter_threshold**
  Threshold below which to compress.
- std::vector< **line_t** > **compression**
  The linear function for applying compression.
- std::vector< **line_t** > **expansion**
  The linear function for applying expansion.
- std::vector< **line_t** > **limiter**
  The linear function for applying limiting.
- std::vector< **mha_real_t** > **maxgain**
  Gain should not exceed this value.
- unsigned int **nbands**
  Number of bands.

Additional Inherited Members

5.94.1 Detailed Description

Runtime config class for **dc_simple** (p. 89) plugin.

5.94.2 Constructor & Destructor Documentation

5.94.2.1 dc_t() dc_simple::dc_t::dc_t (

```
const dc_vars_t & vars,
unsigned int nch )
```

Constructor.

5.94.3 Member Function Documentation

5.94.3.1 process() [1/2] mha_spec_t * dc_simple::dc_t::process (

```
mha_spec_t * s,
mha_wave_t * level_db )
```

Process callback.

Compresses, expands or limits depending on the gain settings and filtered signal levels. Compresses the spectrum input signal in individual bands.

Parameters

s	input/output signal
level_db	smoothed levels of input signal in dB SPL

Returns

s. The input signal is modified in place.

5.94.3.2 process() [2/2]

```
mha_wave_t * dc_simple::dc_t::process (
        mha_wave_t * s,
        mha_wave_t * level_db )
```

Process callback.

Compresses, expands or limits depending on the gain settings and filtered signal levels. Compresses the waveform input signal in individual bands.

Parameters

s	input/output signal
level_db	smoothed levels of input signal in dB SPL

Returns

s. The input signal is modified in place.

5.94.4 Member Data Documentation

5.94.4.1 expansion_threshold

```
std::vector< mha_real_t> dc_simple::dc_t::expansion_threshold [private]
```

Threshold below which to apply expansion.

5.94.4.2 limiter_threshold

```
std::vector< mha_real_t> dc_simple::dc_t::limiter_threshold [private]
```

Threshold below which to compress.

5.94.4.3 compression

```
std::vector< line_t> dc_simple::dc_t::compression [private]
```

The linear function for applying compression.

5.94.4.4 expansion

`std::vector< line_t> dc_simple::dc_t::expansion [private]`

The linear function for applying expansion.

5.94.4.5 limiter

`std::vector< line_t> dc_simple::dc_t::limiter [private]`

The linear function for applying limiting.

5.94.4.6 maxgain

`std::vector< mha_real_t> dc_simple::dc_t::maxgain [private]`

Gain should not exceed this value.

5.94.4.7 nbands

`unsigned int dc_simple::dc_t::nbands [private]`

Number of bands.

5.94.4.8 mon_l

`std::vector<float> dc_simple::dc_t::mon_l`

5.94.4.9 mon_g

`std::vector<float> dc_simple::dc_t::mon_g`

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

- **dc_simple.hh**
- **dc_simple.cpp**

5.95 dc_simple::dc_t::line_t Class Reference

Helper class for usage in computing compression, expansion and limiting.

Public Member Functions

- **line_t** (**mha_real_t** x1, **mha_real_t** y1, **mha_real_t** x2, **mha_real_t** y2)
  Constructor used for compression which takes two x and y coordinates each to find m and y0
- **line_t** (**mha_real_t** x1, **mha_real_t** y1, **mha_real_t** m_)
  Constructor used for expansion and limiting which takes x and y coordinates and a gradient, giving y0.
- **mha_real_t operator()** (**mha_real_t** x)
  Operator overload which returns.

Private Attributes

- **mha_real_t m**
  The gradient and y-intercept.
- **mha_real_t y0**

5.95.1 Detailed Description

Helper class for usage in computing compression, expansion and limiting.

5.95.2 Constructor & Destructor Documentation

5.95.2.1 line_t() [1/2] dc_simple::dc_t::line_t::line_t (

```
mha_real_t x1,
mha_real_t y1,
mha_real_t x2,
mha_real_t y2 )
```

Constructor used for compression which takes two x and y coordinates each to find m and y0

5.95.2.2 line_t() [2/2] dc_simple::dc_t::line_t::line_t (

```
mha_real_t x1,
mha_real_t y1,
mha_real_t m_ )
```

Constructor used for expansion and limiting which takes x and y coordinates and a gradient, giving y0.

5.95.3 Member Function Documentation

5.95.3.1 operator()() `mha_real_t dc_simple::dc_t::line_t::operator() ( mha_real_t x ) [inline]`

Operator overload which returns.

Parameters

x	

Returns

y values mapped to x by a linear equation with gradient m and intercept y0

5.95.4 Member Data Documentation

5.95.4.1 m `mha_real_t dc_simple::dc_t::line_t::m [private]`

The gradient and y-intercept.

5.95.4.2 y0 `mha_real_t dc_simple::dc_t::line_t::y0 [private]`

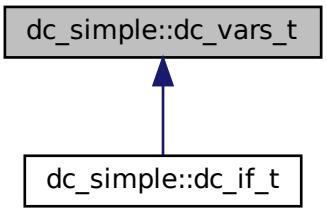
The documentation for this class was generated from the following files:

- **dc_simple.hh**
- **dc_simple.cpp**

5.96 dc_simple::dc_vars_t Class Reference

class for **dc_simple** (p. 89) plugin which registers variables to **MHAParser** (p. 125).

Inheritance diagram for dc_simple::dc_vars_t:



Public Member Functions

- **dc_vars_t** (**MHAParser::parser_t** &p)

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**

5.96.1 Detailed Description

class for **dc_simple** (p. 89) plugin which registers variables to **MHAParser** (p. 125).

5.96.2 Constructor & Destructor Documentation

5.96.2.1 dc_vars_t()

`dc_simple::dc_vars_t::dc_vars_t (`
MHAParser::parser_t & *p*)

5.96.3 Member Data Documentation

5.96.3.1 g50

MHAParser::vfloat_t dc_simple::dc_vars_t::g50

5.96.3.2 g80

MHAParser::vfloat_t dc_simple::dc_vars_t::g80

5.96.3.3 maxgain

MHAParser::vfloat_t dc_simple::dc_vars_t::maxgain

5.96.3.4 expansion_threshold

MHAParser::vfloat_t dc_simple::dc_vars_t::expansion↩
_threshold

5.96.3.5 expansion_slope

MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_↩
slope

5.96.3.6 limiter_threshold

MHAParser::vfloat_t dc_simple::dc_vars_t::limiter_↩
threshold

5.96.3.7 tauattack

MHAParser::vfloat_t dc_simple::dc_vars_t::tauattack

5.96.3.8 taudecay `MHAParser::vfloat_t dc_simple::dc_vars_t::taudecay`

5.96.3.9 bypass `MHAParser::bool_t dc_simple::dc_vars_t::bypass`

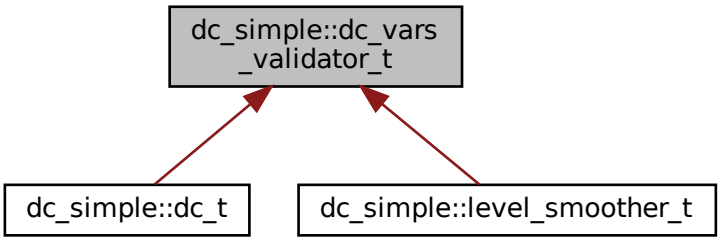
The documentation for this class was generated from the following files:

- **dc_simple.hh**
- **dc_simple.cpp**

5.97 dc_simple::dc_vars_validator_t Class Reference

Helper class to check sizes of configuration variable vectors.

Inheritance diagram for dc_simple::dc_vars_validator_t:



Public Member Functions

- **dc_vars_validator_t** (const **dc_vars_t** &v, unsigned int s)
  Checks that all vectors in `v` *have size* `s` *or size 1.*

5.97.1 Detailed Description

Helper class to check sizes of configuration variable vectors.

5.97.2 Constructor & Destructor Documentation

5.97.2.1 dc_vars_validator_t() dc_simple::dc_vars_validator_t::dc_vars_validator_t (

```
const dc_vars_t & v,
unsigned int s )
```

Checks that all vectors in v have size s or size 1.

Parameters

v	Aggregation of vectors to check the sizes of.
s	Desired vector size for all vectors

Exceptions

MHA_Error *(p. 906)*	if s == 0.
MHA_Error *(p. 906)*	if the size of any vector in v is neither s nor 1.

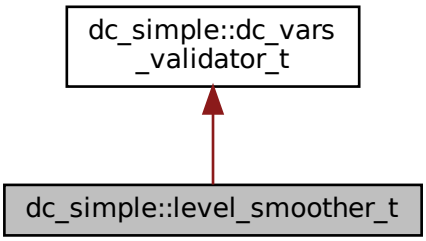
The documentation for this class was generated from the following files:

- **dc_simple.hh**
- **dc_simple.cpp**

5.98 dc_simple::level_smoother_t Class Reference

Class which computes smoothed input levels on individual bands, using an attack and release filter, which are a first order low pass filter and a maximum tracker filter, respectively.

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** *s)
  Process callback.
- **mha_wave_t** * **process** (**mha_wave_t** *s)
  Process callback.

Private Attributes

- **MHAFilter::o1flt_lowpass_t attack**
  first order low pass attack filter
- **MHAFilter::o1flt_maxtrack_t decay**
  maximum tracker decay filter
- unsigned int **nbands**
  Total number of frequency bands of this compressor.
- unsigned int **fftlen**
- **MHASignal::waveform_t level_wave**
- **MHASignal::waveform_t level_spec**

Additional Inherited Members

5.98.1 Detailed Description

Class which computes smoothed input levels on individual bands, using an attack and release filter, which are a first order low pass filter and a maximum tracker filter, respectively.

5.98.2 Constructor & Destructor Documentation

5.98.2.1 level_smoother_t()

```
dc_simple::level_smoother_t::level_smoother_t (
    const dc_vars_t & vars,
    mha_real_t filter_rate,
    mhaconfig_t buscfg )
```

5.98.3 Member Function Documentation

5.98.3.1 process() [1/2] mha_wave_t * dc_simple::level_smoother_t::process (mha_spec_t * s)

Process callback.

Computes smoothed levels from the input mha type spectrum by applying a lowpass and maximum tracker filter

Returns

smoothed input levels in dB SPL

Parameters

s	input signal

5.98.3.2 process() [2/2] mha_wave_t * dc_simple::level_smoother_t::process (mha_wave_t * s)

Process callback.

Computes smoothed levels from the input mha type waveform over individual bands by applying a lowpass and maximum tracker filter

Returns

smoothed input levels in dB SPL

Parameters

s	input/output signal

5.98.4 Member Data Documentation

5.98.4.1 attack `MHAFilter::o1flt_lowpass_t dc_simple::level_smoother_t::attack [private]`

first order low pass attack filter

5.98.4.2 decay `MHAFilter::o1flt_maxtrack_t dc_simple::level_smoother_t::decay [private]`

maximum tracker decay filter

5.98.4.3 nbands `unsigned int dc_simple::level_smoother_t::nbands [private]`

Total number of frequency bands of this compressor.

5.98.4.4 fftlen `unsigned int dc_simple::level_smoother_t::fftlen [private]`

5.98.4.5 level_wave `MHASignal::waveform_t dc_simple::level_smoother_t::level_wave [private]`

5.98.4.6 level_spec `MHASignal::waveform_t dc_simple::level_smoother_t::level_spec [private]`

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

- **dc_simple.hh**
- **dc_simple.cpp**

5.99 DConvLayer Struct Reference

Public Attributes

- const **rnn_bias** * **bias**
- const **rnn_weight** * **input_weights**
- **counter** **nb_lenfilt**
- **counter** **nb_neurons**
- **counter** **activation**

5.99.1 Member Data Documentation

5.99.1.1 bias

`const` **`rnn_bias`** * `DConvLayer::bias`

5.99.1.2 input_weights

`const` **`rnn_weight`** * `DConvLayer::input_weights`

5.99.1.3 nb_lenfilt

`counter` `DConvLayer::nb_lenfilt`

5.99.1.4 nb_neurons

`counter` `DConvLayer::nb_neurons`

5.99.1.5 activation

`counter` `DConvLayer::activation`

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

- **gcfsnet_bin/rnn.h**

5.100 DConvLayer1x1 Struct Reference

Public Attributes

- const **rnn_bias** * **bias**
- const **rnn_weight** * **input_weights**
- **counter nb_inputs**
- **counter nb_neurons**
- **counter activation**

5.100.1 Member Data Documentation

5.100.1.1 bias const **rnn_bias** * DConvLayer1x1::bias

5.100.1.2 input_weights const **rnn_weight** * DConvLayer1x1::input_weights

5.100.1.3 nb_inputs **counter** DConvLayer1x1::nb_inputs

5.100.1.4 nb_neurons **counter** DConvLayer1x1::nb_neurons

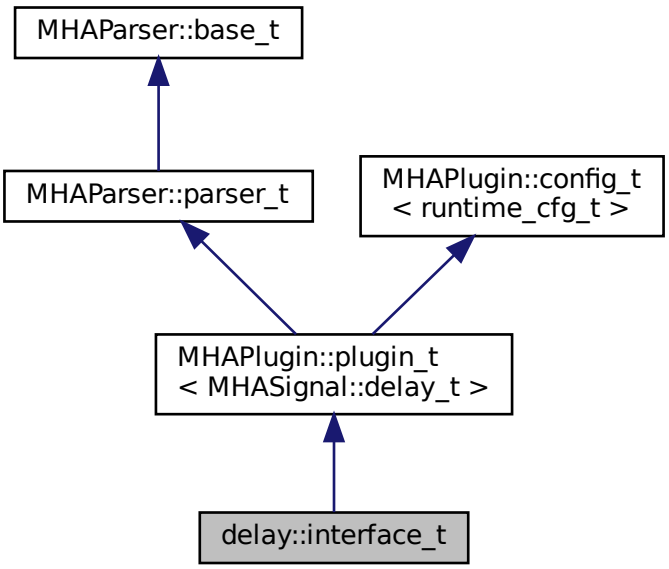
5.100.1.5 activation **counter** DConvLayer1x1::activation

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

- **gcfsnet_bin/rnn.h**

5.101 delay::interface_t Class Reference

Inheritance diagram for delay::interface_t:



Public Member Functions

- **interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.101.1 Constructor & Destructor Documentation

5.101.1.1 interface_t()

```
delay::interface_t::interface_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.101.2 Member Function Documentation

5.101.2.1 prepare()

```
void delay::interface_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< MHASignal::delay_t >** (p. 1301).

5.101.2.2 process()

```
mha_wave_t * delay::interface_t::process (
    mha_wave_t * s )
```

5.101.2.3 update()

```
void delay::interface_t::update ( ) [private]
```

5.101.3 Member Data Documentation

5.101.3.1 delays

```
MHAParser::vint_t delay::interface_t::delays [private]
```

5.101.3.2 patchbay `MHAEvents::patchbay_t< interface_t> delay::interface_t::patchbay [private]`

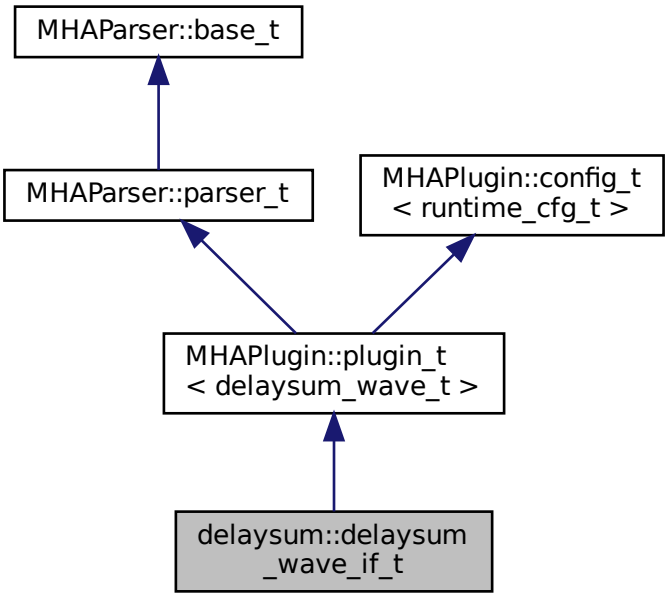
The documentation for this class was generated from the following files:

- **delay.hh**
- **delay.cpp**

5.102 delaysum::delaysum_wave_if_t Class Reference

Interface class for the delaysum plugin.

Inheritance diagram for delaysum::delaysum_wave_if_t:



Public Member Functions

- **delaysum_wave_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
- **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_wave_if_t > patchbay**
  The patchbay to react to config changes.

Additional Inherited Members

5.102.1 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.

5.102.2 Constructor & Destructor Documentation

5.102.2.1 delaysum_wave_if_t() `delaysum::delaysum_wave_if_t::delaysum_wave_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.102.3 Member Function Documentation

5.102.3.1 process() `mha_wave_t * delaysum::delaysum_wave_if_t::process (`

```
mha_wave_t * wave )
```

5.102.3.2 prepare() void delaysum::delaysum_wave_if_t::prepare (mhaconfig_t & tfcfg) [virtual]

Implements **MHAPlugin::plugin_t< delaysum_wave_t >** (p. 1301).

5.102.3.3 release() void delaysum::delaysum_wave_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< delaysum_wave_t >** (p. 1302).

5.102.3.4 update_cfg() void delaysum::delaysum_wave_if_t::update_cfg ( ) [private]

5.102.4 Member Data Documentation

5.102.4.1 weights MHAParser::vfloat_t delaysum::delaysum_wave_if_t::weights [private]

Linear weights to be multiplied with the audio signal, one factor for each channel.

Order is [chan0, chan1, ...]

5.102.4.2 delay MHAParser::vint_t delaysum::delaysum_wave_if_t::delay [private]

vector of channel-specific delays, in samples.

5.102.4.3 patchbay MHAEvents::patchbay_t< delaysum_wave_if_t> delaysum::delaysum_wave_if_t::patchbay [private]

The patchbay to react to config changes.

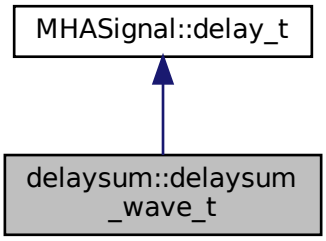
The documentation for this class was generated from the following file:

- **delaysum_wave.cpp**

5.103 delaysum::delaysum_wave_t Class Reference

Runtime configuration of the delaysum_wave plugin.

Inheritance diagram for delaysum::delaysum_wave_t:



Public Member Functions

- **delaysum_wave_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.

5.103.1 Detailed Description

Runtime configuration of the delaysum_wave 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.

5.103.2 Constructor & Destructor Documentation

5.103.2.1 delaysum_wave_t()

```
delaysum::delaysum_wave_t::delaysum_wave_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.

5.103.3 Member Function Documentation

5.103.3.1 process()

```
mha_wave_t * delaysum::delaysum_wave_t::process (
    mha_wave_t * signal )
```

5.103.4 Member Data Documentation

5.103.4.1 weights

```
std::vector< mha_real_t> delaysum::delaysum_wave_t::weights
[private]
```

Relative weights for each channel. Order is [chan0, chan1, ...].

5.103.4.2 out `MHASignal::waveform_t delaysum::delaysum_wave_t::out [private]`

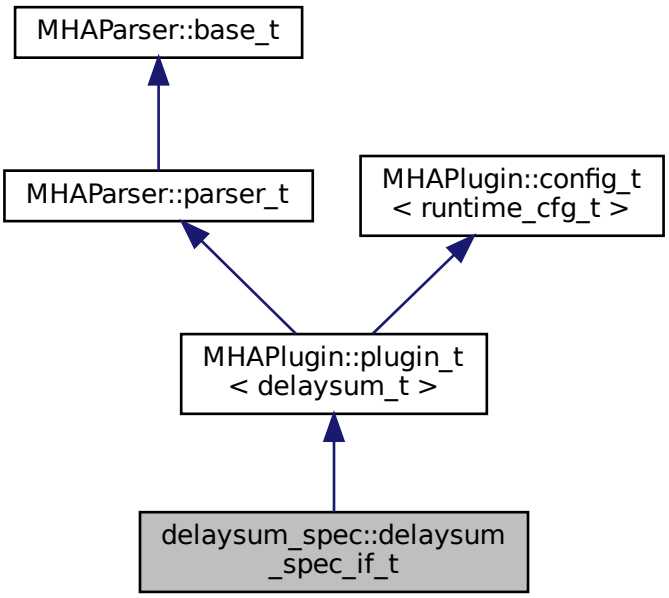
Output waveform.

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

- **delaysum_wave.cpp**

5.104 delaysum_spec::delaysum_spec_if_t Class Reference

Inheritance diagram for delaysum_spec::delaysum_spec_if_t:



Public Member Functions

- **delaysum_spec_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::vfloat_t groupdelay**
- **MHAParser::vfloat_t gain**
- **MHAEvents::patchbay_t< delaysum_spec_if_t > patchbay**

Additional Inherited Members

5.104.1 Constructor & Destructor Documentation

5.104.1.1 delaysum_spec_if_t() `delaysum_spec::delaysum_spec_if_t::delaysum_spec_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.104.2 Member Function Documentation

5.104.2.1 process() `mha_spec_t * delaysum_spec::delaysum_spec_if_t::process (`

```
mha_spec_t * spec )
```

5.104.2.2 prepare() `void delaysum_spec::delaysum_spec_if_t::prepare (`

```
mhaconfig_t & signal_info ) [virtual]
```

Implements **MHAPlugin::plugin_t< delaysum_t >** (p. 1301).

5.104.2.3 update_cfg() `void delaysum_spec::delaysum_spec_if_t::update_cfg ( ) [private]`

5.104.3 Member Data Documentation

5.104.3.1 groupdelay `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::groupdelay [private]`

5.104.3.2 gain `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::gain [private]`

5.104.3.3 patchbay `MHAEvents::patchbay_t< delaysum_spec_if_t> delaysum_spec::delaysum_spec_if_t::patchbay [private]`

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

- **delaysum_spec.cpp**

5.105 delaysum_spec::delaysum_t Class Reference

Public Member Functions

- **delaysum_t** (std::vector< float > groupdelay, std::vector< float > gain, unsigned int nChannels, unsigned int nFFT, float fs)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::spectrum_t scale**
- **MHASignal::spectrum_t output**

5.105.1 Constructor & Destructor Documentation

5.105.1.1 delaysum_t()

```
delaysum_spec::delaysum_t::delaysum_t (
    std::vector< float > groupdelay,
    std::vector< float > gain,
    unsigned int nChannels,
    unsigned int nFFT,
    float fs )
```

5.105.2 Member Function Documentation

5.105.2.1 process() `mha_spec_t * delaysum_spec::delaysum_t::process ( mha_spec_t * spec )`

5.105.3 Member Data Documentation

5.105.3.1 scale `MHASignal::spectrum_t delaysum_spec::delaysum_t::scale [private]`

5.105.3.2 output `MHASignal::spectrum_t delaysum_spec::delaysum_t::output [private]`

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

- **delaysum_spec.cpp**

5.106 DenoiseState Struct Reference

Public Attributes

- **RNNState rnn**
- float * **features**
- float * **filtering_buffer_r**
- float * **filtering_buffer_i**
- float * **filter_b**
- float * **filter_t**
- int **buffer_start_idx**
- int **buffer_write_idx**

5.106.1 Member Data Documentation

5.106.1.1 rnn `RNNState DenoiseState::rnn`

5.106.1.2 features `float * DenoiseState::features`

5.106.1.3 filtering_buffer_r `float * DenoiseState::filtering_buffer_r`

5.106.1.4 filtering_buffer_i `float * DenoiseState::filtering_buffer_i`

5.106.1.5 filter_b `float * DenoiseState::filter_b`

5.106.1.6 filter_t `float * DenoiseState::filter_t`

5.106.1.7 buffer_start_idx `int DenoiseState::buffer_start_idx`

5.106.1.8 buffer_write_idx `int DenoiseState::buffer_write_idx`

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

- **gcfsnet_bin/denoise.c**

5.107 DenseLayer Struct Reference

Public Attributes

- const **rnn_bias** * **bias**
- const **rnn_weight** * **input_weights**
- **counter** **nb_inputs**
- **counter** **nb_neurons**
- **counter** **activation**

5.107.1 Member Data Documentation

5.107.1.1 bias const **rnn_bias** * DenseLayer::bias

5.107.1.2 input_weights const **rnn_weight** * DenseLayer::input_weights

5.107.1.3 nb_inputs **counter** DenseLayer::nb_inputs

5.107.1.4 nb_neurons **counter** DenseLayer::nb_neurons

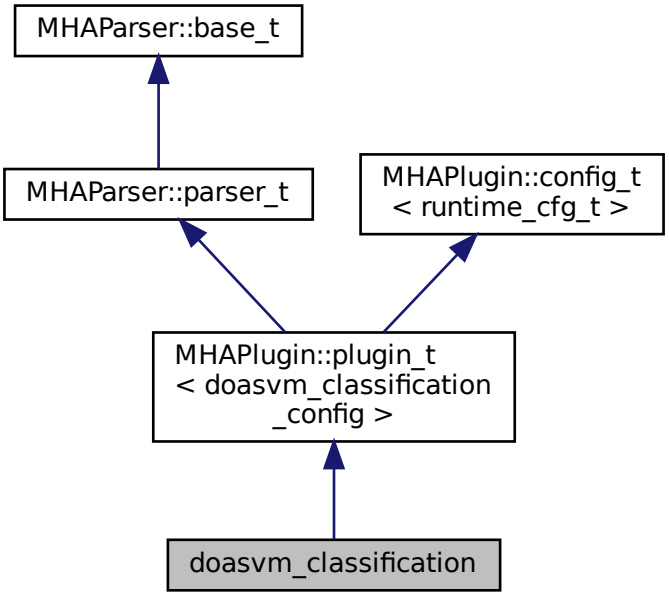
5.107.1.5 activation **counter** DenseLayer::activation

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

- **gcfsnet_bin/rnn.h**

5.108 doasvm_classification Class Reference

Inheritance diagram for doasvm_classification:



Public Member Functions

- **doasvm_classification** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _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

5.108.1 Constructor & Destructor Documentation

5.108.1.1 doasvm_classification()

```
doasvm_classification::doasvm_classification (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

Constructs our plugin.

5.108.1.2 ~doasvm_classification()

```
doasvm_classification::~doasvm_classification (
)
```

5.108.2 Member Function Documentation

5.108.2.1 process()

```
mha_wave_t * doasvm_classification::process (
    mha_wave_t * signal )
```

Checks for the most recent configuration and defers processing to it.

5.108.2.2 prepare() `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. 1301).

5.108.2.3 release() `void doasvm_classification::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< doasvm_classification_config >** (p. 1302).

5.108.2.4 update_cfg() `void doasvm_classification::update_cfg ( ) [private]`

5.108.3 Member Data Documentation

5.108.3.1 angles `MHAParser::vfloat_t doasvm_classification::angles`

5.108.3.2 w `MHAParser::mfloat_t doasvm_classification::w`

5.108.3.3 b `MHAParser::vfloat_t doasvm_classification::b`

5.108.3.4 x MHAParser::vfloat_t doasvm_classification::x

5.108.3.5 y MHAParser::vfloat_t doasvm_classification::y

5.108.3.6 p_name MHAParser::string_t doasvm_classification::p_name

5.108.3.7 max_p_ind_name MHAParser::string_t doasvm_classification::max_p_ind↩ _name

5.108.3.8 vGCC_name MHAParser::string_t doasvm_classification::vGCC_name

5.108.3.9 patchbay 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**

5.109 doasvm_classification_config Class Reference

Public Member Functions

- **doasvm_classification_config** (**MHA_AC::algo_comm_t** & **ac**, **doasvm_↩ classification** ∗_doasvm)
- ∼**doasvm_classification_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **insert_ac_variables** ()

  Insert or reinsert AC variables p, p_max into AC space.

Public Attributes

- **MHA_AC::algo_comm_t & ac**
- **doasvm_classification * doasvm**
- **MHA_AC::waveform_t p**
- **MHA_AC::int_t p_max**
- **mha_wave_t c**

5.109.1 Constructor & Destructor Documentation

5.109.1.1 doasvm_classification_config()

```
doasvm_classification_config::doasvm_classification_config (
            MHA_AC::algo_comm_t & ac,
            doasvm_classification * _doasvm )
```

5.109.1.2 ~doasvm_classification_config()

```
doasvm_classification_config::~doasvm_classification_config ( )
```

5.109.2 Member Function Documentation

5.109.2.1 process()

```
mha_wave_t * doasvm_classification_config::process (
            mha_wave_t * wave )
```

5.109.2.2 insert_ac_variables()

```
void doasvm_classification_config::insert_ac_variables ( )
```

Insert or reinsert AC variables p, p_max into AC space.

5.109.3 Member Data Documentation

5.109.3.1 ac

`MHA_AC::algo_comm_t& doasvm_classification_config::ac`

5.109.3.2 doasvm

`doasvm_classification* doasvm_classification_config::doasvm`

5.109.3.3 p

`MHA_AC::waveform_t doasvm_classification_config::p`

5.109.3.4 p_max

`MHA_AC::int_t doasvm_classification_config::p_max`

5.109.3.5 c

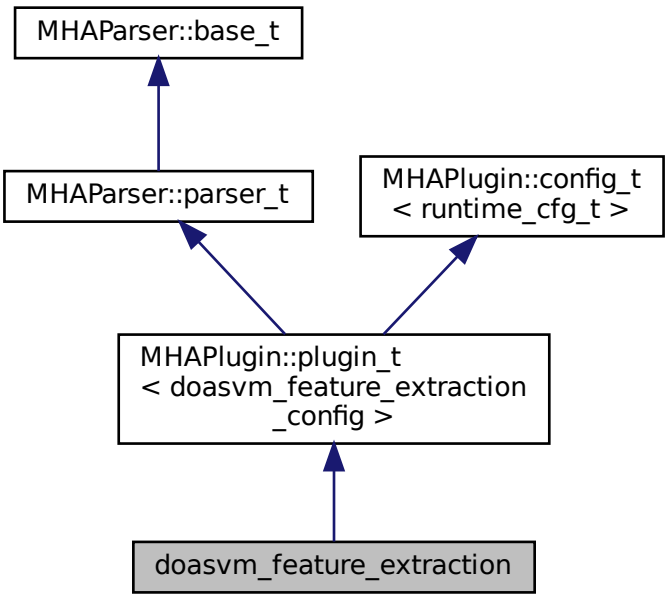
`mha_wave_t doasvm_classification_config::c`

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

5.110 doasvm_feature_extraction Class Reference

Inheritance diagram for doasvm_feature_extraction:



Public Member Functions

- **doasvm_feature_extraction** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.110.1 Constructor & Destructor Documentation

5.110.1.1 doasvm_feature_extraction() `doasvm_feature_extraction::doasvm_feature_extraction (`

```
            MHA_AC::algo_comm_t & iac,
           const std::string & configured_name )
```

Constructs our plugin.

5.110.1.2 ~doasvm_feature_extraction() doasvm_feature_extraction::~doasvm_feature_extraction ( )

5.110.2 Member Function Documentation

5.110.2.1 process() mha_wave_t * doasvm_feature_extraction::process (mha_wave_t * signal)

Checks for the most recent configuration and defers processing to it.

5.110.2.2 prepare() 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. 1301).

5.110.2.3 release() void doasvm_feature_extraction::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< doasvm_feature_extraction_config >** (p. 1302).

5.110.2.4 update_cfg() void doasvm_feature_extraction::update_cfg ( ) [private]

5.110.3 Member Data Documentation

5.110.3.1 fftlen **MHAParser::int_t** doasvm_feature_extraction::fftlen

5.110.3.2 max_lag **MHAParser::int_t** doasvm_feature_extraction::max_lag

5.110.3.3 nupsample **MHAParser::int_t** doasvm_feature_extraction::nupsample

5.110.3.4 vGCC_name **MHAParser::string_t** doasvm_feature_extraction::vGCC_name

5.110.3.5 patchbay **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**

5.111 doasvm_feature_extraction_config Class Reference

Public Member Functions

- **doasvm_feature_extraction_config** (**MHA_AC::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**

5.111.1 Constructor & Destructor Documentation

5.111.1.1 doasvm_feature_extraction_config()

```
doasvm_feature_extraction_config::doasvm_feature_extraction_config (
            MHA_AC::algo_comm_t & ac,
           const  mhaconfig_t in_cfg,
            doasvm_feature_extraction * _doagcc )
```

5.111.1.2 ~doasvm_feature_extraction_config()

```
doasvm_feature_extraction_config::~doasvm_feature_extraction_config ( )
```

5.111.2 Member Function Documentation

5.111.2.1 process()

```
mha_wave_t * doasvm_feature_extraction_config::process (
            mha_wave_t * wave )
```

5.111.3 Member Data Documentation

5.111.3.1 doagcc

`doasvm_feature_extraction* doasvm_feature_extraction_config::doagcc`

5.111.3.2 wndlen

`unsigned int doasvm_feature_extraction_config::wndlen`

5.111.3.3 fftlen

`unsigned int doasvm_feature_extraction_config::fftlen`

5.111.3.4 G_length

`unsigned int doasvm_feature_extraction_config::G_length`

5.111.3.5 GCC_start

`unsigned int doasvm_feature_extraction_config::GCC_start`

5.111.3.6 GCC_end

`unsigned int doasvm_feature_extraction_config::GCC_end`

5.111.3.7 vGCC_ac

`MHA_AC::waveform_t doasvm_feature_extraction_config::vGCC_ac`

5.111.3.8 fft

`mha_fft_t doasvm_feature_extraction_config::fft`

5.111.3.9 ifft `mha_fft_t` doasvm_feature_extraction_config::ifft

5.111.3.10 hifftwin_sum `double` doasvm_feature_extraction_config::hifftwin_sum

5.111.3.11 proc_wave `MHASignal::waveform_t` doasvm_feature_extraction_config↩ ::proc_wave

5.111.3.12 hwin `MHASignal::waveform_t` doasvm_feature_extraction_config::hwin

5.111.3.13 hifftwin `MHASignal::waveform_t` doasvm_feature_extraction_config::hifftwin

5.111.3.14 vGCC `MHASignal::waveform_t` doasvm_feature_extraction_config::vGCC

5.111.3.15 in_spec `MHASignal::spectrum_t` doasvm_feature_extraction_config::in_↩ spec

5.111.3.16 G `MHASignal::spectrum_t` doasvm_feature_extraction_config::G

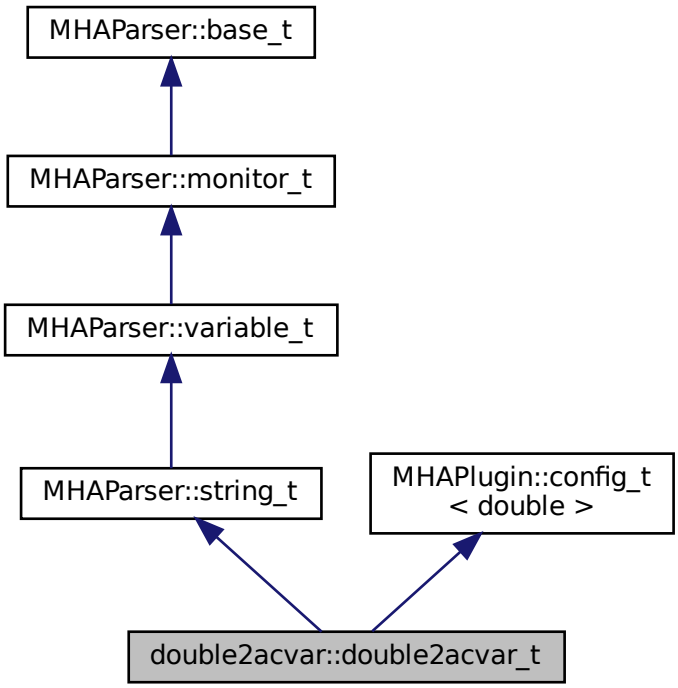
The documentation for this class was generated from the following files:

- **doasvm_feature_extraction.h**
- **doasvm_feature_extraction.cpp**

5.112 double2acvar::double2acvar_t Class Reference

Plugin interface class for **double2acvar** (p. 92).

Inheritance diagram for double2acvar::double2acvar_t:



Public Member Functions

- **double2acvar_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Standard plugin constructor.
- ~**double2acvar_t** ()=default
- template<class T >
  T * **process** (T *s)
  process() *(p. 519) does not alter the signal and has same implementation regardless of signal domain.*
- void **poll_latest_value_and_reinsert** ()
  Called from ***process()*** *(p. 519) and, when allowed, also from* ***on_configuration_update()*** *(p. 520).*
- void **prepare_** (**mhaconfig_t** &)
  Prepare method as expected by the plugin interface macros.
- void **release_** ()
  Release method as expected by the plugin interface macros.
- void **on_configuration_update** ()
  Callback function on write access to the string configuration value.

Private Attributes

- **MHA_AC::double_t ac_double**
  AC variable inserted by this plugin.
- **MHAEvents::patchbay_t< double2acvar_t > patchbay**
  Callback router.
- bool **is_prepared**
  Flag to keep track if we are currently prepared.

Additional Inherited Members

5.112.1 Detailed Description

Plugin interface class for **double2acvar** (p. 92).

5.112.2 Constructor & Destructor Documentation

5.112.2.1 double2acvar_t() `double2acvar::double2acvar_t::double2acvar_t ( MHA_AC::algo_comm_t & iac, const std::string & configured_name )`

Standard plugin constructor.

Parameters

iac	Algorithm communication variable space.
configured_name	Configured name of this plugin, also used as name of the AC variable.

5.112.2.2 ~double2acvar_t() `double2acvar::double2acvar_t::~double2acvar_t ( ) [default]`

5.112.3 Member Function Documentation

5.112.3.1 process() template<class T >

```
T * double2acvar::double2acvar_t::process (
            T * s )
```

process() (p. 519) does not alter the signal and has same implementation regardless of signal domain.

Parameters

s	Pointer to input signal structure, **mha_wave_t** (p. 985) or **mha_spec_t** (p. 937).

Returns

s, unaltered.

5.112.3.2 poll_latest_value_and_reinsert() void double2acvar::double2acvar_t::poll↩ _latest_value_and_reinsert ( )

Called from **process()** (p. 519) and, when allowed, also from **on_configuration_update()** (p. 520).

poll_latest_value_and_reinsert() (p. 520) retrieves the latest configured value and reinserts the AC variable into the AC space.

5.112.3.3 prepare_() void double2acvar::double2acvar_t::prepare_ (**mhaconfig_t** &)

Prepare method as expected by the plugin interface macros.

Parameter is not used nor altered. Sets is_prepared flag.

5.112.3.4 release_() void double2acvar::double2acvar_t::release_ ( )

Release method as expected by the plugin interface macros.

Resets is_prepared flag.

5.112.3.5 on_configuration_update() void double2acvar::double2acvar_t::on_configuration_update ( )

Callback function on write access to the string configuration value.

5.112.4 Member Data Documentation

5.112.4.1 ac_double MHA_AC::double_t double2acvar::double2acvar_t::ac_double [private]

AC variable inserted by this plugin.

5.112.4.2 patchbay MHAEvents::patchbay_t< double2acvar_t> double2acvar::double2acvar_t::patchbay [private]

Callback router.

5.112.4.3 is_prepared bool double2acvar::double2acvar_t::is_prepared [private]

Flag to keep track if we are currently prepared.

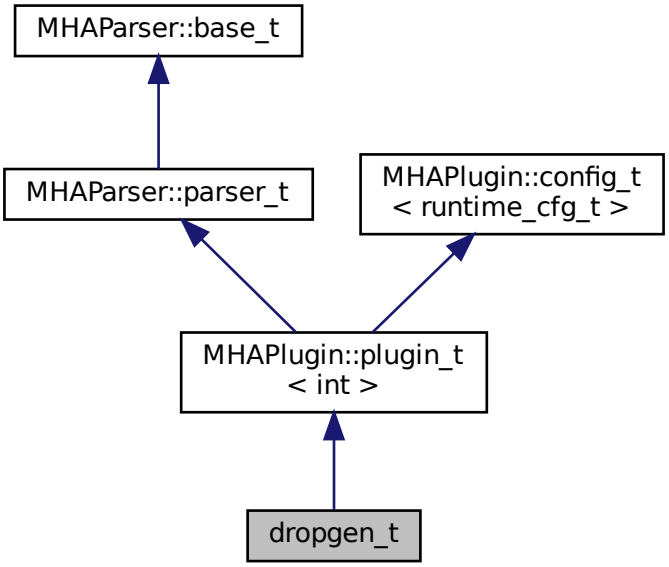
If we are, then signal processing is active and AC variables may only be accessed when MHA is currently executing out **process()** (p. 519) method.

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

- **double2acvar.cpp**

5.113 dropgen_t Class Reference

Inheritance diagram for dropgen_t:



Public Member Functions

- **dropgen_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Public Attributes

- **MHAParser::float_t min_sleep_time**
- **MHAParser::float_t max_sleep_time**
- **MHAParser::float_t chance**
- **MHAEvents::patchbay_t**< **dropgen_t** > **patchbay**
- std::random_device **r**
- std::mt19937 **random_engine**
- std::uniform_real_distribution **dis**

Additional Inherited Members

5.113.1 Constructor & Destructor Documentation

5.113.1.1 dropgen_t() `dropgen_t::dropgen_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.113.2 Member Function Documentation

5.113.2.1 process() [1/2] `mha_wave_t * dropgen_t::process (`

```
mha_wave_t * s )
```

5.113.2.2 process() [2/2] `mha_spec_t * dropgen_t::process (`

```
mha_spec_t * s )
```

5.113.2.3 prepare() `void dropgen_t::prepare (`

```
mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.113.2.4 release() `void dropgen_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.113.3 Member Data Documentation

5.113.3.1 min_sleep_time `MHAParser::float_t dropgen_t::min_sleep_time`

5.113.3.2 max_sleep_time `MHAParser::float_t dropgen_t::max_sleep_time`

5.113.3.3 chance `MHAParser::float_t dropgen_t::chance`

5.113.3.4 patchbay `MHAEvents::patchbay_t< dropgen_t> dropgen_t::patchbay`

5.113.3.5 r `std::random_device dropgen_t::r`

5.113.3.6 random_engine `std::mt19937 dropgen_t::random_engine`

5.113.3.7 dis `std::uniform_real_distribution dropgen_t::dis`

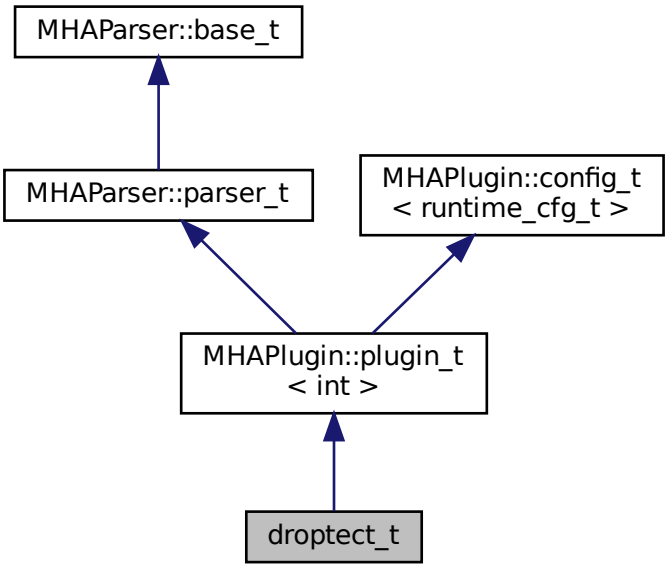
The documentation for this class was generated from the following file:

- **dropgen.cpp**

5.114 droptect_t Class Reference

Detect dropouts in a signal with a constant spectrum.

Inheritance diagram for droptect_t:



Public Member Functions

- **droptect_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
  Allocates and initializes storage for this algorithm.
- void **release** (void)
  Deallocates storage.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗signal)
  Compares current spectrum against history.

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 (duration of fragsize in seconds)
- **MHAParser::mfloat_mon_t filtered_powspec_mon**

  User access to filtered spectrum.
- **MHAParser::vfloat_mon_t level_mon**

  User access to broadband levels.

Additional Inherited Members

5.114.1 Detailed Description

Detect dropouts in a signal with a constant spectrum.

5.114.2 Constructor & Destructor Documentation

5.114.2.1 droptect_t()

```
droptect_t::droptect_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.

5.114.3 Member Function Documentation

5.114.3.1 prepare() void droptect_t::prepare (mhaconfig_t & signal_info) [virtual]

Allocates and initializes storage for this algorithm.

Parameters

signal_info	contains fft length, number of channels, fft length and hop size.

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.114.3.2 release() void droptect_t::release (void) [virtual]

Deallocates storage.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.114.3.3 process() mha_spec_t * droptect_t::process (mha_spec_t * signal)

Compares current spectrum against history.

If spectral power has changed or is below threshold, this is interpreted as dropout occurrence.

5.114.4 Member Data Documentation

5.114.4.1 dropouts MHAParser::vint_mon_t droptect_t::dropouts [private]

5.114.4.2 consecutive_dropouts MHAParser::vint_mon_t droptect_t::consecutive_↩ dropouts [private]

5.114.4.3 blocks `MHAParser::int_mon_t droptect_t::blocks [private]`

5.114.4.4 reset `MHAParser::bool_t droptect_t::reset [private]`

5.114.4.5 threshold `MHAParser::float_t droptect_t::threshold [private]`

5.114.4.6 current_powspec `MHASignal::waveform_t* droptect_t::current_powspec [private]`

5.114.4.7 filtered_powspec `MHASignal::waveform_t* droptect_t::filtered_powspec [private]`

5.114.4.8 tau `MHAParser::float_t droptect_t::tau [private]`

5.114.4.9 filter_activated `std::vector<bool> droptect_t::filter_activated [private]`

5.114.4.10 period `float droptect_t::period [private]`

The period of the process callback (duration of fragsize in seconds)

5.114.4.11 filtered_powspec_mon `MHAParser::mfloat_mon_t droptect_t::filtered_powspec_mon [private]`

User access to filtered spectrum.

5.114.4.12 level_mon `MHAParser::vfloat_mon_t droptect_t::level_mon [private]`

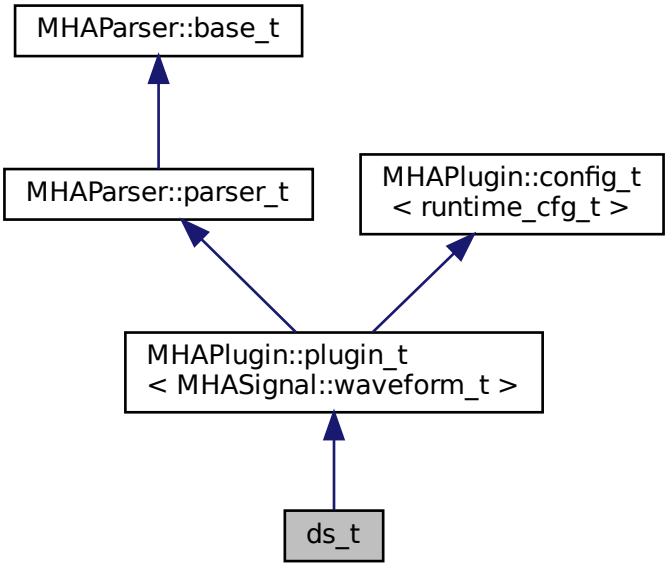
User access to broadband levels.

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

- **droptect.cpp**

5.115 ds_t Class Reference

Inheritance diagram for ds_t:



Public Member Functions

- **ds_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.115.1 Constructor & Destructor Documentation

5.115.1.1 ds_t()

```
ds_t::ds_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.115.2 Member Function Documentation

5.115.2.1 process()

```
mha_wave_t * ds_t::process (
    mha_wave_t * s )
```

5.115.2.2 prepare()

```
void ds_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 1301).

5.115.2.3 release()

```
void ds_t::release (
    void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 1302).

5.115.3 Member Data Documentation

5.115.3.1 ratio `MHAParser::int_t ds_t::ratio [private]`

5.115.3.2 antialias `MHAFilter::iir_filter_t ds_t::antialias [private]`

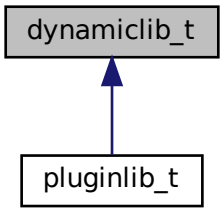
The documentation for this class was generated from the following file:

- **downsample.cpp**

5.116 dynamiclib_t Class Reference

Wrapper class around a shared library.

Inheritance diagram for dynamiclib_t:



Public Member Functions

- **dynamiclib_t** (const std::string &name_)
  C'tor of the wrapper class.
- virtual void * **resolve** (const std::string &name_)
  Resolves the function specified by name_ and returns a pointer to it or a nullptr if the function was not found in the wrapped library.
- virtual void * **resolve_checked** (const std::string &name_)
  Resolves the function specified by name_ and returns a pointer to it or throws an exception if the function was not found.
- virtual ~**dynamiclib_t** ()
  D'tor.
- virtual const std::string & **getmodulename** () const
  Returns unqualified filename of the wrapped library sans file suffix.
- virtual const std::string & **getname** () const

Protected Member Functions

- **dynamiclib_t** ()
  Default constructor.
- void **load_lib** (const std::string &name_)
  Loads the library specified in name_ and saves a handle in h.

Protected Attributes

- std::string **fullname**
  Fully qualified file name of the library.
- std::string **modulename**
  Unqualified file name of the library.
- **mha_libhandle_t h**
  Handle to the shared library.

5.116.1 Detailed Description

Wrapper class around a shared library.

Encapsulates the OS-specific stuff of loading the shared library, resolving functions, etc... Uses the dload API on Linux/macOS and the win32 API on Windows

5.116.2 Constructor & Destructor Documentation

5.116.2.1 dynamiclib_t() [1/2] dynamiclib_t::dynamiclib_t (const std::string & *name_*)

C'tor of the wrapper class.

Takes a the the file name of a shared libary w/o the suffix as argument, searches for the library in the system-dependent standard paths for libraries and in MHA_LIBRARY_PATH. Calls load↩ _lib for the actual work.

Parameters

*name_*	File name of the shared library, without suffix

Exceptions

MHA_Error *(p. 906)*	if the library can not be found or can not be loaded

5.116.2.2 ~dynamiclib_t() `dynamiclib_t::~dynamiclib_t ( ) [virtual]`

D'tor.

Closes the library handle.

5.116.2.3 dynamiclib_t() `[2/2]` `dynamiclib_t::dynamiclib_t ( ) [protected]`

Default constructor.

5.116.3 Member Function Documentation

5.116.3.1 resolve() `void * dynamiclib_t::resolve ( const std::string & name_ ) [virtual]`

Resolves the function specified by name_ and returns a pointer to it or a nullptr if the function was not found in the wrapped library.

Parameters

*name_*	Name of the function to be resolved

Returns

Pointer to the function

Reimplemented in **pluginlib_t** (p. 1516).

5.116.3.2 resolve_checked() `void * dynamiclib_t::resolve_checked ( const std::string & name_ ) [virtual]`

Resolves the function specified by name_ and returns a pointer to it or throws an exception if the function was not found.

Parameters

*name_*	Name of the function to be resolved

Returns

Pointer to the function

5.116.3.3 getmodulename() `virtual const std::string& dynamiclib_t::getmodulename ( ) const [inline], [virtual]`

Returns unqualified filename of the wrapped library sans file suffix.

Returns

Unqualified filename of the wrapped library

5.116.3.4 getname() `virtual const std::string& dynamiclib_t::getname ( ) const [inline], [virtual]`

5.116.3.5 load_lib() `void dynamiclib_t::load_lib ( const std::string & name_ ) [protected]`

Loads the library specified in name_ and saves a handle in h.

Parameters

*name_*	unqualified file name of the shared library w/o suffix

5.116.4 Member Data Documentation

5.116.4.1 fullname `std::string dynamiclib_t::fullname [protected]`

Fully qualified file name of the library.

5.116.4.2 modulename `std::string dynamiclib_t::modulename [protected]`

Unqualified file name of the library.

5.116.4.3 h **`mha_libhandle_t`** `dynamiclib_t::h [protected]`

Handle to the shared library.

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

- **mha_os.h**
- **mha_os.cpp**

5.117 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**

5.117.1 Detailed Description

Real-time class for after burn effect.

The constructor processes the parameters and creates pre-processed variables for efficient realtime processing.

5.117.2 Constructor & Destructor Documentation

5.117.2.1 dc_afterburn_rt_t()

```
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 )
```

5.117.3 Member Function Documentation

5.117.3.1 burn()

```
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.

5.117.4 Member Data Documentation

5.117.4.1 drain_inv `std::vector<float> DynComp::dc_afterburn_rt_t::drain_inv [private]`

5.117.4.2 conflux `std::vector<float> DynComp::dc_afterburn_rt_t::conflux [private]`

5.117.4.3 maxgain `std::vector<float> DynComp::dc_afterburn_rt_t::maxgain [private]`

5.117.4.4 mpo_inv `std::vector<float> DynComp::dc_afterburn_rt_t::mpo_inv [private]`

5.117.4.5 lp `std::vector< MHAFilter::o1flt_lowpass_t> DynComp::dc_afterburn_rt_t::lp [private]`

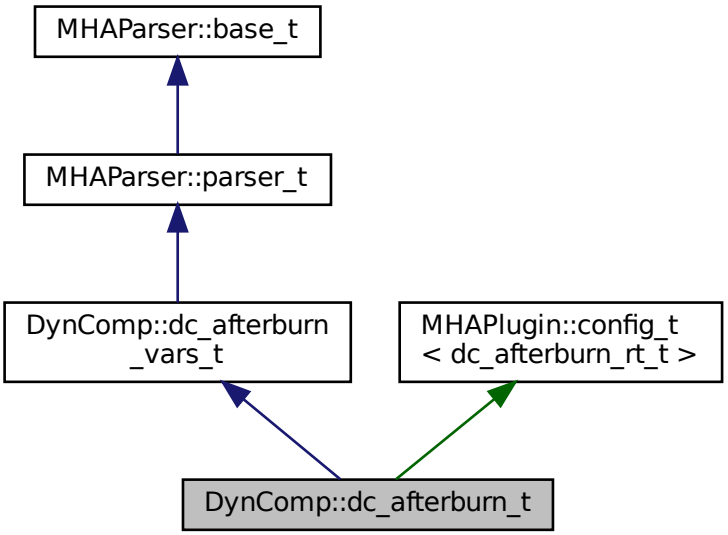
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.118 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

5.118.1 Detailed Description

Afterburn class, to be defined as a member of compressors.

5.118.2 Constructor & Destructor Documentation

5.118.2.1 dc_afterburn_t() `DynComp::dc_afterburn_t::dc_afterburn_t ( )`

5.118.3 Member Function Documentation

5.118.3.1 set_fb_pars()

```
void DynComp::dc_afterburn_t::set_fb_pars (
    const std::vector< float > & cf,
    unsigned int channels,
    float srate )
```

5.118.3.2 unset_fb_pars() `void DynComp::dc_afterburn_t::unset_fb_pars ( )`

5.118.3.3 update_burner() `void DynComp::dc_afterburn_t::update_burner ( ) [inline]`

5.118.3.4 burn()

```
void DynComp::dc_afterburn_t::burn (
    float & Gin,
    float Lin,
    unsigned int band,
    unsigned int channel ) [inline]
```

5.118.3.5 update() void DynComp::dc_afterburn_t::update ( ) [private]

5.118.4 Member Data Documentation

5.118.4.1 patchbay MHAEvents::patchbay_t< dc_afterburn_t> DynComp::dc_afterburn_t::patchbay [private]

5.118.4.2 _cf std::vector<float> DynComp::dc_afterburn_t::_cf [private]

5.118.4.3 _channels unsigned int DynComp::dc_afterburn_t::_channels [private]

5.118.4.4 _srate float DynComp::dc_afterburn_t::_srate [private]

5.118.4.5 commit_pending bool DynComp::dc_afterburn_t::commit_pending [private]

5.118.4.6 fb_pars_configured bool DynComp::dc_afterburn_t::fb_pars_configured [private]

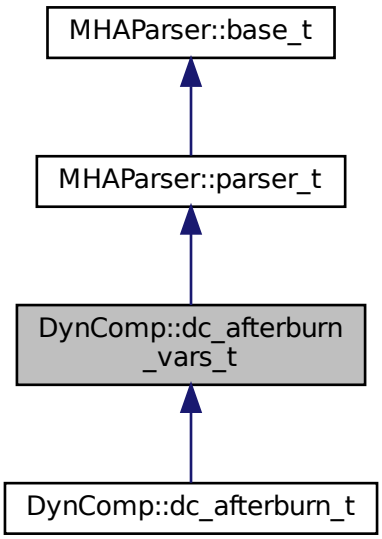
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.119 DynComp::dc_afterburn_vars_t Class Reference

Variables for **dc_afterburn_t** (p. 538) 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

5.119.1 Detailed Description

Variables for **dc_afterburn_t** (p. 538) class.

5.119.2 Constructor & Destructor Documentation

5.119.2.1 dc_afterburn_vars_t() `DynComp::dc_afterburn_vars_t::dc_afterburn_vars_t ( )`

5.119.3 Member Data Documentation

5.119.3.1 f `MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::f`

5.119.3.2 drain `MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::drain`

5.119.3.3 conflux `MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::conflux`

5.119.3.4 maxgain `MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::maxgain`

5.119.3.5 mpo `MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::mpo`

5.119.3.6 taugain `MHAParser::float_t DynComp::dc_afterburn_vars_t::taugain`

5.119.3.7 commit `MHAParser::kw_t DynComp::dc_afterburn_vars_t::commit`

5.119.3.8 bypass `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**

5.120 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**

5.120.1 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. 545) member function via a gain prescription rule from an auditory profile.

5.120.2 Constructor & Destructor Documentation

5.120.2.1 gaintable_t()

```
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).

5.120.2.2 ~gaintable_t()

```
gaintable_t::~gaintable_t ( )
```

5.120.3 Member Function Documentation

5.120.3.1 update()

```
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.

5.120.3.2 get_gain() [1/3]

```
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

5.120.3.3 get_gain() [2/3]

```
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

5.120.3.4 get_gain() [3/3] 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.

5.120.3.5 nbands() unsigned int DynComp::gaintable_t::nbands ( ) const [inline]

Return number of frequency bands.

5.120.3.6 nchannels() unsigned int DynComp::gaintable_t::nchannels ( ) const [inline]

Return number of audio channels.

5.120.3.7 get_iofun() std::vector< std::vector< **mha_real_t** > > gaintable_t::get_iofun ( ) const

Return current input-output function.

5.120.3.8 get_vL() std::vector< **mha_real_t**> DynComp::gaintable_t::get_vL ( ) const [inline]

5.120.3.9 get_vF() std::vector< **mha_real_t**> DynComp::gaintable_t::get_vF ( ) const [inline]

5.120.4 Member Data Documentation

5.120.4.1 num_L unsigned int DynComp::gaintable_t::num_L [private]

5.120.4.2 num_F unsigned int DynComp::gaintable_t::num_F [private]

5.120.4.3 num_channels unsigned int DynComp::gaintable_t::num_channels [private]

5.120.4.4 vL std::vector< **mha_real_t**> DynComp::gaintable_t::vL [private]

5.120.4.5 vF std::vector< **mha_real_t**> DynComp::gaintable_t::vF [private]

5.120.4.6 vFlog std::vector< **mha_real_t**> DynComp::gaintable_t::vFlog [private]

5.120.4.7 data `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**

5.121 equalize::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int infft, int inchannels, std::vector< std::vector< float > > ifgains)
- **cfg_t** (const **cfg_t** &)=delete
- **cfg_t** & **operator=** (const **cfg_t** &)=delete
- ~**cfg_t** ()

Public Attributes

- int **num_bins**
- int **nchannels**
- **mha_real_t** * **fftgains**

5.121.1 Constructor & Destructor Documentation

5.121.1.1 cfg_t() [1/2]

```
cfg_t::cfg_t (
    int infft,
    int inchannels,
    std::vector< std::vector< float > > ifgains )
```

5.121.1.2 cfg_t() [2/2]

```
equalize::cfg_t::cfg_t (
    const cfg_t & ) [delete]
```

5.121.1.3 ~cfg_t() `cfg_t::~cfg_t ( )`

5.121.2 Member Function Documentation

5.121.2.1 operator=() `cfg_t& equalize::cfg_t::operator= ( const cfg_t & ) [delete]`

5.121.3 Member Data Documentation

5.121.3.1 num_bins `int equalize::cfg_t::num_bins`

5.121.3.2 nchannels `int equalize::cfg_t::nchannels`

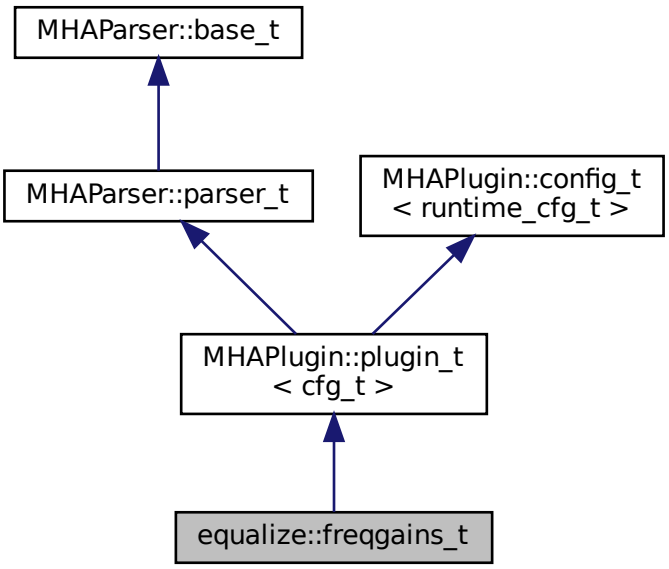
5.121.3.3 fftgains `mha_real_t* equalize::cfg_t::fftgains`

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

- **equalize.cpp**

5.122 equalize::freqgains_t Class Reference

Inheritance diagram for equalize::freqgains_t:



Public Member Functions

- **freqgains_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_gains** ()
- void **update_id** ()

Private Attributes

- **MHAParser::mfloat_t fftgains**
- **MHAParser::string_t id**
- **MHAEvents::patchbay_t**< **freqgains_t** > **patchbay**

Additional Inherited Members

5.122.1 Constructor & Destructor Documentation

5.122.1.1 freqgains_t() equalize::freqgains_t::freqgains_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.122.2 Member Function Documentation

5.122.2.1 process() mha_spec_t * equalize::freqgains_t::process (mha_spec_t * s)

5.122.2.2 prepare() void equalize::freqgains_t::prepare (mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.122.2.3 update_gains() void equalize::freqgains_t::update_gains (void) [private]

5.122.2.4 update_id() void equalize::freqgains_t::update_id (void) [private]

5.122.3 Member Data Documentation

5.122.3.1 fftgains `MHAParser::mfloat_t equalize::freqgains_t::fftgains [private]`

5.122.3.2 id `MHAParser::string_t equalize::freqgains_t::id [private]`

5.122.3.3 patchbay `MHAEvents::patchbay_t< freqgains_t> equalize::freqgains_t::patchbay [private]`

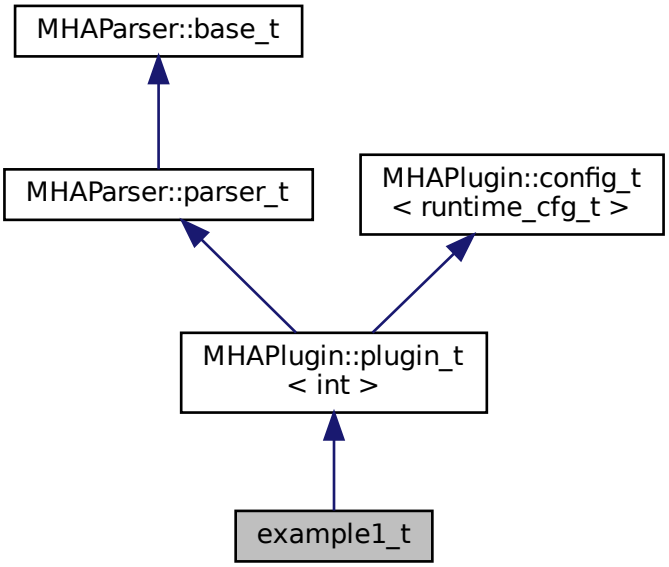
The documentation for this class was generated from the following file:

- **equalize.cpp**

5.123 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** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.123.1 Detailed Description

This C++ class implements the simplest example plugin for the step-by-step tutorial.

It inherits from **MHAPlugin::plugin_t** (p. 1298) for correct integration in the configuration language interface.

5.123.2 Constructor & Destructor Documentation

5.123.2.1 example1_t() `example1_t::example1_t ( MHA_AC::algo_comm_t & iac, const std::string & configured_name ) [inline]`

Do-nothing constructor.

The constructor has to take these two arguments, but it does not have to use them. The base class has to be initialized.

5.123.3 Member Function Documentation

5.123.3.1 release() `void example1_t::release ( void ) [inline], [virtual]`

Release may be empty.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.123.3.2 prepare() `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. 1301).

5.123.3.3 process() `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)

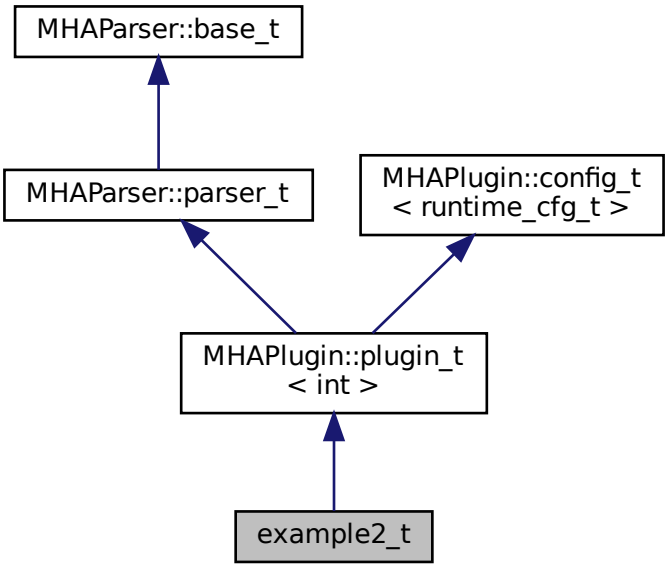
The documentation for this class was generated from the following file:

- **example1.cpp**

5.124 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** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.124.1 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.

5.124.2 Constructor & Destructor Documentation

5.124.2.1 example2_t() `example2_t::example2_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

5.124.3 Member Function Documentation

5.124.3.1 prepare() `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. 1301).

5.124.3.2 release() `void example2_t::release ( void ) [virtual]`

Undo restrictions posed in prepare.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.124.3.3 process() `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)

5.124.4 Member Data Documentation

5.124.4.1 scale_ch `MHAParser::int_t example2_t::scale_ch [private]`

Index of audio channel to scale.

5.124.4.2 factor `MHAParser::float_t example2_t::factor [private]`

The scaling factor applied to the selected channel.

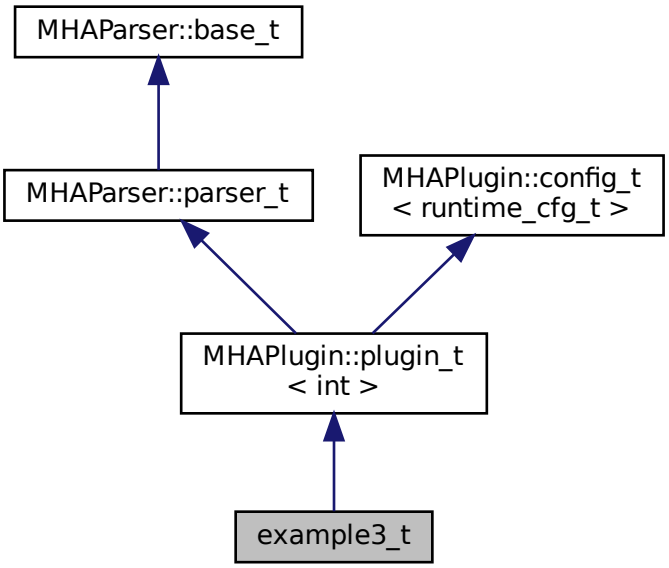
The documentation for this class was generated from the following file:

- **example2.cpp**

5.125 example3_t Class Reference

A Plugin class using the openMHA Event mechanism.

Inheritance diagram for example3_t:



Public Member Functions

- **example3_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.125.1 Detailed Description

A Plugin class using the openMHA Event mechanism.

This is the third example plugin for the step-by-step tutorial.

5.125.2 Constructor & Destructor Documentation

5.125.2.1 example3_t() `example3_t::example3_t ( MHA_AC::algo_comm_t & iac, const std::string & configured_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.

5.125.3 Member Function Documentation

5.125.3.1 on_scale_ch_writeaccess() `void example3_t::on_scale_ch_writeaccess ( ) [private]`

5.125.3.2 on_scale_ch_valuechanged() `void example3_t::on_scale_ch_valuechanged ( ) [private]`

5.125.3.3 on_scale_ch_readaccess() `void example3_t::on_scale_ch_readaccess ( ) [private]`

5.125.3.4 on_prereadaccess() `void example3_t::on_prereadaccess ( ) [private]`

5.125.3.5 prepare() `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. 1301).

5.125.3.6 release() `void example3_t::release ( void ) [virtual]`

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.125.3.7 process() `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)

5.125.4 Member Data Documentation

5.125.4.1 scale_ch `MHAParser::int_t example3_t::scale_ch [private]`

Index of audio channel to scale.

5.125.4.2 factor `MHAParser::float_t example3_t::factor [private]`

The scaling factor applied to the selected channel.

5.125.4.3 prepared `MHAParser::int_mon_t example3_t::prepared [private]`

Keep Track of the prepare/release calls.

5.125.4.4 patchbay `MHAEvents::patchbay_t< example3_t> example3_t::patchbay [private]`

The Event connector.

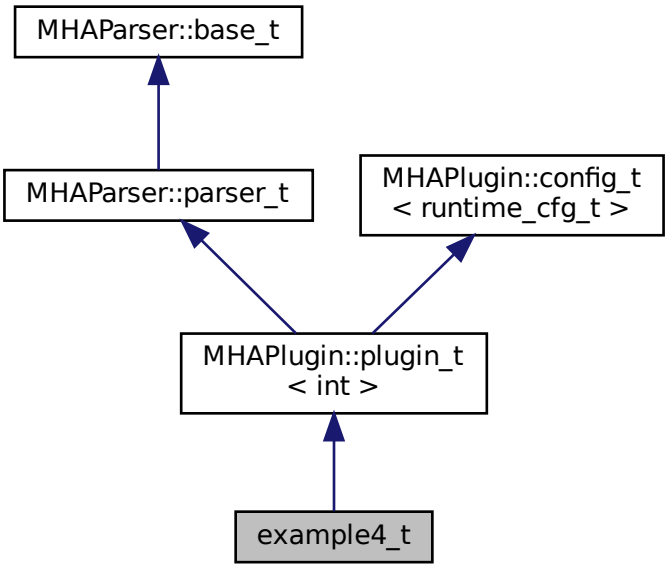
The documentation for this class was generated from the following file:

- **example3.cpp**

5.126 example4_t Class Reference

A Plugin class using the spectral signal.

Inheritance diagram for example4_t:



Public Member Functions

- **example4_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.126.1 Detailed Description

A Plugin class using the spectral signal.

This is the fourth example plugin for the step-by-step tutorial.

5.126.2 Constructor & Destructor Documentation

5.126.2.1 example4_t() `example4_t::example4_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_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.

5.126.3 Member Function Documentation

5.126.3.1 on_scale_ch_writeaccess() `void example4_t::on_scale_ch_writeaccess ( ) [private]`

5.126.3.2 on_scale_ch_valuechanged() `void example4_t::on_scale_ch_valuechanged ( ) [private]`

5.126.3.3 on_scale_ch_readaccess() `void example4_t::on_scale_ch_readaccess ( ) [private]`

5.126.3.4 on_prereadaccess() `void example4_t::on_prereadaccess ( ) [private]`

5.126.3.5 prepare() `void example4_t::prepare ( mhaconfig_t & signal_info ) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the spectral 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. 1301).

5.126.3.6 release() `void example4_t::release ( void ) [virtual]`

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.126.3.7 process() `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)

5.126.4 Member Data Documentation

5.126.4.1 scale_ch `MHAParser::int_t example4_t::scale_ch [private]`

Index of audio channel to scale.

5.126.4.2 factor `MHAParser::float_t example4_t::factor [private]`

The scaling factor applied to the selected channel.

5.126.4.3 prepared `MHAParser::int_mon_t example4_t::prepared [private]`

Keep Track of the prepare/release calls.

5.126.4.4 patchbay `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**

5.127 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**

5.127.1 Constructor & Destructor Documentation

5.127.1.1 example5_t()

```
example5_t::example5_t (
    unsigned int ichannel,
    unsigned int numchannels,
     mha_real_t iscale )
```

5.127.2 Member Function Documentation

5.127.2.1 process() `mha_spec_t * example5_t::process ( mha_spec_t * spec )`

5.127.3 Member Data Documentation

5.127.3.1 channel `unsigned int example5_t::channel [private]`

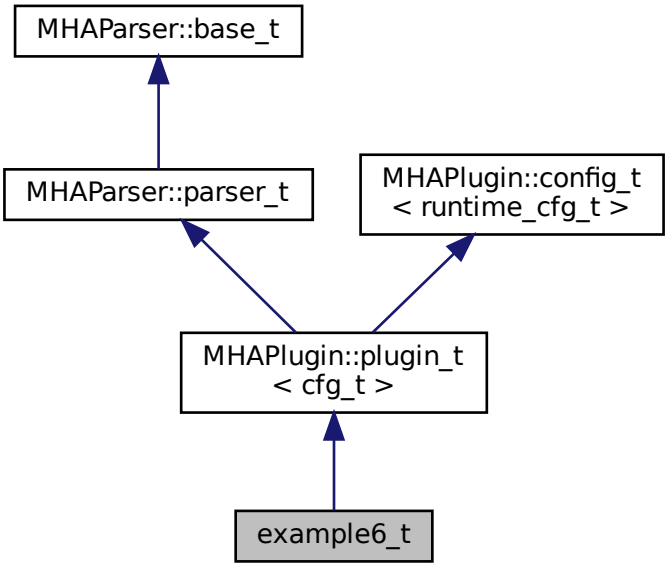
5.127.3.2 scale `mha_real_t example5_t::scale [private]`

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

- **example5.cpp**

5.128 example6_t Class Reference

Inheritance diagram for example6_t:



Public Member Functions

- **example6_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.128.1 Constructor & Destructor Documentation

5.128.1.1 example6_t()

```
example6_t::example6_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.128.2 Member Function Documentation

5.128.2.1 process()

```
mha_wave_t * example6_t::process (
    mha_wave_t * wave )
```

5.128.2.2 prepare() void example6_t::prepare (mhaconfig_t & tfcfg) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.128.2.3 update_cfg() void example6_t::update_cfg ( ) [private]

5.128.3 Member Data Documentation

5.128.3.1 channel_no MHAParser::int_t example6_t::channel_no [private]

5.128.3.2 rmsdb float example6_t::rmsdb [private]

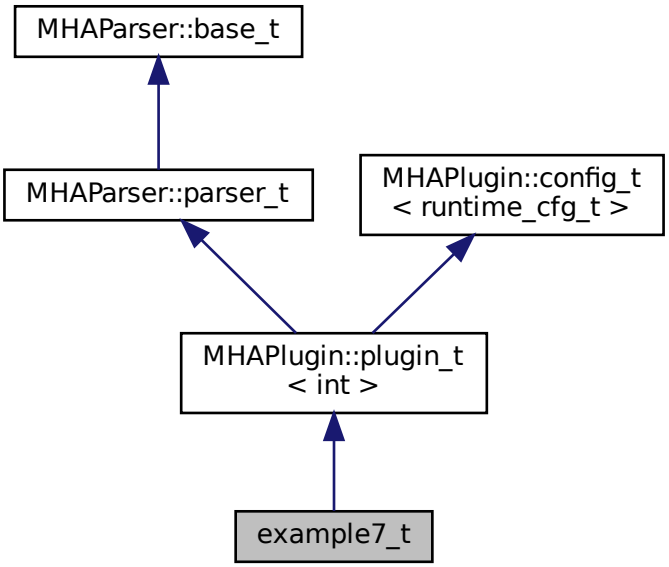
5.128.3.3 patchbay MHAEvents::patchbay_t< example6_t> example6_t::patchbay [private]

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

- **example6.cpp**

5.129 example7_t Class Reference

Inheritance diagram for example7_t:



Public Member Functions

- **example7_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **release** (void)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Additional Inherited Members

5.129.1 Constructor & Destructor Documentation

5.129.1.1 example7_t() example7_t::example7_t (**MHA_AC::algo_comm_t** & *iac*, const std::string & *configured_name*)

5.129.2 Member Function Documentation

5.129.2.1 release() `void example7_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.129.2.2 prepare() `void example7_t::prepare ( mhaconfig_t & signal_info ) [virtual]`

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.129.2.3 process() `mha_wave_t * example7_t::process ( mha_wave_t * signal )`

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

- **example7.hh**
- **example7.cpp**

5.130 expression_t Class Reference

Class for separating a string into a left hand value and a right hand value.

5.130.1 Detailed Description

Class for separating a string into a left hand value and a right hand value.

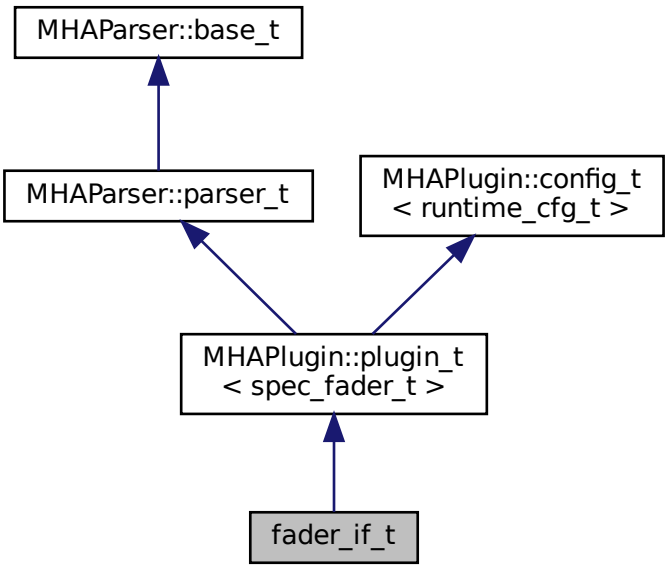
A list of valid operators can be provided. After construction, the class members lval, rval and op contain the apropriate contents.

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

- **mha_parser.cpp**

5.131 fader_if_t Class Reference

Inheritance diagram for fader_if_t:



Public Member Functions

- **fader_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.131.1 Constructor & Destructor Documentation

5.131.1.1 fader_if_t() `fader_if_t::fader_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.131.2 Member Function Documentation

5.131.2.1 process() `mha_spec_t * fader_if_t::process (`

```
mha_spec_t * s )
```

5.131.2.2 prepare() `void fader_if_t::prepare (`

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< spec_fader_t >** (p. 1301).

5.131.2.3 update_cfg() `void fader_if_t::update_cfg (`

```
void ) [private]
```

5.131.3 Member Data Documentation

5.131.3.1 patchbay `MHAEvents::patchbay_t< fader_if_t> fader_if_t::patchbay [private]`

5.131.3.2 tau `MHAParser::float_t fader_if_t::tau [private]`

5.131.3.3 newgains `MHAParser::vfloat_t fader_if_t::newgains [private]`

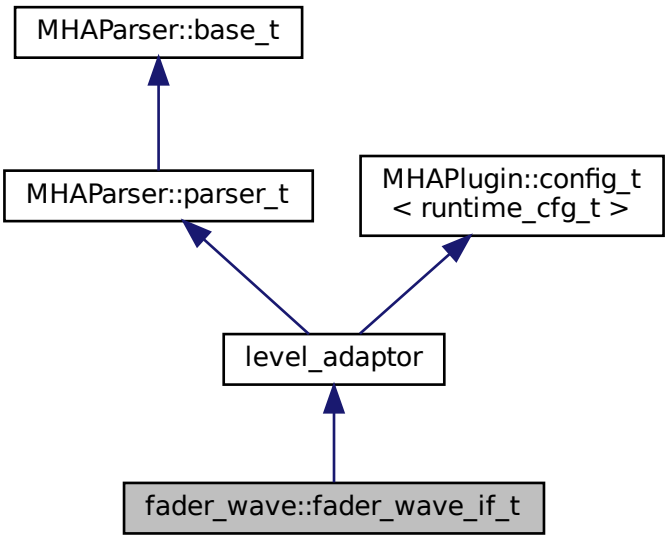
5.131.3.4 actgains `mha_real_t* fader_if_t::actgains [private]`

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

- **fader_spec.cpp**

5.132 fader_wave::fader_wave_if_t Class Reference

Inheritance diagram for fader_wave::fader_wave_if_t:



Public Member Functions

- **fader_wave_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.132.1 Constructor & Destructor Documentation

5.132.1.1 fader_wave_if_t() `fader_wave::fader_wave_if_t::fader_wave_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.132.2 Member Function Documentation

5.132.2.1 process() `mha_wave_t * fader_wave::fader_wave_if_t::process (`

```
mha_wave_t * s )
```

5.132.2.2 prepare() `void fader_wave::fader_wave_if_t::prepare (`

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1301).

5.132.2.3 release() void fader_wave::fader_wave_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1302).

5.132.2.4 set_level() void fader_wave::fader_wave_if_t::set_level ( ) [private]

5.132.3 Member Data Documentation

5.132.3.1 gain MHAParser::vfloat_t fader_wave::fader_wave_if_t::gain [private]

5.132.3.2 ramplen MHAParser::float_t fader_wave::fader_wave_if_t::ramplen [private]

5.132.3.3 patchbay MHAEvents::patchbay_t< fader_wave_if_t> fader_wave::fader_wave_if_t::patchbay [private]

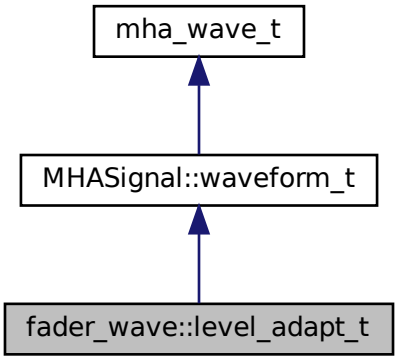
5.132.3.4 prepared bool fader_wave::fader_wave_if_t::prepared [private]

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

- **fader_wave.cpp**

5.133 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

5.133.1 Constructor & Destructor Documentation

5.133.1.1 level_adapt_t()

```
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_ )
```

5.133.2 Member Function Documentation

5.133.2.1 update_frame()

```
void fader_wave::level_adapt_t::update_frame ( )
```

5.133.2.2 get_level()

```
std::vector<float> fader_wave::level_adapt_t::get_level ( )
const [inline]
```

5.133.2.3 can_update()

```
bool fader_wave::level_adapt_t::can_update ( ) const [inline]
```

5.133.3 Member Data Documentation

5.133.3.1 ilen

```
unsigned int fader_wave::level_adapt_t::ilen [private]
```

5.133.3.2 pos

```
unsigned int fader_wave::level_adapt_t::pos [private]
```

5.133.3.3 wnd

```
MHAWindow::fun_t fader_wave::level_adapt_t::wnd [private]
```

5.133.3.4 l_new `std::vector<float> fader_wave::level_adapt_t::l_new [private]`

5.133.3.5 l_old `std::vector<float> fader_wave::level_adapt_t::l_old [private]`

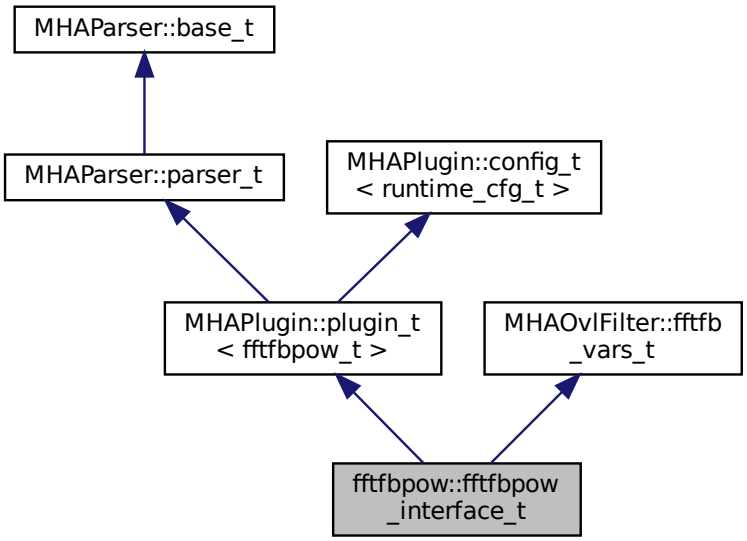
The documentation for this class was generated from the following file:

- **fader_wave.cpp**

5.134 fftfbpow::fftfbpow_interface_t Class Reference

Interface class for fftfbpow plugin.

Inheritance diagram for fftfbpow::fftfbpow_interface_t:



Public Member Functions

- **fftfbpow_interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ 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

5.134.1 Detailed Description

Interface class for fftfbpow plugin.

5.134.2 Constructor & Destructor Documentation

5.134.2.1 fftfbpow_interface_t() `fftfbpow::fftfbpow_interface_t::fftfbpow_interface_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor with standard MHA constructor parameters.

Parameters

iac	Handle to algorithm communication variable space
configured_name	Configured name of this plugin instance

5.134.3 Member Function Documentation

5.134.3.1 prepare() `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. 581) 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. 1301).

5.134.3.2 process() `mha_spec_t * fftfbpow::fftfbpow_interface_t::process ( mha_spec_t * s )`

Standard MHA plugin process fct.

Polls new config and calls **process()** (p. 581) of the runtime configuration.

Parameters

s	Input spectrum

Returns

Unchanged input spectrum

5.134.3.3 update_cfg() `void fftfbpow::fftfbpow_interface_t::update_cfg ( void ) [private]`

Constructs new runtime configuration in thread-safe manner.

5.134.4 Member Data Documentation

5.134.4.1 name `std::string fftfbpow::fftfbpow_interface_t::name [private]`

Configured name of this plugin instance.

5.134.4.2 patchbay `MHAEvents::patchbay_t< fftfbpow_interface_t> fftfbpow::fftfbpow_interface_t::patchbay [private]`

Patchbay to connect to MHA configuration interface.

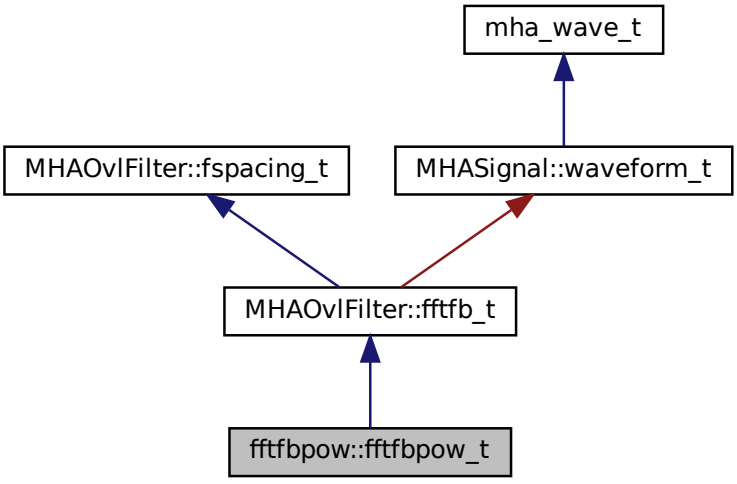
The documentation for this class was generated from the following file:

- **fftfbpow.cpp**

5.135 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, **MHA_AC::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

5.135.1 Detailed Description

Run time configuration for the fftfbpow plugin.

5.135.2 Constructor & Destructor Documentation

5.135.2.1 fftfbpow_t()

```
fftfbpow::fftfbpow_t::fftfbpow_t (
    MHAOvlFilter::fftfb_vars_t & vars,
    unsigned int nch,
    unsigned int nfft,
    mha_real_t fs,
    MHA_AC::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

5.135.3 Member Data Documentation

5.135.3.1 fbpow `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**

5.136 fftfilter::fftfilter_t Class Reference

Public Member Functions

- **fftfilter_t** (const **MHAParser::mfloat_t** &irs, const unsigned int & **fragsize**, const unsigned int & **channels**, const unsigned int & **fftlen**)
  Initialization of new run-time configuration from channel-specific impulse responses.
- **mha_wave_t** $*$ **process** (**mha_wave_t** $*$)

Private Attributes

- unsigned int **irslen**
  Length of the longest impulse response applied.
- unsigned int **fragsize**
  The block size (samples per channel) for waveform audio data.
- unsigned int **fftlen**
  FFT length used for filtering.
- unsigned int **channels**
  Number of prepared audio channels processed by this MHA plugin.
- **MHAFilter::fftfilter_t fftfilt**
  The filter object.

5.136.1 Detailed Description

Run-time configuration class for the fftfilter MHA plugin.

5.136.2 Constructor & Destructor Documentation

5.136.2.1 fftfilter_t()

```
fftfilter::fftfilter_t::fftfilter_t (
    const MHAParser::mfloat_t & irs,
    const unsigned int & fragsize_,
    const unsigned int & channels_,
    const unsigned int & fftlen_ )
```

Initialization of new run-time configuration from channel-specific impulse responses.

Parameters

irs	The matrix containing the impulse responses (one response per channel, or the same response for every channels) as set by the parser.
*fragsize_*	The block size (samples per channel) for waveform audio data
*channels_*	The number of prepared audio channels for this MHA plugin.
*fftlen_*	FFT length used for filtering

5.136.3 Member Function Documentation

5.136.3.1 process()

```
mha_wave_t * fftfilter::fftfilter_t::process (
    mha_wave_t * s )
```

Let fftfiler object handle the filtering

5.136.4 Member Data Documentation

5.136.4.1 irslen

```
unsigned int fftfilter::fftfilter_t::irslen [private]
```

Length of the longest impulse response applied.

5.136.4.2 fragsize `unsigned int fftfilter::fftfilter_t::fragsize [private]`

The block size (samples per channel) for waveform audio data.

5.136.4.3 fftlen `unsigned int fftfilter::fftfilter_t::fftlen [private]`

FFT length used for filtering.

5.136.4.4 channels `unsigned int fftfilter::fftfilter_t::channels [private]`

Number of prepared audio channels processed by this MHA plugin.

5.136.4.5 fftfilt `MHAFilter::fftfilter_t fftfilter::fftfilter_t::fftfilt [private]`

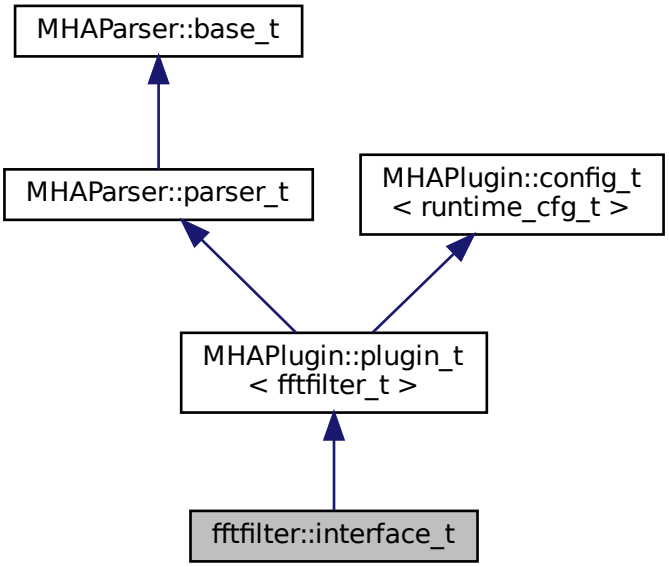
The filter object.

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

- **fftfilter.cpp**

5.137 fftfilter::interface_t Class Reference

Inheritance diagram for fftfilter::interface_t:



Public Member Functions

- **interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::mfloat_t irs**
- **MHAParser::int_t fftlen**
- **MHAParser::int_mon_t fftlen_final**
- **MHAEvents::patchbay_t**< **interface_t** > **patchbay**

Additional Inherited Members

5.137.1 Detailed Description

Implements the MHA plugin interface for FFTFilter

5.137.2 Constructor & Destructor Documentation

5.137.2.1 interface_t() `fftfilter::interface_t::interface_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.137.3 Member Function Documentation

5.137.3.1 process() `mha_wave_t * fftfilter::interface_t::process ( mha_wave_t * s )`

5.137.3.2 prepare() `void fftfilter::interface_t::prepare ( mhaconfig_t & tf ) [virtual]`

Implements **MHAPlugin::plugin_t< fftfilter_t >** (p. 1301).

5.137.3.3 update() `void fftfilter::interface_t::update ( ) [private]`

5.137.4 Member Data Documentation

5.137.4.1 irs `MHAParser::mfloat_t fftfilter::interface_t::irs [private]`

5.137.4.2 fftlen `MHAParser::int_t fftfilter::interface_t::fftlen [private]`

5.137.4.3 fftlen_final `MHAParser::int_mon_t fftfilter::interface_t::fftlen_final [private]`

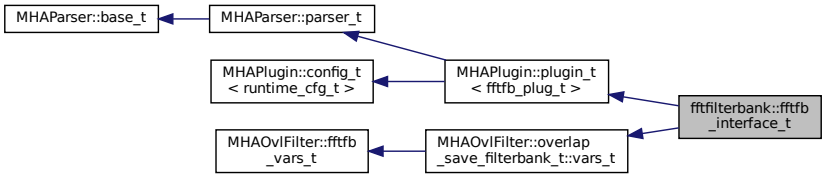
5.137.4.4 patchbay `MHAEvents::patchbay_t< interface_t> fftfilter::interface_t::patchbay [private]`

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

- **fftfilter.cpp**

5.138 fftfilterbank::fftfb_interface_t Class Reference

Inheritance diagram for fftfilterbank::fftfb_interface_t:



Public Member Functions

- **fftfb_interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  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

5.138.1 Constructor & Destructor Documentation

5.138.1.1 fftfb_interface_t()

```
fftfilterbank::fftfb_interface_t::fftfb_interface_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

Default values are set and MHA configuration variables registered into the parser.

Parameters

ac	algorithm communication handle
th	chain name
al	algorithm name

5.138.2 Member Function Documentation

5.138.2.1 prepare()

```
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. 1301).

5.138.2.2 release() void fftfilterbank::fftfb_interface_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 1302).

5.138.2.3 process() [1/2] mha_spec_t * fftfilterbank::fftfb_interface_t::process (mha_spec_t * *s*)

5.138.2.4 process() [2/2] mha_wave_t * fftfilterbank::fftfb_interface_t::process (mha_wave_t * *s*)

5.138.2.5 update_cfg() void fftfilterbank::fftfb_interface_t::update_cfg (void) [private]

5.138.3 Member Data Documentation

5.138.3.1 return_imag MHAParser::bool_t fftfilterbank::fftfb_interface_t::return_imag [private]

5.138.3.2 patchbay MHAEvents::patchbay_t< fftfb_interface_t> fftfilterbank::fftfb_interface_t::patchbay [private]

5.138.3.3 nchannels MHA_AC::int_t fftfilterbank::fftfb_interface_t::nchannels [private]

5.138.3.4 algo `std::string fftfilterbank::fftfb_interface_t::algo [private]`

5.138.3.5 prepared `bool fftfilterbank::fftfb_interface_t::prepared [private]`

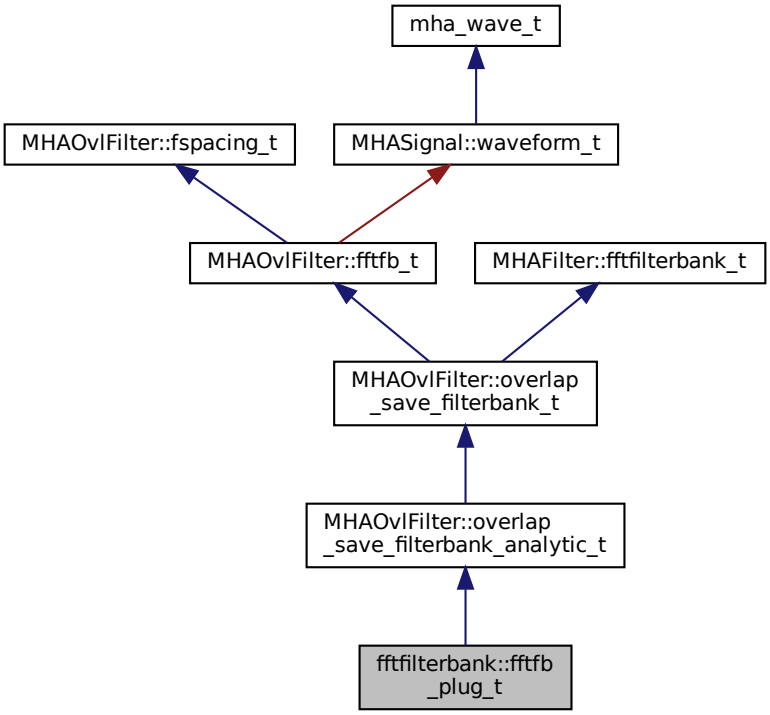
5.138.3.6 nbands `unsigned int fftfilterbank::fftfb_interface_t::nbands [private]`

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

- **fftfilterbank.cpp**

5.139 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, **MHA_AC::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

5.139.1 Constructor & Destructor Documentation

5.139.1.1 fftfb_plug_t() `fftfilterbank::fftfb_plug_t::fftfb_plug_t (`

```
  MHAOvlFilter::overlap_save_filterbank_t::vars_t & vars,
  mhaconfig_t chcfg,
  MHA_AC::algo_comm_t & ac,
 std::string alg,
 bool return_imag )
```

5.139.2 Member Function Documentation

5.139.2.1 process() [1/2] `mha_spec_t * fftfilterbank::fftfb_plug_t::process (`

```
  mha_spec_t * s )
```

5.139.2.2 process() [2/2] `mha_wave_t * fftfilterbank::fftfb_plug_t::process (`

```
  mha_wave_t * s )
```

5.139.2.3 insert() void fftfilterbank::fftfb_plug_t::insert ( )

5.139.3 Member Data Documentation

5.139.3.1 fb_acinfo MHAOvlFilter::fftfb_ac_info_t fftfilterbank::fftfb_plug_t::fb_acinfo [private]

5.139.3.2 s_out MHASignal::spectrum_t fftfilterbank::fftfb_plug_t::s_out [private]

5.139.3.3 imag MHA_AC::waveform_t fftfilterbank::fftfb_plug_t::imag [private]

5.139.3.4 return_imag_ bool fftfilterbank::fftfb_plug_t::return_imag_ [private]

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

- **fftfilterbank.cpp**

5.140 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.

5.140.1 Detailed Description

fshift runtime config class

5.140.2 Constructor & Destructor Documentation

5.140.2.1 fshift_config_t() `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.

5.140.2.2 ~fshift_config_t() `fshift::fshift_config_t::~fshift_config_t ( ) [default]`

5.140.3 Member Function Documentation

5.140.3.1 process() `mha_spec_t * fshift::fshift_config_t::process ( mha_spec_t * in )`

5.140.4 Member Data Documentation

5.140.4.1 kmin `const unsigned int fshift::fshift_config_t::kmin [private]`

FFT bin corresponding to fmin.

5.140.4.2 kmax `const unsigned fshift::fshift_config_t::kmax [private]`

FFT bin corresponding to fmax.

5.140.4.3 df `const int fshift::fshift_config_t::df [private]`

Frequency shift expressed in FFT bins.

5.140.4.4 delta_phi `const mha_complex_t fshift::fshift_config_t::delta_phi [private]`

Phase advance per fft frame.

5.140.4.5 delta_phi_total `mha_complex_t fshift::fshift_config_t::delta_phi_total [private]`

Sum of all phase advances.

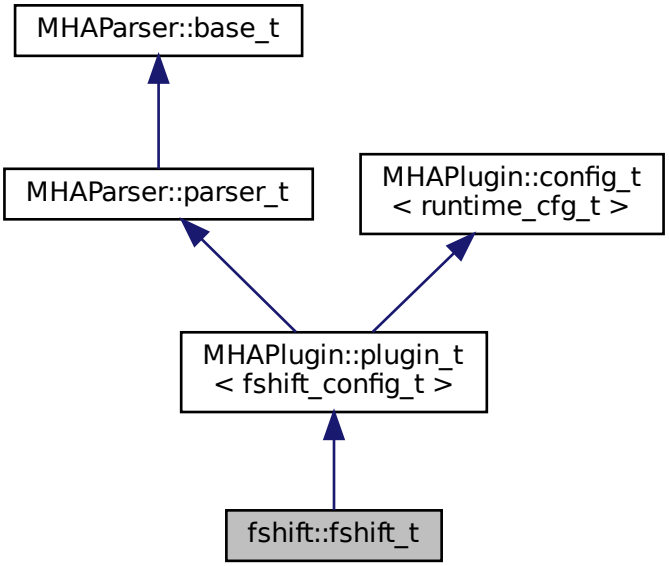
The documentation for this class was generated from the following files:

- **fshift.hh**
- **fshift.cpp**

5.141 fshift::fshift_t Class Reference

fshift plugin interface class

Inheritance diagram for fshift::fshift_t:



Public Member Functions

- **fshift_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.141.1 Detailed Description

fshift plugin interface class

5.141.2 Constructor & Destructor Documentation

5.141.2.1 fshift_t() `fshift::fshift_t::fshift_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.141.2.2 ~fshift_t() `fshift::fshift_t::~fshift_t ( )`

5.141.3 Member Function Documentation

5.141.3.1 process() `mha_spec_t * fshift::fshift_t::process (`

```
  mha_spec_t * signal )
```

Checks for the most recent configuration and defers processing to it.

5.141.3.2 prepare() `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. 1301).

5.141.3.3 release() `void fshift::fshift_t::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< fshift_config_t >** (p. 1302).

5.141.3.4 fmin() `mha_real_t fshift::fshift_t::fmin ( ) const [inline]`

5.141.3.5 fmax() `mha_real_t fshift::fshift_t::fmax ( ) const [inline]`

5.141.3.6 df() `mha_real_t fshift::fshift_t::df ( ) const [inline]`

5.141.3.7 update_cfg() `void fshift::fshift_t::update_cfg ( void ) [private]`

5.141.4 Member Data Documentation

5.141.4.1 patchbay `MHAEvents::patchbay_t< fshift_t> fshift::fshift_t::patchbay [private]`

patch bay for connecting configuration parser events with local member functions:

5.141.4.2 m_fmin `MHAParser::float_t fshift::fshift_t::m_fmin [private]`

5.141.4.3 m_fmax `MHAParser::float_t fshift::fshift_t::m_fmax [private]`

upper boundary for frequency shifter

5.141.4.4 m_df `MHAParser::float_t fshift::fshift_t::m_df [private]`

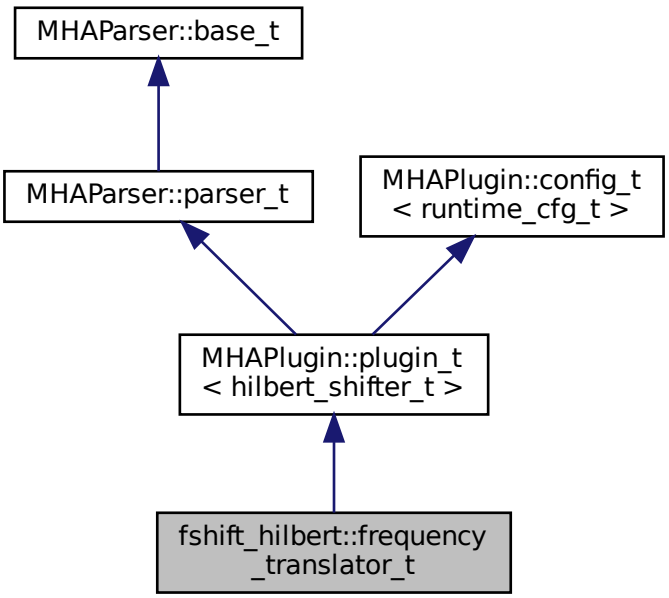
Shift frequency in Hz.

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

- **fshift.hh**
- **fshift.cpp**

5.142 fshift_hilbert::frequency_translator_t Class Reference

Inheritance diagram for fshift_hilbert::frequency_translator_t:



Public Member Functions

- **frequency_translator_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _name)
- **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

5.142.1 Constructor & Destructor Documentation

5.142.1.1 frequency_translator_t()

```
fshift_hilbert::frequency_translator_t::frequency_translator_t (
            MHA_AC::algo_comm_t & iac,
            const std::string & configured_name )
```

5.142.2 Member Function Documentation

5.142.2.1 process()

```
mha_spec_t * fshift_hilbert::frequency_translator_t::process (
            mha_spec_t * s )
```

5.142.2.2 prepare()

```
void fshift_hilbert::frequency_translator_t::prepare (
            mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< hilbert_shifter_t >** (p. 1301).

5.142.2.3 release() void fshift_hilbert::frequency_translator_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< hilbert_shifter_t >** (p. 1302).

5.142.2.4 update() void fshift_hilbert::frequency_translator_t::update ( ) [private]

5.142.3 Member Data Documentation

5.142.3.1 patchbay MHAEvents::patchbay_t< frequency_translator_t> fshift_hilbert::frequency_translator_t::patchbay [private]

5.142.3.2 df MHAParser::vfloat_t fshift_hilbert::frequency_translator_t::df [private]

Vector containing the shift frequencies in Hz.

5.142.3.3 fmin MHAParser::float_t fshift_hilbert::frequency_translator_t::fmin [private]

Lower boundary for frequency shifter.

5.142.3.4 fmax MHAParser::float_t fshift_hilbert::frequency_translator_t::fmax [private]

Upper boundary for frequency shifter.

5.142.3.5 irslen `MHAParser::int_t fshift_hilbert::frequency_translator_t::irslen [private]`

Maximum length of cut off filter response.

5.142.3.6 phasemode `MHAParser::kw_t fshift_hilbert::frequency_translator_t::phasemode [private]`

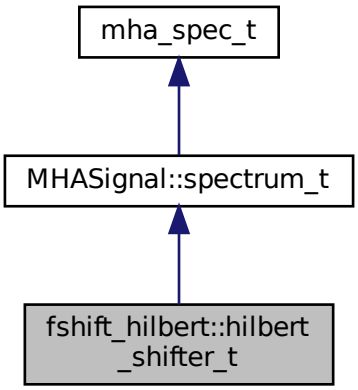
Mode of gain smoothing.

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

- **fshift_hilbert.cpp**

5.143 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

5.143.1 Constructor & Destructor Documentation

5.143.1.1 hilbert_shifter_t() `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 )
```

5.143.1.2 ~hilbert_shifter_t() fshift_hilbert::hilbert_shifter_t::~hilbert_shifter_t ( )

5.143.2 Member Function Documentation

5.143.2.1 process() void fshift_hilbert::hilbert_shifter_t::process (mha_spec_t * s)

5.143.3 Member Data Documentation

5.143.3.1 fullspec MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::fullspec [private]

Part of the spectrum to be frequency shifted.

5.143.3.2 analytic MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::analytic [private]

Analytic signal, defined as a(t)=x(t)+i∗H(x(t))

5.143.3.3 shifted MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::shifted [private]

The frequency shifted signal in the time domain.

5.143.3.4 mixw_shift 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

5.143.3.5 mixw_ref

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

5.143.3.6 plan_spec2analytic

fftw_plan fshift_hilbert::hilbert_shifter_t::plan_spec2analytic [private]

FFT plan for the transformation of fullspec into the time domain.

5.143.3.7 mhafft

mha_fft_t fshift_hilbert::hilbert_shifter_t::mhafft [private]

MHA wrapper object for fftw.

5.143.3.8 df

MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::df [private]

Vector holding one delta f value for every channel.

5.143.3.9 kmin

unsigned int fshift_hilbert::hilbert_shifter_t::kmin [private]

FFT frame that corresponds to f_min.

5.143.3.10 kmax

unsigned int fshift_hilbert::hilbert_shifter_t::kmax [private]

FFT frame that corresponds to f_max.

5.143.3.11 frameshift `unsigned int fshift_hilbert::hilbert_shifter_t::frameshift [private]`

Total phase advance within one fragment.

5.143.3.12 delta_phi `std::vector< mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi [private]`

Phase advance per frame.

5.143.3.13 delta_phi_total `std::vector< mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi_total [private]`

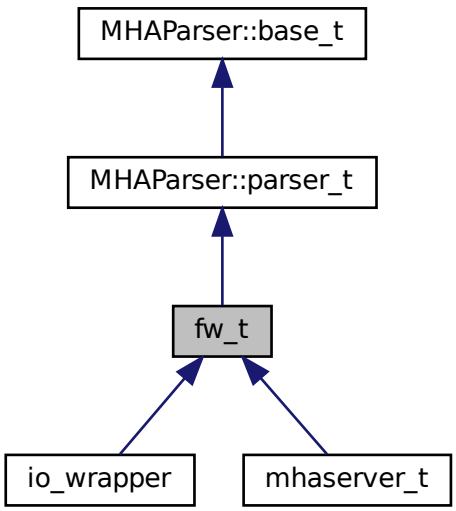
Sum of all phase advances.

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

- **fshift_hilbert.cpp**

5.144 fw_t Class Reference

Inheritance diagram for fw_t:



Public Member Functions

- **fw_t** ()
- ~**fw_t** ()
- bool **exit_request** () const

Protected Attributes

- **io_lib_t** * **io_lib**
- int **proc_error**
- int **io_error**

Private Types

- enum **state_t** {
  fw_unprepared , **fw_stopped** , **fw_starting** , **fw_running** ,
  fw_stopping , **fw_exiting** }

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.
- **MHA_AC::algo_comm_class_t ac**
- **PluginLoader::mhapluginloader_t** * **proc_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

5.144.1 Member Enumeration Documentation

5.144.1.1 state_t enum fw_t::state_t [private]

Enumerator

fw_↩ unprepared	
fw_↩ stopped	
fw_↩ starting	
fw_↩ running	
fw_↩ stopping	
fw_↩ exiting	

5.144.2 Constructor & Destructor Documentation

5.144.2.1 fw_t() `fw_t::fw_t ( )`

5.144.2.2 ~fw_t() `fw_t::~fw_t ( )`

5.144.3 Member Function Documentation

5.144.3.1 exit_request() `bool fw_t::exit_request ( ) const [inline]`

5.144.3.2 prepare() `void fw_t::prepare ( void ) [private]`

preparation for processing

5.144.3.3 start() void fw_t::start ( ) [private]

start of processing

5.144.3.4 stop() void fw_t::stop ( ) [private]

stop/pause of processing

5.144.3.5 release() void fw_t::release ( ) [private]

release of IO device

5.144.3.6 quit() void fw_t::quit ( ) [private]

controlled quit

5.144.3.7 stopped() [1/2] static void fw_t::stopped (

```
void * h,
int proc_err,
int io_err ) [inline], [static], [private]
```

5.144.3.8 started() [1/2] static void fw_t::started (

```
void * h ) [inline], [static], [private]
```

5.144.3.9 process() [1/2] static int fw_t::process (

```
void * h,
 mha_wave_t * sIn,
 mha_wave_t ** sOut ) [inline], [static], [private]
```

5.144.3.10 stopped() [2/2]

```
void fw_t::stopped (
int proc_err,
int io_err ) [private]
```

5.144.3.11 started() [2/2]

```
void fw_t::started ( ) [private]
```

5.144.3.12 process() [2/2]

```
int fw_t::process (
mha_wave_t * s_in,
mha_wave_t ** s_out ) [private]
```

5.144.3.13 exec_fw_command()

```
void fw_t::exec_fw_command ( ) [private]
```

5.144.3.14 load_proc_lib()

```
void fw_t::load_proc_lib ( ) [private]
```

5.144.3.15 load_io_lib()

```
void fw_t::load_io_lib ( ) [private]
```

5.144.3.16 fw_sleep_cmd()

```
void fw_t::fw_sleep_cmd ( ) [private]
```

5.144.3.17 fw_until_cmd()

```
void fw_t::fw_until_cmd ( ) [private]
```

5.144.3.18 get_input_signal_dimension()

```
void fw_t::get_input_signal_dimension ( ) [private]
```

5.144.3.19 async_read() void fw_t::async_read ( ) [inline], [private]

5.144.3.20 async_poll_msg() void fw_t::async_poll_msg ( ) [private]

5.144.3.21 get_parserstate() void fw_t::get_parserstate ( ) [private]

5.144.4 Member Data Documentation

5.144.4.1 prepare_vars **fw_vars_t** fw_t::prepare_vars [private]

5.144.4.2 nchannels_out **MHAParser::int_mon_t** fw_t::nchannels_out [private]

5.144.4.3 proc_name **MHAParser::string_t** fw_t::proc_name [private]

5.144.4.4 io_name **MHAParser::string_t** fw_t::io_name [private]

5.144.4.5 exit_on_stop **MHAParser::bool_t** fw_t::exit_on_stop [private]

5.144.4.6 fw_sleep **MHAParser::int_t** fw_t::fw_sleep [private]

5.144.4.7 fw_until `MHAParser::string_t fw_t::fw_until [private]`

5.144.4.8 fw_cmd `MHAParser::kw_t fw_t::fw_cmd [private]`

5.144.4.9 parserstate `MHAParser::string_mon_t fw_t::parserstate [private]`

5.144.4.10 errorlog `MHAParser::string_t fw_t::errorlog [private]`

5.144.4.11 fatallog `MHAParser::string_t fw_t::fatallog [private]`

5.144.4.12 plugins `MHAParser::vstring_t fw_t::plugins [private]`

5.144.4.13 plugin_paths `MHAParser::vstring_t fw_t::plugin_paths [private]`

5.144.4.14 dump_mha `MHAParser::bool_t fw_t::dump_mha [private]`

5.144.4.15 inst_name `MHAParser::string_t fw_t::inst_name [private]`

A variable for naming MHA instances.

5.144.4.16 ac

MHA_AC::algo_comm_class_t fw_t::ac [private]

5.144.4.17 proc_lib

PluginLoader::mhapluginloader_t* fw_t::proc_lib [private]

5.144.4.18 cfin

mhaconfig_t fw_t::cfin [private]

5.144.4.19 cfout

mhaconfig_t fw_t::cfout [private]

5.144.4.20 state

state_t fw_t::state [private]

5.144.4.21 b_exit_request

bool fw_t::b_exit_request [private]

5.144.4.22 io_lib

io_lib_t* fw_t::io_lib [protected]

5.144.4.23 proc_error

int fw_t::proc_error [protected]

5.144.4.24 io_error

int fw_t::io_error [protected]

5.144.4.25 proc_error_string `MHAParser::string_mon_t fw_t::proc_error_string [private]`

5.144.4.26 patchbay `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**

5.145 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**

5.145.1 Constructor & Destructor Documentation

5.145.1.1 fw_vars_t() `fw_vars_t::fw_vars_t ( MHAParser::parser_t & p ) [explicit]`

5.145.2 Member Function Documentation

5.145.2.1 lock_srate_fragsize() `void fw_vars_t::lock_srate_fragsize ( )`

5.145.2.2 lock_channels() `void fw_vars_t::lock_channels ( )`

5.145.2.3 unlock_srate_fragsize() `void fw_vars_t::unlock_srate_fragsize ( )`

5.145.2.4 unlock_channels() `void fw_vars_t::unlock_channels ( )`

5.145.3 Member Data Documentation

5.145.3.1 pinchannels `MHAParser::int_t fw_vars_t::pinchannels`

5.145.3.2 pfragmentsize `MHAParser::int_t fw_vars_t::pfragmentsize`

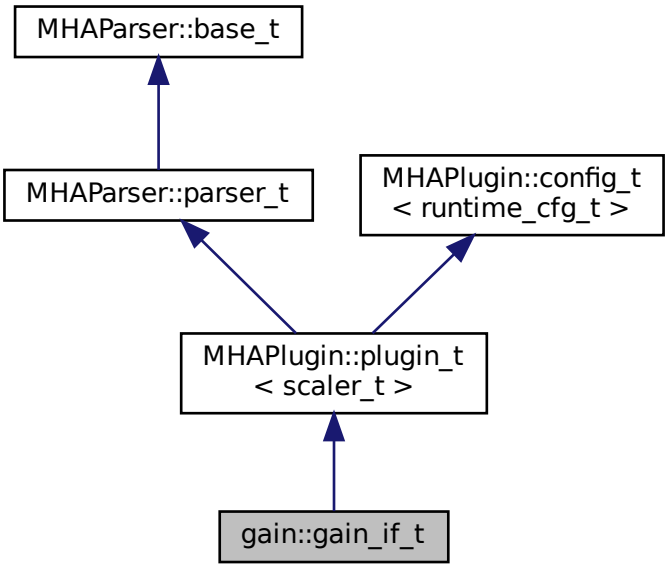
5.145.3.3 psrate `MHAParser::float_t fw_vars_t::psrate`

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

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.146 gain::gain_if_t Class Reference

Inheritance diagram for gain::gain_if_t:



Public Member Functions

- **gain_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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_minmax** ()

Private Attributes

- **MHAEvents::patchbay_t**< **gain_if_t** > **patchbay**
- **MHAParser::vfloat_t gains**
- **MHAParser::float_t vmin**
- **MHAParser::float_t vmax**

Additional Inherited Members

5.146.1 Constructor & Destructor Documentation

5.146.1.1 gain_if_t() `gain::gain_if_t::gain_if_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.146.2 Member Function Documentation

5.146.2.1 process() [1/2] `mha_wave_t * gain::gain_if_t::process (`

```
  mha_wave_t * s )
```

5.146.2.2 process() [2/2] `mha_spec_t * gain::gain_if_t::process (`

```
  mha_spec_t * s )
```

5.146.2.3 prepare() `void gain::gain_if_t::prepare (`

```
  mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< scaler_t >** (p. 1301).

5.146.2.4 release() `void gain::gain_if_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< scaler_t >** (p. 1302).

5.146.2.5 update_gain() void gain::gain_if_t::update_gain ( ) [private]

5.146.2.6 update_minmax() void gain::gain_if_t::update_minmax ( ) [private]

5.146.3 Member Data Documentation

5.146.3.1 patchbay MHAEvents::patchbay_t< gain_if_t> gain::gain_if_t::patchbay [private]

5.146.3.2 gains MHAParser::vfloat_t gain::gain_if_t::gains [private]

5.146.3.3 vmin MHAParser::float_t gain::gain_if_t::vmin [private]

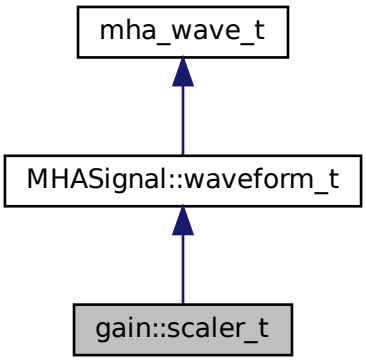
5.146.3.4 vmax MHAParser::float_t gain::gain_if_t::vmax [private]

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

- **gain.cpp**

5.147 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

5.147.1 Constructor & Destructor Documentation

5.147.1.1 scaler_t() `gain::scaler_t::scaler_t (`

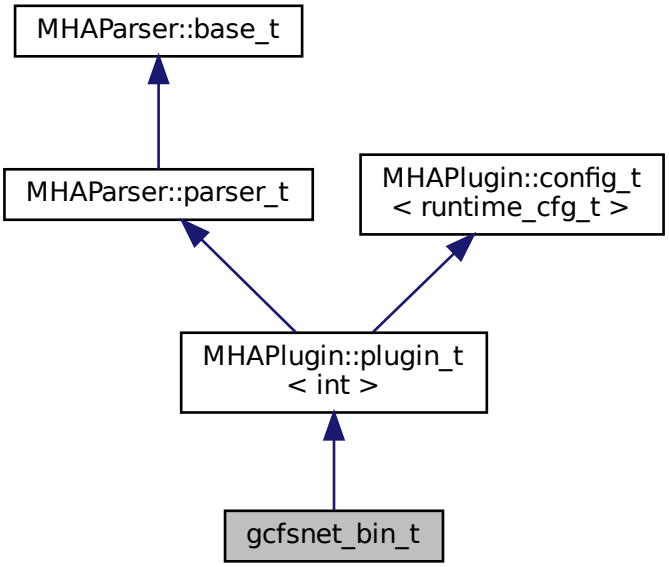
```
const unsigned int & channels,
const  MHAParser::vfloat_t & gains )
```

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

- **gain.cpp**

5.148 gcfsnet_bin_t Class Reference

Inheritance diagram for gcfsnet_bin_t:



Public Member Functions

- **gcfsnet_bin_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗signal)

  Signal processing performed by the plugin.

Private Attributes

- **MHAParser::int_mon_t prepared**
- **MHAParser::float_t remix_factor**
- **MHAParser::float_t calib_factor**
- float **in_frameL_r** [260] = { 0.0 }
- float **in_frameL_i** [260] = { 0.0 }
- float **out_frameL_r** [65] = { 0.0 }

- float **out_frameL_i** [65] = { 0.0 }
- float **in_frameR_r** [260] = { 0.0 }
- float **in_frameR_i** [260] = { 0.0 }
- float **out_frameR_r** [65] = { 0.0 }
- float **out_frameR_i** [65] = { 0.0 }
- unsigned int **num_channels_in** = 4
- **MHASignal::spectrum_t** * **out**
- **DenoiseState** * **state_L**
- **DenoiseState** * **state_R**
- float **calib_fac** = 0

Additional Inherited Members

5.148.1 Constructor & Destructor Documentation

5.148.1.1 gcfsnet_bin_t() `gcfsnet_bin_t::gcfsnet_bin_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

5.148.2 Member Function Documentation

5.148.2.1 prepare() `void gcfsnet_bin_t::prepare (`

```
mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

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. 1301).

5.148.2.2 release() `void gcfsnet_bin_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t**< **int** > (p. 1302).

5.148.2.3 process() `mha_spec_t * gcfsnet_bin_t::process ( mha_spec_t * signal )`

Signal processing performed by the plugin.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the output signal structure.

5.148.3 Member Data Documentation

5.148.3.1 prepared `MHAParser::int_mon_t gcfsnet_bin_t::prepared [private]`

5.148.3.2 remix_factor `MHAParser::float_t gcfsnet_bin_t::remix_factor [private]`

5.148.3.3 calib_factor `MHAParser::float_t gcfsnet_bin_t::calib_factor [private]`

5.148.3.4 in_frameL_r

```
float gcfsnet_bin_t::in_frameL_r[260] = { 0.0 } [private]
```

5.148.3.5 in_frameL_i

```
float gcfsnet_bin_t::in_frameL_i[260] = { 0.0 } [private]
```

5.148.3.6 out_frameL_r

```
float gcfsnet_bin_t::out_frameL_r[65] = { 0.0 } [private]
```

5.148.3.7 out_frameL_i

```
float gcfsnet_bin_t::out_frameL_i[65] = { 0.0 } [private]
```

5.148.3.8 in_frameR_r

```
float gcfsnet_bin_t::in_frameR_r[260] = { 0.0 } [private]
```

5.148.3.9 in_frameR_i

```
float gcfsnet_bin_t::in_frameR_i[260] = { 0.0 } [private]
```

5.148.3.10 out_frameR_r

```
float gcfsnet_bin_t::out_frameR_r[65] = { 0.0 } [private]
```

5.148.3.11 out_frameR_i

```
float gcfsnet_bin_t::out_frameR_i[65] = { 0.0 } [private]
```

5.148.3.12 num_channels_in

```
unsigned int gcfsnet_bin_t::num_channels_in = 4 [private]
```

5.148.3.13 out

```
MHASignal::spectrum_t* gcfsnet_bin_t::out [private]
```

5.148.3.14 state_L

```
DenoiseState* gcfsnet_bin_t::state_L [private]
```

5.148.3.15 state_R

```
DenoiseState* gcfsnet_bin_t::state_R [private]
```

5.148.3.16 calib_fac

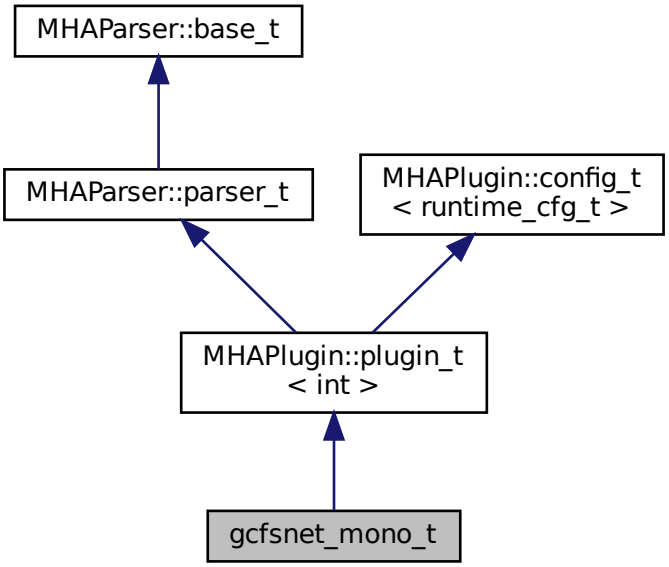
```
float gcfsnet_bin_t::calib_fac = 0 [private]
```

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

- **gcfsnet_bin.cpp**

5.149 gcfsnet_mono_t Class Reference

Inheritance diagram for gcfsnet_mono_t:



Public Member Functions

- **gcfsnet_mono_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗signal)
  Signal processing performed by the plugin.

Private Attributes

- **MHAParser::float_t remix_factor**
  The scaling factor applied to the selected channel.
- **MHAParser::float_t calib_factor**
- **MHAParser::int_mon_t prepared**
  Keep Track of the prepare/release calls.
- float **in_frameL_r** [130] = { 0.0 }
- float **in_frameL_i** [130] = { 0.0 }
- float **out_frameL_r** [65] = { 0.0 }
- float **out_frameL_i** [65] = { 0.0 }
- float **in_frameR_r** [130] = { 0.0 }
- float **in_frameR_i** [130] = { 0.0 }
- float **out_frameR_r** [65] = { 0.0 }
- float **out_frameR_i** [65] = { 0.0 }
- float **calib_fac** = 0
- **MHASignal::spectrum_t** ∗ **out**
- **DenoiseState** ∗ **state_L**
- **DenoiseState** ∗ **state_R**

Additional Inherited Members

5.149.1 Constructor & Destructor Documentation

5.149.1.1 gcfsnet_mono_t() `gcfsnet_mono_t::gcfsnet_mono_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

5.149.2 Member Function Documentation

5.149.2.1 prepare() `void gcfsnet_mono_t::prepare ( mhaconfig_t & signal_info ) [virtual]`

Plugin preparation.

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. 1301).

5.149.2.2 release() `void gcfsnet_mono_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.149.2.3 process() `mha_spec_t * gcfsnet_mono_t::process ( mha_spec_t * signal )`

Signal processing performed by the plugin.

Parameters

signal	Pointer to the input signal structure.

Returns

Returns a pointer to the output signal structure.

5.149.3 Member Data Documentation

5.149.3.1 remix_factor `MHAParser::float_t gcfsnet_mono_t::remix_factor [private]`

The scaling factor applied to the selected channel.

5.149.3.2 calib_factor `MHAParser::float_t gcfsnet_mono_t::calib_factor [private]`

5.149.3.3 prepared `MHAParser::int_mon_t gcfsnet_mono_t::prepared [private]`

Keep Track of the prepare/release calls.

5.149.3.4 in_frameL_r `float gcfsnet_mono_t::in_frameL_r[130] = { 0.0 } [private]`

5.149.3.5 in_frameL_i `float gcfsnet_mono_t::in_frameL_i[130] = { 0.0 } [private]`

5.149.3.6 out_frameL_r `float gcfsnet_mono_t::out_frameL_r[65] = { 0.0 } [private]`

5.149.3.7 out_frameL_i `float gcfsnet_mono_t::out_frameL_i[65] = { 0.0 } [private]`

5.149.3.8 in_frameR_r

```
float gcfsnet_mono_t::in_frameR_r[130] = { 0.0 } [private]
```

5.149.3.9 in_frameR_i

```
float gcfsnet_mono_t::in_frameR_i[130] = { 0.0 } [private]
```

5.149.3.10 out_frameR_r

```
float gcfsnet_mono_t::out_frameR_r[65] = { 0.0 } [private]
```

5.149.3.11 out_frameR_i

```
float gcfsnet_mono_t::out_frameR_i[65] = { 0.0 } [private]
```

5.149.3.12 calib_fac

```
float gcfsnet_mono_t::calib_fac = 0 [private]
```

5.149.3.13 out

```
MHASignal::spectrum_t* gcfsnet_mono_t::out [private]
```

5.149.3.14 state_L

```
DenoiseState* gcfsnet_mono_t::state_L [private]
```

5.149.3.15 state_R

```
DenoiseState* gcfsnet_mono_t::state_R [private]
```

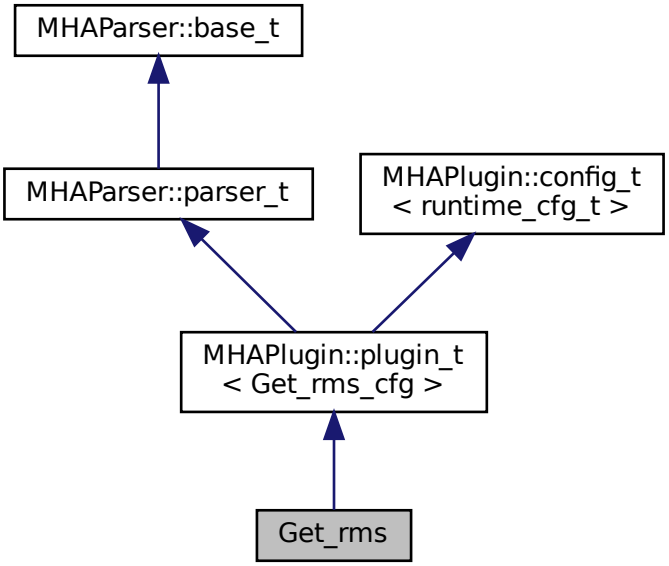
The documentation for this class was generated from the following file:

- **gcfsnet_mono.cpp**

5.150 Get_rms Class Reference

Plugin interface class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.

Inheritance diagram for Get_rms:



Public Member Functions

- **Get_rms** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**

  Name of the algorithm within the plugin chain.
- **MHAParser::float_t tau**

  Time constant for the exponential average / s.
- **MHAEvents::patchbay_t< Get_rms > patchbay**

  Data member connecting an event emitter (i.e.
- std::vector< **mha_real_t** > **rms_prev**

  Vector of RMS values for each channel of the previous input signal fragment.
- **MHA_AC::waveform_t * rms_ac**

  AC variable containing a pointer to the RMS values for each channel of the current input signal fragment.

Additional Inherited Members

5.150.1 Detailed Description

Plugin interface class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.

5.150.2 Constructor & Destructor Documentation

5.150.2.1 Get_rms()

```
Get_rms::Get_rms (
    algo_comm_t & ac,
  const std::string & algo_name )
```

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.150.3 Member Function Documentation

5.150.3.1 prepare() `void Get_rms::prepare (`

`mhaconfig_t` `&` *`signal_info`* `) [virtual]`

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Get_rms_cfg >** (p. 1301).

5.150.3.2 process() **`mha_wave_t`** `* Get_rms::process (`

`mha_wave_t` `*` *`signal`* `)`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.150.3.3 release() `void Get_rms::release (`

`void ) [virtual]`

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Get_rms_cfg >** (p. 1302).

5.150.3.4 update_cfg() `void Get_rms::update_cfg (`

`void ) [private]`

Runtime configuration update function of the plugin interface class.

5.150.4 Member Data Documentation

5.150.4.1 algo_name `std::string Get_rms::algo_name [private]`

Name of the algorithm within the plugin chain.

5.150.4.2 tau `MHAParser::float_t Get_rms::tau [private]`

Time constant for the exponential average / s.

5.150.4.3 patchbay `MHAEvents::patchbay_t< Get_rms> Get_rms::patchbay [private]`

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

5.150.4.4 rms_prev `std::vector< mha_real_t> Get_rms::rms_prev [private]`

Vector of RMS values for each channel of the previous input signal fragment.

5.150.4.5 rms_ac `MHA_AC::waveform_t* Get_rms::rms_ac [private]`

AC variable containing a pointer to the RMS values for each channel of the current input signal fragment.

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

- **get_rms.hh**
- **get_rms.cpp**

5.151 Get_rms_cfg Class Reference

Runtime configuration class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.

Public Member Functions

- **Get_rms_cfg** (**mha_real_t** tau, **Get_rms** ∗ **Get_rms**, std::vector< **mha_real_t** > rms_prev)
  Constructor of the runtime configuration class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal, **Get_rms** ∗ **Get_rms**, **MHA_AC::waveform_t** ∗rms_ac)
  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- **mha_real_t tau_cfg**
  Time constant for the exponential average / s.
- std::vector< **mha_real_t** > **rms_prev_cfg**
  Vector of RMS values for each channel of the previous input signal fragment.

5.151.1 Detailed Description

Runtime configuration class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.

5.151.2 Constructor & Destructor Documentation

5.151.2.1 Get_rms_cfg() `Get_rms_cfg::Get_rms_cfg (`

```
  mha_real_t tau,
  Get_rms * Get_rms,
 std::vector< mha_real_t > rms_prev )
```

Constructor of the runtime configuration class.

Parameters

tau	Time constant for the exponential average / s
Get_rms *(p. 632)*	Pointer to the current instance of the plugin interface class
rms_prev	Vector of RMS values for each channel of the previous input signal fragment

5.151.3 Member Function Documentation

5.151.3.1 process()

```
mha_wave_t * Get_rms_cfg::process (
    mha_wave_t * signal,
    Get_rms * Get_rms,
    MHA_AC::waveform_t * rms_ac )
```

Process function of the runtime configuration class (main signal processing function).

It leaves the input signal fragment unchanged; its purpose is to compute the AC variable rms_ac for further processing in the plugin chain

Parameters

signal	Pointer to the current input signal fragment
Get_rms *(p. 632)*	Pointer to the current instance of the plugin interfa class
rms_ac	AC variable containing a pointer to the RMS value each channel of the current input signal fragment

Returns

Pointer to the (unchanged) output signal fragment

5.151.4 Member Data Documentation

5.151.4.1 tau_cfg

```
mha_real_t Get_rms_cfg::tau_cfg [private]
```

Time constant for the exponential average / s.

5.151.4.2 rms_prev_cfg `std::vector< mha_real_t> Get_rms_cfg::rms_prev_cfg [private]`

Vector of RMS values for each channel of the previous input signal fragment.

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

- **get_rms.hh**
- **get_rms.cpp**

5.152 GRULayer Struct Reference

Public Attributes

- const **rnn_bias** * **bias**
- const **rnn_weight** * **input_weights**
- const **rnn_weight** * **recurrent_weights**
- **counter** **nb_inputs**
- **counter** **nb_neurons**
- **counter** **activation**

5.152.1 Member Data Documentation

5.152.1.1 bias `const rnn_bias * GRULayer::bias`

5.152.1.2 input_weights `const rnn_weight * GRULayer::input_weights`

5.152.1.3 recurrent_weights `const rnn_weight * GRULayer::recurrent_weights`

5.152.1.4 nb_inputs `counter GRULayer::nb_inputs`

5.152.1.5 nb_neurons `counter GRULayer::nb_neurons`

5.152.1.6 activation `counter GRULayer::activation`

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

- **gcfsnet_bin/rnn.h**

5.153 gsc_adaptive_stage::gsc_adaptive_stage Class Reference

Public Member Functions

- **gsc_adaptive_stage** (**MHA_AC::algo_comm_t** & **ac**, const **mhaconfig_t**, int **lenOldSamps**, bool **doCircularComp**, float **mu**, float **alp**, bool **useVAD**, const std::string &vadName_)
  Ctor of the rt processing class.
- ~**gsc_adaptive_stage** ()=default
- **mha_wave_t** * **process** (**mha_wave_t** *wavin)
  Processing callback.

Private Member Functions

- void **insert** ()
  Re-insert all AC variables into the AC space.

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
  Handle to AC space.
- unsigned int **lenOldSamps**
  Number of old samples to buffer.
- unsigned int **lenNewSamps**
  Number of new samples.
- unsigned int **bufSize**
  Total buffer size.
- float **frac_old**
  Fraction of new samples to total buffer size.
- **mha_fft_t** **mha_fft**
  FFT handle.

- unsigned int **nfreq**

  Number of frequency bins.
- unsigned int **nchan**

  Number of channels in input signal.
- unsigned int **desired_chan**

  Channel index containing the desired response.
- bool **doCircularComp**

  Whether to compensate for circular convolution.
- float **mu**

  Linear coefficient for gradient.
- float **alp**

  Autoregressive coefficient for estimating PSD.
- bool **useVAD**

  Wether to use VAD.
- std::string **vadName**

  Name of VAD AC variable.
- **MHA_AC::waveform_t x**

  Buffered input signal.
- **MHA_AC::spectrum_t X**

  FFT of the buffered input signal.
- **MHA_AC::spectrum_t W**

  Time-varying filter.
- **MHA_AC::spectrum_t Y**

  Filter output, frequency domain.
- **MHA_AC::waveform_t y**

  Filter output, time domain.
- **MHA_AC::waveform_t d**

  Desired response.
- **MHA_AC::waveform_t e**

  Error signal, time domain.
- **MHA_AC::spectrum_t E**

  Error signal, frequency domain.
- **MHA_AC::spectrum_t E2**

  Error spectrum multiplied by input spectrum: E2=X∗E.
- **MHA_AC::waveform_t grad**

  Gradient.
- **MHA_AC::spectrum_t Grad**

  FT of the gradient.
- **MHA_AC::waveform_t e_out**

  Error signal.
- **MHA_AC::waveform_t P**

  Signal power estimate.
- **MHA_AC::waveform_t Psum**

  Signal power estimate, summed over all channels.

5.153.1 Constructor & Destructor Documentation

5.153.1.1 gsc_adaptive_stage() gsc_adaptive_stage::gsc_adaptive_stage::gsc_adaptive_stage (

```
 MHA_AC::algo_comm_t & ac,
const  mhaconfig_t in_cfg,
int lenOldSamps,
bool doCircularComp,
float mu,
float alp,
bool useVAD,
const std::string & vadName_ )
```

Ctor of the rt processing class.

Parameters

ac	Handle to AC space
in_cfg	Input signal configuration
lenOldSamps	How many old samples to buffer
doCircularComp	Compensate for circular convolution?
mu	Scalar coefficient for gradient
alp	Autoregressive coefficient for estimating PSD
useVAD	Use voice activity detection?
*vadName_*	Name of the VAD AC variable

5.153.1.2 ~gsc_adaptive_stage() gsc_adaptive_stage::gsc_adaptive_stage::~gsc_adaptive_stage ( ) [default]

5.153.2 Member Function Documentation

5.153.2.1 process() `mha_wave_t * gsc_adaptive_stage::gsc_adaptive_stage::process ( mha_wave_t * wavin )`

Processing callback.

Parameters

wavin	input signal

Returns

Returns a pointer to the output signal

5.153.2.2 insert() `void gsc_adaptive_stage::gsc_adaptive_stage::insert ( ) [private]`

Re-insert all AC variables into the AC space.

5.153.3 Member Data Documentation

5.153.3.1 ac `MHA_AC::algo_comm_t& gsc_adaptive_stage::gsc_adaptive_stage::ac [private]`

Handle to AC space.

5.153.3.2 lenOldSamps `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::lenOldSamps [private]`

Number of old samples to buffer.

5.153.3.3 lenNewSamps `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::lenNewSamps [private]`

Number of new samples.

5.153.3.4 bufSize `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::bufSize [private]`

Total buffer size.

Must be lenOldSamps+lenNewSamps

5.153.3.5 frac_old `float gsc_adaptive_stage::gsc_adaptive_stage::frac_old [private]`

Fraction of new samples to total buffer size.

5.153.3.6 mha_fft `mha_fft_t gsc_adaptive_stage::gsc_adaptive_stage::mha_fft [private]`

FFT handle.

5.153.3.7 nfreq `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::nfreq [private]`

Number of frequency bins.

5.153.3.8 nchan `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::nchan [private]`

Number of channels in input signal.

5.153.3.9 desired_chan `unsigned int gsc_adaptive_stage::gsc_adaptive_stage::desired_chan [private]`

Channel index containing the desired response.

Always last channel by convention

5.153.3.10 doCircularComp `bool gsc_adaptive_stage::gsc_adaptive_stage::doCircular↩ Comp [private]`

Whether to compensate for circular convolution.

5.153.3.11 mu `float gsc_adaptive_stage::gsc_adaptive_stage::mu [private]`

Linear coefficient for gradient.

5.153.3.12 alp `float gsc_adaptive_stage::gsc_adaptive_stage::alp [private]`

Autoregressive coefficient for estimating PSD.

5.153.3.13 useVAD `bool gsc_adaptive_stage::gsc_adaptive_stage::useVAD [private]`

Wether to use VAD.

5.153.3.14 vadName `std::string gsc_adaptive_stage::gsc_adaptive_stage::vadName [private]`

Name of VAD AC variable.

5.153.3.15 x `MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::x [private]`

Buffered input signal.

5.153.3.16 X MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::X [private]

FFT of the buffered input signal.

5.153.3.17 W MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::W [private]

Time-varying filter.

5.153.3.18 Y MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::Y [private]

Filter output, frequency domain.

5.153.3.19 y MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::y [private]

Filter output, time domain.

5.153.3.20 d MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::d [private]

Desired response.

5.153.3.21 e MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::e [private]

Error signal, time domain.

5.153.3.22 E MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::E [private]

Error signal, frequency domain.

5.153.3.23 E2 MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::E2 [private]

Error spectrum multiplied by input spectrum: E2=X∗E.

5.153.3.24 grad MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::grad [private]

Gradient.

5.153.3.25 Grad MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::Grad [private]

FT of the gradient.

5.153.3.26 e_out MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::e_out [private]

Error signal.

5.153.3.27 P MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::P [private]

Signal power estimate.

5.153.3.28 Psum MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::Psum [private]

Signal power estimate, summed over all channels.

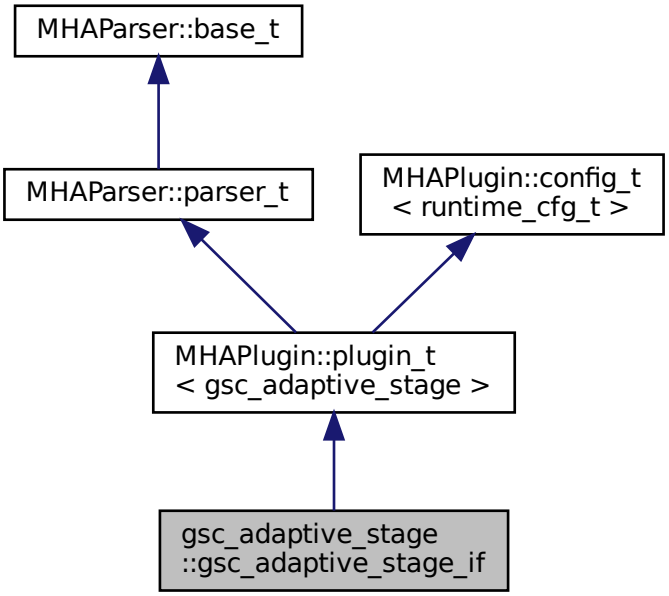
The documentation for this class was generated from the following files:

- **gsc_adaptive_stage.hh**
- **gsc_adaptive_stage.cpp**

5.154 gsc_adaptive_stage::gsc_adaptive_stage_if Class Reference

Plugin interface class.

Inheritance diagram for gsc_adaptive_stage::gsc_adaptive_stage_if:



Public Member Functions

- **gsc_adaptive_stage_if** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _name)

  Constructs the interface to the adaptive filter plugin.
- ∼**gsc_adaptive_stage_if** ()=default
- **mha_wave_t** ∗ **process** (**mha_wave_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** ()

  Update the rt config.
- void **on_model_param_valuechanged** ()

Private Attributes

- **MHAEvents::patchbay_t< gsc_adaptive_stage_if > patchbay**
- **MHAParser::int_t lenOldSamps**
  How many old samples should be buffered per filter block.
- **MHAParser::bool_t doCircularComp**
  Whether to compensate for circular convolution.
- **MHAParser::float_t mu**
  Linear coefficient for gradient used in filter adaption.
- **MHAParser::float_t alp**
  Autoregressive coefficient for PSD estimation.
- **MHAParser::bool_t useVAD**
  Wether to use VAD for conditional filter adaption.
- **MHAParser::string_t vadName**
  Name of VAD AC variable.

Additional Inherited Members

5.154.1 Detailed Description

Plugin interface class.

5.154.2 Constructor & Destructor Documentation

5.154.2.1 gsc_adaptive_stage_if() `gsc_adaptive_stage::gsc_adaptive_stage_if::gsc_adaptive_stage_if (`

```
            MHA_AC::algo_comm_t & iac,
           const std::string & configured_name )
```

Constructs the interface to the adaptive filter plugin.

Parameters

ac	Handle to the ac space

5.154.2.2 ~gsc_adaptive_stage_if() gsc_adaptive_stage::gsc_adaptive_stage_if↩ ::~gsc_adaptive_stage_if ( ) [default]

5.154.3 Member Function Documentation

5.154.3.1 process() mha_wave_t * gsc_adaptive_stage::gsc_adaptive_stage_if::process (mha_wave_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.

5.154.3.2 prepare() void gsc_adaptive_stage::gsc_adaptive_stage_if::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< gsc_adaptive_stage >** (p. 1301).

5.154.3.3 release() void gsc_adaptive_stage::gsc_adaptive_stage_if::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< gsc_adaptive_stage >** (p. 1302).

5.154.3.4 update_cfg() void gsc_adaptive_stage::gsc_adaptive_stage_if::update_cfg (void) [private]

Update the rt config.

5.154.3.5 on_model_param_valuechanged() void gsc_adaptive_stage::gsc_adaptive_stage_if::on_model_param_valuechanged ( ) [private]

5.154.4 Member Data Documentation

5.154.4.1 patchbay **MHAEvents::patchbay_t< gsc_adaptive_stage_if>** gsc_adaptive_stage::gsc_adaptive_stage_if::patchbay [private]

5.154.4.2 lenOldSamps **MHAParser::int_t** gsc_adaptive_stage::gsc_adaptive_stage_if::lenOldSamps [private]

How many old samples should be buffered per filter block.

5.154.4.3 doCircularComp **MHAParser::bool_t** gsc_adaptive_stage::gsc_adaptive_stage_if::doCircularComp [private]

Whether to compensate for circular convolution.

5.154.4.4 mu

MHAParser::float_t gsc_adaptive_stage::gsc_adaptive_stage_if::mu [private]

Linear coefficient for gradient used in filter adaption.

5.154.4.5 alp

MHAParser::float_t gsc_adaptive_stage::gsc_adaptive_stage_if::alp [private]

Autoregressive coefficient for PSD estimation.

5.154.4.6 useVAD

MHAParser::bool_t gsc_adaptive_stage::gsc_adaptive_stage_if↩ ::useVAD [private]

Wether to use VAD for conditional filter adaption.

5.154.4.7 vadName

MHAParser::string_t gsc_adaptive_stage::gsc_adaptive_stage_↩ if::vadName [private]

Name of VAD AC variable.

Ignored if useVAD=no

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

- **gsc_adaptive_stage_if.hh**
- **gsc_adaptive_stage_if.cpp**

5.155 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.

5.155.1 Detailed Description

Configuration for Gammatone Filterbank Analyzer.

5.155.2 Constructor & Destructor Documentation

5.155.2.1 gtfb_analyzer_cfg_t()

```
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.

5.155.2.2 ~gtfb_analyzer_cfg_t()

```
gtfb_analyzer::gtfb_analyzer_cfg_t::~gtfb_analyzer_cfg_t ( ) [inline]
```

5.155.3 Member Function Documentation

5.155.3.1 bands()

```
unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::bands ( ) const [inline]
```

Each band is split into this number of bands.

5.155.3.2 channels()

```
unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::channels ( ) const [inline]
```

The number of separate audio channels.

5.155.3.3 frames() `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::frames ( ) const [inline]`

The number of frames in one chunk.

5.155.3.4 states() `mha_complex_t* gtfb_analyzer::gtfb_analyzer_cfg_t::states ( unsigned channel, unsigned band ) [inline]`

Returns pointer to filter states for that band.

5.155.3.5 cvalue() `mha_complex_t& gtfb_analyzer::gtfb_analyzer_cfg_t::cvalue ( unsigned frame, unsigned channel, unsigned band ) [inline]`

5.155.4 Member Data Documentation

5.155.4.1 order `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::order`

The order of the gammatone filters.

5.155.4.2 coeff `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::coeff`

The complex coefficients of the gammatone filter bands.

5.155.4.3 norm_phase `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::norm_phase`

Combination of normalization and phase correction factor.

5.155.4.4 s_out `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. 985) 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. 99) 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

5.155.4.5 state `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t↩ ::state`

Storage for Filter state.

Holds **channels()** (p. 653) ∗ **bands()** (p. 653) ∗ order complex filter states. Layout↩ : state[(**bands()** (p. 653)∗channel+band)∗order+stage]

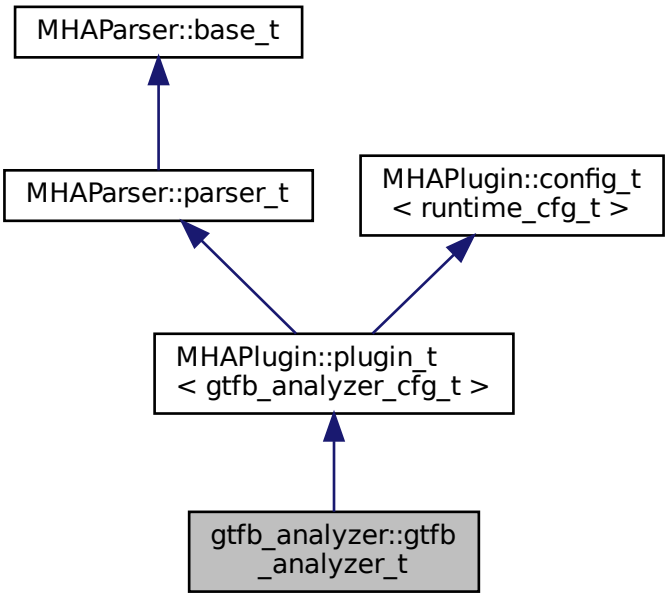
The documentation for this struct was generated from the following file:

- **gtfb_analyzer.cpp**

5.156 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** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

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

5.156.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

5.156.2 Constructor & Destructor Documentation

5.156.2.1 gtfb_analyzer_t()

```
gtfb_analyzer::gtfb_analyzer_t::gtfb_analyzer_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.156.3 Member Function Documentation

5.156.3.1 process()

```
mha_wave_t * gtfb_analyzer::gtfb_analyzer_t::process (
    mha_wave_t * s )
```

5.156.3.2 prepare()

```
void gtfb_analyzer::gtfb_analyzer_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >** (p. 1301).

5.156.3.3 release()

```
void gtfb_analyzer::gtfb_analyzer_t::release (
    void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >** (p. 1302).

5.156.3.4 update_cfg() void gtfb_analyzer::gtfb_analyzer_t::update_cfg (void) [private]

5.156.4 Member Data Documentation

5.156.4.1 patchbay MHAEvents::patchbay_t< gtfb_analyzer_t> gtfb_analyzer::gtfb_analyzer_t::patchbay [private]

5.156.4.2 prepared bool gtfb_analyzer::gtfb_analyzer_t::prepared [private]

5.156.4.3 order MHAParser::int_t gtfb_analyzer::gtfb_analyzer_t::order [private]

5.156.4.4 coeff MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t::coeff [private]

5.156.4.5 norm_phase 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**

5.157 gtfb_simd_cfg_t Class Reference

Public Member Functions

- unsigned **get_bands** () const
  Each band is split into this number of bands.
- unsigned **get_channels** () const
  The number of separate audio channels.
- unsigned **get_frames** () const
  The number of frames in one chunk.
- **gtfb_simd_cfg_t** (**gtfb_simd_cfg_t** &&)=delete
- **gtfb_simd_cfg_t** & **operator=** (**gtfb_simd_cfg_t** &&)=delete
- **gtfb_simd_cfg_t** (const **gtfb_simd_cfg_t** &)=delete
- **gtfb_simd_cfg_t** & **operator=** (const **gtfb_simd_cfg_t** &)=delete
- **gtfb_simd_cfg_t** (unsigned ch, unsigned frames, unsigned ord, const std::vector< **mha_complex_t** > &_coeff, const std::vector< **mha_complex_t** > &_norm_phase)
- ~**gtfb_simd_cfg_t** ()
- **mha_wave_t** * **process** (**mha_wave_t** *s_in)

Private Attributes

- unsigned **order**
  The order of the gammatone filters.
- unsigned **bands**
  Number of frequency bands per channel.
- unsigned **channels**
  Number of input audio channels.
- unsigned **bandsXchannels**
  Product of bands and channels.
- std::vector< **mha_complex_t** > **norm_phase**
  Combination of normalization and phase correction factor.
- float * **rinputs**
  input signal (1 sample) multiplied with norm_phase
- float * **iinputs**
- float * **rcoefficients**
  The complex coefficients of the gammatone filter bands.
- float * **icoefficients**
- float * **rstates**
- float * **istates**
- float * **sout_buf**
  output signal buffer, used by s_out.
- float * **large_array**
  Large float array.
- **mha_wave_t** **s_out**

5.157.1 Detailed Description

Configuration for Gammatone Filterbank SIMD Analyzer.

5.157.2 Constructor & Destructor Documentation

5.157.2.1 gtfb_simd_cfg_t() [1/3]

```
gtfb_simd_cfg_t::gtfb_simd_cfg_t (
    gtfb_simd_cfg_t && ) [delete]
```

5.157.2.2 gtfb_simd_cfg_t() [2/3]

```
gtfb_simd_cfg_t::gtfb_simd_cfg_t (
    const gtfb_simd_cfg_t & ) [delete]
```

5.157.2.3 gtfb_simd_cfg_t() [3/3]

```
gtfb_simd_cfg_t::gtfb_simd_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.

5.157.2.4 ~gtfb_simd_cfg_t() gtfb_simd_cfg_t::~gtfb_simd_cfg_t ( ) [inline]

5.157.3 Member Function Documentation

5.157.3.1 get_bands() unsigned gtfb_simd_cfg_t::get_bands ( ) const [inline]

Each band is split into this number of bands.

5.157.3.2 get_channels() unsigned gtfb_simd_cfg_t::get_channels ( ) const [inline]

The number of separate audio channels.

5.157.3.3 get_frames() unsigned gtfb_simd_cfg_t::get_frames ( ) const [inline]

The number of frames in one chunk.

5.157.3.4 operator=() [1/2] gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= (gtfb_simd_cfg_t &&) [delete]

5.157.3.5 operator=() [2/2] gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= (const gtfb_simd_cfg_t &) [delete]

5.157.3.6 process() mha_wave_t* gtfb_simd_cfg_t::process (mha_wave_t * *s_in*) [inline]

5.157.4 Member Data Documentation

5.157.4.1 order `unsigned gtfb_simd_cfg_t::order [private]`

The order of the gammatone filters.

5.157.4.2 bands `unsigned gtfb_simd_cfg_t::bands [private]`

Number of frequency bands per channel.

5.157.4.3 channels `unsigned gtfb_simd_cfg_t::channels [private]`

Number of input audio channels.

5.157.4.4 bandsXchannels `unsigned gtfb_simd_cfg_t::bandsXchannels [private]`

Product of bands and channels.

5.157.4.5 norm_phase `std::vector< mha_complex_t> gtfb_simd_cfg_t::norm_phase [private]`

Combination of normalization and phase correction factor.

5.157.4.6 rinputs `float* gtfb_simd_cfg_t::rinputs [private]`

input signal (1 sample) multiplied with norm_phase

5.157.4.7 iinputs `float* gtfb_simd_cfg_t::iinputs [private]`

5.157.4.8 rcoefficients `float* gtfb_simd_cfg_t::rcoefficients [private]`

The complex coefficients of the gammatone filter bands.

5.157.4.9 icoefficients `float* gtfb_simd_cfg_t::icoefficients [private]`

5.157.4.10 rstates `float* gtfb_simd_cfg_t::rstates [private]`

Storage for Filter state. Holds **channels()** (p. 662) * **bands()** (p. 662) * order complex filter states. Layout: state[stage * bandsXchannels + **bands()** (p. 662)*channel+band]

5.157.4.11 istates `float* gtfb_simd_cfg_t::istates [private]`

5.157.4.12 sout_buf `float* gtfb_simd_cfg_t::sout_buf [private]`

output signal buffer, used by s_out.

Contains bandsXchannels * fragsize *2 entries. all real parts come before all complex parts. Order is: channel 0 band 0 real, channel 0 band 1 real, ..., channel 1 band 0 real channel 1 band 1 real, ... channel 0 band 0 imag ... then next sample

5.157.4.13 large_array `float* gtfb_simd_cfg_t::large_array [private]`

Large float array.

All previous pointers point into this array. Contains 3 unused entries to be able to adjust for alignment.

5.157.4.14 s_out `mha_wave_t gtfb_simd_cfg_t::s_out [private]`

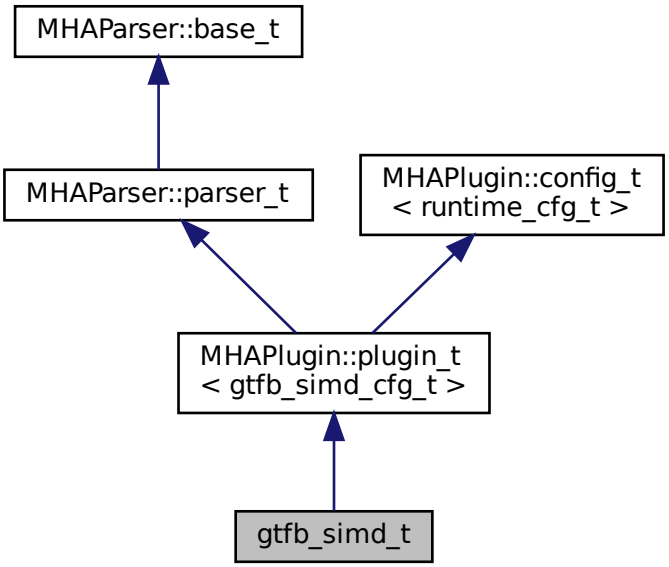
Storage for the (complex) output signal. Each of the real input audio channels is split into frequency bands with complex time signal output. The split complex time signal is again stored in a **mha_wave_t** (p. 985) buffer. Each complex time signal is stored as adjacent real and imaginary channels. Complex output from one source channel is stored in adjacent complex output channels.

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

- **gtfb_simd.cpp**

5.158 gtfb_simd_t Class Reference

Inheritance diagram for gtfb_simd_t:



Public Member Functions

- **gtfb_simd_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< gtfb_simd_t > patchbay**
- bool **prepared**
- **MHAParser::int_t order**
- **MHAParser::vcomplex_t coeff**
- **MHAParser::vcomplex_t norm_phase**

Additional Inherited Members

5.158.1 Constructor & Destructor Documentation

5.158.1.1 gtfb_simd_t()

```
gtfb_simd_t::gtfb_simd_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.158.2 Member Function Documentation

5.158.2.1 process()

```
mha_wave_t * gtfb_simd_t::process (
    mha_wave_t * s )
```

5.158.2.2 prepare()

```
void gtfb_simd_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< gtfb_simd_cfg_t >** (p. 1301).

5.158.2.3 update_cfg() void gtfb_simd_t::update_cfg (void) [private]

5.158.3 Member Data Documentation

5.158.3.1 patchbay MHAEvents::patchbay_t< gtfb_simd_t> gtfb_simd_t::patchbay [private]

5.158.3.2 prepared bool gtfb_simd_t::prepared [private]

5.158.3.3 order MHAParser::int_t gtfb_simd_t::order [private]

5.158.3.4 coeff MHAParser::vcomplex_t gtfb_simd_t::coeff [private]

5.158.3.5 norm_phase MHAParser::vcomplex_t gtfb_simd_t::norm_phase [private]

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

- **gtfb_simd.cpp**

5.159 gtfb_simple_rt_t Class Reference

Runtime configuration class of gtfb_simple_bridge plugin.

Public Member Functions

- **gtfb_simple_rt_t** (**MHA_AC::algo_comm_t** &ac, const std::string &name, **mhaconfig_t** &chcfg, std::vector< **mha_real_t** > cf, std::vector< **mha_real_t** > bw, unsigned int order, unsigned int pre_stages, unsigned int desired_delay, std::vector< **mha_real_t** > &vcltass, std::vector< **mha_real_t** > &resynthesis_gain, const std::string &element_gain_name)
- **mha_wave_t** * **pre_plugin** (**mha_wave_t** *s)

  Filter real input signal s with the pre_stages filter orders in each gammatone filter band.
- **mha_wave_t** * **post_plugin** (**mha_wave_t** *s)

  Post-filter the complex-valued filter-bank signal s after it has been processed by the loaded plugin.
- void **insert_ac_variables** ()

  (Re-)insert all AC variables into the AC space.
- const **MHAFilter::gamma_flt_t** & **get_gf** () const

  Const-accessor to contained gammatone filterbank object.

Static Private Member Functions

- static std::vector< **mha_real_t** > **duplicate_vector** (const std::vector< **mha_real_t** > &src, unsigned int nchannels)

  Helper function to repeat the elements in a vector.

Private Attributes

- unsigned int **_order**

  Total number of gammatone filter orders.
- unsigned int **_pre_stages**

  Number of filter orders applied before the loaded plugin is invoked.
- unsigned int **nbands**

  Number of frequency bands to produce = number of gammatone filters.
- **MHA_AC::waveform_t imag**

  Storage for the imaginary part of the filterbank signal.
- **MHA_AC::waveform_t accf**

  AC variable to publish the center frequencies of the gammatone filters.
- **MHA_AC::waveform_t acbw**

  AC variable to publish the bandwiths of the gammatone filters.
- **MHASignal::waveform_t input**

  Real part of the filterbank signal.
- **MHASignal::waveform_t output**

  Resynthesized broadband signal, used as the output signal of this plugin.
- **MHAFilter::gamma_flt_t gf**

  The gammatone filter bank implementation.
- **MHA_AC::waveform_t cLTASS**

AC variable to publish band-specific LTASS level correction values.

- **MHA_AC::waveform_t ac_resynthesis_gain**

  AC variable to publish the configured per-frequency resyntesis gains.

- std::string **element_gain_name_**

  Either an empty string, or the name of an AC variable from which element-wise linear factors are read.

- **MHA_AC::algo_comm_t** & **_ac**

  Algorithm Communication Variable space.

5.159.1 Detailed Description

Runtime configuration class of gtfb_simple_bridge plugin.

5.159.2 Constructor & Destructor Documentation

5.159.2.1 gtfb_simple_rt_t() `gtfb_simple_rt_t::gtfb_simple_rt_t (`

```
 MHA_AC::algo_comm_t & ac,
const std::string & name,
 mhaconfig_t & chcfg,
std::vector< mha_real_t > cf,
std::vector< mha_real_t > bw,
unsigned int order,
unsigned int pre_stages,
unsigned int desired_delay,
std::vector< mha_real_t > & vcltass,
std::vector< mha_real_t > & resynthesis_gain,
const std::string & element_gain_name )
```

5.159.3 Member Function Documentation

5.159.3.1 pre_plugin() `mha_wave_t * gtfb_simple_rt_t::pre_plugin ( mha_wave_t * s )`

Filter real input signal s with the pre_stages filter orders in each gammatone filter band.

The real part of the complex output is returned in the return value of the method, the imaginary part is stored into the AC variable.

Parameters

s	real-valued, broad-band input signal

Returns

real part of complex-valued output signal after pre_stages gammatone filter orders have been applied in each band. Order of output bands in real and imaginary output are: (channel0,band0), (channel0,band1), ..., (cannel1,band0), ...

5.159.3.2 post_plugin() `mha_wave_t * gtfb_simple_rt_t::post_plugin ( mha_wave_t * s )`

Post-filter the complex-valued filter-bank signal s after it has been processed by the loaded plugin.

The remaining gammatone filter orders are applied to restrict the loaded plugin's output signal to the respective bands. After

Parameters

s	complex-valued, filter-bank signal. This signal is produced by letting the loaded plugin process the output signal of the pre_plugin method.

Returns

real part of complex-valued output signal after pre_stages gammatone filter orders have been applied in each band. Order of output bands in real and imaginary output are: (channel0,band0), (channel0,band1), ..., (cannel1,band0), ...

5.159.3.3 insert_ac_variables() void gtfb_simple_rt_t::insert_ac_variables ( )

(Re-)insert all AC variables into the AC space.

Must be called during each prepare() and process() callback of the plugin. For the process() callback, this method is called from **pre_plugin()** (p. 668). For the prepare() callback, it must be called by the plugin interface.

5.159.3.4 get_gf() const MHAFilter::gamma_flt_t& gtfb_simple_rt_t::get_gf ( ) const [inline]

Const-accessor to contained gammatone filterbank object.

5.159.3.5 duplicate_vector() std::vector< mha_real_t > gtfb_simple_rt_t::duplicate_vector (

```
const std::vector< mha_real_t > & src,
unsigned int nchannels ) [static], [private]
```

Helper function to repeat the elements in a vector.

Parameters

src	vector to repeat
nchannels	number of times to repeat input vector

Returns

a vector that returns nchannels repetitions of input vector.

5.159.4 Member Data Documentation

5.159.4.1 _order unsigned int gtfb_simple_rt_t::_order [private]

Total number of gammatone filter orders.

5.159.4.2 _pre_stages unsigned int gtfb_simple_rt_t::_pre_stages [private]

Number of filter orders applied before the loaded plugin is invoked.

5.159.4.3 nbands unsigned int gtfb_simple_rt_t::nbands [private]

Number of frequency bands to produce = number of gammatone filters.

5.159.4.4 imag MHA_AC::waveform_t gtfb_simple_rt_t::imag [private]

Storage for the imaginary part of the filterbank signal.

It is used as the imaginary input signal for the loaded plugin. Furthermore, it is expected that the loaded plugin processes the imaginary part of the data in place.

5.159.4.5 accf MHA_AC::waveform_t gtfb_simple_rt_t::accf [private]

AC variable to publish the center frequencies of the gammatone filters.

5.159.4.6 acbw MHA_AC::waveform_t gtfb_simple_rt_t::acbw [private]

AC variable to publish the bandwiths of the gammatone filters.

5.159.4.7 input MHASignal::waveform_t gtfb_simple_rt_t::input [private]

Real part of the filterbank signal.

It is used as the real input signal to the loaded plugin.

5.159.4.8 output MHASignal::waveform_t gtfb_simple_rt_t::output [private]

Resynthesized broadband signal, used as the output signal of this plugin.

5.159.4.9 gf `MHAFilter::gamma_flt_t gtfb_simple_rt_t::gf [private]`

The gammatone filter bank implementation.

5.159.4.10 cLTASS `MHA_AC::waveform_t gtfb_simple_rt_t::cLTASS [private]`

AC variable to publish band-specific LTASS level correction values.

5.159.4.11 ac_resynthesis_gain `MHA_AC::waveform_t gtfb_simple_rt_t::ac_resynthesis_gain [private]`

AC variable to publish the configured per-frequency resyntesis gains.

5.159.4.12 element_gain_name_ `std::string gtfb_simple_rt_t::element_gain_name_ [private]`

Either an empty string, or the name of an AC variable from which element-wise linear factors are read.

5.159.4.13 _ac `MHA_AC::algo_comm_t& gtfb_simple_rt_t::_ac [private]`

Algorithm Communication Variable space.

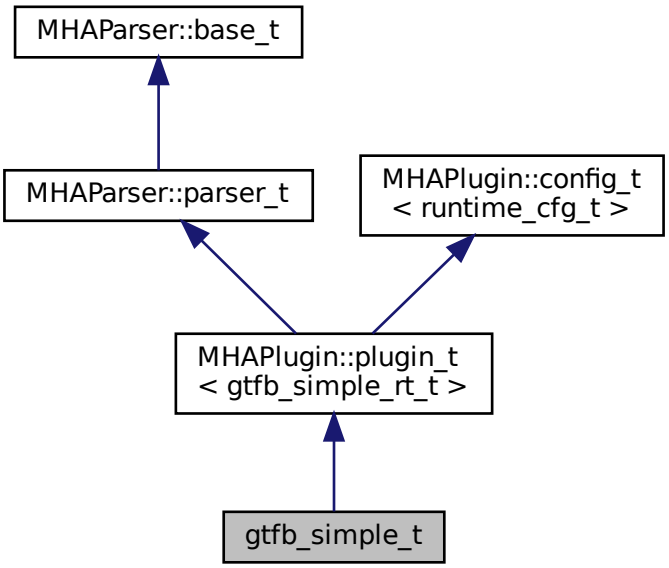
The documentation for this class was generated from the following file:

- **gtfb_simple_bridge.cpp**

5.160 gtfb_simple_t Class Reference

Interface class of gtfb_simple_bridge plugin.

Inheritance diagram for gtfb_simple_t:



Public Member Functions

- **gtfb_simple_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)

  Constructor.
- void **prepare** (**mhaconfig_t** &chcfg)

  Prepare contained plugin for signal processing.
- void **release** ()

  Releases contained plugin.
- **mha_wave_t** * **process** (**mha_wave_t** *sln)

  Process the input signal: Computes the filterbank with the pre-stages, calls the loaded plugin for processing of the filterbank signal, and resynthesizes the result to a modified broadband signal.
- void **setlock** (bool b)

  Locks / unlocks all configuration variables before / after signal processing.

Private Attributes

- **MHAParser::mhapluginloader_t plug**

  Handle to the loaded plugin.
- **MHAOvlFilter::fscale_bw_t fscale**

  Object determines the filterbank frequencies.
- **MHAParser::int_t order**

  total order of gammatone filter
- **MHAParser::int_t prestages**

  Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.
- **MHAParser::int_t desired_delay**

  Desired group delay in audio samples.
- **MHAParser::string_t element_gain_name**

  Number of AC variable to take element-wise gain factors from for resynthesis.
- **MHAParser::vfloat_mon_t cLTASS**

  Monitoring of LTASS correction values / dB.
- **MHAParser::vfloat_mon_t resynthesis_gain**

  configured frequency-specific resynthesis gains
- **MHAParser::string_mon_t gf_internals**

  For tests and debugging: a serialization of the gammatone filter internals.
- std::string **name_**

  Configured algorithm name, used to name the AC variables.

Additional Inherited Members

5.160.1 Detailed Description

Interface class of gtfb_simple_bridge plugin.

5.160.2 Constructor & Destructor Documentation

5.160.2.1 gtfb_simple_t() gtfb_simple_t::gtfb_simple_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

Constructor.

Registers parser variables.

Parameters

ac	Algorithm Communication Variable space
chain	chain name
algo	configured name of this plugin instance

5.160.3 Member Function Documentation

5.160.3.1 prepare() void gtfb_simple_t::prepare (mhaconfig_t & chcfg) [virtual]

Prepare contained plugin for signal processing.

Allocates the runtime configuration instance. Locks all variables.

Implements **MHAPlugin::plugin_t< gtfb_simple_rt_t >** (p. 1301).

5.160.3.2 release() void gtfb_simple_t::release (void) [virtual]

Releases contained plugin.

Unlocks all variables.

Reimplemented from **MHAPlugin::plugin_t< gtfb_simple_rt_t >** (p. 1302).

5.160.3.3 process() `mha_wave_t * gtfb_simple_t::process ( mha_wave_t * sIn )`

Process the input signal: Computes the filterbank with the pre-stages, calls the loaded plugin for processing of the filterbank signal, and resynthesizes the result to a modified broadband signal.

5.160.3.4 setlock() `void gtfb_simple_t::setlock ( bool b ) [inline]`

Locks / unlocks all configuration variables before / after signal processing.

5.160.4 Member Data Documentation

5.160.4.1 plug `MHAParser::mhapluginloader_t gtfb_simple_t::plug [private]`

Handle to the loaded plugin.

5.160.4.2 fscale `MHAOvlFilter::fscale_bw_t gtfb_simple_t::fscale [private]`

Object determines the filterbank frequencies.

5.160.4.3 order `MHAParser::int_t gtfb_simple_t::order [private]`

total order of gammatone filter

5.160.4.4 prestages `MHAParser::int_t gtfb_simple_t::prestages [private]`

Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.

5.160.4.5 desired_delay

`MHAParser::int_t gtfb_simple_t::desired_delay [private]`

Desired group delay in audio samples.

5.160.4.6 element_gain_name

`MHAParser::string_t gtfb_simple_t::element_gain_name [private]`

Number of AC variable to take element-wise gain factors from for resynthesis.

5.160.4.7 cLTASS

`MHAParser::vfloat_mon_t gtfb_simple_t::cLTASS [private]`

Monitoring of LTASS correction values / dB.

5.160.4.8 resynthesis_gain

`MHAParser::vfloat_mon_t gtfb_simple_t::resynthesis_gain [private]`

configured frequency-specific resynthesis gains

5.160.4.9 gf_internals

`MHAParser::string_mon_t gtfb_simple_t::gf_internals [private]`

For tests and debugging: a serialization of the gammatone filter internals.

5.160.4.10 name_

`std::string gtfb_simple_t::name_ [private]`

Configured algorithm name, used to name the AC variables.

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

- **gtfb_simple_bridge.cpp**

5.161 hanning_ramps_t Class Reference

Class for storing two hanning ramps: a rising ramp a and a falling ramp b.

Public Member Functions

- **hanning_ramps_t** (unsigned int **len_a**, unsigned int **len_b**)
  Initialize two arrays with hanning ramp factors.
- ∼**hanning_ramps_t** ()
- void **operator()** (**MHASignal::waveform_t** &b)
  Apply the hanning ramps to a waveform.

Private Attributes

- unsigned int **len_a**
  Length of the rising ramp a.
- unsigned int **len_b**
  Length of the falling ramp b.
- **mha_real_t** ∗ **ramp_a**
  Factors of the rising hanning ramp from 0 (inclusive) to 1 (exclusive).
- **mha_real_t** ∗ **ramp_b**
  Factors of the falling hanning ramp from 1 (inclusive) to 0(exclusive).

5.161.1 Detailed Description

Class for storing two hanning ramps: a rising ramp a and a falling ramp b.

5.161.2 Constructor & Destructor Documentation

5.161.2.1 hanning_ramps_t() `hanning_ramps_t::hanning_ramps_t (`

```
unsigned int len_a,
unsigned int len_b )
```

Initialize two arrays with hanning ramp factors.

Parameters

len_a	Length of the rising ramp a from 0 (inclusive) to 1 (exclusive).
len_b	Length of the falling ramp b from 1 (inclusive) to 0 (exclusive).

5.161.2.2 ~hanning_ramps_t() `hanning_ramps_t::~hanning_ramps_t ( void )`

5.161.3 Member Function Documentation

5.161.3.1 operator()() `void hanning_ramps_t::operator() ( MHASignal::waveform_t & b )`

Apply the hanning ramps to a waveform.

Parameters

b	The waveform to which the ramps are applied. ramp_a is applied to the first len_a frames of b, ramp_b is applied to the last len_b frames of b. Ramps are applied to all channels. If the waveform has fewer frames than len_a or len_b, then the respective ramp is truncated. The audio samples stored by b are modified in place.

5.161.4 Member Data Documentation

5.161.4.1 len_a `unsigned int hanning_ramps_t::len_a [private]`

Length of the rising ramp a.

5.161.4.2 len_b `unsigned int hanning_ramps_t::len_b [private]`

Length of the falling ramp b.

5.161.4.3 ramp_a `mha_real_t* hanning_ramps_t::ramp_a [private]`

Factors of the rising hanning ramp from 0 (inclusive) to 1 (exclusive).

5.161.4.4 ramp_b `mha_real_t* hanning_ramps_t::ramp_b [private]`

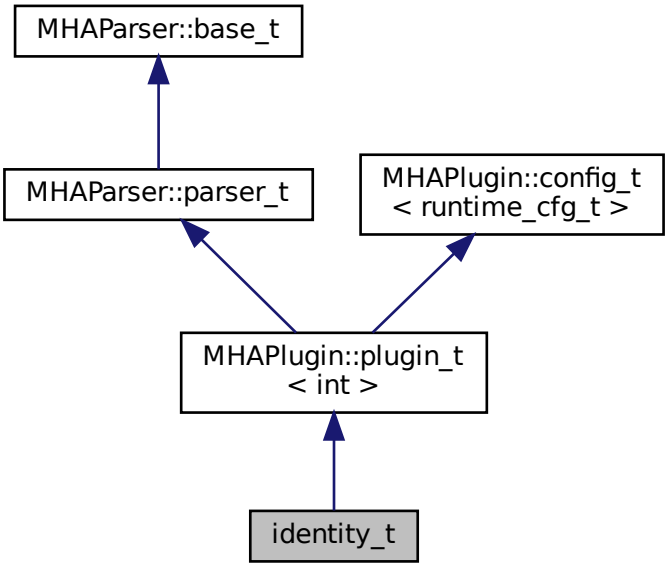
Factors of the falling hanning ramp from 1 (inclusive) to 0(exclusive).

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

- **spec2wave.cpp**

5.162 identity_t Class Reference

Inheritance diagram for identity_t:



Public Member Functions

- **identity_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Additional Inherited Members

5.162.1 Constructor & Destructor Documentation

5.162.1.1 identity_t() `identity_t::identity_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.162.2 Member Function Documentation

5.162.2.1 process() [1/2] `mha_wave_t * identity_t::process (`

```
mha_wave_t * s )
```

5.162.2.2 process() [2/2] `mha_spec_t * identity_t::process (`

```
mha_spec_t * s )
```

5.162.2.3 prepare() `void identity_t::prepare (`

```
mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin_t**< **int** > (p. 1301).

5.162.2.4 release() `void identity_t::release ( void ) [virtual]`

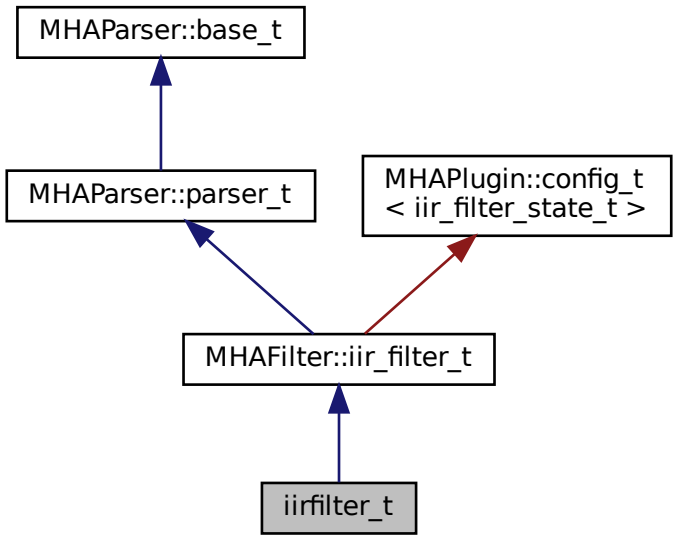
Reimplemented from **MHAPlugin::plugin_t**< **int** > (p. 1302).

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

- **identity.cpp**

5.163 iirfilter_t Class Reference

Inheritance diagram for iirfilter_t:



Public Member Functions

- **iirfilter_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare_** (**mhaconfig_t** &)
- void **release_** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Additional Inherited Members

5.163.1 Constructor & Destructor Documentation

5.163.1.1 iirfilter_t() `iirfilter_t::iirfilter_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.163.2 Member Function Documentation

5.163.2.1 prepare_() `void iirfilter_t::prepare_ (`

```
mhaconfig_t & tf )
```

5.163.2.2 release_() `void iirfilter_t::release_ ( ) [inline]`

5.163.2.3 process() `mha_wave_t * iirfilter_t::process (`

```
mha_wave_t * s )
```

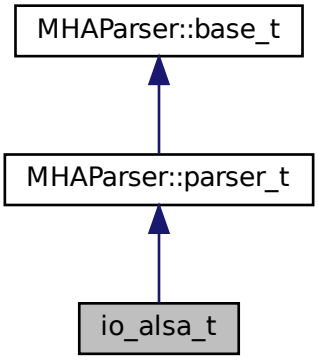
The documentation for this class was generated from the following file:

- **iirfilter.cpp**

5.164 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**, **IOStopped↩ Event_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

5.164.1 Detailed Description

MHA IO interface class for ALSA IO.

5.164.2 Constructor & Destructor Documentation

5.164.2.1 io_alsa_t() 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.

5.164.3 Member Function Documentation

5.164.3.1 prepare() [1/2] 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

5.164.3.2 release() void io_alsa_t::release (

```
            void  )
```

MHA framework leaves prepared state.

5.164.3.3 start() void io_alsa_t::start ( )

MHA framework calls this function when signal processing should start.

5.164.3.4 stop() void io_alsa_t::stop ( )

MHA framework calls this function when signal processing should stop.

5.164.3.5 thread_start() 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.

5.164.3.6 process() void io_alsa_t::process (void) [private]

5.164.3.7 prepare() [2/2] template<>
void io_alsa_t::prepare (int *nch_in*, int *nch_out*)

5.164.4 Member Data Documentation

5.164.4.1 b_process bool io_alsa_t::b_process [private]

5.164.4.2 fw_fragsize unsigned int io_alsa_t::fw_fragsize [private]

5.164.4.3 fw_samplerate unsigned int io_alsa_t::fw_samplerate [private]

5.164.4.4 proc_event **IOProcessEvent_t** io_alsa_t::proc_event [private]

5.164.4.5 proc_handle `void* io_alsa_t::proc_handle [private]`

5.164.4.6 start_event `IOStartedEvent_t io_alsa_t::start_event [private]`

5.164.4.7 start_handle `void* io_alsa_t::start_handle [private]`

5.164.4.8 stop_event `IOStoppedEvent_t io_alsa_t::stop_event [private]`

5.164.4.9 stop_handle `void* io_alsa_t::stop_handle [private]`

5.164.4.10 dev_in `alsa_base_t* io_alsa_t::dev_in [private]`

5.164.4.11 dev_out `alsa_base_t* io_alsa_t::dev_out [private]`

5.164.4.12 proc_thread `pthread_t io_alsa_t::proc_thread [private]`

5.164.4.13 p_in `alsa_dev_par_parser_t io_alsa_t::p_in [private]`

5.164.4.14 p_out

`alsa_dev_par_parser_t io_alsa_t::p_out [private]`

5.164.4.15 pcmlink

`MHAParser::bool_t io_alsa_t::pcmlink [private]`

5.164.4.16 priority

`MHAParser::int_t io_alsa_t::priority [private]`

5.164.4.17 format

`MHAParser::kw_t io_alsa_t::format [private]`

5.164.4.18 alsa_start_counter

`MHAParser::int_mon_t io_alsa_t::alsa_start_counter [private]`

5.164.4.19 patchbay

`MHAEvents::patchbay_t< io_alsa_t> io_alsa_t::patchbay [private]`

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

- **MHAIOalsa.cpp**

5.165 io_asterisk_fwcb_t Class Reference

TCP sound-io library's interface to the framework callbacks.

Public Member Functions

- **io_asterisk_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_asterisk_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**

5.165.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

5.165.2 Constructor & Destructor Documentation

5.165.2.1 io_asterisk_fwcb_t() io_asterisk_fwcb_t::io_asterisk_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.

5.165.2.2 ~io_asterisk_fwcb_t() virtual io_asterisk_fwcb_t::~io_asterisk_fwcb_t ( ) [inline], [virtual]

Do-nothing destructor.

5.165.3 Member Function Documentation

5.165.3.1 start() void io_asterisk_fwcb_t::start ( ) [virtual]

Call the framework's start callback.

5.165.3.2 process() int io_asterisk_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.

5.165.3.3 set_errnos() void io_asterisk_fwcb_t::set_errnos (int *proc_err*, int *io_err*) [virtual]

Save error numbers to use during.

See also

stop (p. 692)

Parameters

proc_err	The error number from the

See also

process (p. 691) callback.

Parameters

io_err	The error number from the io library itself.

5.165.3.4 stop() void io_asterisk_fwcb_t::stop ( ) [virtual]

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

set_errnos (p. 691).

5.165.4 Member Data Documentation

5.165.4.1 proc_event `IOProcessEvent_t io_asterisk_fwcb_t::proc_event [private]`

Pointer to signal processing callback function.

5.165.4.2 start_event `IOStartedEvent_t io_asterisk_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.

5.165.4.3 stop_event `IOStoppedEvent_t io_asterisk_fwcb_t::stop_event [private]`

Pointer to stop notification callback function.

Called when the connection is closed.

5.165.4.4 proc_handle `void* io_asterisk_fwcb_t::proc_handle [private]`

Handles belonging to framework.

5.165.4.5 start_handle `void * io_asterisk_fwcb_t::start_handle [private]`

5.165.4.6 stop_handle `void * io_asterisk_fwcb_t::stop_handle [private]`

5.165.4.7 proc_err `int io_asterisk_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. 692) from that callback. Within **stop** (p. 692), these error numbers are read again and transmitted to the framework.

5.165.4.8 io_err `int io_asterisk_fwcb_t::io_err [private]`

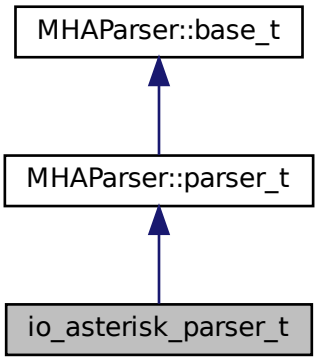
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

5.166 io_asterisk_parser_t Class Reference

The parser interface of the IOAsterisk library.

Inheritance diagram for io_asterisk_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_asterisk_parser_t** ()

  Constructor initializes parser variables.
- virtual ∼**io_asterisk_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

5.166.1 Detailed Description

The parser interface of the IOAsterisk library.

5.166.2 Constructor & Destructor Documentation

5.166.2.1 io_asterisk_parser_t() `io_asterisk_parser_t::io_asterisk_parser_t ( )`

Constructor initializes parser variables.

5.166.2.2 ~io_asterisk_parser_t() `virtual io_asterisk_parser_t::~io_asterisk_parser_t ( ) [inline], [virtual]`

Do-nothing destructor.

5.166.3 Member Function Documentation

5.166.3.1 get_local_address() `virtual const std::string& io_asterisk_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.

5.166.3.2 get_local_port() `unsigned short io_asterisk_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.

5.166.3.3 set_local_port() `void io_asterisk_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 over 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. 696) currently returns 0.

5.166.3.4 get_server_port_open() `bool io_asterisk_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. 697).

5.166.3.5 set_server_port_open() `void io_asterisk_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. 697) currently returns false.

Postcondition

See also

get_server_port_open (p. 697) returns the **value** (p. 48) of open.

5.166.3.6 get_connected() `bool io_asterisk_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. 698).

5.166.3.7 set_connected() `void io_asterisk_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. 698).

Postcondition

See also

get_connected (p. 698) returns the **value** (p. 48) of open.

5.166.3.8 set_new_peer()

```
void io_asterisk_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. 698) currently returns false.

Postcondition

See also

get_connected (p. 698) returns true.

5.166.3.9 debug() virtual void io_asterisk_parser_t::debug (const std::string & *message*) [inline], [virtual]

5.166.4 Member Data Documentation

5.166.4.1 local_address MHAParser::string_t io_asterisk_parser_t::local_address [private]

Lets the user set the local network interface to listen on.

5.166.4.2 local_port MHAParser::int_t io_asterisk_parser_t::local_port [private]

Lets the user choose the local tcp port to listen on.

5.166.4.3 server_port_open MHAParser::int_mon_t io_asterisk_parser_t::server_↩ port_open [private]

Indicates wether the TCP server socket is currently open.

5.166.4.4 connected MHAParser::int_mon_t io_asterisk_parser_t::connected [private]

Indicator if there currently is a sound data connection over TCP.

5.166.4.5 peer_address MHAParser::string_mon_t io_asterisk_parser_t::peer_↩ address [private]

Display the ip address of the currently connected sound data client.

5.166.4.6 peer_port `MHAParser::int_mon_t io_asterisk_parser_t::peer_port [private]`

Display the tcp port used by the current sound data client.

5.166.4.7 debug_filename `MHAParser::string_t io_asterisk_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

5.166.4.8 debug_file `FILE* io_asterisk_parser_t::debug_file [private]`

file handle to write debugging info to

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

- **MHAIOAsterisk.cpp**

5.167 io_asterisk_sound_t Class Reference

Sound data handling of io tcp library.

Public Member Functions

- **io_asterisk_sound_t** (int **fragsize**, float **samplerate**)
  Initialize sound data handling.
- virtual ~**io_asterisk_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.
- std::string & **hton** (const **mha_wave_t** *s_out)
  Serialize data for network transfer.
- **mha_wave_t** * **ntoh** (const std::string &data)
  Deserialize data from network.

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.
- std::string **output_data**
  Serialized data for network trnasfer.

5.167.1 Detailed Description

Sound data handling of io tcp library.

5.167.2 Constructor & Destructor Documentation

5.167.2.1 io_asterisk_sound_t() `io_asterisk_sound_t::io_asterisk_sound_t ( int fragsize, float samplerate )`

Initialize sound data handling.

Parameters

fragsize	Number of sound samples in each channel expected and returned from processing callback.
samplerate	Number of samples per second in each channel.

5.167.2.2 ~io_asterisk_sound_t() `virtual io_asterisk_sound_t::~io_asterisk_sound_t ( ) [inline], [virtual]`

Do-nothing destructor.

5.167.3 Member Function Documentation

5.167.3.1 prepare() `void io_asterisk_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.

5.167.3.2 release() `void io_asterisk_sound_t::release ( void ) [virtual]`

Called during release.

Deletes sound data storage.

5.167.3.3 chunkbytes_in() `int io_asterisk_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.

5.167.3.4 header() `std::string io_asterisk_sound_t::header ( ) const [virtual]`

Create the tcp sound header lines.

5.167.3.5 hton() `std::string & io_asterisk_sound_t::hton ( const mha_wave_t * s_out )`

Serialize data for network transfer.

5.167.3.6 ntoh() `mha_wave_t * io_asterisk_sound_t::ntoh ( const std::string & data )`

Deserialize data from network.

5.167.4 Member Data Documentation

5.167.4.1 fragsize `int io_asterisk_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

5.167.4.2 samplerate `float io_asterisk_sound_t::samplerate [private]`

Sampling rate.

Number of samples per second in each channel.

5.167.4.3 num_inchannels `int io_asterisk_sound_t::num_inchannels [private]`

Number of input channels.

Number of channels expected from and returned by signal processing callback.

5.167.4.4 num_outchannels `int io_asterisk_sound_t::num_outchannels [private]`

5.167.4.5 s_in `MHASignal::waveform_t* io_asterisk_sound_t::s_in [private]`

Storage for input signal.

5.167.4.6 output_data `std::string io_asterisk_sound_t::output_data [private]`

Serialized data for network trnasfer.

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

- **MHAIOAsterisk.cpp**

5.168 io_asterisk_t Class Reference

The tcp sound io library.

Public Member Functions

- **io_asterisk_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_asterisk_t** ()

Private Attributes

- **io_asterisk_parser_t parser**
- **io_asterisk_sound_t sound**
- **io_asterisk_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**

5.168.1 Detailed Description

The tcp sound io library.

5.168.2 Constructor & Destructor Documentation

5.168.2.1 io_asterisk_t()

```
io_asterisk_t::io_asterisk_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 )
```

5.168.2.2 ~io_asterisk_t()

```
virtual io_asterisk_t::~io_asterisk_t ( ) [inline], [virtual]
```

5.168.3 Member Function Documentation

5.168.3.1 prepare()

```
void io_asterisk_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

5.168.3.2 start()

```
void io_asterisk_t::start ( )
```

Call frameworks start callback if there is a sound data connection at the moment.

5.168.3.3 stop()

```
void io_asterisk_t::stop ( )
```

Close the current connection if there is one.

stop IO thread

5.168.3.4 release()

```
void io_asterisk_t::release (
    void )
```

Close the current connection and close the server socket.

Stop IO thread and close server socket.

5.168.3.5 accept_loop()

```
void io_asterisk_t::accept_loop ( ) [virtual]
```

IO thread executes this method.

5.168.3.6 connection_loop()

```
void io_asterisk_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.

5.168.3.7 parse()

```
virtual void io_asterisk_t::parse (
    const char * cmd,
    char * retval,
    unsigned int len )  [inline], [virtual]
```

Parser interface.

5.168.4 Member Data Documentation

5.168.4.1 parser

io_asterisk_parser_t io_asterisk_t::parser [private]

5.168.4.2 sound

io_asterisk_sound_t io_asterisk_t::sound [private]

5.168.4.3 fwcb

io_asterisk_fwcb_t io_asterisk_t::fwcb [private]

5.168.4.4 server

MHA_TCP::Server* io_asterisk_t::server [private]

5.168.4.5 thread

MHA_TCP::Thread* io_asterisk_t::thread [private]

5.168.4.6 notify_start `MHA_TCP::Async_Notify io_asterisk_t::notify_start [private]`

5.168.4.7 notify_stop `MHA_TCP::Async_Notify io_asterisk_t::notify_stop [private]`

5.168.4.8 notify_release `MHA_TCP::Async_Notify io_asterisk_t::notify_release [private]`

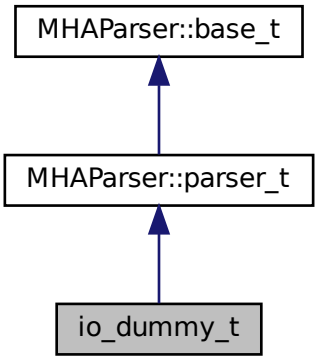
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

5.169 io_dummy_t Class Reference

Dummy sound io library.

Inheritance diagram for io_dummy_t:



Public Member Functions

- **io_dummy_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 nch_in)
  Prepare.
- void **release** ()
  Release.
- void **start** ()
  Initialize main_loop and start the loop immediately, stop on error.
- void **stop** ()
  Send stop request to main loop and join thread.

Private Attributes

- float **samplerate**
  The framework's sampling rate.
- unsigned int **fragsize**
  The framework's frag size.
- **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**
- std::unique_ptr< **MHASignal::waveform_t** > **in**
  Input sound for framework, always zero.
- **mha_wave_t** * **out**
  Output from framework, always ignored.
- std::thread **main_loop**
  Main event loop.
- std::atomic< bool > **stop_request**
  Stop request flag for main_loop.

Additional Inherited Members

5.169.1 Detailed Description

Dummy sound io library.

Simulates real time sound. Input is always zero, output is always ignored.

5.169.2 Constructor & Destructor Documentation

5.169.2.1 io_dummy_t() io_dummy_t::io_dummy_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 )
```

5.169.3 Member Function Documentation

5.169.3.1 prepare() void io_dummy_t::prepare (

```
int nch_in )
```

Prepare.

Initialized the input buffer

5.169.3.2 release() void io_dummy_t::release (

```
void  )
```

Release.

Frees the input buffer

5.169.3.3 start() void io_dummy_t::start ( )

Initialize main_loop and start the loop immediately, stop on error.

5.169.3.4 stop() void io_dummy_t::stop ( )

Send stop request to main loop and join thread.

5.169.4 Member Data Documentation

5.169.4.1 samplerate `float io_dummy_t::samplerate [private]`

The framework's sampling rate.

5.169.4.2 fragsize `unsigned int io_dummy_t::fragsize [private]`

The framework's frag size.

5.169.4.3 proc_event `IOProcessEvent_t io_dummy_t::proc_event [private]`

Pointer to signal processing callback function.

5.169.4.4 start_event `IOStartedEvent_t io_dummy_t::start_event [private]`

Pointer to start notification callback function.

Called when the user issues the start command.

5.169.4.5 stop_event `IOStoppedEvent_t io_dummy_t::stop_event [private]`

Pointer to stop notification callback function.

Called when the user issues a stop request

5.169.4.6 proc_handle `void* io_dummy_t::proc_handle [private]`

Handles belonging to framework.

5.169.4.7 start_handle `void * io_dummy_t::start_handle [private]`

5.169.4.8 stop_handle `void * io_dummy_t::stop_handle [private]`

5.169.4.9 in `std::unique_ptr< MHASignal::waveform_t> io_dummy_t::in [private]`

Input sound for framework, always zero.

5.169.4.10 out `mha_wave_t* io_dummy_t::out [private]`

Output from framework, always ignored.

5.169.4.11 main_loop `std::thread io_dummy_t::main_loop [private]`

Main event loop.

Calls the framework's process chain periodically according to sampling rate and fragsize

5.169.4.12 stop_request `std::atomic<bool> io_dummy_t::stop_request [private]`

Stop request flag for main_loop.

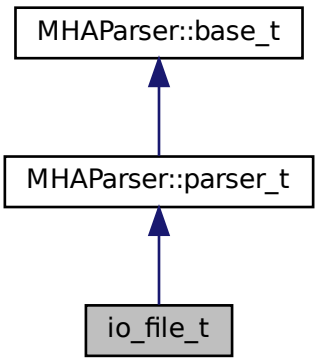
The documentation for this class was generated from the following file:

- **MHAIODummy.cpp**

5.170 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**, **IOStopped↩ Event_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

5.170.1 Detailed Description

File IO.

5.170.2 Constructor & Destructor Documentation

5.170.2.1 io_file_t()

```
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 )
```

5.170.2.2 ~io_file_t()

```
io_file_t::~io_file_t ( )
```

5.170.3 Member Function Documentation

5.170.3.1 prepare()

```
void io_file_t::prepare (
 int nch_in,
 int nch_out )
```

Allocate buffers, activate FILE client and install internal ports.

5.170.3.2 start()

```
void io_file_t::start ( )
```

5.170.3.3 stop()

```
void io_file_t::stop ( )
```

5.170.3.4 release()

```
void io_file_t::release (
 void  )
```

Remove FILE client and deallocate internal ports and buffers.

5.170.3.5 stopped() `void io_file_t::stopped (`

```
int proc_err,
int io_err ) [private]
```

5.170.3.6 setlock() `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.

5.170.4 Member Data Documentation

5.170.4.1 fragsize `int io_file_t::fragsize [private]`

5.170.4.2 samplerate `float io_file_t::samplerate [private]`

5.170.4.3 nchannels_in `int io_file_t::nchannels_in [private]`

5.170.4.4 nchannels_file_in `int io_file_t::nchannels_file_in [private]`

5.170.4.5 nchannels_out `int io_file_t::nchannels_out [private]`

5.170.4.6 proc_event `IOProcessEvent_t io_file_t::proc_event [private]`

5.170.4.7 proc_handle `void* io_file_t::proc_handle [private]`

5.170.4.8 start_event `IOStartedEvent_t io_file_t::start_event [private]`

5.170.4.9 start_handle `void* io_file_t::start_handle [private]`

5.170.4.10 stop_event `IOStoppedEvent_t io_file_t::stop_event [private]`

5.170.4.11 stop_handle `void* io_file_t::stop_handle [private]`

5.170.4.12 filename_input `MHAParser::string_t io_file_t::filename_input [private]`

5.170.4.13 filename_output `MHAParser::string_t io_file_t::filename_output [private]`

5.170.4.14 output_sample_format

`MHAParser::kw_t io_file_t::output_sample_format [private]`

5.170.4.15 startsample

`MHAParser::int_t io_file_t::startsample [private]`

5.170.4.16 length

`MHAParser::int_t io_file_t::length [private]`

5.170.4.17 strict_channel_match

`MHAParser::bool_t io_file_t::strict_channel_match [private]`

5.170.4.18 strict_srate_match

`MHAParser::bool_t io_file_t::strict_srate_match [private]`

5.170.4.19 s_in

`MHASignal::waveform_t* io_file_t::s_in [private]`

5.170.4.20 s_file_in

`MHASignal::waveform_t* io_file_t::s_file_in [private]`

5.170.4.21 s_out

`mha_wave_t* io_file_t::s_out [private]`

5.170.4.22 b_prepared

`bool io_file_t::b_prepared [private]`

5.170.4.23 sf_in SNDFILE* io_file_t::sf_in [private]

5.170.4.24 sf_out SNDFILE* io_file_t::sf_out [private]

5.170.4.25 sfinf_in SF_INFO io_file_t::sfinf_in [private]

5.170.4.26 sfinf_out SF_INFO io_file_t::sfinf_out [private]

5.170.4.27 total_read sf_count_t io_file_t::total_read [private]

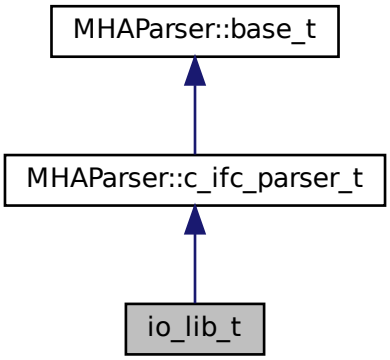
The documentation for this class was generated from the following file:

- **MHAIOFile.cpp**

5.171 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)
- std::string **get_documentation** () const
- std::vector< std::string > **get_categories** () const

Protected Member Functions

- void **test_error** ()

Protected Attributes

- int **lib_err**
- **pluginlib_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**
- std::string **plugin_documentation**
- std::vector< std::string > **plugin_categories**

Additional Inherited Members

5.171.1 Detailed Description

Class for loading MHA sound IO module.

5.171.2 Constructor & Destructor Documentation

5.171.2.1 io_lib_t() `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.

5.171.2.2 ~io_lib_t() `io_lib_t::~io_lib_t ( )`

Deinitialize and unload this MHA sound io module.

5.171.3 Member Function Documentation

5.171.3.1 prepare() `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

Parameters

outch	number of output channels

5.171.3.2 start() `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.

5.171.3.3 stop() `void io_lib_t::stop ( )`

5.171.3.4 release() `void io_lib_t::release ( )`

5.171.3.5 lib_str_error() `std::string io_lib_t::lib_str_error ( int err )`

5.171.3.6 get_documentation() `std::string io_lib_t::get_documentation ( ) const [inline]`

5.171.3.7 get_categories() `std::vector<std::string> io_lib_t::get_categories ( ) const [inline]`

5.171.3.8 test_error() `void io_lib_t::test_error ( ) [protected]`

5.171.4 Member Data Documentation

5.171.4.1 lib_err `int io_lib_t::lib_err [protected]`

5.171.4.2 lib_handle **`pluginlib_t`** `io_lib_t::lib_handle [protected]`

5.171.4.3 lib_data `void* io_lib_t::lib_data [protected]`

5.171.4.4 IOInit_cb **`IOInit_t`** `io_lib_t::IOInit_cb [protected]`

5.171.4.5 IOPrepare_cb **`IOPrepare_t`** `io_lib_t::IOPrepare_cb [protected]`

5.171.4.6 IOStart_cb **`IOStart_t`** `io_lib_t::IOStart_cb [protected]`

5.171.4.7 IOStop_cb **`IOStop_t`** `io_lib_t::IOStop_cb [protected]`

5.171.4.8 IORelease_cb **`IORelease_t`** `io_lib_t::IORelease_cb [protected]`

5.171.4.9 IOSetVar_cb `IOSetVar_t io_lib_t::IOSetVar_cb [protected]`

5.171.4.10 IOStrError_cb `IOStrError_t io_lib_t::IOStrError_cb [protected]`

5.171.4.11 IODestroy_cb `IODestroy_t io_lib_t::IODestroy_cb [protected]`

5.171.4.12 plugin_documentation `std::string io_lib_t::plugin_documentation [protected]`

5.171.4.13 plugin_categories `std::vector<std::string> io_lib_t::plugin_categories [protected]`

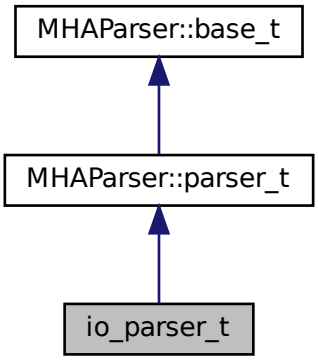
The documentation for this class was generated from the following files:

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.172 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

5.172.1 Detailed Description

Main class for Parser IO.

5.172.2 Constructor & Destructor Documentation

5.172.2.1 io_parser_t() 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 )
```

5.172.2.2 ∼io_parser_t() io_parser_t::∼io_parser_t ( )

5.172.3 Member Function Documentation

5.172.3.1 prepare() void io_parser_t::prepare (

```
int nch_in,
int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

5.172.3.2 start() void io_parser_t::start ( )

5.172.3.3 stop() void io_parser_t::stop ( )

5.172.3.4 release() void io_parser_t::release (void)

Remove JACK client and deallocate internal ports and buffers.

5.172.3.5 stopped() void io_parser_t::stopped (int *proc_err*, int *io_err*) [private]

5.172.3.6 started() void io_parser_t::started ( ) [private]

5.172.3.7 process_frame() void io_parser_t::process_frame ( ) [private]

5.172.4 Member Data Documentation

5.172.4.1 fragsize unsigned int io_parser_t::fragsize [private]

5.172.4.2 nchannels_in unsigned int io_parser_t::nchannels_in [private]

5.172.4.3 nchannels_out unsigned int io_parser_t::nchannels_out [private]

5.172.4.4 proc_event **IOProcessEvent_t** io_parser_t::proc_event [private]

5.172.4.5 proc_handle

`void* io_parser_t::proc_handle [private]`

5.172.4.6 start_event

`IOStartedEvent_t io_parser_t::start_event [private]`

5.172.4.7 start_handle

`void* io_parser_t::start_handle [private]`

5.172.4.8 stop_event

`IOStoppedEvent_t io_parser_t::stop_event [private]`

5.172.4.9 stop_handle

`void* io_parser_t::stop_handle [private]`

5.172.4.10 input

`MHAParser::mfloat_t io_parser_t::input [private]`

5.172.4.11 output

`MHAParser::mfloat_mon_t io_parser_t::output [private]`

5.172.4.12 s_in

`MHASignal::waveform_t* io_parser_t::s_in [private]`

5.172.4.13 s_out

`mha_wave_t* io_parser_t::s_out [private]`

5.172.4.14 b_fw_started `bool io_parser_t::b_fw_started [private]`

5.172.4.15 b_stopped `bool io_parser_t::b_stopped [private]`

5.172.4.16 b_prepared `bool io_parser_t::b_prepared [private]`

5.172.4.17 b_starting `bool io_parser_t::b_starting [private]`

5.172.4.18 patchbay `MHAEvents::patchbay_t< io_parser_t> io_parser_t::patchbay [private]`

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

- **MHAIOParser.cpp**

5.173 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**, **IOStarted↩ Event_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**

5.173.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

5.173.2 Constructor & Destructor Documentation

5.173.2.1 io_tcp_fwcb_t() `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.

5.173.2.2 ~io_tcp_fwcb_t() `virtual io_tcp_fwcb_t::~io_tcp_fwcb_t ( ) [inline], [virtual]`

Do-nothing destructor.

5.173.3 Member Function Documentation

5.173.3.1 start() `void io_tcp_fwcb_t::start ( ) [virtual]`

Call the framework's start callback.

5.173.3.2 process() `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.

5.173.3.3 set_errnos() `void io_tcp_fwcb_t::set_errnos (`

```
int proc_err,
int io_err ) [virtual]
```

Save error numbers to use during.

See also

stop (p. 733)

Parameters

proc_err	The error number from the

See also

process (p. 732) callback.

Parameters

io_err	The error number from the io library itself.

5.173.3.4 stop() `void io_tcp_fwcb_t::stop ( ) [virtual]`

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

set_errnos (p. 732).

5.173.4 Member Data Documentation

5.173.4.1 proc_event `IOProcessEvent_t io_tcp_fwcb_t::proc_event [private]`

Pointer to signal processing callback function.

5.173.4.2 start_event `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.

5.173.4.3 stop_event `IOStoppedEvent_t io_tcp_fwcb_t::stop_event [private]`

Pointer to stop notification callback function.

Called when the connection is closed.

5.173.4.4 proc_handle `void* io_tcp_fwcb_t::proc_handle [private]`

Handles belonging to framework.

5.173.4.5 start_handle `void * io_tcp_fwcb_t::start_handle [private]`

5.173.4.6 stop_handle `void * io_tcp_fwcb_t::stop_handle [private]`

5.173.4.7 proc_err `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. 733) from that callback. Within **stop** (p. 733), these error numbers are read again and transmitted to the framework.

5.173.4.8 io_err `int io_tcp_fwcb_t::io_err [private]`

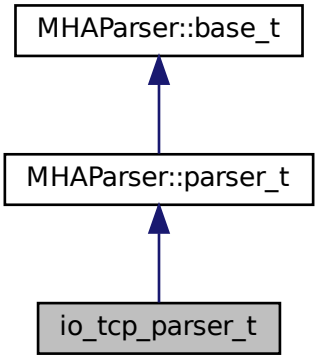
The documentation for this class was generated from the following file:

- **MHAIOTCP.cpp**

5.174 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

5.174.1 Detailed Description

The parser interface of the IOTCP library.

5.174.2 Constructor & Destructor Documentation

5.174.2.1 io_tcp_parser_t() `io_tcp_parser_t::io_tcp_parser_t ( )`

Constructor initializes parser variables.

5.174.2.2 ~io_tcp_parser_t() `virtual io_tcp_parser_t::~io_tcp_parser_t ( ) [inline], [virtual]`

Do-nothing destructor.

5.174.3 Member Function Documentation

5.174.3.1 get_local_address() 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.

5.174.3.2 get_local_port() 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.

5.174.3.3 set_local_port() 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. 737) currently returns 0.

5.174.3.4 get_server_port_open() `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. 738).

5.174.3.5 set_server_port_open() `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. 738) currently returns false.

Postcondition

See also

get_server_port_open (p. 738) returns the **value** (p. 48) of open.

5.174.3.6 get_connected() `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. 739).

5.174.3.7 set_connected() `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. 738).

Postcondition

See also

get_connected (p. 738) returns the **value** (p. 48) of open.

5.174.3.8 set_new_peer() 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. 738) currently returns false.

Postcondition

See also

get_connected (p. 738) returns true.

5.174.3.9 debug() virtual void io_tcp_parser_t::debug (const std::string & *message*) [inline], [virtual]

5.174.4 Member Data Documentation

5.174.4.1 local_address

MHAParser::string_t io_tcp_parser_t::local_address [private]

Lets the user set the local network interface to listen on.

5.174.4.2 local_port

MHAParser::int_t io_tcp_parser_t::local_port [private]

Lets the user choose the local tcp port to listen on.

5.174.4.3 server_port_open

MHAParser::int_mon_t io_tcp_parser_t::server_port_↩ open [private]

Indicates wether the TCP server socket is currently open.

5.174.4.4 connected

MHAParser::int_mon_t io_tcp_parser_t::connected [private]

Indicator if there currently is a sound data connection over TCP.

5.174.4.5 peer_address

MHAParser::string_mon_t io_tcp_parser_t::peer_address [private]

Display the ip address of the currently connected sound data client.

5.174.4.6 peer_port

MHAParser::int_mon_t io_tcp_parser_t::peer_port [private]

Display the tcp port used by the current sound data client.

5.174.4.7 debug_filename `MHAParser::string_t io_tcp_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

5.174.4.8 debug_file `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**

5.175 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. 985) 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.

5.175.1 Detailed Description

Sound data handling of io tcp library.

5.175.2 Constructor & Destructor Documentation

5.175.2.1 io_tcp_sound_t() `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. 744).

Parameters

fragsize	Number of sound samples in each channel expected and returned from processing callback.
samplerate	Number of samples per second in each channel.

5.175.2.2 ~io_tcp_sound_t() `virtual io_tcp_sound_t::~io_tcp_sound_t ( ) [inline], [virtual]`

Do-nothing destructor.

5.175.3 Member Function Documentation

5.175.3.1 check_sound_data_type() `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. 906)*	if mha_real_t is not compatible to 32-bit float.

5.175.3.2 prepare() `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.

Parameters

num_outchannels	Number of output audio channels.

5.175.3.3 release() `void io_tcp_sound_t::release ( void ) [virtual]`

Called during release.

Deletes sound data storage.

5.175.3.4 chunkbytes_in() `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.

5.175.3.5 header() `std::string io_tcp_sound_t::header ( ) const [virtual]`

Create the tcp sound header lines.

5.175.3.6 ntoh() `mha_wave_t * io_tcp_sound_t::ntoh ( const std::string & data ) [virtual]`

Copy data received from tcp into **mha_wave_t** (p. 985) structure.

Doing network-to-host byte order swapping in the process.

Parameters

data	One chunk (

See also

chunkbytes_in (p. 745)) of sound data to process.

Returns

Pointer to the sound data storage.

5.175.3.7 hton() `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.

5.175.4 Member Data Documentation

5.175.4.1 fragsize `int io_tcp_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

5.175.4.2 samplerate `float io_tcp_sound_t::samplerate [private]`

Sampling rate.

Number of samples per second in each channel.

5.175.4.3 num_inchannels `int io_tcp_sound_t::num_inchannels [private]`

Number of input channels.

Number of channels expected from and returned by signal processing callback.

5.175.4.4 num_outchannels `int io_tcp_sound_t::num_outchannels [private]`

5.175.4.5 s_in `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**

5.176 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]

5.176.1 Detailed Description

This union helps in conversion of floats from host byte order to network byte order and back again.

5.176.2 Member Data Documentation

5.176.2.1 f `float io_tcp_sound_t::float_union::f`

5.176.2.2 i `unsigned int io_tcp_sound_t::float_union::i`

5.176.2.3 c `char io_tcp_sound_t::float_union::c[4]`

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

- **MHAIOTCP.cpp**

5.177 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**

5.177.1 Detailed Description

The tcp sound io library.

5.177.2 Constructor & Destructor Documentation

5.177.2.1 io_tcp_t()

```
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 )
```

5.177.2.2 ~io_tcp_t()

```
virtual io_tcp_t::~io_tcp_t ( ) [inline], [virtual]
```

5.177.3 Member Function Documentation

5.177.3.1 prepare() 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

5.177.3.2 start() void io_tcp_t::start ( )

Call frameworks start callback if there is a sound data connection at the moment.

5.177.3.3 stop() void io_tcp_t::stop ( )

Close the current connection if there is one.

stop IO thread

5.177.3.4 release() void io_tcp_t::release (void)

Close the current connection and close the server socket.

Stop IO thread and close server socket.

5.177.3.5 accept_loop() void io_tcp_t::accept_loop ( ) [virtual]

IO thread executes this method.

5.177.3.6 connection_loop() 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.

5.177.3.7 parse()

```
virtual void io_tcp_t::parse (
    const char * cmd,
    char * retval,
    unsigned int len ) [inline], [virtual]
```

Parser interface.

5.177.4 Member Data Documentation

5.177.4.1 parser

`io_tcp_parser_t io_tcp_t::parser [private]`

5.177.4.2 sound

`io_tcp_sound_t io_tcp_t::sound [private]`

5.177.4.3 fwcb

`io_tcp_fwcb_t io_tcp_t::fwcb [private]`

5.177.4.4 server

`MHA_TCP::Server* io_tcp_t::server [private]`

5.177.4.5 thread

`MHA_TCP::Thread* io_tcp_t::thread [private]`

5.177.4.6 notify_start `MHA_TCP::Async_Notify io_tcp_t::notify_start [private]`

5.177.4.7 notify_stop `MHA_TCP::Async_Notify io_tcp_t::notify_stop [private]`

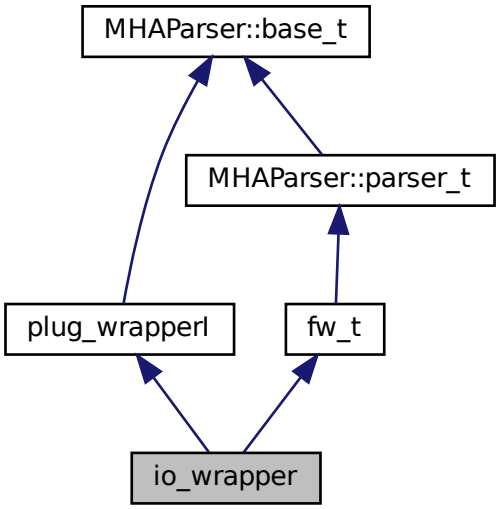
5.177.4.8 notify_release `MHA_TCP::Async_Notify io_tcp_t::notify_release [private]`

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

- **MHAIOTCP.cpp**

5.178 io_wrapper Class Reference

Inheritance diagram for io_wrapper:



Public Member Functions

- **io_wrapper** (const std::string &libname)
- virtual ~**io_wrapper** ()=default
- virtual std::vector< std::string > **get_categories** ()
- virtual std::string **parse** (const std::string &str)
- virtual bool **has_parser** ()
- virtual std::string **get_documentation** ()
- virtual bool **has_process** (**mha_domain_t** in, **mha_domain_t** out)

Additional Inherited Members

5.178.1 Constructor & Destructor Documentation

5.178.1.1 io_wrapper() `io_wrapper::io_wrapper ( const std::string & libname ) [inline]`

5.178.1.2 ~io_wrapper() `virtual io_wrapper::~io_wrapper ( ) [virtual], [default]`

5.178.2 Member Function Documentation

5.178.2.1 get_categories() `virtual std::vector<std::string> io_wrapper::get_categories ( ) [inline], [virtual]`

Implements **plug_wrapperl** (p. 1508).

5.178.2.2 parse() `virtual std::string io_wrapper::parse ( const std::string & str ) [inline], [virtual]`

Implements **plug_wrapperl** (p. 1508).

5.178.2.3 has_parser() virtual bool io_wrapper::has_parser ( ) [inline], [virtual]

Implements **plug_wrapperI** (p. 1508).

5.178.2.4 get_documentation() virtual std::string io_wrapper::get_documentation ( ) [inline], [virtual]

Implements **plug_wrapperI** (p. 1508).

5.178.2.5 has_process() virtual bool io_wrapper::has_process (**mha_domain_t** *in*, **mha_domain_t** *out*) [inline], [virtual]

Implements **plug_wrapperI** (p. 1508).

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

- **generatemhaplugindoc.cpp**

5.179 latex_doc_t Class Reference

Class to access the information stored in the plugin source code's MHAPLUGIN_↩ DOCUMENTATION 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** (**MHA_AC::algo_comm_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**
- **MHA_AC::algo_comm_class_t ac**
- std::unique_ptr< **plug_wrapperl** > **loader**
- const std::string **plugin_macro**

5.179.1 Detailed Description

Class to access the information stored in the plugin source code's MHAPLUGIN_↩ DOCUMENTATION macro.

5.179.2 Constructor & Destructor Documentation

5.179.2.1 latex_doc_t() `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", ...)

5.179.3 Member Function Documentation

5.179.3.1 get_latex_doc() `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

5.179.3.2 get_main_category() `std::string latex_doc_t::get_main_category ( ) const`

Returns

the first word of the categories string in the MHAPLUGIN_DOCUMENTATION macro

5.179.3.3 get_categories() `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

5.179.3.4 strdom() `std::string latex_doc_t::strdom ( mha_domain_t d ) const [private]`

5.179.3.5 get_ac()

```
std::string latex_doc_t::get_ac (
    MHA_AC::algo_comm_t & ac,
    std::string txt ) const  [private]
```

5.179.3.6 parsername()

```
std::string latex_doc_t::parsername (
    std::string s ) const  [private]
```

5.179.3.7 get_parser_var()

```
std::string latex_doc_t::get_parser_var (
    MHAParser::base_t * p,
    std::string name ) const  [private]
```

5.179.3.8 get_parser_tab()

```
std::string latex_doc_t::get_parser_tab (
    MHAParser::base_t * p,
    const std::string & prefix,
    const std::string & latex_macro ) const  [private]
```

5.179.4 Member Data Documentation

5.179.4.1 plugname

```
const std::string latex_doc_t::plugname  [private]
```

5.179.4.2 latex_plugname

```
const std::string latex_doc_t::latex_plugname  [private]
```

5.179.4.3 ac

```
MHA_AC::algo_comm_class_t latex_doc_t::ac  [private]
```

5.179.4.4 loader `std::unique_ptr< plug_wrapperI> latex_doc_t::loader [private]`

5.179.4.5 plugin_macro `const std::string latex_doc_t::plugin_macro [private]`

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

- **generatemhaplugindoc.cpp**

5.180 level_matching::channel_pair Class Reference

Public Member Functions

- **channel_pair** (const std::pair< int, int > &idx_, **mha_real_t** filter_rate_, **mha_real_t** mismatch_time_constant_)
  Channel pair ctor.
- **channel_pair** (int idx1_, int idx2_, **mha_real_t** filter_rate_, **mha_real_t** mismatch_↩ time_constant_)
  Channel pair ctor.
- **mha_real_t update_mismatch** (const **mha_wave_t** &signal_)
  Calculates the filtered rms level mismatch.
- **mha_real_t update_mismatch** (const **mha_spec_t** &signal_, unsigned fftlen_)
  Calculates the filtered rms level mismatch.
- **mha_real_t get_mismatch** () const
  Get last filter result.
- const std::pair< int, int > & **get_idx** () const

Private Attributes

- std::pair< int, int > **idx**
  Indices of channels.
- **MHAFilter::o1flt_lowpass_t mismatch**
  Low-pass filtered level mismatch.

5.180.1 Constructor & Destructor Documentation

5.180.1.1 channel_pair() [1/2]

```
level_matching::channel_pair::channel_pair (
    const std::pair< int, int > & idx_,
    mha_real_t filter_rate_,
    mha_real_t mismatch_time_constant_ )
```

Channel pair ctor.

Parameters

*idx_*	Pair of channel indices
*filter_rate_*	Filter rate of low pass filter
*mismatch_time_constant_*	Time constant of low pass filter

5.180.1.2 channel_pair() [2/2]

```
level_matching::channel_pair::channel_pair (
    int idx1_,
    int idx2_,
    mha_real_t filter_rate_,
    mha_real_t mismatch_time_constant_ )
```

Channel pair ctor.

Parameters

idx1	First channel index. Used as reference
idx2	Second channel index
*filter_rate_*	Filter rate of low pass filter
*mismatch_time_constant_*	Time constant of low pass filter

5.180.2 Member Function Documentation

5.180.2.1 update_mismatch() [1/2]

```
mha_real_t level_matching::channel_pair::update_mismatch (
    const mha_wave_t & signal_ )
```

Calculates the filtered rms level mismatch.

Parameters

signal	Input signal

Returns

Low-pass filtered mismatch ratio

5.180.2.2 update_mismatch() [2/2] mha_real_t level_matching::channel_pair::update_mismatch (

```
const mha_spec_t & signal_,
unsigned fftlen_ )
```

Calculates the filtered rms level mismatch.

Parameters

signal	Input signal

Returns

Low-pass filtered mismatch ratio

5.180.2.3 get_mismatch() mha_real_t level_matching::channel_pair::get_mismatch ( ) const

Get last filter result.

Returns

Last filter result

5.180.2.4 get_idx() const std::pair<int,int>& level_matching::channel_pair::get_idx ( ) const [inline]

5.180.3 Member Data Documentation

5.180.3.1 idx `std::pair<int,int> level_matching::channel_pair::idx [private]`

Indices of channels.

5.180.3.2 mismatch `MHAFilter::o1flt_lowpass_t level_matching::channel_pair::mismatch [private]`

Low-pass filtered level mismatch.

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

- **level_matching.hh**
- **level_matching.cpp**

5.181 level_matching::level_matching_config_t Class Reference

Public Member Functions

- **level_matching_config_t** (const **mhaconfig_t** &cfg_, const std::vector< std::vector< int >> &pairs_, **mha_real_t** signal_lp_fpass_, **mha_real_t** signal_fstop_, unsigned signal_lp_order_, **mha_real_t** level_tau_, **mha_real_t** range_)
  RT-config c'tor.
- ~**level_matching_config_t** ()=default
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::waveform_t tmp**
  Temporary storage for input signal to use waveform_t helper functions.
- std::vector< **channel_pair** > **pairings**
  Channel pairings.
- **MHAFilter::iir_filter_state_t lp**
  Nth-order low pass filter used to exclude high frequencies from level matching.
- **mha_real_t range**
  Maximum matching range. Maximum adjustment done.
- unsigned **fftlen**
  fftlen in spectral domain

5.181.1 Constructor & Destructor Documentation

5.181.1.1 level_matching_config_t() `level_matching::level_matching_config_t::level_matching_config_t (`

```
            const  mhaconfig_t & cfg_,
            const std::vector< std::vector< int >> & pairs_,
             mha_real_t signal_lp_fpass_,
             mha_real_t signal_lp_fstop_,
            unsigned signal_lp_order_,
             mha_real_t level_tau_,
             mha_real_t range_ )
```

RT-config c'tor.

Parameters

*cfg_*	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate.

Precondition

Size of param pairs must be two

Parameters

*pairs_*	Matrix containing the channel indices of the microphone pairs. Two entries per row.
signal_tau	Time constant of the signal low pass filter in seconds.
level_tau	Time constant of the level mismatch averaging filter in seconds.

5.181.1.2 ~level_matching_config_t() `level_matching::level_matching_config_t::~level_matching_config_t ( ) [default]`

5.181.2 Member Function Documentation

5.181.2.1 process() [1/2]

```
mha_wave_t * level_matching::level_matching_config_t↩
::process (
            mha_wave_t * s )
```

5.181.2.2 process() [2/2]

```
mha_spec_t * level_matching::level_matching_config_t↩
::process (
            mha_spec_t * s )
```

5.181.3 Member Data Documentation

5.181.3.1 tmp

```
MHASignal::waveform_t level_matching::level_matching_config_t::tmp
[private]
```

Temporary storage for input signal to use waveform_t helper functions.

5.181.3.2 pairings

```
std::vector< channel_pair> level_matching::level_matching_↩
config_t::pairings [private]
```

Channel pairings.

5.181.3.3 lp

```
MHAFilter::iir_filter_state_t level_matching::level_matching_config↩
_t::lp [private]
```

Nth-order low pass filter used to exclude high frequencies from level matching.

5.181.3.4 range

```
mha_real_t level_matching::level_matching_config_t::range [private]
```

Maximum matching range. Maximum adjustment done.

5.181.3.5 fftlen `unsigned level_matching::level_matching_config_t::fftlen [private]`

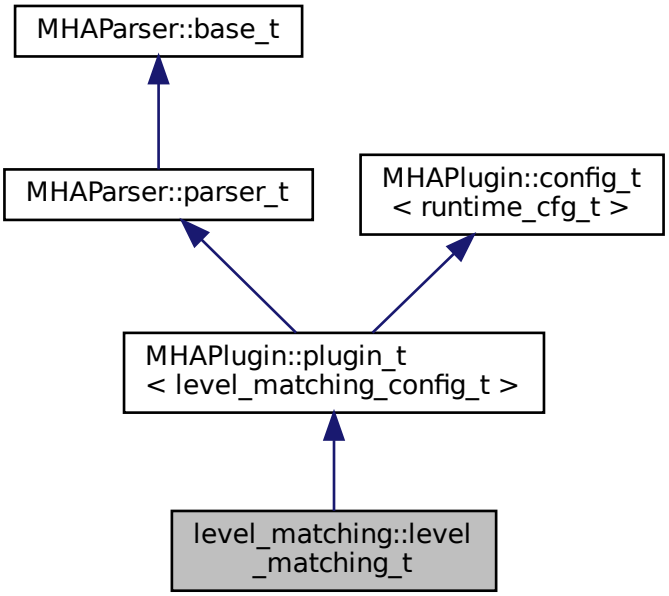
fftlen in spectral domain

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

- **level_matching.hh**
- **level_matching.cpp**

5.182 level_matching::level_matching_t Class Reference

Inheritance diagram for level_matching::level_matching_t:



Public Member Functions

- **level_matching_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Plugin interface ctor.
- ∼**level_matching_t** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Processing callback.
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
  Processing callback.
- void **prepare** (**mhaconfig_t** &)
  Plugin preparation callback.
- void **release** (void)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::mint_t channels** ={"channels","[[0 1]]"}
- **MHAParser::float_t range** ={"Maximum matching range in dB","4","[0,["}
- **MHAParser::float_t lp_signal_fpass** ={"Upper edge of the lp pass band for the signal in Hz","4000","[0,["}
- **MHAParser::float_t lp_signal_fstop** ={"Stop band lower edge frequency for the signal in Hz","8000","[0,["}
- **MHAParser::float_t lp_signal_order** ={"Signal lp order","24","[1,["}
- **MHAParser::float_t lp_level_tau** ={"Low pass time constant for the mismatch in s","1","[0,["}
- **MHAEvents::patchbay_t**< **level_matching_t** > **patchbay**

Additional Inherited Members

5.182.1 Constructor & Destructor Documentation

5.182.1.1 level_matching_t() `level_matching::level_matching_t::level_matching_t (`

MHA_AC::algo_comm_t & *iac*,
const std::string & *configured_name*)

Plugin interface ctor.

Parameters

ac	

5.182.1.2 ~level_matching_t() `level_matching::level_matching_t::~level_matching_t ( )`

5.182.2 Member Function Documentation

5.182.2.1 process() [1/2] mha_wave_t * level_matching::level_matching_t::process (mha_wave_t * signal_)

Processing callback.

Parameters

signal	Input signal

5.182.2.2 process() [2/2] mha_spec_t * level_matching::level_matching_t::process (mha_spec_t * signal_)

Processing callback.

Parameters

signal	Input signal

5.182.2.3 prepare() void level_matching::level_matching_t::prepare (mhaconfig_t &) [virtual]

Plugin preparation callback.

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< level_matching_config_t >** (p. 1301).

5.182.2.4 release() void level_matching::level_matching_t::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< level_matching_config_t >** (p. 1302).

5.182.2.5 update_cfg() void level_matching::level_matching_t::update_cfg (void) [private]

5.182.3 Member Data Documentation

5.182.3.1 channels MHAParser::mint_t level_matching::level_matching_t::channels ={"channels","[[0 1]]"} [private]

5.182.3.2 range MHAParser::float_t level_matching::level_matching_t::range ={"Maximum matching range in dB","4","[0,["} [private]

5.182.3.3 lp_signal_fpass MHAParser::float_t level_matching::level_matching_t↩ ::lp_signal_fpass ={"Upper edge of the lp pass band for the signal in Hz","4000","[0,["} [private]

5.182.3.4 lp_signal_fstop MHAParser::float_t level_matching::level_matching_t↩ ::lp_signal_fstop ={"Stop band lower edge frequency for the signal in Hz","8000","[0,["} [private]

5.182.3.5 lp_signal_order MHAParser::float_t level_matching::level_matching_t↩ ::lp_signal_order ={"Signal lp order","24","[1,["} [private]

5.182.3.6 lp_level_tau MHAParser::float_t level_matching::level_matching_t::lp_↩ level_tau ={"Low pass time constant for the mismatch in s","1","[0,["} [private]

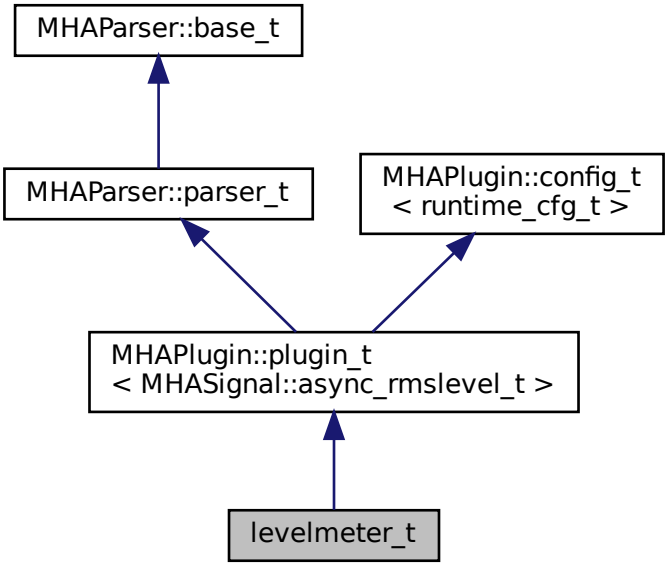
5.182.3.7 patchbay `MHAEvents::patchbay_t< level_matching_t> level_matching::level_matching_t::patchbay [private]`

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

- **level_matching.hh**
- **level_matching.cpp**

5.183 levelmeter_t Class Reference

Inheritance diagram for levelmeter_t:



Public Member Functions

- **levelmeter_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_tau** ()
- void **query_rms** ()
- void **query_peak** ()

Private Attributes

- **MHAParser::float_t tau**
- **MHAParser::kw_t mode**
- **MHAParser::vfloat_mon_t rms**
- **MHAParser::vfloat_mon_t peak**
- **MHAEvents::patchbay_t< levelmeter_t > patchbay**

Additional Inherited Members

5.183.1 Constructor & Destructor Documentation

5.183.1.1 levelmeter_t() `levelmeter_t::levelmeter_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.183.2 Member Function Documentation

5.183.2.1 process() `mha_wave_t * levelmeter_t::process (`

```
mha_wave_t * s )
```

5.183.2.2 prepare() `void levelmeter_t::prepare (`

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< MHASignal::async_rmslevel_t >** (p. 1301).

5.183.2.3 update_tau() `void levelmeter_t::update_tau ( ) [private]`

5.183.2.4 query_rms() void levelmeter_t::query_rms ( ) [private]

5.183.2.5 query_peak() void levelmeter_t::query_peak ( ) [private]

5.183.3 Member Data Documentation

5.183.3.1 tau MHAParser::float_t levelmeter_t::tau [private]

5.183.3.2 mode MHAParser::kw_t levelmeter_t::mode [private]

5.183.3.3 rms MHAParser::vfloat_mon_t levelmeter_t::rms [private]

5.183.3.4 peak MHAParser::vfloat_mon_t levelmeter_t::peak [private]

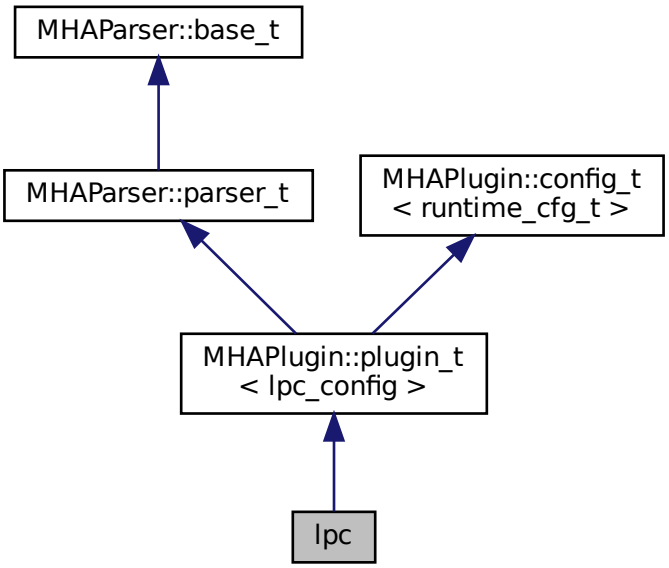
5.183.3.5 patchbay MHAEvents::patchbay_t< levelmeter_t> levelmeter_t::patchbay [private]

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

- **levelmeter.cpp**

5.184 lpc Class Reference

Inheritance diagram for lpc:



Public Member Functions

- **lpc** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.184.1 Constructor & Destructor Documentation

5.184.1.1 lpc() `lpc::lpc (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.184.1.2 ~lpc() `lpc::~lpc ( )`

5.184.2 Member Function Documentation

5.184.2.1 process() `mha_wave_t * lpc::process (`

```
  mha_wave_t * signal )
```

Checks for the most recent configuration and defers processing to it.

5.184.2.2 prepare() 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. 1301).

5.184.2.3 release() void lpc::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< lpc_config >** (p. 1302).

5.184.2.4 update_cfg() void lpc::update_cfg (void) [private]

5.184.3 Member Data Documentation

5.184.3.1 algo_name std::string lpc::algo_name [private]

5.184.3.2 lpc_order MHAParser::int_t lpc::lpc_order [private]

5.184.3.3 lpc_buffer_size `MHAParser::int_t lpc::lpc_buffer_size [private]`

5.184.3.4 shift `MHAParser::bool_t lpc::shift [private]`

5.184.3.5 comp_each_iter `MHAParser::int_t lpc::comp_each_iter [private]`

5.184.3.6 norm `MHAParser::bool_t lpc::norm [private]`

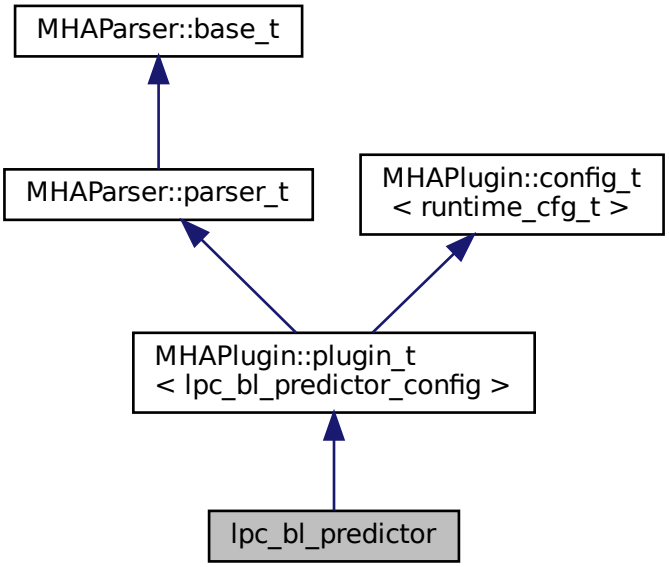
5.184.3.7 patchbay `MHAEvents::patchbay_t< lpc> lpc::patchbay [private]`

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

- **lpc.h**
- **lpc.cpp**

5.185 lpc_bl_predictor Class Reference

Inheritance diagram for lpc_bl_predictor:



Public Member Functions

- **lpc_bl_predictor** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.185.1 Constructor & Destructor Documentation

5.185.1.1 lpc_bl_predictor() `lpc_bl_predictor::lpc_bl_predictor (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.185.1.2 ~lpc_bl_predictor() `lpc_bl_predictor::~lpc_bl_predictor ( )`

5.185.2 Member Function Documentation

5.185.2.1 process() `mha_wave_t * lpc_bl_predictor::process ( mha_wave_t * signal )`

Checks for the most recent configuration and defers processing to it.

5.185.2.2 prepare() `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. 1301).

5.185.2.3 release() `void lpc_bl_predictor::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< lpc_bl_predictor_config >** (p. 1302).

5.185.2.4 update_cfg() `void lpc_bl_predictor::update_cfg ( void ) [private]`

5.185.3 Member Data Documentation

5.185.3.1 lpc_order `MHAParser::int_t lpc_bl_predictor::lpc_order`

5.185.3.2 name_kappa `MHAParser::string_t lpc_bl_predictor::name_kappa`

5.185.3.3 name_lpc_f `MHAParser::string_t lpc_bl_predictor::name_lpc_f`

5.185.3.4 name_lpc_b `MHAParser::string_t lpc_bl_predictor::name_lpc_b`

5.185.3.5 name_f `MHAParser::string_t lpc_bl_predictor::name_f`

5.185.3.6 name_b `MHAParser::string_t lpc_bl_predictor::name_b`

5.185.3.7 patchbay `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**

5.186 lpc_bl_predictor_config Class Reference

Public Member Functions

- **lpc_bl_predictor_config** (**MHA_AC::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

- **MHA_AC::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**

5.186.1 Constructor & Destructor Documentation

5.186.1.1 lpc_bl_predictor_config() lpc_bl_predictor_config::lpc_bl_predictor_config (

```
MHA_AC::algo_comm_t & iac,
const mhaconfig_t in_cfg,
lpc_bl_predictor * _lpc )
```

5.186.1.2 ~lpc_bl_predictor_config() lpc_bl_predictor_config::~lpc_bl_predictor_config ( )

5.186.2 Member Function Documentation

5.186.2.1 process()

```
mha_wave_t * lpc_bl_predictor_config::process (
        mha_wave_t * wave )
```

5.186.3 Member Data Documentation

5.186.3.1 ac

```
MHA_AC::algo_comm_t& lpc_bl_predictor_config::ac [private]
```

5.186.3.2 f_est

```
MHA_AC::waveform_t lpc_bl_predictor_config::f_est [private]
```

5.186.3.3 b_est

```
MHA_AC::waveform_t lpc_bl_predictor_config::b_est [private]
```

5.186.3.4 forward

```
MHASignal::waveform_t lpc_bl_predictor_config::forward [private]
```

5.186.3.5 backward

```
MHASignal::waveform_t lpc_bl_predictor_config::backward [private]
```

5.186.3.6 lpc_order

```
int lpc_bl_predictor_config::lpc_order [private]
```

5.186.3.7 name_km

```
std::string lpc_bl_predictor_config::name_km [private]
```

5.186.3.8 name_f `std::string lpc_bl_predictor_config::name_f [private]`

5.186.3.9 name_b `std::string lpc_bl_predictor_config::name_b [private]`

5.186.3.10 km `mha_wave_t lpc_bl_predictor_config::km [private]`

5.186.3.11 s_f `mha_wave_t lpc_bl_predictor_config::s_f [private]`

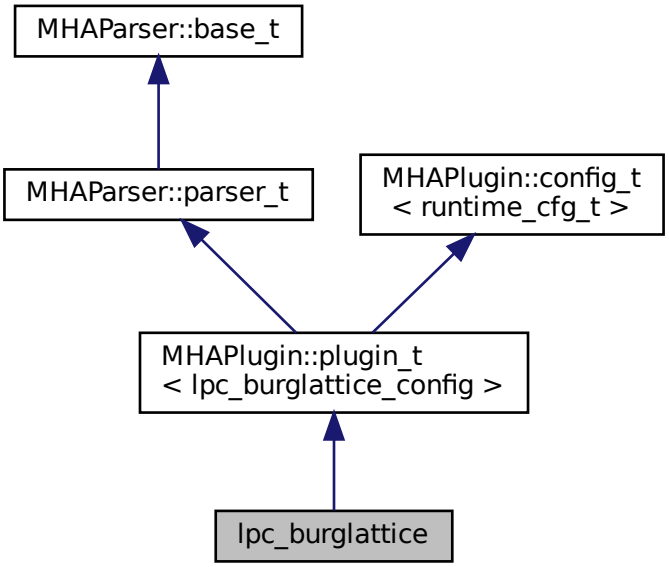
5.186.3.12 s_b `mha_wave_t lpc_bl_predictor_config::s_b [private]`

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

- **lpc_bl_predictor.h**
- **lpc_bl_predictor.cpp**

5.187 lpc_burglattice Class Reference

Inheritance diagram for lpc_burglattice:



Public Member Functions

- **lpc_burglattice** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.187.1 Constructor & Destructor Documentation

5.187.1.1 lpc_burglattice() `lpc_burglattice::lpc_burglattice (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructs our plugin.

5.187.1.2 ~lpc_burglattice() lpc_burglattice::~lpc_burglattice ( )

5.187.2 Member Function Documentation

5.187.2.1 process() mha_wave_t * lpc_burglattice::process (mha_wave_t * signal)

Checks for the most recent configuration and defers processing to it.

5.187.2.2 prepare() 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. 1301).

5.187.2.3 release() void lpc_burglattice::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< lpc_burglattice_config >** (p. 1302).

5.187.2.4 update_cfg() void lpc_burglattice::update_cfg (void) [private]

5.187.3 Member Data Documentation

5.187.3.1 lpc_order **MHAParser::int_t** lpc_burglattice::lpc_order

5.187.3.2 name_kappa **MHAParser::string_t** lpc_burglattice::name_kappa

5.187.3.3 name_f **MHAParser::string_t** lpc_burglattice::name_f

5.187.3.4 name_b **MHAParser::string_t** lpc_burglattice::name_b

5.187.3.5 lambda **MHAParser::float_t** lpc_burglattice::lambda

5.187.3.6 patchbay **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**

5.188 lpc_burglattice_config Class Reference

Public Member Functions

- **lpc_burglattice_config** (**MHA_AC::algo_comm_t** &iac, const **mhaconfig_t** in_cfg, **lpc_burglattice** ∗_lpc)
- ∼**lpc_burglattice_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Attributes

- **MHA_AC::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**

5.188.1 Constructor & Destructor Documentation

5.188.1.1 lpc_burglattice_config()

```
lpc_burglattice_config::lpc_burglattice_config (
    MHA_AC::algo_comm_t & iac,
    const mhaconfig_t in_cfg,
    lpc_burglattice * _lpc )
```

5.188.1.2 ~lpc_burglattice_config()

```
lpc_burglattice_config::~lpc_burglattice_config
( )
```

5.188.2 Member Function Documentation

5.188.2.1 process()

```
mha_wave_t * lpc_burglattice_config::process (
    mha_wave_t * wave )
```

5.188.3 Member Data Documentation

5.188.3.1 ac

`MHA_AC::algo_comm_t& lpc_burglattice_config::ac [private]`

5.188.3.2 forward

`MHASignal::waveform_t lpc_burglattice_config::forward [private]`

5.188.3.3 backward

`MHASignal::waveform_t lpc_burglattice_config::backward [private]`

5.188.3.4 kappa

`MHASignal::waveform_t lpc_burglattice_config::kappa [private]`

5.188.3.5 kappa_block

`MHA_AC::waveform_t lpc_burglattice_config::kappa_block [private]`

5.188.3.6 dm

`MHASignal::waveform_t lpc_burglattice_config::dm [private]`

5.188.3.7 nm

`MHASignal::waveform_t lpc_burglattice_config::nm [private]`

5.188.3.8 lambda

`mha_real_t lpc_burglattice_config::lambda [private]`

5.188.3.9 lpc_order

`int lpc_burglattice_config::lpc_order [private]`

5.188.3.10 name_f `std::string lpc_burglattice_config::name_f [private]`

5.188.3.11 name_b `std::string lpc_burglattice_config::name_b [private]`

5.188.3.12 s_f `mha_wave_t lpc_burglattice_config::s_f [private]`

5.188.3.13 s_b `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**

5.189 lpc_config Class Reference

Public Member Functions

- **lpc_config** (**MHA_AC::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**

5.189.1 Constructor & Destructor Documentation

5.189.1.1 lpc_config() lpc_config::lpc_config (
MHA_AC::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*)

5.189.1.2 ~lpc_config() lpc_config::~lpc_config ( )

5.189.2 Member Function Documentation

5.189.2.1 process() **mha_wave_t** * lpc_config::process (
mha_wave_t * *wave*)

5.189.2.2 insert() void lpc_config::insert ( )

5.189.3 Member Data Documentation

5.189.3.1 norm bool lpc_config::norm [private]

5.189.3.2 shift bool lpc_config::shift [private]

5.189.3.3 comp_each_iter unsigned int lpc_config::comp_each_iter [private]

5.189.3.4 order unsigned int lpc_config::order [private]

5.189.3.5 lpc_buffer_size unsigned int lpc_config::lpc_buffer_size [private]

5.189.3.6 N unsigned int lpc_config::N [private]

5.189.3.7 comp_iter unsigned int lpc_config::comp_iter [private]

5.189.3.8 sample **mha_wave_t** lpc_config::sample [private]

5.189.3.9 R std::vector< **mha_real_t**> lpc_config::R [private]

5.189.3.10 A std::vector< **mha_real_t**> lpc_config::A [private]

5.189.3.11 inwave `MHASignal::ringbuffer_t lpc_config::inwave [private]`

5.189.3.12 lpc_out `MHA_AC::waveform_t lpc_config::lpc_out [private]`

5.189.3.13 corr_out `MHA_AC::waveform_t lpc_config::corr_out [private]`

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

- **lpc.h**
- **lpc.cpp**

5.190 lsl2ac::cfg_t Class Reference

Runtime configuration class of the **lsl2ac** (p. 100) plugin.

Public Member Functions

- **cfg_t** (**MHA_AC::algo_comm_t** &ac_, **overrun_behavior** overrun_, int bufsize_↩ , int chunksize_, const std::vector< std::string > &streamnames_, int nchannels_, int nsamples_)

  *C'tor of **lsl2ac** (p. 100) run time configuration.*
- void **process** ()

Private Attributes

- std::map< std::string, std::unique_ptr< **save_var_base_t** > > **varlist**

  Maps variable name to unique ptr's of lsl to ac bridges.

5.190.1 Detailed Description

Runtime configuration class of the **lsl2ac** (p. 100) plugin.

5.190.2 Constructor & Destructor Documentation

5.190.2.1 cfg_t() cfg_t::cfg_t (

```
MHA_AC::algo_comm_t & ac_,
overrun_behavior overrun_,
int bufsize_,
int chunksize_,
const std::vector< std::string > & streamnames_,
int nchannels_,
int nsamples_ )
```

C'tor of **lsl2ac** (p. 100) run time configuration.

Parameters

*ac_*	AC space, data from LSL will be inserted as AC variables
*overrun_*	Overrun behavior
*bufsize_*	Buffer size of the LSL stream inlet
*chunksize_*	Chunk size of the LSL stream
*streamnames_*	Names of LSL streams to be subscribed to
*nchannels_*	Number of channels to expect in the the LSL streams. Zero means accept any.
*nsamples_*	Number of samples per channel in the AC variable. Zero means resize as needed.

5.190.3 Member Function Documentation

5.190.3.1 process() void cfg_t::process (

```
void )
```

5.190.4 Member Data Documentation

5.190.4.1 varlist `std::map<std::string, std::unique_ptr< save_var_base_t> > lsl2ac↩ ::cfg_t::varlist [private]`

Maps variable name to unique ptr's of lsl to ac bridges.

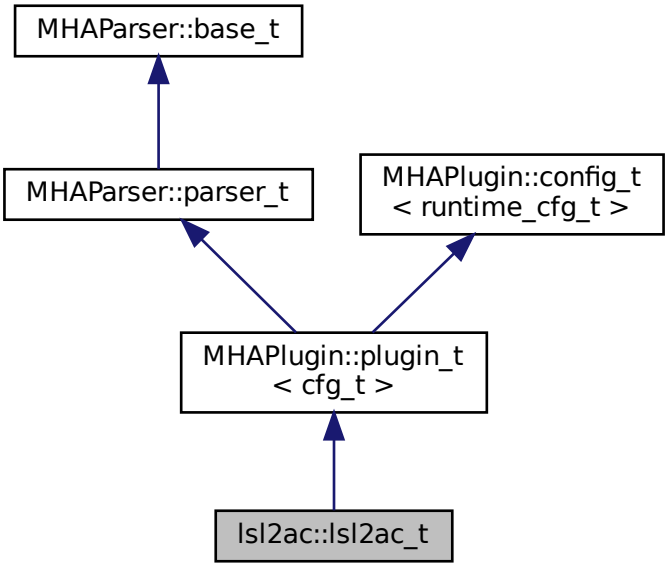
The documentation for this class was generated from the following files:

- **lsl2ac.hh**
- **lsl2ac.cpp**

5.191 lsl2ac::lsl2ac_t Class Reference

Plugin class of **lsl2ac** (p. 100).

Inheritance diagram for lsl2ac::lsl2ac_t:



Public Member Functions

- **lsl2ac_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)

  *Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 794).*
- **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

- void **get_all_stream_names** ()
  Retrieves all stream names from LSL and fills them into available_streams.
- void **update** ()
  Construct new runtime configuration.
- void **setlock** (bool lock_)
  Convencience function lock/unlock all config variables.

Private Attributes

- **MHAParser::vstring_t streams** ={"List of LSL streams to be saved, empty for all.","[ ]"}
  Config variable for list of streams to be saved.
- **MHAParser::bool_t activate** ={"Receive from network?","yes"}
  Config variable for activation/deactivation of plugin.
- **MHAParser::kw_t overrun_behavior** ={"How to handle overrun","discard","[discard ignore]"}
  Config variable for overrun behavior.
- **MHAParser::int_t buffersize**
  Config variable for maximum buffer size of LSL.
- **MHAParser::int_t chunksize**
  Config variable for maximum chunk size of LSL.
- **MHAParser::int_t nchannels**
  Config variable for number of channels to expect from LSL stream.
- **MHAParser::int_t nsamples**
  Config variable for maximum chunk size of LSL.
- **MHAEvents::patchbay_t**< **lsl2ac_t** > **patchbay**
  Patchbay for configuration callbacks.
- **MHAParser::vstring_mon_t available_streams** ={"List of all available LSL **streams**"}
  Monitor variable containing all available streams.

Additional Inherited Members

5.191.1 Detailed Description

Plugin class of **lsl2ac** (p. 100).

5.191.2 Constructor & Destructor Documentation

5.191.2.1 lsl2ac_t() `lsl2ac::lsl2ac_t::lsl2ac_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.191.3 Member Function Documentation

5.191.3.1 prepare() `void lsl2ac::lsl2ac_t::prepare (`

```
mhaconfig_t & ) [virtual]
```

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 794).

Implements **MHAPlugin::plugin_t**< **cfg_t** > (p. 1301).

5.191.3.2 process() [1/3] `mha_wave_t* lsl2ac::lsl2ac_t::process (`

```
mha_wave_t * s ) [inline]
```

Processing fct for waveforms.

Calls **process(void)** (p. 793).

5.191.3.3 process() [2/3] `mha_spec_t* lsl2ac::lsl2ac_t::process (`

```
mha_spec_t * s ) [inline]
```

Processing fct for spectra.

Calls **process(void)** (p. 793).

5.191.3.4 process() [3/3] `void lsl2ac::lsl2ac_t::process (`

```
void )
```

Process function.

5.191.3.5 release() void lsl2ac::lsl2ac_t::release (void) [virtual]

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1302).

5.191.3.6 get_all_stream_names() void lsl2ac::lsl2ac_t::get_all_stream_names ( ) [private]

Retrieves all stream names from LSL and fills them into available_streams.

5.191.3.7 update() void lsl2ac::lsl2ac_t::update ( ) [private]

Construct new runtime configuration.

5.191.3.8 setlock() void lsl2ac::lsl2ac_t::setlock (bool *lock_*) [private]

Convencience function lock/unlock all config variables.

Parameters

*lock_*	True to lock, False for unlock

5.191.4 Member Data Documentation

5.191.4.1 streams

`MHAParser::vstring_t lsl2ac::lsl2ac_t::streams ={"List of LSL streams to be saved, empty for all.","[]"} [private]`

Config variable for list of streams to be saved.

5.191.4.2 activate

`MHAParser::bool_t lsl2ac::lsl2ac_t::activate ={"Receive from network?","yes"} [private]`

Config variable for activation/deactivation of plugin.

5.191.4.3 overrun_behavior

`MHAParser::kw_t lsl2ac::lsl2ac_t::overrun_behavior ={"How to handle overrun","discard","[discard ignore]"} [private]`

Config variable for overrun behavior.

5.191.4.4 buffersize

`MHAParser::int_t lsl2ac::lsl2ac_t::buffersize [private]`

Config variable for maximum buffer size of LSL.

5.191.4.5 chunksize

`MHAParser::int_t lsl2ac::lsl2ac_t::chunksize [private]`

Config variable for maximum chunk size of LSL.

5.191.4.6 nchannels

`MHAParser::int_t lsl2ac::lsl2ac_t::nchannels [private]`

Config variable for number of channels to expect from LSL stream.

5.191.4.7 nsamples `MHAParser::int_t lsl2ac::lsl2ac_t::nsamples [private]`

Config variable for maximum chunk size of LSL.

5.191.4.8 patchbay `MHAEvents::patchbay_t< lsl2ac_t> lsl2ac::lsl2ac_t::patchbay [private]`

Patchbay for configuration callbacks.

5.191.4.9 available_streams `MHAParser::vstring_mon_t lsl2ac::lsl2ac_t::available_streams ={"List of all available LSL streams"} [private]`

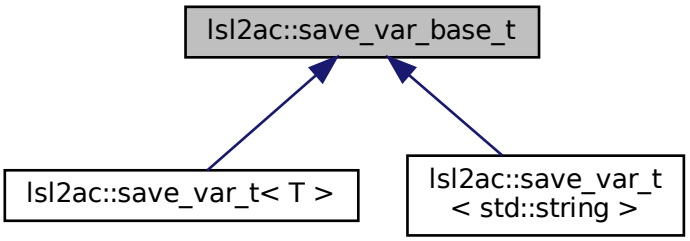
Monitor variable containing all available streams.

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

- **lsl2ac.hh**
- **lsl2ac.cpp**

5.192 lsl2ac::save_var_base_t Class Reference

Inheritance diagram for lsl2ac::save_var_base_t:



Public Member Functions

- virtual ~**save_var_base_t** ()=default
- virtual lsl::stream_info **info** ()=0
  Get stream info object from stream inlet.
- virtual void **receive_frame** ()=0
  Receive a samples from lsl and copy to AC space.

5.192.1 Constructor & Destructor Documentation

5.192.1.1 ~**save_var_base_t()** `virtual lsl2ac::save_var_base_t::~save_var_base_t ( ) [virtual], [default]`

5.192.2 Member Function Documentation

5.192.2.1 **info()** `virtual lsl::stream_info lsl2ac::save_var_base_t::info ( ) [pure virtual]`

Get stream info object from stream inlet.

Implemented in **lsl2ac::save_var_t**< **std::string** > (p. 809), and **lsl2ac::save_var_t**< **T** > (p. 801).

5.192.2.2 **receive_frame()** `virtual void lsl2ac::save_var_base_t::receive_frame ( ) [pure virtual]`

Receive a samples from lsl and copy to AC space.

Handling of underrun is configuration-dependent

Implemented in **lsl2ac::save_var_t**< **std::string** > (p. 809), and **lsl2ac::save_var_t**< **T** > (p. 802).

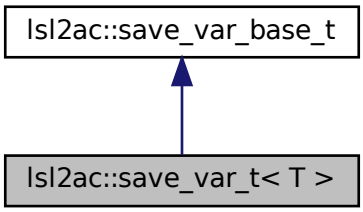
The documentation for this class was generated from the following file:

- **lsl2ac.hh**

5.193 lsl2ac::save_var_t< T > Class Template Reference

LSL to AC bridge variable.

Inheritance diagram for lsl2ac::save_var_t< T >:



Public Member Functions

- **save_var_t** (const **save_var_t** &)=delete
- **save_var_t** (**save_var_t** &&)=delete
- virtual ~**save_var_t** ()=default
- **save_var_t** & **operator=** (const **save_var_t** &)=delete
- **save_var_t** & **operator=** (**save_var_t** &&)=delete
- **save_var_t** (const lsl::stream_info &info_, **MHA_AC::algo_comm_t** &ac_, **overrun_behavior** ob_, int type_, int buflen_, int chunksize_, int nchannels_, int nsamples_)
  C'tor of lsl to ac bridge.
- lsl::stream_info **info** () override
  Get stream info object from stream inlet.
- void **receive_frame** () override
  Receive a samples from lsl and copy to AC space.

Private Member Functions

- void **pull_samples_ignore** ()
  Pull new samples, ignore overrun.
- void **pull_samples_discard** ()
  Pull new samples as long as there are samples ready for pickup in the LSL buffers.
- void **get_time_correction** ()
  Refresh time correction value every 5s.
- void **insert_vars** ()
  Insert stream value, time stamp and time offset into ac space.

Private Attributes

- lsl::stream_inlet **stream**
  LSL stream outlet.
- std::vector< T > **buf**
  Data buffer of the ac variable.
- std::vector< double > **ts_buf**
  Timestamp buffer.
- std::vector< double > **tc_buf**
  Clock correction buffer.
- **MHA_AC::algo_comm_t** & **ac**
  Handle to AC space.
- **MHA_AC::comm_var_t cv**
  Timeseries AC variable.
- **MHA_AC::comm_var_t ts**
  Timestamp AC variable.
- **MHA_AC::comm_var_t tc**
  Time correction AC variable.
- std::string **ts_name**
  Timestamp AC variable name.
- std::string **tc_name**
  Time correction AC variable name.
- std::string **new_name**
  Number of new samples AC variable name.
- std::chrono::time_point< std::chrono::steady_clock > **tic**
  time point of last time correction pull
- bool **skip** =false
  Should the variable be skipped in future process calls? Only true when error occured.
- **overrun_behavior ob**
  Behavior on stream overrun.
- const std::string **name**
  Name of stream.
- int32_t **nchannels**
  Number of channels.
- std::size_t **nsamples**
  Number of samples per channel in the AC variable.
- std::size_t **chunksize**
  Maximal chunk size of lsl stream.
- std::size_t **bufsize**
  Total buffer size of the ac variable buffer.
- int **n_new_samples**
  Number of most recently pulled samples per channel.

5.193.1 Detailed Description

template<typename T>
class lsl2ac::save_var_t< T >

LSL to AC bridge variable.

5.193.2 Constructor & Destructor Documentation

5.193.2.1 save_var_t() [1/3] template<typename T >

```
lsl2ac::save_var_t< T >::  save_var_t (
            const  save_var_t< T > &  ) [delete]
```

5.193.2.2 save_var_t() [2/3] template<typename T >

```
lsl2ac::save_var_t< T >::  save_var_t (
             save_var_t< T > &&  ) [delete]
```

5.193.2.3 ~save_var_t() template<typename T >

```
virtual  lsl2ac::save_var_t< T >::~ save_var_t ( ) [virtual], [default]
```

5.193.2.4 save_var_t() [3/3] template<typename T >

```
lsl2ac::save_var_t< T >::  save_var_t (
            const lsl::stream_info & info_,
             MHA_AC::algo_comm_t & ac_,
             overrun_behavior ob_,
            int type_,
            int buflen_,
            int chunksize_,
            int nchannels_,
            int nsamples_ )  [inline]
```

C'tor of lsl to ac bridge.

Parameters

*name_*	Name of LSL stream to be received
*info_*	LSL stream info object containing metadata
*ac_*	Handle to ac space
*ob_*	Overrun behavior. 0=Discard oldest, 1=Ignore
*type_*	Type tag of the AC variable
*buflen_*	LSL buffer size
*chunksize_*	LSL chunk size
*nchannels_*	Number of channels in the AC variable must be zero or equal to number of channels in LSL stream
*nsamples_*	Number of samples per channel in the AC variable. Zero means resize as needed

5.193.3 Member Function Documentation

5.193.3.1 operator=() [1/2] template<typename T >

```
save_var_t&  lsl2ac::save_var_t< T >::operator= (
            const  save_var_t< T > &  )  [delete]
```

5.193.3.2 operator=() [2/2] template<typename T >

```
save_var_t&  lsl2ac::save_var_t< T >::operator= (
             save_var_t< T > &&  )  [delete]
```

5.193.3.3 info() template<typename T >

lsl::stream_info **lsl2ac::save_var_t**< T >::info ( ) [inline], [override], [virtual]

Get stream info object from stream inlet.

Implements **lsl2ac::save_var_base_t** (p. 797).

5.193.3.4 receive_frame() template<typename T >

void **lsl2ac::save_var_t**< T >::receive_frame ( ) [inline], [override], [virtual]

Receive a samples from lsl and copy to AC space.

Handling of underrun is configuration-dependent

Implements **lsl2ac::save_var_base_t** (p. 797).

5.193.3.5 pull_samples_ignore() template<typename T >

void **lsl2ac::save_var_t**< T >::pull_samples_ignore ( ) [inline], [private]

Pull new samples, ignore overrun.

If nsamples=0, leaves the buffers in a state where the newest samples are at the beginning of the buffers, the state of the older samples is undefined and n_new_samples contains the number of new samples per channel. If nsamples is non-zero, the buffers are rotated so the oldest samples are the first in the buffer and n_new_samples contains the number of new samples per channel.

5.193.3.6 pull_samples_discard() template<typename T >

void **lsl2ac::save_var_t**< T >::pull_samples_discard ( ) [inline], [private]

Pull new samples as long as there are samples ready for pickup in the LSL buffers.

If nsamples=0, leaves the buffers in a state where the newest samples are at the beginning of the buffers, the state of the older samples is undefined and n_new_samples contains total number of new samples per channel. If nsamples is non-zero, the buffers are rotated so the oldest samples come first. n_samples_new then contains the total number of new samples per channel. If n_new_samples is larger than the number of samples in the AC variable that means samples had be discarded.

5.193.3.7 get_time_correction() template<typename T >

void **lsl2ac::save_var_t**< T >::get_time_correction ( ) [inline], [private]

Refresh time correction value every 5s.

5.193.3.8 insert_vars() template<typename T >

void **lsl2ac::save_var_t**< T >::insert_vars ( ) [inline], [private]

Insert stream value, time stamp and time offset into ac space.

5.193.4 Member Data Documentation

5.193.4.1 stream template<typename T >

lsl::stream_inlet **lsl2ac::save_var_t**< T >::stream [private]

LSL stream outlet.

Interface to lsl

5.193.4.2 buf template<typename T >

std::vector<T> **lsl2ac::save_var_t**< T >::buf [private]

Data buffer of the ac variable.

5.193.4.3 ts_buf template<typename T >

std::vector<double> **lsl2ac::save_var_t**< T >::ts_buf [private]

Timestamp buffer.

5.193.4.4 tc_buf template<typename T >

std::vector<double> **lsl2ac::save_var_t**< T >::tc_buf [private]

Clock correction buffer.

5.193.4.5 ac template<typename T >

MHA_AC::algo_comm_t& **lsl2ac::save_var_t**< T >::ac [private]

Handle to AC space.

5.193.4.6 cv template<typename T >

MHA_AC::comm_var_t **lsl2ac::save_var_t**< T >::cv [private]

Timeseries AC variable.

5.193.4.7 ts template<typename T >

MHA_AC::comm_var_t **lsl2ac::save_var_t**< T >::ts [private]

Timestamp AC variable.

5.193.4.8 tc template<typename T >

MHA_AC::comm_var_t **lsl2ac::save_var_t**< T >::tc [private]

Time correction AC variable.

5.193.4.9 ts_name template<typename T >

std::string **lsl2ac::save_var_t**< T >::ts_name [private]

Timestamp AC variable name.

5.193.4.10 tc_name template<typename T >

std::string **lsl2ac::save_var_t**< T >::tc_name [private]

Time correction AC variable name.

5.193.4.11 new_name template<typename T >

std::string **lsl2ac::save_var_t**< T >::new_name [private]

Number of new samples AC variable name.

5.193.4.12 tic template<typename T >

std::chrono::time_point<std::chrono::steady_clock> **lsl2ac::save_var_t**< T >::tic [private]

time point of last time correction pull

5.193.4.13 skip template<typename T >

bool **lsl2ac::save_var_t**< T >::skip =false [private]

Should the variable be skipped in future process calls? Only true when error occured.

5.193.4.14 ob template<typename T >

overrun_behavior **lsl2ac::save_var_t**< T >::ob [private]

Behavior on stream overrun.

5.193.4.15 name template<typename T >

const std::string **lsl2ac::save_var_t**< T >::name [private]

Name of stream.

Must be saved separately because the stream info might be unrecoverable in error cases

5.193.4.16 nchannels template<typename T >

int32_t **lsl2ac::save_var_t**< T >::nchannels [private]

Number of channels.

5.193.4.17 nsamples template<typename T >

std::size_t **lsl2ac::save_var_t**< T >::nsamples [private]

Number of samples per channel in the AC variable.

Zero means resize as needed.

5.193.4.18 chunksize template<typename T >

std::size_t **lsl2ac::save_var_t**< T >::chunksize [private]

Maximal chunk size of lsl stream.

5.193.4.19 bufsize template<typename T >

std::size_t **lsl2ac::save_var_t**< T >::bufsize [private]

Total buffer size of the ac variable buffer.

nsamples∗nchannels if nsamples is set, chunksize∗nchannels otherwise if chunksize is set, nchannels if neither nsamples and chunksize are set.

5.193.4.20 n_new_samples template<typename T >

int **lsl2ac::save_var_t**< T >::n_new_samples [private]

Number of most recently pulled samples per channel.

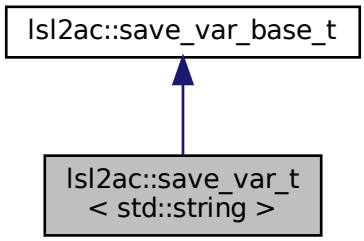
The documentation for this class was generated from the following file:

- **lsl2ac.hh**

5.194 lsl2ac::save_var_t< std::string > Class Reference

Specialication for marker streams.

Inheritance diagram for lsl2ac::save_var_t< std::string >:



Public Member Functions

- **save_var_t** (const **save_var_t** &)=delete
- **save_var_t** (**save_var_t** &&)=delete
- virtual ~**save_var_t** ()=default
- **save_var_t** & **operator=** (const **save_var_t** &)=delete
- **save_var_t** & **operator=** (**save_var_t** &&)=delete
- **save_var_t** (const lsl::stream_info &info_, **MHA_AC::algo_comm_t** &ac_, **overrun_behavior** ob_, int buflen_, int chunksize_, int strlen_)

  C'tor of lsl to ac bridge.
- lsl::stream_info **info** () override

  Get stream info object from stream inlet.
- void **receive_frame** () override

  Receive a samples from lsl and copy to AC space.

Private Member Functions

- std::size_t **copy_string_safe** ()

  Copy string to AC buffer, stop at the latest at buffer end, make sure that the string is always zero-terminated.
- void **pull_samples_ignore** ()

  Pull new samples, ignore overrun, i.e.
- void **pull_samples_discard** ()

  Pull new samples as long as there are samples ready for pickup in the LSL buffers.
- void **get_time_correction** ()

  Refresh time correction value every 5s.
- void **insert_vars** ()

  Insert stream value, time stamp and time offset into ac space.

Private Attributes

- lsl::stream_inlet **stream**

  LSL stream outlet.
- std::string **str**

  Temporary storage for marker string.
- std::vector< char > **buf**

  Data buffer of the ac variable.
- double **ts**

  Timestamp.
- **MHA_AC::algo_comm_t** & **ac**

  Handle to AC space.
- **MHA_AC::comm_var_t** **cv**

  Timeseries AC variable.
- double **tc** =0.0

Current time correction.

- std::string **ts_name**

  Timestamp AC variable name.
- std::string **tc_name**

  Time correction AC variable name.
- std::string **new_name**

  Number of new samples AC variable name.
- std::chrono::time_point< std::chrono::steady_clock > **tic**

  time point of last time correction pull
- bool **skip** =false

  Should the variable be skipped in future process calls? Only true when error occured.
- **overrun_behavior ob**

  Behavior on stream overrun.
- const std::string **name**

  Name of stream.

5.194.1 Detailed Description

Specialication for marker streams.

5.194.2 Constructor & Destructor Documentation

5.194.2.1 save_var_t() [1/3] lsl2ac::save_var_t< std::string >:: save_var_t (const save_var_t< std::string > &) [delete]

5.194.2.2 save_var_t() [2/3] lsl2ac::save_var_t< std::string >:: save_var_t (save_var_t< std::string > &&) [delete]

5.194.2.3 ~save_var_t() virtual lsl2ac::save_var_t< std::string >::~ save_var_t ( ) [virtual], [default]

5.194.2.4 save_var_t() [3/3] lsl2ac::save_var_t< std::string >:: save_var_t (const lsl::stream_info & *info_*, MHA_AC::algo_comm_t & *ac_*, overrun_behavior *ob_*, int *buflen_*, int *chunksize_*, int *strlen_*) [inline]

C'tor of lsl to ac bridge.

Parameters

*name_*	Name of LSL stream to be received
*info_*	LSL stream info object containing metadata
*ac_*	Handle to ac space
*ob_*	Overrun behavior. 0=Discard oldest, 1=Ignore
*buflen_*	LSL buffer size
*chunksize_*	LSL chunk size
*strlen_*	Length string buffer.

5.194.3 Member Function Documentation

5.194.3.1 operator=() [1/2] save_var_t& lsl2ac::save_var_t< std::string >::operator= (const save_var_t< std::string > &) [delete]

5.194.3.2 operator=() [2/2] save_var_t& lsl2ac::save_var_t< std::string >::operator= (save_var_t< std::string > &&) [delete]

5.194.3.3 info() lsl::stream_info lsl2ac::save_var_t< std::string >::info ( ) [inline], [override], [virtual]

Get stream info object from stream inlet.

Implements **lsl2ac::save_var_base_t** (p. 797).

5.194.3.4 receive_frame() void lsl2ac::save_var_t< std::string >::receive_frame ( ) [inline], [override], [virtual]

Receive a samples from lsl and copy to AC space.

Handling of underrun is configuration-dependent

Implements **lsl2ac::save_var_base_t** (p. 797).

5.194.3.5 copy_string_safe() std::size_t lsl2ac::save_var_t< std::string >::copy_string_safe ( ) [inline], [private]

Copy string to AC buffer, stop at the latest at buffer end, make sure that the string is always zero-terminated.

Returns

The number of characters copied into the AC buffer, also the number entries in the AC variable

5.194.3.6 pull_samples_ignore() void lsl2ac::save_var_t< std::string >::pull_samples_ignore ( ) [inline], [private]

Pull new samples, ignore overrun, i.e.

only pull one sample from the buffer, do not check if there are newer ones waiting

5.194.3.7 pull_samples_discard() void lsl2ac::save_var_t< std::string >::pull_samples_discard ( ) [inline], [private]

Pull new samples as long as there are samples ready for pickup in the LSL buffers.

Overwrite old markers

5.194.3.8 get_time_correction() void lsl2ac::save_var_t< std::string >::get_time_correction ( ) [inline], [private]

Refresh time correction value every 5s.

5.194.3.9 insert_vars() void lsl2ac::save_var_t< std::string >::insert_vars ( ) [inline], [private]

Insert stream value, time stamp and time offset into ac space.

5.194.4 Member Data Documentation

5.194.4.1 stream lsl::stream_inlet lsl2ac::save_var_t< std::string >::stream [private]

LSL stream outlet.

Interface to lsl

5.194.4.2 str std::string lsl2ac::save_var_t< std::string >::str [private]

Temporary storage for marker string.

5.194.4.3 buf std::vector<char> lsl2ac::save_var_t< std::string >::buf [private]

Data buffer of the ac variable.

5.194.4.4 ts double lsl2ac::save_var_t< std::string >::ts [private]

Timestamp.

5.194.4.5 ac MHA_AC::algo_comm_t& lsl2ac::save_var_t< std::string >::ac [private]

Handle to AC space.

5.194.4.6 cv `MHA_AC::comm_var_t lsl2ac::save_var_t< std::string >::cv [private]`

Timeseries AC variable.

5.194.4.7 tc `double lsl2ac::save_var_t< std::string >::tc =0.0 [private]`

Current time correction.

5.194.4.8 ts_name `std::string lsl2ac::save_var_t< std::string >::ts_name [private]`

Timestamp AC variable name.

5.194.4.9 tc_name `std::string lsl2ac::save_var_t< std::string >::tc_name [private]`

Time correction AC variable name.

5.194.4.10 new_name `std::string lsl2ac::save_var_t< std::string >::new_name [private]`

Number of new samples AC variable name.

5.194.4.11 tic `std::chrono::time_point<std::chrono::steady_clock> lsl2ac::save_var_t< std::string >::tic [private]`

time point of last time correction pull

5.194.4.12 skip `bool lsl2ac::save_var_t< std::string >::skip =false [private]`

Should the variable be skipped in future process calls? Only true when error occured.

5.194.4.13 ob `overrun_behavior lsl2ac::save_var_t< std::string >::ob [private]`

Behavior on stream overrun.

5.194.4.14 name `const std::string lsl2ac::save_var_t< std::string >::name [private]`

Name of stream.

Must be saved separately because the stream info might be unrecoverable in error cases

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

- **lsl2ac.hh**

5.195 matlab_wrapper::callback Class Reference

Utility class connecting a user_config_t instance to its corresponding configuration parser.

Public Member Functions

- **callback** (**matlab_wrapper_t** *parent_, user_config_t *user_config_, **MHAParser::mfloat_t** *var_)
  Ctor.
- void **on_writeaccess** ()
  Write access callback.

Private Attributes

- user_config_t * **user_config**
  Ptr to user_config_t.
- **MHAParser::mfloat_t** * **var**
  Ptr to parser.
- **matlab_wrapper_t** * **parent**
  Ptr to parent plugin.

5.195.1 Detailed Description

Utility class connecting a user_config_t instance to its corresponding configuration parser.

Provides the callback. Every time the a user defined configuration parser is changed, this class makes sure that the corresponding 'matlab-side' struct is also changed and a new real time config is created and pushed.

5.195.2 Constructor & Destructor Documentation

5.195.2.1 callback() `matlab_wrapper::callback::callback (`

```
  matlab_wrapper_t * parent_,
 user_config_t * user_config_,
  MHAParser::mfloat_t * var_ )
```

Ctor.

Parameters

*parent_*	Ptr to the parent parser. Used to signal finished change.
*user_config_*	Ptr to user_config_t struct holding the 'matlab side' of the configuration variable
*var_*	Ptr to the configuration parser corresponding to user_config_

5.195.3 Member Function Documentation

5.195.3.1 on_writeaccess() `void matlab_wrapper::callback::on_writeaccess ( )`

Write access callback.

To be called on every writeaccess of ∗var. Synchronises ∗var and ∗user_config.

5.195.4 Member Data Documentation

5.195.4.1 user_config `user_config_t* matlab_wrapper::callback::user_config [private]`

Ptr to user_config_t.

5.195.4.2 var `MHAParser::mfloat_t* matlab_wrapper::callback::var [private]`

Ptr to parser.

5.195.4.3 parent `matlab_wrapper_t* matlab_wrapper::callback::parent [private]`

Ptr to parent plugin.

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

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

5.196 matlab_wrapper::matlab_wrapper_rt_cfg_t Class Reference

Thin wrapper around the emxArray containing the user defined configuration variables.

Public Member Functions

- **matlab_wrapper_rt_cfg_t** (emxArray_user_config_t *user_config_)
  Ctor of the real time configuration class.
- ~**matlab_wrapper_rt_cfg_t** ()
  Dtor.

Public Attributes

- emxArray_user_config_t * **user_config**
  User configuration to be used in the process callback.

5.196.1 Detailed Description

Thin wrapper around the emxArray containing the user defined configuration variables.

This wrapper holds a copy of the user configuration which is used by the process callback. On write access to a variable, the user configuration is changed and a new copy is created and exchanged in a real time safe manner.

5.196.2 Constructor & Destructor Documentation

5.196.2.1 matlab_wrapper_rt_cfg_t() `matlab_wrapper::matlab_wrapper_rt_cfg_t::matlab_wrapper_rt_cfg_t (emxArray_user_config_t * user_config_ ) [explicit]`

Ctor of the real time configuration class.

Creates a local copy of the user config

Parameters

*user_config_*	Pointer to the array containing the user config

5.196.2.2 ~matlab_wrapper_rt_cfg_t() `matlab_wrapper::matlab_wrapper_rt_cfg_t::~matlab_wrapper_rt_cfg_t ( )`

Dtor.

Calls the appropriate c-style destructors of the emx_ types

5.196.3 Member Data Documentation

5.196.3.1 user_config `emxArray_user_config_t* matlab_wrapper::matlab_wrapper_rt_↩ cfg_t::user_config`

User configuration to be used in the process callback.

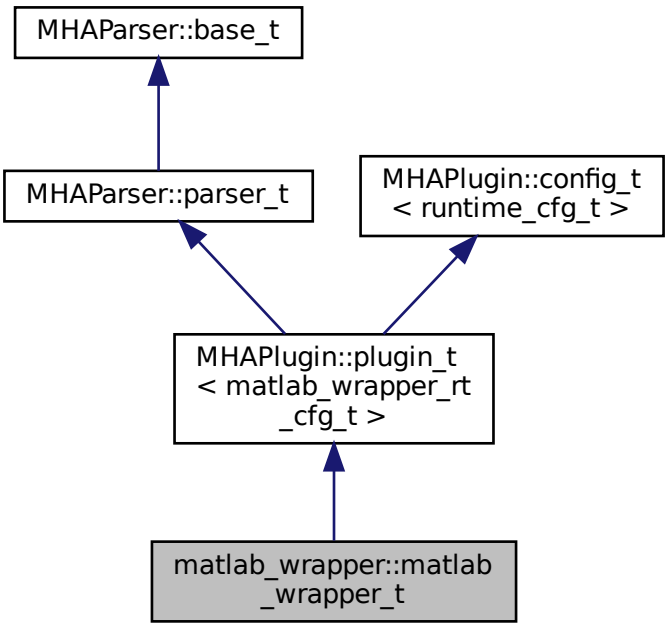
The documentation for this class was generated from the following files:

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

5.197 matlab_wrapper::matlab_wrapper_t Class Reference

Matlab wraper plugin interface class.

Inheritance diagram for matlab_wrapper::matlab_wrapper_t:



Classes

- class **wrapped_plugin_t**
  Wrapper class around the matlab-generated library.

Public Member Functions

- **matlab_wrapper_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
  Ctor of the plugin interface class.
- void **process** (**mha_wave_t** ∗sin, **mha_wave_t** ∗∗sout)
  Pure waveform processing.
- void **process** (**mha_spec_t** ∗sin, **mha_spec_t** ∗∗sout)
  Pure spectrum processing.
- void **process** (**mha_wave_t** ∗sin, **mha_spec_t** ∗∗sout)
  Signal processing with domain transformation from waveform to spectrum.
- void **process** (**mha_spec_t** ∗sin, **mha_wave_t** ∗∗sout)
  Signal processing with domain transformation from spectrum to waveform.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare callback.
- void **release** ()
  Release callback.

Private Member Functions

- void **insert_monitors** ()
- void **update_monitors** ()
- void **insert_config_vars** ()
- void **load_lib** ()
  Create new library wrapper and load library.
- void **update** ()
  Create new real time safe user config from user config.

Private Attributes

- friend **callback**
- **MHAParser::string_t library_name** {"Name of matlab generated library",""}
  Configuration variable holding the file name of the matlab generated library.
- **MHAEvents::patchbay_t**< **matlab_wrapper_t** > **patchbay**
  Patchbay for the interface plugins.
- **MHAEvents::patchbay_t**< **callback** > **cb_patchbay**
  Patchbay for the custom callbacks.
- std::unique_ptr< **wrapped_plugin_t** > **plug**
  Unique ptr holding the instance of the plugin wrapper class.
- std::deque< **MHAParser::mfloat_t** > **vars**
  Deque holding the user defined configuration variables.
- std::deque< **callback** > **callbacks**
  Deque holding the callbacks for the user defined variables' write access.
- std::deque< **MHAParser::mfloat_mon_t** > **monitors**
  Deque holding the monitors corresponding to user state.

Additional Inherited Members

5.197.1 Detailed Description

Matlab wraper plugin interface class.

5.197.2 Constructor & Destructor Documentation

5.197.2.1 matlab_wrapper_t() `matlab_wrapper::matlab_wrapper_t::matlab_wrapper_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Ctor of the plugin interface class.

Parameters

iac	Handle to ac space
configured_name	Configured name of this plugin

5.197.3 Member Function Documentation

5.197.3.1 process() [1/4] `void matlab_wrapper::matlab_wrapper_t::process (`

```
mha_wave_t * sin,
mha_wave_t ** sout )
```

Pure waveform processing.

Parameters

sin	input waveform signal

Parameters

sout	output waveform signal

5.197.3.2 process() [2/4] void matlab_wrapper::matlab_wrapper_t::process (

```
mha_spec_t * sin,
mha_spec_t ** sout )
```

Pure spectrum processing.

Parameters

sin	input spectrum signal
sout	output spectrum signal

5.197.3.3 process() [3/4] void matlab_wrapper::matlab_wrapper_t::process (

```
mha_wave_t * sin,
mha_spec_t ** sout )
```

Signal processing with domain transformation from waveform to spectrum.

Parameters

sin	input waveform signal
sout	output spectrum signal

5.197.3.4 process() [4/4] void matlab_wrapper::matlab_wrapper_t::process (mha_spec_t * sin, mha_wave_t ** sout)

Signal processing with domain transformation from spectrum to waveform.

Parameters

sin	input spectrum signal
sout	output waveform signal

5.197.3.5 prepare() void matlab_wrapper::matlab_wrapper_t::prepare (mhaconfig_t & signal_info) [virtual]

Prepare callback.

Parameters

signal_info	struct holding the input/output signal dimensions

Implements **MHAPlugin::plugin_t< matlab_wrapper_rt_cfg_t >** (p. 1301).

5.197.3.6 release() void matlab_wrapper::matlab_wrapper_t::release (void) [virtual]

Release callback.

Reimplemented from **MHAPlugin::plugin_t< matlab_wrapper_rt_cfg_t >** (p. 1302).

5.197.3.7 insert_monitors() void matlab_wrapper::matlab_wrapper_t::insert_monitors ( ) [private]

5.197.3.8 update_monitors() void matlab_wrapper::matlab_wrapper_t::update_monitors ( ) [private]

5.197.3.9 insert_config_vars() void matlab_wrapper::matlab_wrapper_t::insert_↩ config_vars ( ) [private]

5.197.3.10 load_lib() void matlab_wrapper::matlab_wrapper_t::load_lib ( ) [private]

Create new library wrapper and load library.

To be called write access to library_name.

5.197.3.11 update() void matlab_wrapper::matlab_wrapper_t::update ( ) [private]

Create new real time safe user config from user config.

Called by the custom callbacks.

5.197.4 Member Data Documentation

5.197.4.1 callback friend matlab_wrapper::matlab_wrapper_t::callback [private]

5.197.4.2 library_name **MHAParser::string_t** matlab_wrapper::matlab_wrapper_t↩ ::library_name {"Name of matlab generated library",""} [private]

Configuration variable holding the file name of the matlab generated library.

5.197.4.3 patchbay

`MHAEvents::patchbay_t< matlab_wrapper_t> matlab_wrapper::matlab_wrapper_t::patchbay [private]`

Patchbay for the interface plugins.

5.197.4.4 cb_patchbay

`MHAEvents::patchbay_t< callback> matlab_wrapper::matlab_wrapper_t::cb_patchbay [private]`

Patchbay for the custom callbacks.

Can use normal patchbay bc/ of the interface of patchbay_t

5.197.4.5 plug

`std::unique_ptr< wrapped_plugin_t> matlab_wrapper::matlab_wrapper_t::plug [private]`

Unique ptr holding the instance of the plugin wrapper class.

5.197.4.6 vars

`std::deque< MHAParser::mfloat_t> matlab_wrapper::matlab_wrapper_t::vars [private]`

Deque holding the user defined configuration variables.

Deque is used because we need an indexable container that does not invalidate pointers

5.197.4.7 callbacks

`std::deque< callback> matlab_wrapper::matlab_wrapper_t::callbacks [private]`

Deque holding the callbacks for the user defined variables' write access.

Deque is used because we need an indexable container that does not invalidate pointers

5.197.4.8 monitors

`std::deque< MHAParser::mfloat_mon_t> matlab_wrapper::matlab_wrapper_t::monitors [private]`

Deque holding the monitors corresponding to user state.

Deque is used because we need an indexable container that does not invalidate pointers.

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

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

5.198 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t Class Reference

Wrapper class around the matlab-generated library.

Public Member Functions

- **wrapped_plugin_t** ()=delete
- **wrapped_plugin_t** (const char *name_)
  Ctor.
- virtual ~**wrapped_plugin_t** ()
  Dtor.
- **mha_wave_t** * **process_ww** (**mha_wave_t** *s, emxArray_user_config_t *user_config_)
  Process callback.
- **mha_spec_t** * **process_ss** (**mha_spec_t** *s, emxArray_user_config_t *user_config_)
  Process callback.
- **mha_spec_t** * **process_ws** (**mha_wave_t** *s, emxArray_user_config_t *user_config_)
  Process callback.
- **mha_wave_t** * **process_sw** (**mha_spec_t** *s, emxArray_user_config_t *user_config_)
  Process callback.
- void **prepare** (**mhaconfig_t** &config)
  Prepare callback.
- void **release** ()
  Release callback.

Public Attributes

- emxArray_user_config_t * **user_config** =nullptr
  Ptr to user config array.
- emxArray_user_config_t * **state** =nullptr
  Ptr to state array.

Private Attributes

- **dynamiclib_t library_handle**
  Handle to matlab generated shared library.
- void(* **fcn_terminate**)() =nullptr
  Handle to matlab generated cleanup function.
- void(* **fcn_init**)(emxArray_user_config_t *, emxArray_user_config_t *) =nullptr
  Handle to matlab generated init function.

- void(∗ **fcn_process_ww**)(const emxArray_real_T ∗, const signal_dimensions_t ∗, const emxArray_user_config_t ∗, emxArray_user_config_t ∗, emxArray_real_T ∗) =nullptr

  Handle to matlab generated wave to wave process function.
- void(∗ **fcn_process_ss**)(const emxArray_creal_T ∗, const signal_dimensions_t ∗, const emxArray_user_config_t ∗, emxArray_user_config_t ∗, emxArray_creal_T ∗) =nullptr

  Handle to matlab generated spectrum to spectrum process function.
- void(∗ **fcn_process_ws**)(const emxArray_real_T ∗, const signal_dimensions_t ∗, const emxArray_user_config_t ∗, emxArray_user_config_t ∗, emxArray_creal_T ∗) =nullptr

  Handle to matlab generated wave to spectrum process function.
- void(∗ **fcn_process_sw**)(const emxArray_creal_T ∗, const signal_dimensions_t ∗, const emxArray_user_config_t ∗, emxArray_user_config_t ∗, emxArray_real_T ∗) =nullptr

  Handle to matlab generated spectrum to wave process function.
- void(∗ **fcn_prepare**)(signal_dimensions_t ∗, emxArray_user_config_t ∗, emxArray_↩ user_config_t ∗) =nullptr

  Handle to matlab generated prepare function.
- void(∗ **fcn_release**)() =nullptr
- signal_dimensions_t **signal_dimensions** {0,'U',0,0,0,0}

  Signal dimensions.
- emxArray_real_T ∗ **wave_in** =nullptr

  Ptr to emxArray holding the input signal.
- emxArray_real_T ∗ **wave_out** =nullptr

  Ptr to emxArray holding the output signal.
- emxArray_creal_T ∗ **spec_in** =nullptr

  Ptr to emxArray holding the input signal.
- emxArray_creal_T ∗ **spec_out** =nullptr

  Ptr to emxArray holding the output signal.
- std::unique_ptr< **MHASignal::waveform_t** > **mha_wave_out**

  MHA waveform holding the output signal.
- std::unique_ptr< **MHASignal::spectrum_t** > **mha_spec_out**

  MHA waveform holding the output signal.

5.198.1 Detailed Description

Wrapper class around the matlab-generated library.

5.198.2 Constructor & Destructor Documentation

5.198.2.1 wrapped_plugin_t() [1/2] matlab_wrapper::matlab_wrapper_t::wrapped_↩ plugin_t::wrapped_plugin_t ( ) [delete]

5.198.2.2 wrapped_plugin_t() [2/2] matlab_wrapper::matlab_wrapper_t::wrapped_↩ plugin_t::wrapped_plugin_t (

```
            const char * name_ )  [explicit]
```

Ctor.

Opens matlab-generated library and resolves functions

Parameters

*name_*	file name of shared library

5.198.2.3 ∼wrapped_plugin_t() matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t↩ ::∼wrapped_plugin_t ( ) [virtual]

Dtor.

5.198.3 Member Function Documentation

5.198.3.1 process_ww() **mha_wave_t** * matlab_wrapper::matlab_wrapper_t::wrapped_↩ plugin_t::process_ww (

```
             mha_wave_t * s,
            emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type wave to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

s	input/output signal
*user_config_*	Ptr to user configuration array

5.198.3.2 process_ss()

```
mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::process_ss (
            mha_spec_t * s,
            emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type spectrum to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

s	input/output signal
*user_config_*	Ptr to user configuration array

5.198.3.3 process_ws()

```
mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::process_ws (
            mha_wave_t * s,
            emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type wave to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

s	input/output signal
*user_config_*	Ptr to user configuration array

5.198.3.4 process_sw()

```
mha_wave_t * matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::process_sw (
            mha_spec_t * s,
            emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type spectrum to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

s	input/output signal
*user_config_*	Ptr to user configuration array

5.198.3.5 prepare() `void matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::prepare (`

```
mhaconfig_t & config )
```

Prepare callback.

Calls wrapped prepare function if necessary and determines output signal dimensions

5.198.3.6 release() `void matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::release (`

```
void  )
```

Release callback.

Cleans up io arrays and calls wrapped release if necessary.

5.198.4 Member Data Documentation

5.198.4.1 user_config `emxArray_user_config_t* matlab_wrapper::matlab_wrapper_t↩ ::wrapped_plugin_t::user_config =nullptr`

Ptr to user config array.

5.198.4.2 state `emxArray_user_config_t* matlab_wrapper::matlab_wrapper_t::wrapped↩ _plugin_t::state =nullptr`

Ptr to state array.

5.198.4.3 library_handle

`dynamiclib_t matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::library_handle [private]`

Handle to matlab generated shared library.

5.198.4.4 fcn_terminate

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_terminate) () =nullptr [private]`

Handle to matlab generated cleanup function.

5.198.4.5 fcn_init

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_init) (emxArray_user_config_t *, emxArray_user_config_t *) =nullptr [private]`

Handle to matlab generated init function.

5.198.4.6 fcn_process_ww

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ww) (const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr [private]`

Handle to matlab generated wave to wave process function.

5.198.4.7 fcn_process_ss

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ss) (const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr [private]`

Handle to matlab generated spectrum to spectrum process function.

5.198.4.8 fcn_process_ws

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ws) (const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr [private]`

Handle to matlab generated wave to spectrum process function.

5.198.4.9 fcn_process_sw

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_sw) (const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr [private]`

Handle to matlab generated spectrum to wave process function.

5.198.4.10 fcn_prepare

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_prepare) (signal_dimensions_t *, emxArray_user_config_t *, emxArray_user_config_t *) =nullptr [private]`

Handle to matlab generated prepare function.

5.198.4.11 fcn_release

`void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_release) () =nullptr [private]`

5.198.4.12 signal_dimensions

`signal_dimensions_t matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::signal_dimensions {0,'U',0,0,0,0} [private]`

Signal dimensions.

Matlab-equivalent to mha_config_t

5.198.4.13 wave_in

`emxArray_real_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wave_in =nullptr [private]`

Ptr to emxArray holding the input signal.

5.198.4.14 wave_out

`emxArray_real_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wave_out =nullptr [private]`

Ptr to emxArray holding the output signal.

5.198.4.15 spec_in

`emxArray_creal_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::spec_in =nullptr [private]`

Ptr to emxArray holding the input signal.

5.198.4.16 spec_out

`emxArray_creal_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::spec_out =nullptr [private]`

Ptr to emxArray holding the output signal.

5.198.4.17 mha_wave_out

`std::unique_ptr< MHASignal::waveform_t> matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::mha_wave_out [private]`

MHA waveform holding the output signal.

5.198.4.18 mha_spec_out

`std::unique_ptr< MHASignal::spectrum_t> matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::mha_spec_out [private]`

MHA waveform holding the output signal.

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

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

5.199 matlab_wrapper::types< T > Struct Template Reference

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

- **matlab_wrapper.hh**

5.200 matlab_wrapper::types< MHA_SPECTRUM > Struct Reference

Public Types

- typedef emxArray_creal_T **array_type**
- typedef **mha_spec_t signal_type**
- typedef **MHASignal::spectrum_t class_signal_type**

5.200.1 Member Typedef Documentation

5.200.1.1 array_type typedef emxArray_creal_T matlab_wrapper::types< MHA_SPECTRUM >:: array_type

5.200.1.2 signal_type typedef mha_spec_t matlab_wrapper::types< MHA_SPECTRUM >:: signal_type

5.200.1.3 class_signal_type typedef MHASignal::spectrum_t matlab_wrapper::types< MHA_SPECTRUM >:: class_signal_type

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

- **matlab_wrapper.hh**

5.201 matlab_wrapper::types< MHA_WAVEFORM > Struct Reference

Public Types

- typedef emxArray_real_T **array**
- typedef **mha_wave_t signal_type**
- typedef **MHASignal::waveform_t class_signal_type**

5.201.1 Member Typedef Documentation

5.201.1.1 array typedef emxArray_real_T matlab_wrapper::types< MHA_WAVEFORM >:: array

5.201.1.2 signal_type typedef mha_wave_t matlab_wrapper::types< MHA_WAVEFORM >:: signal_type

5.201.1.3 class_signal_type typedef MHASignal::waveform_t matlab_wrapper::types< MHA_WAVEFORM >:: class_signal_type

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

- **matlab_wrapper.hh**

5.202 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**

5.202.1 Constructor & Destructor Documentation

5.202.1.1 cfg_t()

```
cfg_t::cfg_t (
    std::vector< std::vector< float > > imixer,
    unsigned int ci,
    unsigned int co,
    unsigned int fragsize,
    unsigned int nfft )
```

5.202.2 Member Function Documentation

5.202.2.1 process() [1/2]

```
mha_wave_t * cfg_t::process (
    mha_wave_t * s )
```

5.202.2.2 process() [2/2]

```
mha_spec_t * cfg_t::process (
    mha_spec_t * s )
```

5.202.3 Member Data Documentation

5.202.3.1 m

```
MHASignal::waveform_t matrixmixer::cfg_t::m [private]
```

5.202.3.2 wout

```
MHASignal::waveform_t matrixmixer::cfg_t::wout [private]
```

5.202.3.3 sout

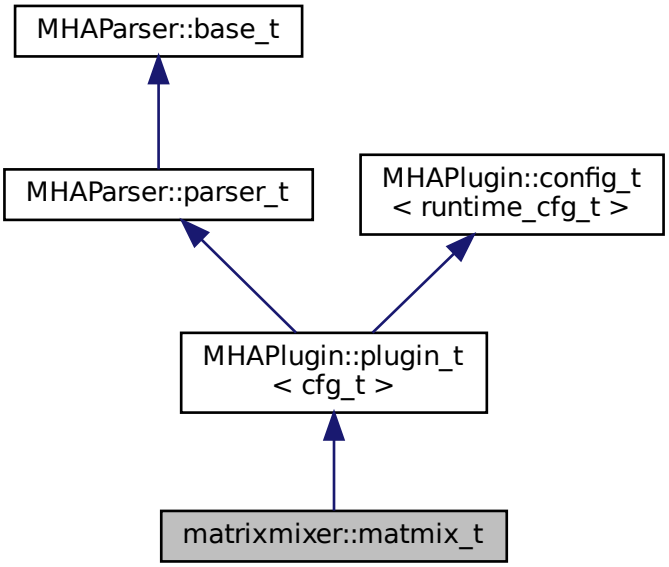
```
MHASignal::spectrum_t matrixmixer::cfg_t::sout [private]
```

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

- **matrixmixer.cpp**

5.203 matrixmixer::matmix_t Class Reference

Inheritance diagram for matrixmixer::matmix_t:



Public Member Functions

- **matmix_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.203.1 Constructor & Destructor Documentation

5.203.1.1 matmix_t() matrixmixer::matmix_t::matmix_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.203.2 Member Function Documentation

5.203.2.1 prepare() void matrixmixer::matmix_t::prepare (

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.203.2.2 process() [1/2] mha_wave_t * matrixmixer::matmix_t::process (

```
mha_wave_t * s )
```

5.203.2.3 process() [2/2] mha_spec_t * matrixmixer::matmix_t::process (

```
mha_spec_t * s )
```

5.203.2.4 update_m() void matrixmixer::matmix_t::update_m (

```
void ) [private]
```

5.203.3 Member Data Documentation

5.203.3.1 patchbay

`MHAEvents::patchbay_t< matmix_t> matrixmixer::matmix_t::patchbay [private]`

5.203.3.2 mixer

`MHAParser::mfloat_t matrixmixer::matmix_t::mixer [private]`

5.203.3.3 ci

`unsigned int matrixmixer::matmix_t::ci [private]`

5.203.3.4 co

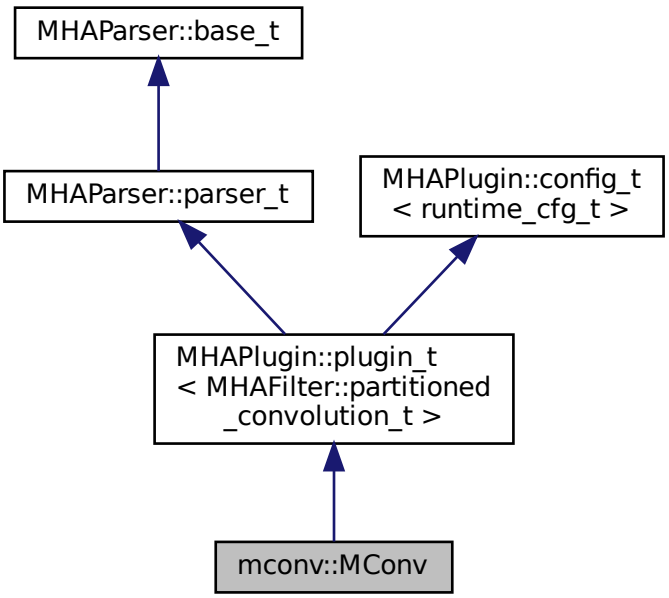
`unsigned int matrixmixer::matmix_t::co [private]`

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

- **matrixmixer.cpp**

5.204 mconv::MConv Class Reference

Inheritance diagram for mconv::MConv:



Public Member Functions

- **MConv** (**MHA_AC::algo_comm_t** &iac, const std::string &algoname)
  Plugin constructor.
- void **prepare** (**mhaconfig_t** &mhaconfig)
  Prepare this plugin for processing.
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)

Private Member Functions

- void **update** ()
  *Update is needed only once, since this plugin allows only change of irs after **prepare()** (p. 839).*
- void **update_irs** ()
  *This function updates the irs without allowing a change of its size after **prepare()** (p. 839).*

Private Attributes

- **MHAParser::int_t nchannels_out**
  Number of output channels to produce.
- **MHAParser::vint_t inch**
  Vector of input channel indices.
- **MHAParser::vint_t outch**
  Vector of output channel indices.
- **MHAParser::mfloat_t irs**
  Impulse responses, one per row.
- unsigned int **nchannels_in**
  Number of input channels, set during prepare.
- unsigned int **fragsize**
  Fragsize, set during prepare, is used as the partition length in the partitioned convolution.
- **MHAEvents::patchbay_t**< **MConv** > **patchbay**

Additional Inherited Members

5.204.1 Detailed Description

class implements plugin for partitioned convolution. A matrix of impulse responses, filtering n input channels to m output channels, is supported.

5.204.2 Constructor & Destructor Documentation

5.204.2.1 MConv() mconv::MConv::MConv (MHA_AC::algo_comm_t & iac, const std::string & algoname)

Plugin constructor.

Parameters

iac	handle and function pointers for algorithm communication
configured_name	The name by which e.g an mhachain refers to this instance of the mconv plugin

5.204.3 Member Function Documentation

5.204.3.1 prepare() void mconv::MConv::prepare (mhaconfig_t & mhaconfig) [virtual]

Prepare this plugin for processing.

Parameters

mhaconfig	Configuration for this plugin (Input/Output parameter) Sample rate, fragment size, number of channels are detailed here.

Implements **MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >** (p. 1301).

5.204.3.2 release() void mconv::MConv::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >** (p. 1302).

5.204.3.3 process() `mha_wave_t * mconv::MConv::process ( mha_wave_t * s_in )`

5.204.3.4 update() `void mconv::MConv::update ( ) [private]`

Update is needed only once, since this plugin allows only change of irs after **prepare()** (p. 839).

5.204.3.5 update_irs() `void mconv::MConv::update_irs ( ) [private]`

This function updates the irs without allowing a change of its size after **prepare()** (p. 839).

5.204.4 Member Data Documentation

5.204.4.1 nchannels_out `MHAParser::int_t mconv::MConv::nchannels_out [private]`

Number of output channels to produce.

5.204.4.2 inch `MHAParser::vint_t mconv::MConv::inch [private]`

Vector of input channel indices.

Each element in this vector identifies the input channel to which to apply the corresponding impulse response in irs.

5.204.4.3 outch `MHAParser::vint_t mconv::MConv::outch [private]`

Vector of output channel indices.

Each element in this vector identifies the output channel to which the result of filtering with the corresponding impulse response in irs is mixed.

5.204.4.4 irs `MHAParser::mfloat_t mconv::MConv::irs [private]`

Impulse responses, one per row.

For each row, the corresponding element of inch identifies the source channel, and the corresponding element of outch identifies the target channel.

5.204.4.5 nchannels_in `unsigned int mconv::MConv::nchannels_in [private]`

Number of input channels, set during prepare.

5.204.4.6 fragsize `unsigned int mconv::MConv::fragsize [private]`

Fragsize, set during prepare, is used as the partition length in the partitioned convolution.

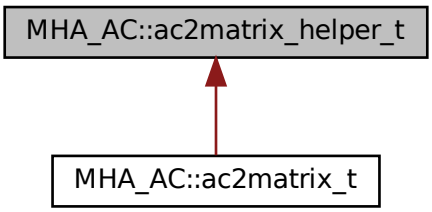
5.204.4.7 patchbay `MHAEvents::patchbay_t< MConv> mconv::MConv::patchbay [private]`

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

- **mconv.cpp**

5.205 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**

5.205.1 Constructor & Destructor Documentation

5.205.1.1 ac2matrix_helper_t()

```
MHA_AC::ac2matrix_helper_t::ac2matrix_helper_t (
    algo_comm_t & iac,
    const std::string & iname )
```

5.205.2 Member Function Documentation

5.205.2.1 getvar()

```
void MHA_AC::ac2matrix_helper_t::getvar ( )
```

5.205.3 Member Data Documentation

5.205.3.1 ac

`algo_comm_t* MHA_AC::ac2matrix_helper_t::ac`

5.205.3.2 name

`std::string MHA_AC::ac2matrix_helper_t::name`

5.205.3.3 username

`std::string MHA_AC::ac2matrix_helper_t::username`

5.205.3.4 size

`MHASignal::uint_vector_t MHA_AC::ac2matrix_helper_t::size`

5.205.3.5 is_complex

`bool MHA_AC::ac2matrix_helper_t::is_complex`

5.205.3.6 acvar

`comm_var_t MHA_AC::ac2matrix_helper_t::acvar [protected]`

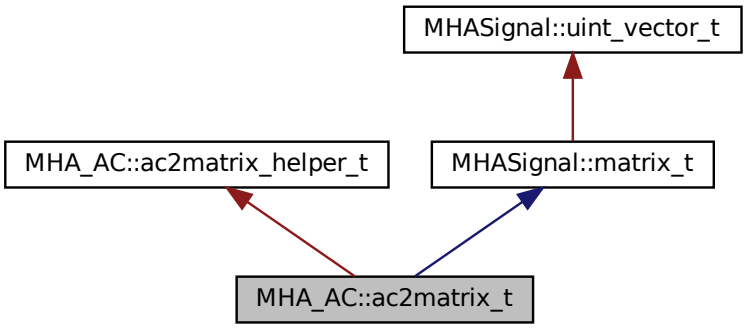
The documentation for this class was generated from the following files:

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.206 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

5.206.1 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. 845) function is real-time safe.

5.206.2 Constructor & Destructor Documentation

5.206.2.1 ac2matrix_t() `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

5.206.3 Member Function Documentation

5.206.3.1 update() `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.

5.206.3.2 getname() `const std::string& MHA_AC::ac2matrix_t::getname ( ) const [inline]`

Return name of AC variable/matrix.

5.206.3.3 getusername() `const std::string& MHA_AC::ac2matrix_t::getusername ( ) const [inline]`

Return user specified name of AC variable/matrix.

5.206.3.4 insert() `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.hh**
- **mha_algo_comm.cpp**

5.207 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**

5.207.1 Detailed Description

Copy all or a subset of all numeric AC variables into an array of matrixes.

5.207.2 Constructor & Destructor Documentation

5.207.2.1 acspace2matrix_t() [1/2] 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.

5.207.2.2 acspace2matrix_t() [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.

5.207.2.3 ~acspace2matrix_t() MHA_AC::acspace2matrix_t::~acspace2matrix_t ( )

5.207.3 Member Function Documentation

5.207.3.1 operator=()

```
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.

5.207.3.2 operator[]() [1/2]

```
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. 849)-1.

Return values

Reference	to matrix.

5.207.3.3 operator[]() [2/2]

```
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. 849)-1.

Return values

Constant	reference to matrix.

5.207.3.4 update() `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.

5.207.3.5 size() `unsigned int MHA_AC::acspace2matrix_t::size ( ) const [inline]`

Number of matrixes in AC space.

5.207.3.6 frame() `unsigned int MHA_AC::acspace2matrix_t::frame ( ) const [inline]`

Actual frame number.

5.207.3.7 insert() `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

5.207.4 Member Data Documentation

5.207.4.1 len `unsigned int MHA_AC::acspace2matrix_t::len [private]`

5.207.4.2 data `MHA_AC::ac2matrix_t** MHA_AC::acspace2matrix_t::data [private]`

5.207.4.3 frameno `unsigned int MHA_AC::acspace2matrix_t::frameno [private]`

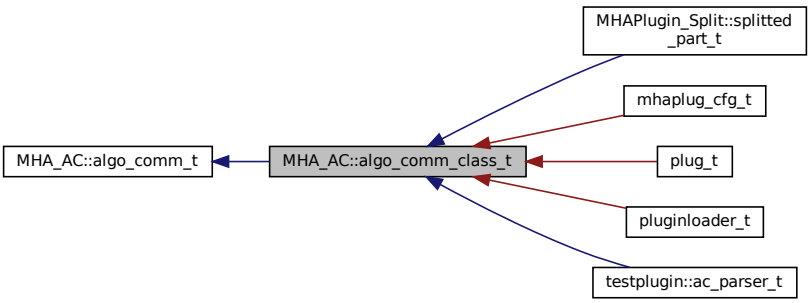
The documentation for this class was generated from the following files:

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.208 MHA_AC::algo_comm_class_t Class Reference

AC variable space implementation.

Inheritance diagram for MHA_AC::algo_comm_class_t:



Public Member Functions

- void **insert_var** (const std::string &name, **comm_var_t** cv) override
- void **insert_var_int** (const std::string &name, int *ptr) override
- void **insert_var_vfloat** (const std::string &name, std::vector< float > &vec) override
- void **insert_var_float** (const std::string &name, float *ptr) override
- void **insert_var_double** (const std::string &name, double *ptr) override
- void **remove_var** (const std::string &name) override
- void **remove_ref** (void *addr) override
- bool **is_var** (const std::string &name) const override
- **comm_var_t** **get_var** (const std::string &name) const override
- int **get_var_int** (const std::string &name) const override
- float **get_var_float** (const std::string &name) const override
- double **get_var_double** (const std::string &name) const override
- const std::vector< std::string > & **get_entries** () const override
- size_t **size** () const override
- virtual void **set_prepared** (bool prepared)

  The provider of this AC space must set the AC space to prepared at the end of its own prepare() operation and to not prepared at the beginning of its own release() operation.

Private Attributes

- **comm_var_map_t** **vars**

  Storage.

5.208.1 Detailed Description

AC variable space implementation.

5.208.2 Member Function Documentation

5.208.2.1 insert_var()

```
void MHA_AC::algo_comm_class_t::insert_var (
    const std::string & name,
    comm_var_t cv ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 856).

5.208.2.2 insert_var_int()

```
void MHA_AC::algo_comm_class_t::insert_var_int (
    const std::string & name,
    int * ptr ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 856).

5.208.2.3 insert_var_vfloat()

```
void MHA_AC::algo_comm_class_t::insert_var_vfloat (
    const std::string & name,
    std::vector< float > & vec ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 857).

5.208.2.4 insert_var_float()

```
void MHA_AC::algo_comm_class_t::insert_var_float (
    const std::string & name,
    float * ptr ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 858).

5.208.2.5 insert_var_double()

```
void MHA_AC::algo_comm_class_t::insert_var_double (
    const std::string & name,
    double * ptr ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 859).

5.208.2.6 remove_var()

```
void MHA_AC::algo_comm_class_t::remove_var (
    const std::string & name ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 859).

5.208.2.7 remove_ref()

```
void MHA_AC::algo_comm_class_t::remove_ref (
    void * addr ) [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 860).

5.208.2.8 is_var()

```
bool MHA_AC::algo_comm_class_t::is_var (
        const std::string & name ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 860).

5.208.2.9 get_var()

```
MHA_AC::comm_var_t MHA_AC::algo_comm_class_t::get_var (
        const std::string & name ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 861).

5.208.2.10 get_var_int()

```
int MHA_AC::algo_comm_class_t::get_var_int (
        const std::string & name ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 861).

5.208.2.11 get_var_float()

```
float MHA_AC::algo_comm_class_t::get_var_float (
        const std::string & name ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 862).

5.208.2.12 get_var_double()

```
double MHA_AC::algo_comm_class_t::get_var_double (
        const std::string & name ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 863).

5.208.2.13 get_entries()

```
const std::vector< std::string > & MHA_AC::algo_comm_class_t::get_entries ( ) const  [override], [virtual]
```

Implements **MHA_AC::algo_comm_t** (p. 863).

5.208.2.14 size() `size_t MHA_AC::algo_comm_class_t::size ( ) const [override], [virtual]`

Implements **MHA_AC::algo_comm_t** (p. 863).

5.208.2.15 set_prepared() `void MHA_AC::algo_comm_class_t::set_prepared ( bool prepared ) [virtual]`

The provider of this AC space must set the AC space to prepared at the end of its own prepare() operation and to not prepared at the beginning of its own release() operation.

5.208.3 Member Data Documentation

5.208.3.1 vars `comm_var_map_t MHA_AC::algo_comm_class_t::vars [private]`

Storage.

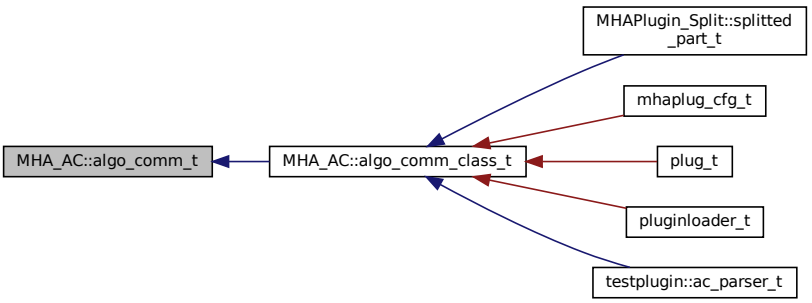
The documentation for this class was generated from the following files:

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.209 MHA_AC::algo_comm_t Class Reference

Algorithm communication variable space interface.

Inheritance diagram for MHA_AC::algo_comm_t:



Public Member Functions

- virtual ∼**algo_comm_t** ()=default
- virtual void **insert_var** (const std::string &name, **comm_var_t** cv)=0

  Interacts with AC space storage to create or replace an AC variable.
- virtual void **insert_var_int** (const std::string &name, int ∗ptr)=0

  Convenience method for inserting or replacing a scalar integer AC variable into AC space.
- virtual void **insert_var_vfloat** (const std::string &name, std::vector< float > &vec)=0

  Convenience function for inserting or replacing a vector of floats as an AC variable into the AC space.
- virtual void **insert_var_float** (const std::string &name, float ∗ptr)=0

  Convenience method for inserting or replacing a scalar float AC variable into AC space.
- virtual void **insert_var_double** (const std::string &name, double ∗ptr)=0

  Convenience method for inserting or replacing a scalar double AC variable into AC space.
- virtual void **remove_var** (const std::string &name)=0

  Remove an AC variable from AC space by name.
- virtual void **remove_ref** (void ∗addr)=0

  Remove all AC variables from AC space that point to the given memory address.
- virtual bool **is_var** (const std::string &name) const =0

  Interacts with AC space storage to check if an AC variable with the given name exists.
- virtual **comm_var_t** **get_var** (const std::string &name) const =0

  Interacts with AC space storage to retrieve the metadata for an AC variable with the given name.
- virtual int **get_var_int** (const std::string &name) const =0

  Convenience method for retrieving a scalar integer AC variable from AC space.
- virtual float **get_var_float** (const std::string &name) const =0

  Convenience method for retrieving a scalar float AC variable from AC space.
- virtual double **get_var_double** (const std::string &name) const =0

  Convenience method for retrieving a scalar double AC variable from AC space.
- virtual const std::vector< std::string > & **get_entries** () const =0
- virtual size_t **size** () const =0

  Interacts with AC space storage to return the number of AC variables currently stored in the AC space.

5.209.1 Detailed Description

Algorithm communication variable space interface.

5.209.2 Constructor & Destructor Documentation

5.209.2.1 ~algo_comm_t() `virtual MHA_AC::algo_comm_t::~algo_comm_t ( ) [virtual], [default]`

5.209.3 Member Function Documentation

5.209.3.1 insert_var() `virtual void MHA_AC::algo_comm_t::insert_var ( const std::string & name, comm_var_t cv ) [pure virtual]`

Interacts with AC space storage to create or replace an AC variable.

When the AC space is already prepared, only replacing existing variables is permitted, not creating new ones. An AC space becomes prepared only after the plugin's prepare() method has finished executing, and becomes unprepared again before the plugin's release() method starts executing. During signal processing, which starts after all plugins have executed their prepare() methods and terminates before any plugin executes its release() method, the AC space stays in prepared state.

Plugins calling this method must ensure that it is called directly or indirectly from every single invocation of their prepare() and their process() methods for each AC variable that they choose to publish. During prepare(), plugins must decide which AC variables to publish and stick to this decision until the next invocation of release().

Parameters

name	Name of the AC variable to create or to replace. May not be empty. Must not contain space character.
cv	Descriptor of AC variable. The `data` pointer of this struct must remain valid until at least the next invocation of the calling plugin's process() or release() method, the other fields must correctly describe the data.

Exceptions

MHA_Error *(p. 906)*	If the AC space is already prepared and no AC variable with name `name` exists yet.
MHA_Error *(p. 906)*	if name is emtpy or contains space.

Implemented in **MHA_AC::algo_comm_class_t** (p. 851).

5.209.3.2 insert_var_int() `virtual void MHA_AC::algo_comm_t::insert_var_int ( const std::string & name, int * ptr ) [pure virtual]`

Convenience method for inserting or replacing a scalar integer AC variable into AC space.

Creates suitable **comm_var_t** (p. 868) and forwards to **insert_var()** (p. 856), therefore see also the documentation of **insert_var()** (p. 856). When the AC space is already prepared, only replacing existing variables is permitted, not creating new ones.

Parameters

name	Name of the AC variable to create or to replace. May not be empty. Must not contain space character.
ptr	Pointer to an int variable owned by the calling plugin. The pointer must remain valid until at least the next invocation of the calling plugin's process() or release() method.

Exceptions

MHA_Error *(p. 906)*	If the AC space is already prepared and no AC variable with name name exists yet.
MHA_Error *(p. 906)*	if name is emtpy or contains space.

Implemented in **MHA_AC::algo_comm_class_t** (p. 851).

5.209.3.3 insert_var_vfloat() `virtual void MHA_AC::algo_comm_t::insert_var_vfloat ( const std::string & name, std::vector< float > & vec ) [pure virtual]`

Convenience function for inserting or replacing a vector of floats as an AC variable into the AC space.

Creates suitable **comm_var_t** (p. 868) and forwards to **insert_var()** (p. 856), therefore see also the documentation of **insert_var()** (p. 856). When the AC space is prepared, only replacing existing variables is permitted, not creating new ones.

Parameters

name	Name of the AC variable to create or to replace. May not be empty. Must not contain space character.

Parameters

vec	Reference to a float vector owned by the calling plugin. The internal storage of this vector must remain valid until at least the next invocation of the calling plugin's process() or release() method. No methods that could cause iterator invalidation may be called on this vector until at least then.

Exceptions

MHA_Error *(p. 906)*	If the AC space is already prepared and no AC variable with name `name` exists yet.
MHA_Error *(p. 906)*	if name is emtpy or contains space.
MHA_Error *(p. 906)*	if vec contains more elements than can be represented by **comm_var_t::num_entries** (p. 869

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.4 insert_var_float()

```
virtual void MHA_AC::algo_comm_t::insert_var_float (
    const std::string & name,
    float * ptr ) [pure virtual]
```

Convenience method for inserting or replacing a scalar float AC variable into AC space.

Creates suitable **comm_var_t** (p. 868) and forwards to **insert_var()** (p. 856), therefore see also the documentation of **insert_var()** (p. 856). When the AC space is prepared, only replacing existing variables is permitted, not creating new ones.

Parameters

name	Name of the AC variable to create or to replace. May not be empty. Must not contain space character.
ptr	Pointer to a float variable owned by the calling plugin. The pointer must remain valid until at least the next invocation of the calling plugin's process() or release() method.

Exceptions

MHA_Error *(p. 906)*	If the AC space is already prepared and no AC variable with name `name` exists yet.

Exceptions

MHA_Error *(p. 906)*	if name is emtpy or contains space.

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.5 insert_var_double()

```
virtual void MHA_AC::algo_comm_t::insert_var_double (
    const std::string & name,
    double * ptr ) [pure virtual]
```

Convenience method for inserting or replacing a scalar double AC variable into AC space.

Creates suitable **comm_var_t** (p. 868) and forwards to **insert_var()** (p. 856), therefore see also the documentation of **insert_var()** (p. 856). When the AC space is prepared, only replacing existing variables is permitted, not creating new ones.

Parameters

name	Name of the AC variable to create or to replace. May not be empty. Must not contain space character.
ptr	Pointer to a double variable owned by the calling plugin. The pointer must remain valid until at least the next invocation of the calling plugin's process() or release() method.

Exceptions

MHA_Error *(p. 906)*	If the AC space is already prepared and no AC variable with name `name` exists yet.
MHA_Error *(p. 906)*	if name is emtpy or contains space.

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.6 remove_var() `virtual void MHA_AC::algo_comm_t::remove_var ( const std::string & name ) [pure virtual]`

Remove an AC variable from AC space by name.

Only permitted when AC space is not prepared. Trying to remove a non-existing AC variable from AC space is not by itself an error. Calling this method while the AC space is prepared is an error, because it is not permitted to remove AC variables during signal processing, only to update them.

Parameters

name	Name of the AC variable to remove.

Exceptions

MHA_Error *(p. 906)*	if called while prepared, and then regardless of w an AC variable with name `name` exists or not.

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.7 remove_ref() `virtual void MHA_AC::algo_comm_t::remove_ref ( void * addr ) [pure virtual]`

Remove all AC variables from AC space that point to the given memory address.

Only permitted when AC space is not prepared. While not prepared, it is not an error if no AC variables or if multiple AC variables actually point to addr. All matching variables are removed.

Parameters

addr	Memory address where the data of the AC variable(s) to remove is or was stored.

Exceptions

MHA_Error *(p. 906)*	if called while prepared, regardless whether any A variables currently point to addr or not.

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.8 is_var() `virtual bool MHA_AC::algo_comm_t::is_var ( const std::string & name ) const [pure virtual]`

Interacts with AC space storage to check if an AC variable with the given name exists.

Parameters

name	Name of the AC variable to check.

Implemented in **MHA_AC::algo_comm_class_t** (p. 852).

5.209.3.9 get_var() `virtual comm_var_t MHA_AC::algo_comm_t::get_var ( const std::string & name ) const [pure virtual]`

Interacts with AC space storage to retrieve the metadata for an AC variable with the given name.

Parameters

name	Name of the AC variable to retrieve.

Returns

a struct describing the AC variable's data type, memory location and size.

Exceptions

MHA_Error *(p. 906)*	if no AC variable with the given name exists.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

5.209.3.10 get_var_int() `virtual int MHA_AC::algo_comm_t::get_var_int ( const std::string & name ) const [pure virtual]`

Convenience method for retrieving a scalar integer AC variable from AC space.

Checks data type and size.

Parameters

name	Name of the AC variable to read.

Returns

Value of the AC varible.

Exceptions

MHA_Error *(p. 906)*	if no AC variable with the given name exists.
MHA_Error *(p. 906)*	if AC variable `name` is not an integer or not a scalar.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

5.209.3.11 get_var_float() `virtual float MHA_AC::algo_comm_t::get_var_float ( const std::string & name ) const [pure virtual]`

Convenience method for retrieving a scalar float AC variable from AC space.

Checks data type and size.

Parameters

name	Name of the AC variable to read.

Returns

Value of the AC varible.

Exceptions

MHA_Error *(p. 906)*	if no AC variable with the given name exists.
MHA_Error *(p. 906)*	if AC variable `name` is not a float or not a scalar.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

5.209.3.12 get_var_double()

`virtual double MHA_AC::algo_comm_t::get_var_double ( const std::string & name ) const [pure virtual]`

Convenience method for retrieving a scalar double AC variable from AC space.

Checks data type and size.

Parameters

name	Name of the AC variable to read.

Returns

Value of the AC varible.

Exceptions

MHA_Error *(p. 906)*	if no AC variable with the given name exists.
MHA_Error *(p. 906)*	if AC variable `name` is not a double or not a scalar.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

5.209.3.13 get_entries()

`virtual const std::vector<std::string>& MHA_AC::algo_comm_t::get_entries ( ) const [pure virtual]`

Returns

a list of the names of all existing AC variables.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

5.209.3.14 size() `virtual size_t MHA_AC::algo_comm_t::size ( ) const [pure virtual]`

Interacts with AC space storage to return the number of AC variables currently stored in the AC space.

Always permitted.

Implemented in **MHA_AC::algo_comm_class_t** (p. 853).

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

- **mha_algo_comm.hh**

5.210 MHA_AC::comm_var_map_t Class Reference

Storage class for the AC variable space.

Public Member Functions

- bool **has_key** (const std::string &name) const
  Query the map if some AC variable name is present in the AC space.
- void **insert** (const std::string &name, const **comm_var_t** &var)
  Create or replace variable.
- void **erase_by_name** (const std::string &name)
  Remove variable.
- void **erase_by_pointer** (void *ptr)
  Find variables that point to the given address.
- const **comm_var_t** & **retrieve** (const std::string &name) const
  *Get the **comm_var_t** (p. 868) of an existing variable.*
- const std::vector< std::string > & **get_entries** () const
- size_t **size** () const

Public Attributes

- bool **is_prepared** = {false}
  is_prepared stores whether the provider of the AC space has entered MHA state "prepared" or not.

Private Member Functions

- void **update_entries** ()
  *Update the member variable **entries** (p. 868) because an AC variable has been inserted or removed.*

Private Attributes

- std::map< std::string, **comm_var_t** > **map**
  The std::map used for organizing the AC space.
- std::vector< std::string > **entries**
  A list containing the names of all AC variables.

5.210.1 Detailed Description

Storage class for the AC variable space.

Uses an std::map for associating AC variable names with AC variable metadata (**comm_var_t** (p. 868)). Acts as a delegator for the std::map storage. Allows operations that may require memory allocations/deallocations only when is_prepared == false.

5.210.2 Member Function Documentation

5.210.2.1 update_entries() `void MHA_AC::comm_var_map_t::update_entries ( ) [private]`

Update the member variable **entries** (p. 868) because an AC variable has been inserted or removed.

Only permitted if is_prepared == false.

5.210.2.2 has_key() `bool MHA_AC::comm_var_map_t::has_key ( const std::string & name ) const [inline]`

Query the map if some AC variable name is present in the AC space.

Parameters

name	Name of AC variable to check.

Returns

true if the variable is present in the AC space.
false if no variable with this name exists in the AC space.

5.210.2.3 insert()

```
void MHA_AC::comm_var_map_t::insert (
    const std::string & name,
    const  comm_var_t & var )
```

Create or replace variable.

Creating is only permitted if is_prepared == false.

Parameters

name	Name of the AC variable to create or update. May not be empty. Must not contain space character.
var	Metadata of the AC variable.

Exceptions

MHA_Error *(p. 906)*	if asked to create in prepared state.
MHA_Error *(p. 906)*	if name is emtpy or contains space.

5.210.2.4 erase_by_name()

```
void MHA_AC::comm_var_map_t::erase_by_name (
    const std::string & name )
```

Remove variable.

Only permitted if is_prepared == false.

Parameters

name	Name of the AC variable to remove.

Exceptions

MHA_Error *(p. 906)*	if called while prepared.

5.210.2.5 erase_by_pointer() `void MHA_AC::comm_var_map_t::erase_by_pointer ( void * ptr )`

Find variables that point to the given address.

Erase all that are found. It is not an error if no variable points there. Only permitted if is_↩ prepared == false. @ptr Pointer to memory where the variables data is stored.

Exceptions

MHA_Error *(p. 906)*	if called while prepared.

5.210.2.6 retrieve() `const MHA_AC::comm_var_t & MHA_AC::comm_var_map_t::retrieve ( const std::string & name ) const`

Get the **comm_var_t** (p. 868) of an existing variable.

Parameters

name	The name of the AC variable.

Exceptions

MHA_Error *(p. 906)*	if no such variable exists in the AC space.

5.210.2.7 get_entries() `const std::vector< std::string > & MHA_AC::comm_var_map_t↩ ::get_entries ( ) const`

Returns

A list of names of all AC variables in this AC space.

5.210.2.8 size() `size_t MHA_AC::comm_var_map_t::size ( ) const [inline]`

Returns

number of stored AC variables

5.210.3 Member Data Documentation

5.210.3.1 map `std::map<std::string, comm_var_t> MHA_AC::comm_var_map_t::map [private]`

The std::map used for organizing the AC space.

5.210.3.2 entries `std::vector<std::string> MHA_AC::comm_var_map_t::entries [private]`

A list containing the names of all AC variables.

5.210.3.3 is_prepared `bool MHA_AC::comm_var_map_t::is_prepared = {false}`

is_prepared stores whether the provider of the AC space has entered MHA state "prepared" or not.

Operations on `map` that require memory allocations or deallocations are only allowed when not prepared. Needs to be set by the containing
algo_comm_class_t (p. 850) AC space instance.

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

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.211 MHA_AC::comm_var_t Struct Reference

Algorithm communication variable structure.

Public Attributes

- unsigned int **data_type**

  Type of data.
- unsigned int **num_entries**

  *The number of elements of data type **data_type** (p. 869) stored at the pointer address **data** (p. 870).*
- unsigned int **stride**

  This data member can be used to describe the extent of one dimension if the data should be interpreted as a two-dimensional matrix.
- void * **data**

  `data` *is a pointer to where the AC variable's* `data` *is stored in memory.*

5.211.1 Detailed Description

Algorithm communication variable structure.

Algorithm communication variables (AC variables) are described by objects of this type. AC variables can be published to the algorithm variable space with the **algo_comm_class_t::insert_var** (p. 851) method so that other plugins can read and modify their values.

5.211.2 Member Data Documentation

5.211.2.1 data_type `unsigned int MHA_AC::comm_var_t::data_type`

Type of data.

`data_type` can be one of the predefined types or any user defined type. The pre-defined types are:

- **MHA_AC_CHAR** (p. 1796)
- **MHA_AC_INT** (p. 1796)
- **MHA_AC_MHAREAL** (p. 1796)
- **MHA_AC_FLOAT** (p. 1796)
- **MHA_AC_DOUBLE** (p. 1796)
- **MHA_AC_MHACOMPLEX** (p. 1796)
- or any user defined type with a value >= **MHA_AC_USER** (p. 1796)

5.211.2.2 num_entries `unsigned int MHA_AC::comm_var_t::num_entries`

The number of elements of data type **data_type** (p. 869) stored at the pointer address **data** (p. 870).

5.211.2.3 stride `unsigned int MHA_AC::comm_var_t::stride`

This data member can be used to describe the extent of one dimension if the data should be interpreted as a two-dimensional matrix.

The extent in the other dimension then is **num_entries** (p. 869) / **stride** (p. 870). When downstream plugins could interpret the AC variable as a (possibly 1x1) matrix, then it should be avoided to set **stride** (p. 870) to 0.

5.211.2.4 data `void* MHA_AC::comm_var_t::data`

`data` is a pointer to where the AC variable's `data` is stored in memory.

This pointer has to be valid for the lifetime of this AC variable.

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

- **mha.hh**

5.212 MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE > Class Template Reference

Template for convenience classes for inserting a numeric scalar into the AC space.

Public Member Functions

- **scalar_t** (**algo_comm_t** & **ac**, const std::string & **name**, numeric_t val=0, bool insert↩ _now=true)
  Initialize memory and metadata of the AC variable.
- ~**scalar_t** ()
  Destroy the AC variable: deallocate its memory.
- void **insert** ()
  Insert or re-insert AC variable into AC space.
- void **remove** ()
  Remove the AC variable by reference from the AC variable space.

Public Attributes

- numeric_t **data**
  Numeric value of this AC variable.

Private Attributes

- **algo_comm_t** & **ac**
  AC variable space.
- const std::string **name**
  Name of this AC variable in the AC variable space.
- const bool **remove_during_destructor**
  flag whether to remove from AC variable space in destructor.

5.212.1 Detailed Description

template<typename numeric_t, unsigned int MHA_AC_TYPECODE>
class MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE >

Template for convenience classes for inserting a numeric scalar into the AC space.

5.212.2 Constructor & Destructor Documentation

5.212.2.1 scalar_t() template<typename numeric_t , unsigned int MHA_AC_TYPECODE>

```
MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE >::  scalar_t (
             algo_comm_t & ac,
            const std::string & name,
            numeric_t val = 0,
            bool insert_now = true )  [inline]
```

Initialize memory and metadata of the AC variable.

Parameters

ac	AC handle
name	Name of variable in AC space
val	Initial value

Parameters

insert_now	If true, then the constructor inserts the new variable into the AC space, and the destructor will remove the variable from AC space when it executes.

5.212.2.2 ~scalar_t() template<typename numeric_t , unsigned int MHA_AC_TYPECODE> MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE >::~ scalar_t ( ) [inline]

Destroy the AC variable: deallocate its memory.

If the constructor parameter insert_now was true, then the destruc- tor removes the AC variable from AC space when it executes.

5.212.3 Member Function Documentation

5.212.3.1 insert() template<typename numeric_t , unsigned int MHA_AC_TYPECODE> void MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE >::insert ( ) [inline]

Insert or re-insert AC variable into AC space.

Plugins should call this method from their prepare() and process() functions.

5.212.3.2 remove() template<typename numeric_t , unsigned int MHA_AC_TYPECODE> void MHA_AC::scalar_t< numeric_t, MHA_AC_TYPECODE >::remove ( ) [inline]

Remove the AC variable by reference from the AC variable space.

Plugins may call this method only from their prepare(), release() methods or their plugin destructor. It is not necessary to remove the AC variable from AC space at all if either another AC variable with the same name has replaced this variable before this variable is destroyed, or if no plugin will access this variable between its destruction and either its replacement or the MHA exit.

5.212.4 Member Data Documentation

5.212.4.1 data

`template<typename numeric_t , unsigned int MHA_AC_TYPECODE>`
`numeric_t` **MHA_AC::scalar_t**< numeric_t, MHA_AC_TYPECODE >::data

Numeric value of this AC variable.

5.212.4.2 ac

`template<typename numeric_t , unsigned int MHA_AC_TYPECODE>`
algo_comm_t& **MHA_AC::scalar_t**< numeric_t, MHA_AC_TYPECODE >::ac [private]

AC variable space.

5.212.4.3 name

`template<typename numeric_t , unsigned int MHA_AC_TYPECODE>`
`const std::string` **MHA_AC::scalar_t**< numeric_t, MHA_AC_TYPECODE >::name [private]

Name of this AC variable in the AC variable space.

5.212.4.4 remove_during_destructor

`template<typename numeric_t , unsigned int MHA_AC_TYPECODE>`
`const bool` **MHA_AC::scalar_t**< numeric_t, MHA_AC_TYPECODE >::remove_during_destructor [private]

flag whether to remove from AC variable space in destructor.

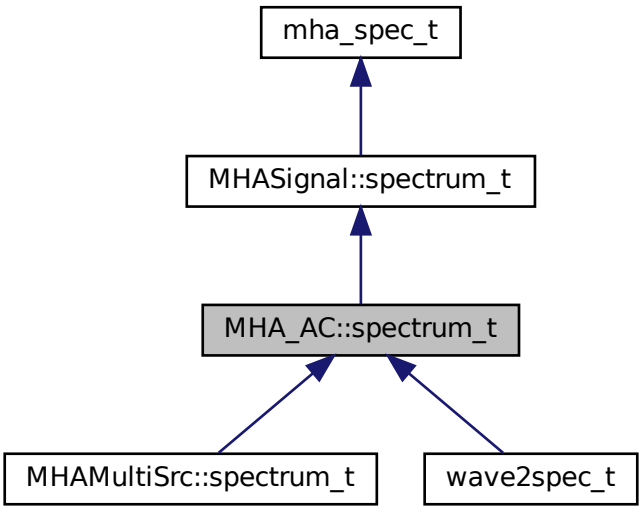
The documentation for this class was generated from the following file:

- **mha_algo_comm.hh**

5.213 MHA_AC::spectrum_t Class Reference

Convenience class for inserting a spectrum into the AC space.

Inheritance diagram for MHA_AC::spectrum_t:



Public Member Functions

- **spectrum_t** (**algo_comm_t** & **ac**, const std::string & **name**, unsigned int bins, unsigned int **channels**, bool insert_now)
  Initialize memory and metadata of the AC variable.
- ~**spectrum_t** ()
  Destroy the AC variable: deallocate its memory.
- void **insert** ()
  Insert or re-insert AC variable into AC space.
- void **remove** ()
  Remove the AC variable by reference from the AC variable space.

Protected Attributes

- **algo_comm_t** & **ac**
  AC variable space.
- const std::string **name**
  Name of this AC variable in the AC variable space.
- const bool **remove_during_destructor**
  flag whether to remove from AC variable space in destructor.

Additional Inherited Members

5.213.1 Detailed Description

Convenience class for inserting a spectrum into the AC space.

In MHA, spectra are stored non-interleaved: First all bins of the first channel are stored, then all bins of the second channel, etc.

The stride of the AC variable is set to the number of stored bins, which is equal to floor(fftlen/2)+1 in MHA (negative frequency bins are not stored).

5.213.2 Constructor & Destructor Documentation

5.213.2.1 spectrum_t()

```
MHA_AC::spectrum_t::spectrum_t (
    algo_comm_t & ac,
    const std::string & name,
    unsigned int bins,
    unsigned int channels,
    bool insert_now )
```

Initialize memory and metadata of the AC variable.

All spectral bins are initially set to 0.

Parameters

ac	AC handle
name	Name of variable in AC space
bins	Number of FFT bins per channel in the **spectrum_t** (p. 874) class
channels	Number of audio channels in the **spectrum_t** (p. 874) class
insert_now	If true, then the constructor inserts the new variable into the AC space, and the destructor will remove the variable from AC space when it executes.

5.213.2.2 ~spectrum_t() `MHA_AC::spectrum_t::~spectrum_t ( void ) [virtual]`

Destroy the AC variable: deallocate its memory.

If the constructor parameter insert_now was true, then the destruc- tor removes the AC variable from AC space when it executes.

Reimplemented from **MHASignal::spectrum_t** (p. 1401).

5.213.3 Member Function Documentation

5.213.3.1 insert() `void MHA_AC::spectrum_t::insert ( )`

Insert or re-insert AC variable into AC space.

Plugins should call this method from their prepare() and process() functions.

5.213.3.2 remove() `void MHA_AC::spectrum_t::remove ( )`

Remove the AC variable by reference from the AC variable space.

Plugins may call this method only from their prepare(), release() methods or their plugin destructor. It is not necessary to remove the AC variable from AC space at all if either another AC variable with the same name has replaced this variable before this variable is destroyed, or if no plugin will access this variable between its destruction and either its replacement or the MHA exit.

5.213.4 Member Data Documentation

5.213.4.1 ac `algo_comm_t& MHA_AC::spectrum_t::ac [protected]`

AC variable space.

5.213.4.2 name `const std::string MHA_AC::spectrum_t::name [protected]`

Name of this AC variable in the AC variable space.

5.213.4.3 remove_during_destructor `const bool MHA_AC::spectrum_t::remove_during↩_destructor [protected]`

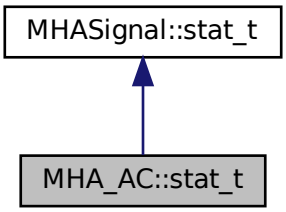
flag whether to remove from AC variable space in destructor.

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

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.214 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**

5.214.1 Constructor & Destructor Documentation

5.214.1.1 stat_t() 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 )
```

5.214.2 Member Function Documentation

5.214.2.1 update() void MHA_AC::stat_t::update ( )

5.214.2.2 insert() void MHA_AC::stat_t::insert ( )

5.214.3 Member Data Documentation

5.214.3.1 mean MHA_AC::waveform_t MHA_AC::stat_t::mean [private]

5.214.3.2 std MHA_AC::waveform_t MHA_AC::stat_t::std [private]

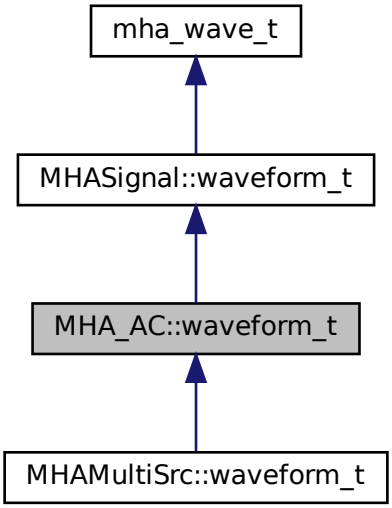
The documentation for this class was generated from the following files:

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.215 MHA_AC::waveform_t Class Reference

Convenience class for inserting a waveform (a block of time-domain audio signal) into the AC space.

Inheritance diagram for MHA_AC::waveform_t:



Public Member Functions

- **waveform_t** (**algo_comm_t** & **ac**, const std::string & **name**, unsigned int frames, unsigned int **channels**, bool insert_now)
  Initialize memory and metadata of the AC variable.
- ~**waveform_t** ()
  Destroy the AC variable: deallocate its memory.
- void **insert** ()
  Insert or re-insert AC variable into AC space.
- void **remove** ()
  Remove the AC variable by reference from the AC variable space.

Protected Attributes

- **algo_comm_t** & **ac**
  AC variable space.
- const std::string **name**
  Name of this AC variable in the AC variable space.
- const bool **remove_during_destructor**
  flag whether to remove from AC variable space in destructor.

Additional Inherited Members

5.215.1 Detailed Description

Convenience class for inserting a waveform (a block of time-domain audio signal) into the AC space.

In MHA, waveforms are stored interleaved: The first sample of the first is followed by the first samples of all other channels before the second sample of the first sample is stored, etc.

The stride of the AC variable is set to the number of audio channels.

5.215.2 Constructor & Destructor Documentation

5.215.2.1 waveform_t()

```
MHA_AC::waveform_t::waveform_t (
    algo_comm_t & ac,
  const std::string & name,
  unsigned int frames,
  unsigned int channels,
  bool insert_now )
```

Initialize memory and metadata of the AC variable.

All audio samples are initially set to 0.

Parameters

ac	AC handle
name	Name of variable in AC space
frames	Number of samples per channel in the **waveform_t** (p. 879) class
channels	Number of audio channels in the **waveform_t** (p. 879) class
insert_now	If true, then the constructor inserts the new variable into the AC space, and the destructor will remove the variable from AC space when it executes.

5.215.2.2 ~waveform_t() `MHA_AC::waveform_t::~waveform_t ( void ) [virtual]`

Destroy the AC variable: deallocate its memory.

If the constructor parameter insert_now was true, then the destruc- tor removes the AC variable from AC space when it executes.

Reimplemented from **MHASignal::waveform_t** (p. 1417).

5.215.3 Member Function Documentation

5.215.3.1 insert() `void MHA_AC::waveform_t::insert ( )`

Insert or re-insert AC variable into AC space.

Plugins should call this method from their prepare() and process() functions.

5.215.3.2 remove() `void MHA_AC::waveform_t::remove ( )`

Remove the AC variable by reference from the AC variable space.

Plugins may call this method only from their prepare(), release() methods or their plugin destructor. It is not necessary to remove the AC variable from AC space at all if either another AC variable with the same name has replaced this variable before this variable is destroyed, or if no plugin will access this variable between its destruction and either its replacement or the MHA exit.

5.215.4 Member Data Documentation

5.215.4.1 ac `algo_comm_t& MHA_AC::waveform_t::ac [protected]`

AC variable space.

5.215.4.2 name `const std::string MHA_AC::waveform_t::name [protected]`

Name of this AC variable in the AC variable space.

5.215.4.3 remove_during_destructor `const bool MHA_AC::waveform_t::remove_during_destructor [protected]`

flag whether to remove from AC variable space in destructor.

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

- **mha_algo_comm.hh**
- **mha_algo_comm.cpp**

5.216 mha_audio_descriptor_t Struct Reference

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 996)).

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.

5.216.1 Detailed Description

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 996)).

5.216.2 Member Data Documentation

5.216.2.1 n_samples `unsigned int mha_audio_descriptor_t::n_samples`

Number of samples.

5.216.2.2 n_channels `unsigned int mha_audio_descriptor_t::n_channels`

Number of audio channels.

5.216.2.3 n_freqs `unsigned int mha_audio_descriptor_t::n_freqs`

Number of frequency bands.

5.216.2.4 is_complex `unsigned int mha_audio_descriptor_t::is_complex`

Flag about sample type.

5.216.2.5 dt `mha_real_t mha_audio_descriptor_t::dt`

Time distance between samples (only equidistant samples allowed)

5.216.2.6 cf `mha_real_t* mha_audio_descriptor_t::cf`

Center frequencies of frequency bands.

5.216.2.7 chdir `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**

5.217 mha_audio_t Struct Reference

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937)).

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. 883) is unset.*
- **mha_complex_t * cdata**
  *Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 883) is set.*

5.217.1 Detailed Description

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937)).

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. 985) and **mha_spec_t** (p. 937) data pointers into corresponding **mha_audio_t** (p. 884) objects.

5.217.2 Member Data Documentation

5.217.2.1 descriptor `mha_audio_descriptor_t mha_audio_t::descriptor`

Dimension and description of the data.

5.217.2.2 rdata `mha_real_t* mha_audio_t::rdata`

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 883) is unset.

5.217.2.3 cdata `mha_complex_t* mha_audio_t::cdata`

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 883) is set.

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

- **mha.hh**

5.218 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.

5.218.1 Detailed Description

Channel information structure.

5.218.2 Member Data Documentation

5.218.2.1 id `int mha_channel_info_t::id`

channel id

5.218.2.2 idstr `char mha_channel_info_t::idstr[32]`

channel id

5.218.2.3 side `unsigned int mha_channel_info_t::side`

side (left/right)

5.218.2.4 dir **`mha_direction_t`** `mha_channel_info_t::dir`

source direction

5.218.2.5 peaklevel **`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**

5.219 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.

5.219.1 Detailed Description

Type for complex floating point values.

5.219.2 Member Data Documentation

5.219.2.1 re `mha_real_t mha_complex_t::re`

Real part.

5.219.2.2 im `mha_real_t mha_complex_t::im`

Imaginary part.

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

- **mha.hh**

5.220 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. 886) 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]

5.220.1 Detailed Description

Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 886) 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.

5.220.2 Member Data Documentation

5.220.2.1 c `mha_complex_t mha_complex_test_array_t::c[2]`

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

- **mha.hh**

5.221 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.

5.221.1 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.

5.221.2 Member Typedef Documentation

5.221.2.1 value_type `template<class FIFO >`

`typedef FIFO::value_type` **`mha_dblbuf_t`**`< FIFO >::` **`value_type`**

The datatype exchanged by the FIFO and this doublebuffer.

5.221.3 Constructor & Destructor Documentation

5.221.3.1 mha_dblbuf_t() 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.

5.221.3.2 ~mha_dblbuf_t() template<class FIFO >

```
mha_dblbuf_t< FIFO >::~ mha_dblbuf_t  [virtual]
```

5.221.4 Member Function Documentation

5.221.4.1 get_inner_size() template<class FIFO >

```
virtual unsigned  mha_dblbuf_t< FIFO >::get_inner_size ( ) const  [inline], [virtual]
```

5.221.4.2 get_outer_size() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_outer_size ( ) const [inline], [virtual]

5.221.4.3 get_delay() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_delay ( ) const [inline], [virtual]

5.221.4.4 get_fifo_size() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_fifo_size ( ) const [inline], [virtual]

5.221.4.5 get_input_channels() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_input_channels ( ) const [inline], [virtual]

5.221.4.6 get_output_channels() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_output_channels ( ) const [inline], [virtual]

5.221.4.7 get_input_fifo_fill_count() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_input_fifo_fill_count ( ) const [inline], [virtual]

5.221.4.8 get_output_fifo_fill_count() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_output_fifo_fill_count ( ) const [inline], [virtual]

5.221.4.9 get_input_fifo_space() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_input_fifo_space ( ) const [inline], [virtual]

5.221.4.10 get_output_fifo_space() template<class FIFO >

virtual unsigned **mha_dblbuf_t**< FIFO >::get_output_fifo_space ( ) const [inline], [virtual]

5.221.4.11 get_inner_error() template<class FIFO >

virtual **MHA_Error*** **mha_dblbuf_t**< FIFO >::get_inner_error ( ) const [inline], [virtual]

5.221.4.12 provoke_inner_error() template<class FIFO >

void **mha_dblbuf_t**< FIFO >::provoke_inner_error (
const **MHA_Error** & *error*) [virtual]

5.221.4.13 provoke_outer_error() template<class FIFO >

void **mha_dblbuf_t**< FIFO >::provoke_outer_error (
const **MHA_Error** & *error*) [virtual]

5.221.4.14 process() 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.

Parameters

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. 906)*	When count is > outer_size as given to constructor the underlying fifo implementation detects an error.

5.221.4.15 input() `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. 906)*	When the underlying fifo implementation detects an error.

5.221.4.16 output() `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. 906)*	When the underlying fifo implementation detects an error.

5.221.5 Member Data Documentation

5.221.5.1 outer_size `template<class FIFO >`

`unsigned` **`mha_dblbuf_t`**`< FIFO >::outer_size [private]`

The block size used by the outer process.

5.221.5.2 inner_size `template<class FIFO >`

`unsigned` **`mha_dblbuf_t`**`< FIFO >::inner_size [private]`

The block size used by the inner process.

5.221.5.3 delay `template<class FIFO >`

`unsigned` **`mha_dblbuf_t`**`< FIFO >::delay [private]`

The delay introduced by bidirectional buffer size adaptation.

5.221.5.4 fifo_size `template<class FIFO >`

`unsigned` **`mha_dblbuf_t`**`< FIFO >::fifo_size [private]`

The size of each of the FIFOs.

5.221.5.5 input_channels template<class FIFO >

unsigned **mha_dblbuf_t**< FIFO >::input_channels [private]

The number of input channels.

5.221.5.6 output_channels template<class FIFO >

unsigned **mha_dblbuf_t**< FIFO >::output_channels [private]

The number of output channels.

5.221.5.7 input_fifo 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.

5.221.5.8 output_fifo 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.

5.221.5.9 inner_error template<class FIFO >

MHA_Error* **mha_dblbuf_t**< FIFO >::inner_error [private]

Owned copy of exception to be thrown in inner thread.

5.221.5.10 outer_error 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**

5.222 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

5.222.1 Detailed Description

Channel source direction structure.

5.222.2 Member Data Documentation

5.222.2.1 azimuth **mha_real_t** mha_direction_t::azimuth

azimuth in radiants

5.222.2.2 elevation **mha_real_t** mha_direction_t::elevation

elevation in radiants

5.222.2.3 distance **mha_real_t** mha_direction_t::distance

distance in meters

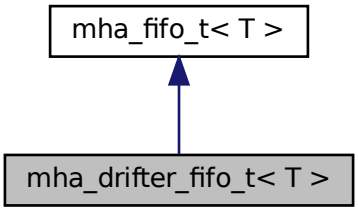
The documentation for this struct was generated from the following file:

- **mha.hh**

5.223 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. 905) 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. 905) 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. 901) or **mha_drifter_fifo_t<T>::write** (p. 900) when their xrun in succession counter exceeds its limit.*
- virtual void **starting** ()
  *Called by **mha_drifter_fifo_t<T>::read** (p. 901) or **mha_drifter_fifo_t<T>::write** (p. 900) when the respective flag (**mha_drifter_fifo_t<T>::reader_started** (p. 903) or **mha_drifter_fifo_t<T>::writer_started** (p. 903)) 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. 903) and **mha_drifter_fifo_t<T>::writer_started** (p. 903) are true, then first **mha_drifter_fifo_t<T>::desired_fill_count** (p. 903) instances of **mha_drifter_fifo_t<T>::null_data** (p. 905) are delivered to the reader.*

Additional Inherited Members

5.223.1 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.

5.223.2 Constructor & Destructor Documentation

5.223.2.1 mha_drifter_fifo_t() [1/2] template<class T >

```
mha_drifter_fifo_t< T >::  mha_drifter_fifo_t (
            unsigned min_fill_count,
            unsigned desired_fill_count,
            unsigned max_fill_count )
```

Create drifter FIFO.

5.223.2.2 mha_drifter_fifo_t() [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.

5.223.3 Member Function Documentation

5.223.3.1 write() template<class T >

```
void  mha_drifter_fifo_t< T >::write (
            const T * data,
            unsigned count )  [virtual]
```

write data to fifo

Sets **writer_started** (p. 903) to true.

When processing has started, i.e. both **reader_started** (p. 903) and **writer_started** (p. 903) 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. 904) exceeds **maximum_writer_xruns_in_succession_before_stop** (p. 904).

Parameters

data	Pointer to source data.
count	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 922).

5.223.3.2 read()

```
template<class T >
void mha_drifter_fifo_t< T >::read (
            T * buf,
            unsigned count ) [virtual]
```

Read data from fifo.

Sets **reader_started** (p. 903) to true.

When processing has started, i.e. both **reader_started** (p. 903) and **writer_started** (p. 903) are true, then read specified ammount of data from the fifo. As long as **startup_zeros** (p. 905) is > 0, **null_data** (p. 905) is delivered to the reader and **startup_zeros** (p. 905) is diminished. Only when **startup_zeros** (p. 905) 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. 903), then only so much data are read that **minimum_fill_count** (p. 903) entries remain in the fifo, the missing data is replaced with **null_data** (p. 905), and the reader xrun counters are increased.

Processing is stopped when **reader_xruns_in_succession** (p. 904) exceeds **maximum_↩ reader_xruns_in_succession_before_stop** (p. 905).

Parameters

buf	Pointer to the target buffer
count	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.223.3.3 get_fill_count() template<class T >

unsigned **mha_drifter_fifo_t**< T >::get_fill_count [virtual]

Return fill_count, adding **mha_drifter_fifo_t<T>::startup_zeros** (p. 905) to the number of samples actually in the fifo's buffer.

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.223.3.4 get_available_space() template<class T >

unsigned **mha_drifter_fifo_t**< T >::get_available_space [virtual]

Return available space, subtracting number of **mha_drifter_fifo_t<T>::startup_zeros** (p. 905) 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. 923).

5.223.3.5 get_des_fill_count() template<class T >

virtual unsigned **mha_drifter_fifo_t**< T >::get_des_fill_count ( ) const [inline], [virtual]

The desired fill count of this fifo.

5.223.3.6 get_min_fill_count() template<class T >

virtual unsigned **mha_drifter_fifo_t**< T >::get_min_fill_count ( ) const [inline], [virtual]

The minimum fill count of this fifo.

5.223.3.7 stop() template<class T >

void **mha_drifter_fifo_t**< T >::stop [virtual]

Called by **mha_drifter_fifo_t<T>::read** (p. 901) or **mha_drifter_fifo_t<T>::write** (p. 900) when their xrun in succession counter exceeds its limit.

Called by **read** (p. 901) or **write** (p. 900) when their xrun in succession counter exceeds its limit.

May also be called explicitly.

5.223.3.8 starting() template<class T >

void **mha_drifter_fifo_t**< T >::starting [virtual]

Called by **mha_drifter_fifo_t<T>::read** (p. 901) or **mha_drifter_fifo_t<T>::write** (p. 900) when the respective flag (**mha_drifter_fifo_t<T>::reader_started** (p. 903) or **mha_drifter_↩ fifo_t<T>::writer_started** (p. 903)) is about to be toggled from false to true.

The fifo's buffer is emptied, this method resets **startup_zeros** (p. 905) to **desired_fill_count** (p. 903), and it also resets **reader_xruns_since_start** (p. 904) and **writer_xruns_since_start** (p. 904) to 0.

5.223.4 Member Data Documentation

5.223.4.1 minimum_fill_count template<class T >

const unsigned **mha_drifter_fifo_t**< T >::minimum_fill_count [private]

The minimum fill count of this fifo.

5.223.4.2 desired_fill_count 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.

5.223.4.3 writer_started template<class T >

bool **mha_drifter_fifo_t**< T >::writer_started [private]

Flag set to true when write is called the first time.

5.223.4.4 reader_started template<class T >

bool **mha_drifter_fifo_t**< T >::reader_started [private]

Flag set to true when read is called for the first time.

5.223.4.5 writer_xruns_total template<class T >

unsigned **mha_drifter_fifo_t**< T >::writer_xruns_total [private]

The number of xruns seen by the writer since object instantiation.

5.223.4.6 reader_xruns_total template<class T >

unsigned **mha_drifter_fifo_t**< T >::reader_xruns_total [private]

The number of xruns seen by the reader since object instantiation.

5.223.4.7 writer_xruns_since_start 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.

5.223.4.8 reader_xruns_since_start 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.

5.223.4.9 writer_xruns_in_succession 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.

5.223.4.10 reader_xruns_in_succession 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.

5.223.4.11 maximum_writer_xruns_in_succession_before_stop

`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.

5.223.4.12 maximum_reader_xruns_in_succession_before_stop

`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.

5.223.4.13 null_data

`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.

5.223.4.14 startup_zeros

`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. 903) and **mha_drifter_fifo_t<T>::writer_started** (p. 903) are true, then first **mha_drifter_fifo_t<T>::desired_fill_count** (p. 903) instances of **mha_drifter_fifo_t<T>::null_data** (p. 905) are delivered to the reader.

These **null_data** (p. 905) instances are not transmitted through the fifo because filling the fifo with enough **null_data** (p. 905) might not be realtime safe and this filling has to be initiated by **starting** (p. 902) or **stop** (p. 902) (this implementation: **starting** (p. 902)) which are be called with realtime constraints.

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

- **mha_fifo.h**

5.224 MHA_Error Class Reference

Error reporting exception class.

Inherits std::exception.

Public Member Functions

- **MHA_Error** (const char *file, int line, const char *fmt,...) __attribute__((__format__(printf
  *Create an instance of a **MHA_Error** (p. 906).*
- **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**

5.224.1 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. 906) class holds source file name, line number and an error message.

5.224.2 Constructor & Destructor Documentation

5.224.2.1 MHA_Error() [1/2]

```
MHA_Error::MHA_Error (
    const char * s_file,
    int l,
    const char * fmt,
    ... )
```

Create an instance of a **MHA_Error** (p. 906).

Parameters

s_file	source file name (**FILE**)
l	source line (**LINE**)
fmt	format string for error message (as in printf)

5.224.2.2 MHA_Error() [2/2]

```
MHA_Error::MHA_Error (
    const MHA_Error & p )
```

5.224.2.3 ~MHA_Error()

```
MHA_Error::~MHA_Error ( ) throw ( )
```

5.224.3 Member Function Documentation

5.224.3.1 operator=()

```
MHA_Error & MHA_Error::operator= (
    const MHA_Error & p )
```

5.224.3.2 get_msg()

```
const char* MHA_Error::get_msg ( ) const [inline]
```

Return the error message without source position.

5.224.3.3 get_longmsg() const char* MHA_Error::get_longmsg ( ) const [inline]

Return the error message with source position.

5.224.3.4 what() const char* MHA_Error::what ( ) const throw ( ) [inline]

overwrite std::execption::what()

5.224.4 Member Data Documentation

5.224.4.1 msg char* MHA_Error::msg [private]

5.224.4.2 longmsg char* MHA_Error::longmsg [private]

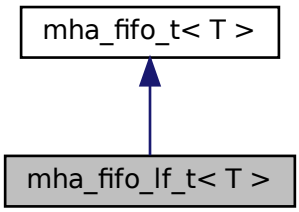
The documentation for this class was generated from the following files:

- **mha_error.hh**
- **mha_error.cpp**

5.225 mha_fifo_lf_t< T > Class Template Reference

A lock-free FIFO class for transferring data from a producer thread to a consumer thread.

Inheritance diagram for mha_fifo_lf_t< T >:



Public Member Functions

- virtual void **write** (const T *data, unsigned count) override
  Write specified ammount of data to the fifo.
- virtual void **read** (T *outbuf, unsigned count) override
  Read data from fifo.
- virtual unsigned **get_fill_count** () const override
  get_fill_count() *(p. 911) must only be called by the reader thread*
- virtual unsigned **get_available_space** () const override
  get_available_space() *(p. 911) must only be called by the writer thread*
- **mha_fifo_lf_t** (unsigned max_fill_count, const T &t=T())
  Create FIFO with fixed buffer size.

Private Attributes

- std::atomic< const T * > **atomic_write_ptr**
  atomic copy of the write_ptr, only modified by the producer thread
- std::atomic< const T * > **atomic_read_ptr**
  atomic copy of the read ptr, only modified by the consumer thread

Additional Inherited Members

5.225.1 Detailed Description

template<class T>
class mha_fifo_lf_t< T >

A lock-free FIFO class for transferring data from a producer thread to a consumer thread.

Inherits basic functionality from **mha_fifo_t** (p. 919), adds release-acquire semantics to ensure consumer that the fill count or free space deduced from read and write pointers is consistent with the actual data. Copying, moving, and assignment of FIFO are forbidden by base class.

5.225.2 Constructor & Destructor Documentation

5.225.2.1 mha_fifo_lf_t() template<class T >

```
mha_fifo_lf_t< T >::  mha_fifo_lf_t (
            unsigned max_fill_count,
            const T & t = T() )  [inline], [explicit]
```

Create FIFO with fixed buffer size.

All (initially unused) instances of T are initialized as copies of t

5.225.3 Member Function Documentation

5.225.3.1 write() template<class T >

```
virtual void  mha_fifo_lf_t< T >::write (
            const T * data,
            unsigned count )  [inline], [override], [virtual]
```

Write specified ammount of data to the fifo.

Must only be called by the writer thread.

Parameters

data	Pointer to source data.
count	Number of instances to copy.

Exceptions

MHA_Error *(p. 906)*	when there is not enough space available.

Reimplemented from **mha_fifo_t< T >** (p. 922).

5.225.3.2 read() template<class T >

```
virtual void mha_fifo_lf_t< T >::read (
            T * outbuf,
            unsigned count ) [inline], [override], [virtual]
```

Read data from fifo.

Must only be called by the reader thread.

Parameters

outbuf	Pointer to the target buffer.
count	Number of instances to copy.

Exceptions

MHA_Error *(p. 906)*	when there is not enough data available.

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.225.3.3 get_fill_count() template<class T >

```
virtual unsigned mha_fifo_lf_t< T >::get_fill_count ( ) const [inline], [override],
[virtual]
```

get_fill_count() (p. 911) must only be called by the reader thread

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.225.3.4 get_available_space() template<class T >

```
virtual unsigned mha_fifo_lf_t< T >::get_available_space ( ) const [inline], [override],
[virtual]
```

get_available_space() (p. 911) must only be called by the writer thread

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.225.4 Member Data Documentation

5.225.4.1 atomic_write_ptr `template<class T >`

`std::atomic<const T *> mha_fifo_lf_t< T >::atomic_write_ptr [private]`

atomic copy of the write_ptr, only modified by the producer thread

5.225.4.2 atomic_read_ptr `template<class T >`

`std::atomic<const T *> mha_fifo_lf_t< T >::atomic_read_ptr [private]`

atomic copy of the read ptr, only modified by the consumer thread

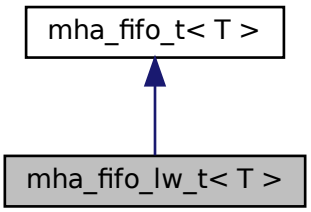
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.226 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

5.226.1 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.

5.226.2 Constructor & Destructor Documentation

5.226.2.1 mha_fifo_lw_t() template<class T >

```
mha_fifo_lw_t< T >::  mha_fifo_lw_t (
            unsigned max_fill_count )  [explicit]
```

Create FIFO with fixed buffer size.

5.226.2.2 ~mha_fifo_lw_t() template<class T >

mha_fifo_lw_t< T >::~ **mha_fifo_lw_t** [virtual]

release synchronization object

5.226.3 Member Function Documentation

5.226.3.1 write() 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. 906)*	when detecting a deadlock situation.

Reimplemented from **mha_fifo_t**< **T** > (p. 922).

5.226.3.2 read() 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. 906)*	when detecting a deadlock situation.

Reimplemented from **mha_fifo_t< T >** (p. 923).

5.226.3.3 set_error() 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. 906) to be thrown

5.226.4 Member Data Documentation

5.226.4.1 sync template<class T >

mha_fifo_thread_platform_t* **mha_fifo_lw_t**< T >::sync [private]

platform specific thread synchronization

5.226.4.2 error template<class T >

MHA_Error* **mha_fifo_lw_t**< T >::error[2] [private]

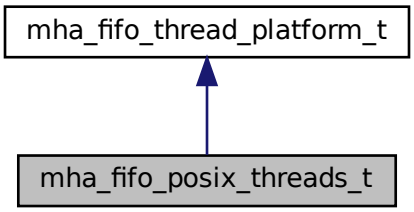
If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

5.227 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**

5.227.1 Constructor & Destructor Documentation

5.227.1.1 mha_fifo_posix_threads_t() `mha_fifo_posix_threads_t::mha_fifo_posix_↩ threads_t ( ) [inline]`

5.227.1.2 ~mha_fifo_posix_threads_t() `virtual mha_fifo_posix_threads_t::~mha_↩ fifo_posix_threads_t ( ) [inline], [virtual]`

5.227.2 Member Function Documentation

5.227.2.1 aquire_mutex() `virtual void mha_fifo_posix_threads_t::aquire_mutex ( ) [inline], [virtual]`

Implements **mha_fifo_thread_platform_t** (p. 929).

5.227.2.2 release_mutex() `virtual void mha_fifo_posix_threads_t::release_mutex ( ) [inline], [virtual]`

Implements **mha_fifo_thread_platform_t** (p. 929).

5.227.2.3 wait_for_decrease() virtual void mha_fifo_posix_threads_t::wait_for_↩ decrease ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 929).

5.227.2.4 wait_for_increase() virtual void mha_fifo_posix_threads_t::wait_for_↩ increase ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 930).

5.227.2.5 increment() virtual void mha_fifo_posix_threads_t::increment ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 930).

5.227.2.6 decrement() virtual void mha_fifo_posix_threads_t::decrement ( ) [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 930).

5.227.3 Member Data Documentation

5.227.3.1 mutex pthread_mutex_t mha_fifo_posix_threads_t::mutex [private]

5.227.3.2 decrease_condition pthread_cond_t mha_fifo_posix_threads_t::decrease_↩ condition [private]

5.227.3.3 increase_condition `pthread_cond_t mha_fifo_posix_threads_t::increase_condition [private]`

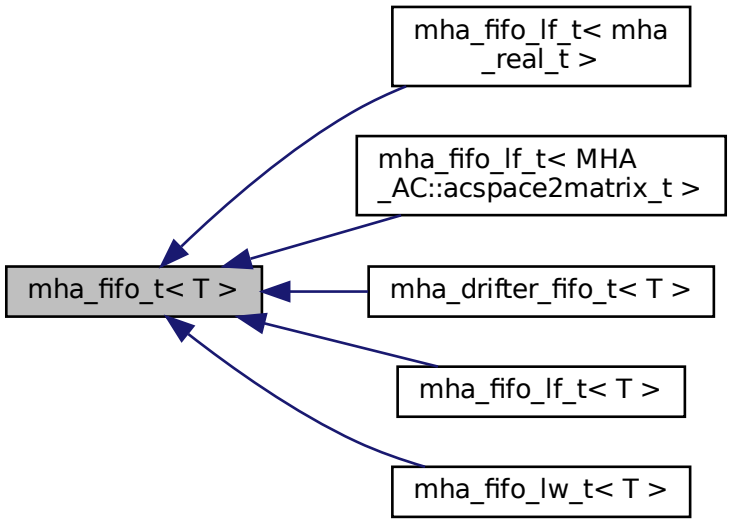
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.228 mha_fifo_t< T > Class Template Reference

A FIFO class.

Inheritance diagram for mha_fifo_t< T >:



Public Types

- typedef std::vector< T >:: **value_type** **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 *outbuf, 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, const T &t=T())
  Create FIFO with fixed buffer size, where all (initially unused) instances of T are initialized as copies of t.
- virtual ~**mha_fifo_t** ()=default
  Make destructor virtual.
- **mha_fifo_t** (const **mha_fifo_t** &)=delete
  Copy constructor.
- **mha_fifo_t** (**mha_fifo_t** &&)=delete
  Move constructor.
- **mha_fifo_t**< T > & **operator=** (const **mha_fifo_t**< T > &)=delete
  Assignment operator.
- **mha_fifo_t**< T > & **operator=** (**mha_fifo_t**< T > &&)=delete
  Move assignment operator.

Protected Member Functions

- void **clear** ()
  Empty the fifo at once.
- const T * **get_write_ptr** () const
  read-only access to the write pointer for derived classes
- const T * **get_read_ptr** () const
  read-only access to read pointer for derived classes
- unsigned **get_fill_count** (const T *wp, const T *rp) const
  Compute fill count from given write pointer and read pointer.

Private Attributes

- std::vector< T > **buf**
  The memory allocated to store the data in the fifo.
- T * **write_ptr**
  points to location where to write next
- const T * **read_ptr**
  points to location where to read next

5.228.1 Detailed Description

template<class T>
class mha_fifo_t< T >

A FIFO class.

Synchronization: None. Use external synchronisation or synchronization in inheriting class. Assignment, copy and move constructors are disabled.

5.228.2 Member Typedef Documentation

5.228.2.1 value_type `template<class T >`

```
typedef std::vector<T>::  value_type  mha_fifo_t< T >::  value_type
```

The data type exchanged by this fifo.

5.228.3 Constructor & Destructor Documentation

5.228.3.1 mha_fifo_t() [1/3] `template<class T >`

```
mha_fifo_t< T >::  mha_fifo_t (
            unsigned max_fill_count,
            const T & t = T() )  [explicit]
```

Create FIFO with fixed buffer size, where all (initially unused) instances of T are initialized as copies of t.

Parameters

max_fill_count	The maximum number of instances of T that can be held at the same time inside the fifo.
The	fifo allocates a vector of max_fill_count+1 instances of T for storage, one of which is always unused.

5.228.3.2 ~mha_fifo_t() template<class T >

```
virtual mha_fifo_t< T >::~ mha_fifo_t ( ) [virtual], [default]
```

Make destructor virtual.

5.228.3.3 mha_fifo_t() [2/3] template<class T >

```
mha_fifo_t< T >:: mha_fifo_t (
            const mha_fifo_t< T > & ) [delete]
```

Copy constructor.

5.228.3.4 mha_fifo_t() [3/3] template<class T >

```
mha_fifo_t< T >:: mha_fifo_t (
             mha_fifo_t< T > && ) [delete]
```

Move constructor.

5.228.4 Member Function Documentation

5.228.4.1 write() 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. 906)*	when there is not enough space available.

Reimplemented in **mha_fifo_lf_t< T >** (p. 910), **mha_fifo_lw_t< T >** (p. 914), and **mha_drifter_fifo_t< T >** (p. 900).

5.228.4.2 read()

```
template<class T >
void mha_fifo_t< T >::read (
            T * outbuf,
            unsigned count ) [virtual]
```

read data from fifo

Parameters

outbuf	Pointer to the target buffer.
count	Number of instances to copy.

Exceptions

MHA_Error *(p. 906)*	when there is not enough data available.

Reimplemented in **mha_fifo_lf_t< T >** (p. 910), **mha_fifo_lw_t< T >** (p. 914), and **mha_drifter_fifo_t< T >** (p. 901).

5.228.4.3 get_fill_count() [1/2]

```
template<class T >
virtual unsigned mha_fifo_t< T >::get_fill_count ( ) const [inline], [virtual]
```

Read-only access to fill_count.

Reimplemented in **mha_fifo_lf_t< T >** (p. 911), **mha_fifo_lf_t< mha_real_t >** (p. 911), **mha_fifo_lf_t< MHA_AC::acspace2matrix_t >** (p. 911), and **mha_drifter_fifo_t< T >** (p. 901).

5.228.4.4 get_available_space() template<class T >

unsigned **mha_fifo_t**< T >::get_available_space [virtual]

Read-only access to available_space.

Reimplemented in **mha_fifo_lf_t**< **T** > (p. 911), **mha_fifo_lf_t**< **mha_real_t** > (p. 911), **mha_fifo_lf_t**< **MHA_AC::acspace2matrix_t** > (p. 911), and **mha_drifter_fifo_t**< **T** > (p. 902).

5.228.4.5 get_max_fill_count() template<class T >

virtual unsigned **mha_fifo_t**< T >::get_max_fill_count ( ) const [inline], [virtual]

The capacity of this fifo.

5.228.4.6 operator=() [1/2] template<class T >

mha_fifo_t<T>& **mha_fifo_t**< T >::operator= (
const **mha_fifo_t**< T > &) [delete]

Assignment operator.

5.228.4.7 operator=() [2/2] template<class T >

mha_fifo_t<T>& **mha_fifo_t**< T >::operator= (
mha_fifo_t< T > &&) [delete]

Move assignment operator.

5.228.4.8 clear() 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.

5.228.4.9 get_write_ptr() template<class T >

```
const T* mha_fifo_t< T >::get_write_ptr ( ) const [inline], [protected]
```

read-only access to the write pointer for derived classes

5.228.4.10 get_read_ptr() template<class T >

```
const T* mha_fifo_t< T >::get_read_ptr ( ) const [inline], [protected]
```

read-only access to read pointer for derived classes

5.228.4.11 get_fill_count() [2/2] template<class T >

```
unsigned mha_fifo_t< T >::get_fill_count (
            const T * wp,
            const T * rp ) const [inline], [protected]
```

Compute fill count from given write pointer and read pointer.

Parameters

wp	Write pointer.
rp	Read pointer.

Precondition

wp and rp must point to an instance of T inside buf.

Returns

Number of elements that can be read from the fifo when wp and rp have the given values.

5.228.5 Member Data Documentation

5.228.5.1 buf template<class T >

std::vector<T> **mha_fifo_t**< T >::buf [private]

The memory allocated to store the data in the fifo.

At least one location in buf is always unused, because we have max_fill_count + 1 possible fillcounts [0:max_fill_count] that we need to distinguish.

5.228.5.2 write_ptr template<class T >

T* **mha_fifo_t**< T >::write_ptr [private]

points to location where to write next

5.228.5.3 read_ptr template<class T >

const T* **mha_fifo_t**< T >::read_ptr [private]

points to location where to read next

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

- **mha_fifo.h**

5.229 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**

5.229.1 Detailed Description

Simple Mutex Guard Class.

5.229.2 Constructor & Destructor Documentation

5.229.2.1 mha_fifo_thread_guard_t() mha_fifo_thread_guard_t::mha_fifo_thread_↩ guard_t (mha_fifo_thread_platform_t * sync) [inline]

5.229.2.2 ∼mha_fifo_thread_guard_t() mha_fifo_thread_guard_t::∼mha_fifo_thread_↩ guard_t ( ) [inline]

5.229.3 Member Data Documentation

5.229.3.1 sync mha_fifo_thread_platform_t* mha_fifo_thread_guard_t::sync [private]

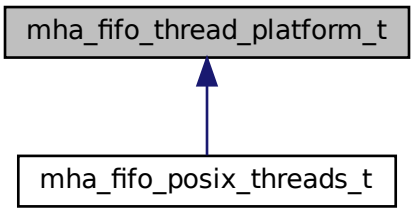
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.230 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** &)

5.230.1 Detailed Description

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Works only with single producer and single consumer.

5.230.2 Constructor & Destructor Documentation

5.230.2.1 ~**mha_fifo_thread_platform_t()** `virtual mha_fifo_thread_platform_t::~mha↩_fifo_thread_platform_t ( ) [inline], [virtual]`

Make destructor virtual.

5.230.2.2 mha_fifo_thread_platform_t() [1/2] mha_fifo_thread_platform_t::mha_fifo↩ _thread_platform_t (const mha_fifo_thread_platform_t &) [private]

5.230.2.3 mha_fifo_thread_platform_t() [2/2] mha_fifo_thread_platform_t::mha_fifo↩ _thread_platform_t ( ) [inline]

Make default constructor accessible.

5.230.3 Member Function Documentation

5.230.3.1 aquire_mutex() 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. 917).

5.230.3.2 release_mutex() 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. 917).

5.230.3.3 wait_for_decrease() `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. 917).

5.230.3.4 wait_for_increase() `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. 918).

5.230.3.5 increment() `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. 918).

5.230.3.6 decrement() `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. 918).

5.230.3.7 operator=() `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**

5.231 mha_real_test_array_t Struct Reference

Public Attributes

- **mha_real_t r** [4]

5.231.1 Member Data Documentation

5.231.1.1 r `mha_real_t mha_real_test_array_t::r[4]`

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

- **mha.hh**

5.232 mha_rt_fifo_element_t< T > Class Template Reference

Object wrapper for **mha_rt_fifo_t** (p. 933).

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.

5.232.1 Detailed Description

template<**class T**>
class mha_rt_fifo_element_t< **T** >

Object wrapper for **mha_rt_fifo_t** (p. 933).

5.232.2 Constructor & Destructor Documentation

5.232.2.1 mha_rt_fifo_element_t() `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.

5.232.2.2 ~mha_rt_fifo_element_t() `template<class T >`

mha_rt_fifo_element_t< T >::~ **mha_rt_fifo_element_t** ( ) [inline]

5.232.3 Member Data Documentation

5.232.3.1 next `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.

5.232.3.2 abandonned `template<class T >`

`bool mha_rt_fifo_element_t< T >::abandonned`

Indicates that this element will no longer be used and may be deleted.

5.232.3.3 data `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**

5.233 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. 935) or **poll_1** (p. 935).*

5.233.1 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. 935). The push operation is not real time safe, it allocates and deallocates memory. The latest element is retrieved by calling **poll** (p. 935). This operation will skip fifo elements if more than one **push** (p. 935) has been occured since the last poll. To avoid skipping, call the **poll_1** (p. 935) operation instead.

5.233.2 Constructor & Destructor Documentation

5.233.2.1 mha_rt_fifo_t() `template<class T >`

mha_rt_fifo_t< T >:: **mha_rt_fifo_t** ( ) [inline]

Construct empty fifo.

5.233.2.2 ~mha_rt_fifo_t() template<class T >

`mha_rt_fifo_t< T >::~ mha_rt_fifo_t ( ) [inline]`

Destructor will delete all data currently in the fifo.

5.233.3 Member Function Documentation

5.233.3.1 poll() 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.

5.233.3.2 poll_1() 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. 935) or **poll_1()** (p. 935) invocation.

Returns

The next element in this Fifo, if there is one, or the same as before. Returns NULL if the Fifo is empty.

5.233.3.3 push() 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.

5.233.3.4 remove_abandonned() template<class T >

void **mha_rt_fifo_t**< T >::remove_abandonned ( ) [inline], [private]

Deletes abandonned elements.

5.233.3.5 remove_all() template<class T >

void **mha_rt_fifo_t**< T >::remove_all ( ) [inline], [private]

Deletes all elements.

5.233.4 Member Data Documentation

5.233.4.1 root 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.

5.233.4.2 current `template<class T >`

`mha_rt_fifo_element_t<T>* mha_rt_fifo_t< T >::current [private]`

The element most recently returned by **poll** (p. 935) or **poll_1** (p. 935).

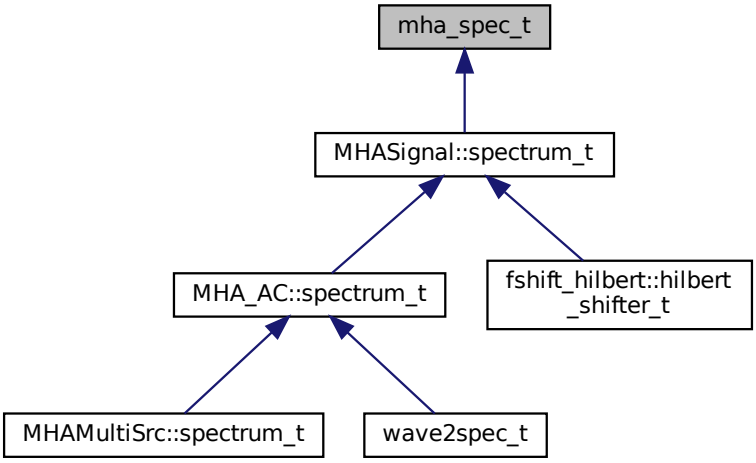
Searching for new elements starts here.

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

- **mha_fifo.h**

5.234 mha_spec_t Struct Reference

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

5.234.1 Detailed Description

```
\ingroup mhasignal
\brief 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.

Only the FFT bins for the positive frequencies, 0, and the Nyquist frequency are stored in this structure. The negative frequencies are not stored, because for a real-valued time signal they are the complex conjugates of the positive frequencies.

The negative frequencies still contribute to the signal's level. Refer to **Central Calibration** (p. 3) for a description of the scaling and how the level would be computed from the spectrum. It is recommended to use the library function **MHASignal::rmslevel** (p. 53) to compute the unweighted level correctly in Pascal, or **MHASignal::colored_intensity** (p. 54) to compute a possibly weighted intensity.

5.234.2 Member Data Documentation

5.234.2.1 buf `mha_complex_t* mha_spec_t::buf`

signal buffer

5.234.2.2 num_channels `unsigned int mha_spec_t::num_channels`

number of channels

5.234.2.3 num_frames `unsigned int mha_spec_t::num_frames`

number of frames in each channel

5.234.2.4 channel_info `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**

5.235 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.

5.235.1 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.

5.235.2 Constructor & Destructor Documentation

5.235.2.1 mha_stash_environment_variable_t() mha_stash_environment_variable_t↩ ::mha_stash_environment_variable_t (

```
            const std::string & variable_name,
            const std::string & new_content ) [inline]
```

5.235.2.2 ∼mha_stash_environment_variable_t() mha_stash_environment_variable_t↩ ::∼mha_stash_environment_variable_t ( ) [inline]

5.235.3 Member Data Documentation

5.235.3.1 existed_before const bool mha_stash_environment_variable_t::existed_↩ before [private]

Flag indicates if the environment variable existed before constructor.

5.235.3.2 variable_name const std::string mha_stash_environment_variable_t::variable↩ _name [private]

Name of environment variable.

5.235.3.3 original_content const std::string mha_stash_environment_variable_t↩ ::original_content [private]

Content of environment variable before constructor executed.

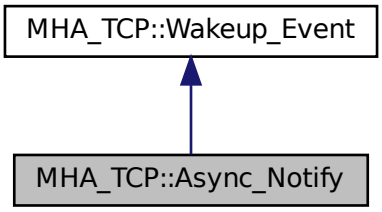
The documentation for this class was generated from the following file:

- **mha_os.h**

5.236 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

5.236.1 Detailed Description

Portable Multiplexable cross-thread notification.

5.236.2 Constructor & Destructor Documentation

5.236.2.1 Async_Notify() `Async_Notify::Async_Notify ( )`

5.236.2.2 ~Async_Notify() `Async_Notify::~Async_Notify ( ) [virtual]`

5.236.3 Member Function Documentation

5.236.3.1 reset() `void Async_Notify::reset ( ) [virtual]`

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 983).

5.236.3.2 set() `void Async_Notify::set ( ) [virtual]`

5.236.4 Member Data Documentation

5.236.4.1 pipe `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**

5.237 mha_tcp::buffered_socket_t Class Reference

An asio TCP socket with an associated streambuf buffer for receiving lines of text, as well as string buffers for sending responses.

Inherits asio::ip::tcp::socket, and std::enable_shared_from_this< buffered_socket_t >.

Public Member Functions

- asio::streambuf & **get_buffer** ()
  Access to associated streambuf.
- void **queue_write** (const std::string &message)
  Send the given message through this connection to the client asynchronously.

Private Attributes

- asio::streambuf **streambuf**
  associated streambuf object to collect received pieces into lines
- std::string **current_message**
  The message that is currently sent back to the client.
- std::string **next_message**
  A buffer for the next message(s) that must be sent back to the client after the sending of current_message has completed.

5.237.1 Detailed Description

An asio TCP socket with an associated streambuf buffer for receiving lines of text, as well as string buffers for sending responses.

Used for communicating with MHA TCP clients. The life time of the connection objects is managed with shared pointers registered together with callbacks in the asio event loop. This is a common idiom in asio. To support this, we inherit from enable_shared_from_this which is also common in code that uses asio.

5.237.2 Member Function Documentation

5.237.2.1 get_buffer() `asio::streambuf& mha_tcp::buffered_socket_t::get_buffer ( ) [inline]`

Access to associated streambuf.

Needed to invoke async_read.

Returns

associated streambuf object by reference

5.237.2.2 queue_write() `void mha_tcp::buffered_socket_t::queue_write ( const std::string & message )`

Send the given message through this connection to the client asynchronously.

Parameters

message	The text to send. Method copies the message before returning.

5.237.3 Member Data Documentation

5.237.3.1 streambuf `asio::streambuf mha_tcp::buffered_socket_t::streambuf [private]`

associated streambuf object to collect received pieces into lines

5.237.3.2 current_message `std::string mha_tcp::buffered_socket_t::current_message [private]`

The message that is currently sent back to the client.

5.237.3.3 next_message `std::string mha_tcp::buffered_socket_t::next_message [private]`

A buffer for the next message(s) that must be sent back to the client after the sending of current_message has completed.

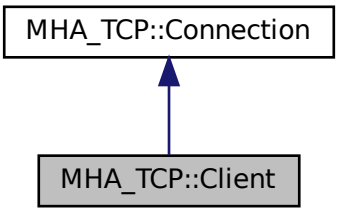
The documentation for this class was generated from the following files:

- **mha_tcp_server.hh**
- **mha_tcp_server.cpp**

5.238 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

5.238.1 Detailed Description

A portable class for a tcp client connections.

5.238.2 Constructor & Destructor Documentation

5.238.2.1 Client() [1/2] 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. 960).
port	The port or the TCP **Server** (p. 960).

5.238.2.2 Client() [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. 960).
port	The port or the TCP **Server** (p. 960).
timeout_watcher	an Event watcher that implements a timeout.

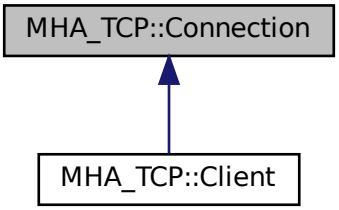
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.239 MHA_TCP::Connection Class Reference

Connection (p. 946) 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**

5.239.1 Detailed Description

Connection (p. 946) handles Communication between client and server, is used on both sides.

5.239.2 Constructor & Destructor Documentation

5.239.2.1 Connection() `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. 906)*	If the file descriptor is < 0.

5.239.2.2 ~Connection() `Connection::~Connection ( ) [virtual]`

Destructor closes the underlying file descriptor.

5.239.3 Member Function Documentation

5.239.3.1 init_peer_data() `void MHA_TCP::Connection::init_peer_data ( ) [private]`

determine peer address and port

5.239.3.2 can_sysread() `bool Connection::can_sysread ( ) [private]`

Determine wether at least 1 byte can be read without blocking.

5.239.3.3 can_syswrite() `bool Connection::can_syswrite ( ) [private]`

Determine wether at least 1 byte can be written without blocking.

5.239.3.4 sysread() `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.

5.239.3.5 syswrite() `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.

5.239.3.6 get_read_event() `Sockread_Event * Connection::get_read_event ( )`

5.239.3.7 get_write_event() `Sockwrite_Event * Connection::get_write_event ( )`

5.239.3.8 get_peer_address() `std::string Connection::get_peer_address ( )`

Get peer's IP Address.

5.239.3.9 get_peer_port() `unsigned short Connection::get_peer_port ( )`

Get peer's TCP port.

5.239.3.10 get_fd() `SOCKET MHA_TCP::Connection::get_fd ( ) const [inline]`

Return the (protected) file descriptor of the connection.

Will be required for SSL.

5.239.3.11 eof() `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.

5.239.3.12 can_read_line() `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.

5.239.3.13 can_read_bytes() `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

5.239.3.14 read_line() `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.

5.239.3.15 read_bytes() `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.

5.239.3.16 try_write() `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.

5.239.3.17 write() `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.

5.239.3.18 needs_write() `bool Connection::needs_write ( )`

Checks if the internal "outgoing" buffer contains data.

5.239.3.19 buffered_incoming_bytes() `unsigned Connection::buffered_incoming_bytes ( ) const`

Returns the number of bytes in the internal "incoming" buffer.

5.239.3.20 buffered_outgoing_bytes() `unsigned Connection::buffered_outgoing_bytes ( ) const`

Returns the number of bytes in the internal "outgoing" buffer.

5.239.4 Member Data Documentation

5.239.4.1 outbuf `std::string MHA_TCP::Connection::outbuf [private]`

5.239.4.2 inbuf `std::string MHA_TCP::Connection::inbuf [private]`

5.239.4.3 read_event `Sockread_Event* MHA_TCP::Connection::read_event [private]`

5.239.4.4 write_event `Sockwrite_Event* MHA_TCP::Connection::write_event [private]`

5.239.4.5 closed `bool MHA_TCP::Connection::closed [private]`

5.239.4.6 peer_addr `struct sockaddr_in MHA_TCP::Connection::peer_addr [private]`

5.239.4.7 fd `SOCKET MHA_TCP::Connection::fd [protected]`

The file descriptor of the TCP Socket.

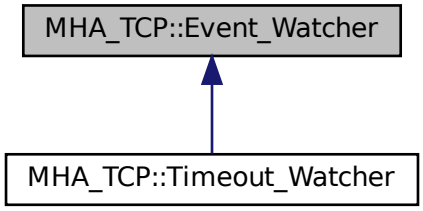
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.240 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.

5.240.1 Detailed Description

OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.

5.240.2 Member Typedef Documentation

5.240.2.1 Events typedef std::set< Wakeup_Event* > MHA_TCP::Event_Watcher::↩ Events

5.240.2.2 iterator typedef std::set< Wakeup_Event* >:: iterator MHA_TCP::Event_↩ Watcher::iterator

5.240.3 Constructor & Destructor Documentation

5.240.3.1 ~Event_Watcher() Event_Watcher::~Event_Watcher ( ) [virtual]

5.240.4 Member Function Documentation

5.240.4.1 observe() void Event_Watcher::observe (Wakeup_Event * *event*)

Add an event to this observer.

5.240.4.2 ignore() void Event_Watcher::ignore (Wakeup_Event * *event*)

Remove an event from this observer.

5.240.4.3 wait() std::set< Wakeup_Event * > Event_Watcher::wait ( )

\ Wait for some event to occur.

Return all events that are ready

5.240.5 Member Data Documentation

5.240.5.1 events 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**

5.241 MHA_TCP::OS_EVENT_TYPE Struct Reference

Public Types

- enum { **R** =0 , **W** =1 , **X** =2 , **T** }

Public Attributes

- enum MHA_TCP::OS_EVENT_TYPE:: { ... } **mode**
- union {
  int **fd**
  double **timeout**
  };

5.241.1 Member Enumeration Documentation

5.241.1.1 anonymous enum `anonymous enum`

Enumerator

R	
W	
X	
T	

5.241.2 Member Data Documentation

5.241.2.1 `enum { ... } MHA_TCP::OS_EVENT_TYPE::mode`

5.241.2.2 fd `int MHA_TCP::OS_EVENT_TYPE::fd`

5.241.2.3 timeout `double MHA_TCP::OS_EVENT_TYPE::timeout`

5.241.2.4 `union { ... }`

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

- **mha_tcp.hh**

5.242 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. 956).
- **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**

5.242.1 Constructor & Destructor Documentation

5.242.1.1 Server() [1/2] 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.

5.242.1.2 Server() [2/2] 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.

5.242.1.3 ~Server() Server::~Server ( )

Close the TCP server socket.

5.242.2 Member Function Documentation

5.242.2.1 initialize() void Server::initialize (const std::string & *iface*, unsigned short *port*) [private]

5.242.2.2 get_interface() `std::string Server::get_interface ( ) const`

Get the name given in the constructor for the network interface.

5.242.2.3 get_port() `unsigned short Server::get_port ( ) const`

Get the port that the TCP server socket currently listens to.

5.242.2.4 get_accept_event() `Sockaccept_Event * Server::get_accept_event ( )`

Produces an event that can be observed by an **Event_Watcher** (p. 956).

This event signals incoming connections that can be accepted.

5.242.2.5 accept() `Connection * Server::accept ( )`

Accept an incoming connection.

blocks if necessary.

Returns

The new TCP connection. The connection has to be deleted by the caller.

5.242.2.6 try_accept() `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.

5.242.3 Member Data Documentation

5.242.3.1 sock_addr `sockaddr_in MHA_TCP::Server::sock_addr [private]`

5.242.3.2 serversocket `SOCKET MHA_TCP::Server::serversocket [private]`

5.242.3.3 iface `std::string MHA_TCP::Server::iface [private]`

5.242.3.4 port `unsigned short MHA_TCP::Server::port [private]`

5.242.3.5 accept_event `Sockaccept_Event* MHA_TCP::Server::accept_event [private]`

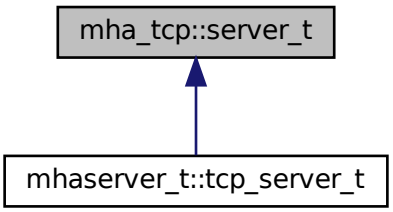
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.243 mha_tcp::server_t Class Reference

Class for accepting TCP connections from clients.

Inheritance diagram for mha_tcp::server_t:



Public Member Functions

- **server_t** (const std::string &interface, uint16_t port)
  Allocates a TCP server.
- uint16_t **get_port** () const
- asio::ip::tcp::endpoint **get_endpoint** () const
- asio::ip::address **get_address** () const
- size_t **get_num_accepted_connections** () const
- void **run** ()
  Accepts connections on the TCP port and serves them.
- virtual bool **on_received_line** (std::shared_ptr< **buffered_socket_t** > c, const std↩ ::string &l)
  This method is invoked when a line of text is received on one of the accepted connections.
- virtual void **shutdown** ()
  *Shuts down the server: Close the acceptor (no new connections), shuts down the receiving direction of all accepted connections (no new commands, but responses can still be finished), registers a timer event that will cause event loop termination and return from the **run()** (p. 966) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.*
- virtual ~**server_t** ()=default
  Make destructor virtual.
- asio::io_context & **get_context** ()

Private Member Functions

- void **trigger_accept** ()
  Triggers the acceptance of the next connection.
- void **post_trigger_read_line** (std::shared_ptr< **buffered_socket_t** > c)
  Call trigger_read_line with a single detour through asio's event loop to avoid starving other connections when one connection floods us.
- void **trigger_read_line** (std::shared_ptr< **buffered_socket_t** > c)
  Triggers the reading of the next line from a connection.
- void **add_connection** (std::shared_ptr< **buffered_socket_t** > connection)
  Add new connection to the list of connections, retire stale pointers.

Private Attributes

- asio::io_context **io_context**
  The io context used to run event loops.
- std::shared_ptr< asio::ip::tcp::acceptor > **acceptor**
  The underlying asio object used to accept incoming TCP connections.
- bool **async_accept_has_been_triggered** = false
  Set to true when async_acceptance is triggered in trigger_accept (Only one accept can be in process at any time).
- size_t **num_accepted_connections** = 0U
  Number of accepted connections (not necessarily still existing)
- std::vector< std::weak_ptr< **buffered_socket_t** > > **connections**
  Weak pointers to the existing connections.

5.243.1 Detailed Description

Class for accepting TCP connections from clients.

5.243.2 Constructor & Destructor Documentation

5.243.2.1 server_t() `mha_tcp::server_t::server_t (`

```
const std::string & interface,
uint16_t port )
```

Allocates a TCP server.

Parameters

interface	Host name of the network interface to listen on. Can be "localhost" or "127.0.0.1" for localhost, "0.0.0.0" for any ipv4 interface, ...
port	TCP port to open for incoming connections. If port==0, then the operating system will select a free port.

Exceptions

system_error	if the name given in interface cannot be resolved
system_error	if we cannot bind to the requested interface

5.243.2.2 ~server_t() `virtual mha_tcp::server_t::~server_t ( ) [virtual], [default]`

Make destructor virtual.

5.243.3 Member Function Documentation

5.243.3.1 get_port() `uint16_t mha_tcp::server_t::get_port ( ) const`

Returns

The port number of the TCP port that has been opened. If the port specified in the constructor was 0, this will return the port that the operating system has selected.

5.243.3.2 get_endpoint() `asio::ip::tcp::endpoint mha_tcp::server_t::get_endpoint ( ) const`

Returns

The local endpoint of the acceptor.

5.243.3.3 get_address() `asio::ip::address mha_tcp::server_t::get_address ( ) const`

Returns

The ip adress that the server is bound to.

5.243.3.4 get_num_accepted_connections() `size_t mha_tcp::server_t::get_num_↩ accepted_connections ( ) const`

Returns

the number of TCP connections that ha ve been accepted

5.243.3.5 run() `void mha_tcp::server_t::run ( )`

Accepts connections on the TCP port and serves them.

Triggers the acceptance of the next connection to start things off.

5.243.3.6 on_received_line() `bool mha_tcp::server_t::on_received_line ( std::shared_ptr< buffered_socket_t > c, const std::string & l ) [virtual]`

This method is invoked when a line of text is received on one of the accepted connections.

Override this method to process the communication with the client.

Parameters

c	the connection that has received this line
l	the line that has been received, without the line ending

Returns

client should return true when client wants to read another line of text, else false.

Reimplemented in **mhaserver_t::tcp_server_t** (p. 1346).

5.243.3.7 shutdown() `void mha_tcp::server_t::shutdown ( ) [virtual]`

Shuts down the server: Close the acceptor (no new connections), shuts down the receiving direction of all accepted connections (no new commands, but responses can still be finished), registers a timer event that will cause event loop termination and return from the **run()** (p. 966) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.

5.243.3.8 get_context() `asio::io_context & mha_tcp::server_t::get_context ( )`

Returns

the asio io context used to run the event loop

5.243.3.9 trigger_accept() void mha_tcp::server_t::trigger_accept ( ) [private]

Triggers the acceptance of the next connection.

Called from run to accept the first connection and from the accept handler to accept each next connection. Once a connection from a client is accepted, the accept handler will register it with the event loop for receiving a line of text.

5.243.3.10 post_trigger_read_line() void mha_tcp::server_t::post_trigger_read_line (std::shared_ptr< **buffered_socket_t** > *c*) [private]

Call trigger_read_line with a single detour through asio's event loop to avoid starving other connections when one connection floods us.

5.243.3.11 trigger_read_line() void mha_tcp::server_t::trigger_read_line (std::shared_ptr< **buffered_socket_t** > *c*) [private]

Triggers the reading of the next line from a connection.

Parameters

c	The connection where the incoming data is expected from.

5.243.3.12 add_connection() void mha_tcp::server_t::add_connection (std::shared_ptr< **buffered_socket_t** > *connection*) [inline], [private]

Add new connection to the list of connections, retire stale pointers.

5.243.4 Member Data Documentation

5.243.4.1 io_context `asio::io_context mha_tcp::server_t::io_context [private]`

The io context used to run event loops.

5.243.4.2 acceptor `std::shared_ptr<asio::ip::tcp::acceptor> mha_tcp::server_t::acceptor [private]`

The underlying asio object used to accept incoming TCP connections.

5.243.4.3 async_accept_has_been_triggered `bool mha_tcp::server_t::async_accept_has_been_triggered = false [private]`

Set to true when async_acceptance is triggered in trigger_accept (Only one accept can be in process at any time).

5.243.4.4 num_accepted_connections `size_t mha_tcp::server_t::num_accepted_connections = 0U [private]`

Number of accepted connections (not necessarily still existing)

5.243.4.5 connections `std::vector<std::weak_ptr< buffered_socket_t> > mha_tcp::server_t::connections [private]`

Weak pointers to the existing connections.

Needed to shutdown the active connections for incoming data when server shuts down.

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

- **mha_tcp_server.hh**
- **mha_tcp_server.cpp**

5.244 MHA_TCP::sock_init_t Class Reference

Public Member Functions

- **sock_init_t** ()

5.244.1 Constructor & Destructor Documentation

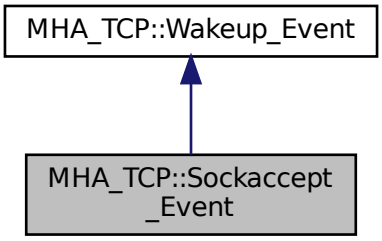
5.244.1.1 sock_init_t() `MHA_TCP::sock_init_t::sock_init_t ( ) [inline]`

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

- **mha_tcp.cpp**

5.245 MHA_TCP::Sockaccept_Event Class Reference

Inheritance diagram for MHA_TCP::Sockaccept_Event:



Public Member Functions

- **Sockaccept_Event** (**SOCKET**)

Additional Inherited Members

5.245.1 Constructor & Destructor Documentation

5.245.1.1 Sockaccept_Event() `MHA_TCP::Sockaccept_Event::Sockaccept_Event ( SOCKET s )`

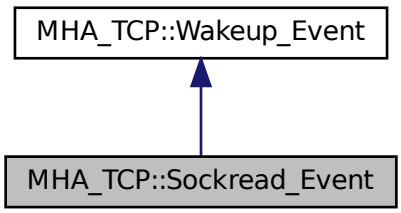
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.246 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

5.246.1 Detailed Description

Watch socket for incoming data.

5.246.2 Constructor & Destructor Documentation

5.246.2.1 Sockread_Event() `MHA_TCP::Sockread_Event::Sockread_Event ( SOCKET s )`

Set socket to watch for.

Parameters

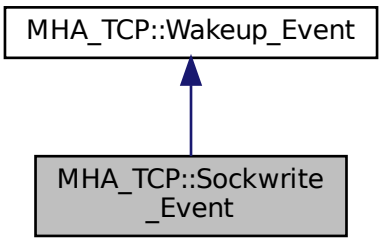
s	The socket to observe incoming data on.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.247 MHA_TCP::Sockwrite_Event Class Reference

Inheritance diagram for MHA_TCP::Sockwrite_Event:



Public Member Functions

- **Sockwrite_Event** (**SOCKET** s)

Additional Inherited Members

5.247.1 Constructor & Destructor Documentation

5.247.1.1 Sockwrite_Event() `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**

5.248 MHA_TCP::Thread Class Reference

A very simple class for portable threads.

Public Types

- enum { **PREPARED** , **RUNNING** , **FINISHED** }
  The current state of the thread.
- 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. 973) is finished.*
- virtual void **run** ()
  *The internal method that delegated the new thread to the registered **Thread** (p. 973) 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. 906) 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.

5.248.1 Detailed Description

A very simple class for portable threads.

5.248.2 Member Typedef Documentation

5.248.2.1 thr_f `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.

5.248.3 Member Enumeration Documentation

5.248.3.1 anonymous enum `anonymous enum`

The current state of the thread.

Enumerator

PREPARED	
RUNNING	
FINISHED	

5.248.4 Constructor & Destructor Documentation

5.248.4.1 Thread() `[1/2]` `MHA_TCP::Thread::Thread ( ) [protected]`

Default constructor may only be used by derived classes that want to start the thread themselves.

5.248.4.2 Thread() [2/2] 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.

5.248.4.3 ~Thread() Thread::~Thread ( ) [virtual]

The destructor should only be called when the **Thread** (p. 973) is finished.

There is preliminary support for forceful thread cancellation in the destructor, but probably not very robust or portable..

5.248.5 Member Function Documentation

5.248.5.1 run() void Thread::run ( ) [virtual]

The internal method that delegated the new thread to the registered **Thread** (p. 973) function.

5.248.6 Member Data Documentation

5.248.6.1 thread_handle pthread_t MHA_TCP::Thread::thread_handle [private]

The posix thread handle.

5.248.6.2 thread_attr `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.

5.248.6.3 arg `void* MHA_TCP::Thread::arg [protected]`

The argument for the client's thread function.

5.248.6.4 return_value `void* MHA_TCP::Thread::return_value [protected]`

The return value from the client's thread function is stored here When that function returns.

5.248.6.5 thread_finish_event **`Async_Notify`** `MHA_TCP::Thread::thread_finish_event`

Event will be triggered when the thread exits.

5.248.6.6 `enum { ... } MHA_TCP::Thread::state`

The current state of the thread.

5.248.6.7 thread_func **`thr_f`** `MHA_TCP::Thread::thread_func`

The thread function that the client has registered.

5.248.6.8 thread_arg `void* MHA_TCP::Thread::thread_arg`

The argument that the client wants to be handed through to the thread function.

5.248.6.9 error `MHA_Error* MHA_TCP::Thread::error`

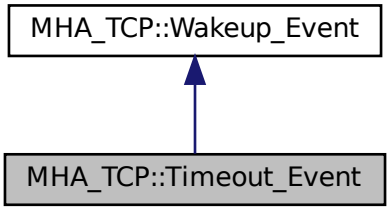
The **MHA_Error** (p. 906) that caused the thread to abort, if any.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.249 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

5.249.1 Constructor & Destructor Documentation

5.249.1.1 Timeout_Event() `Timeout_Event::Timeout_Event ( double interval )`

5.249.2 Member Function Documentation

5.249.2.1 get_os_event() `OS_EVENT_TYPE Timeout_Event::get_os_event ( ) [virtual]`

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 983).

5.249.3 Member Data Documentation

5.249.3.1 end_time `double MHA_TCP::Timeout_Event::end_time [private]`

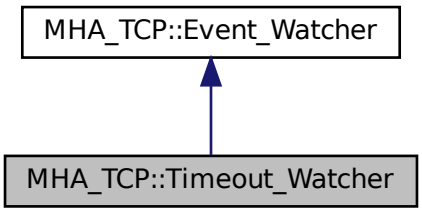
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.250 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

5.250.1 Detailed Description

OS-independent event watcher with internal fixed-end-time timeout.

5.250.2 Constructor & Destructor Documentation

5.250.2.1 Timeout_Watcher() `Timeout_Watcher::Timeout_Watcher ( double interval ) [explicit]`

5.250.2.2 ∼Timeout_Watcher() `Timeout_Watcher::∼Timeout_Watcher ( ) [virtual]`

5.250.3 Member Data Documentation

5.250.3.1 timeout **`Timeout_Event`** `MHA_TCP::Timeout_Watcher::timeout [private]`

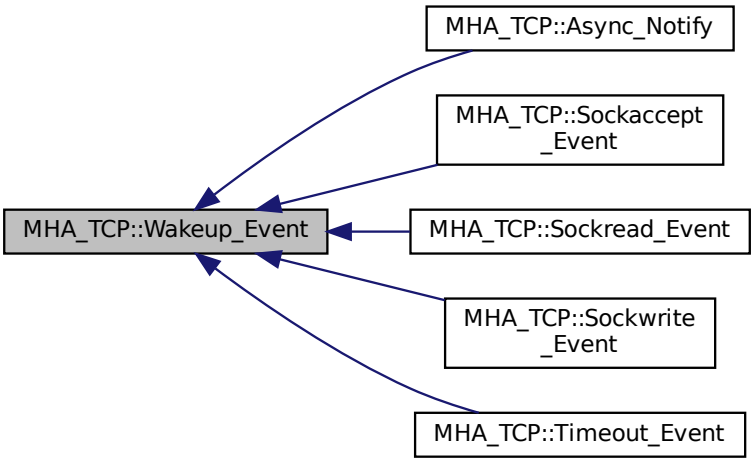
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.251 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. 956) *when this event is added to its list of observed events.*
- virtual void **ignored_by** (**Event_Watcher** ∗observer)
  *Called by the **Event_Watcher*** (p. 956) *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. 956) *instances that this **Wakeup_Event*** (p. 981) *is observed by (stored here for proper deregistering).*

5.251.1 Detailed Description

A base class for asynchronous wakeup events.

5.251.2 Constructor & Destructor Documentation

5.251.2.1 Wakeup_Event() `Wakeup_Event::Wakeup_Event ( )`

Event Constructor.

The new event has invalid state.

5.251.2.2 ~Wakeup_Event() `Wakeup_Event::~Wakeup_Event ( ) [virtual]`

Destructor deregisters from observers.

5.251.3 Member Function Documentation

5.251.3.1 observed_by() `void Wakeup_Event::observed_by ( Event_Watcher * observer ) [virtual]`

Called by the **Event_Watcher** (p. 956) when this event is added to its list of observed events.

5.251.3.2 ignored_by() void Wakeup_Event::ignored_by (Event_Watcher * observer) [virtual]

Called by the **Event_Watcher** (p. 956) when this event is removed from its list of observed events.

5.251.3.3 get_os_event() OS_EVENT_TYPE Wakeup_Event::get_os_event ( ) [virtual]

Get necessary information for the Event Watcher.

Reimplemented in **MHA_TCP::Timeout_Event** (p. 979).

5.251.3.4 reset() void Wakeup_Event::reset ( ) [virtual]

For pure notification events, reset the "signalled" status.

Reimplemented in **MHA_TCP::Async_Notify** (p. 942).

5.251.3.5 status() bool Wakeup_Event::status ( ) [virtual]

Query wether the event is in signalled state now.

5.251.4 Member Data Documentation

5.251.4.1 observers std::set<class Event_Watcher *> MHA_TCP::Wakeup_Event↩ ::observers [private]

A list of all **Event_Watcher** (p. 956) instances that this **Wakeup_Event** (p. 981) is observed by (stored here for proper deregistering).

5.251.4.2 os_event `OS_EVENT_TYPE MHA_TCP::Wakeup_Event::os_event [protected]`

5.251.4.3 os_event_valid `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**

5.252 mha_tictoc_t Struct Reference

Public Attributes

- struct timeval **tv1**
- struct timeval **tv2**
- struct timezone **tz**
- float **t**

5.252.1 Member Data Documentation

5.252.1.1 tv1 `struct timeval mha_tictoc_t::tv1`

5.252.1.2 tv2 `struct timeval mha_tictoc_t::tv2`

5.252.1.3 tz `struct timezone mha_tictoc_t::tz`

5.252.1.4 t `float mha_tictoc_t::t`

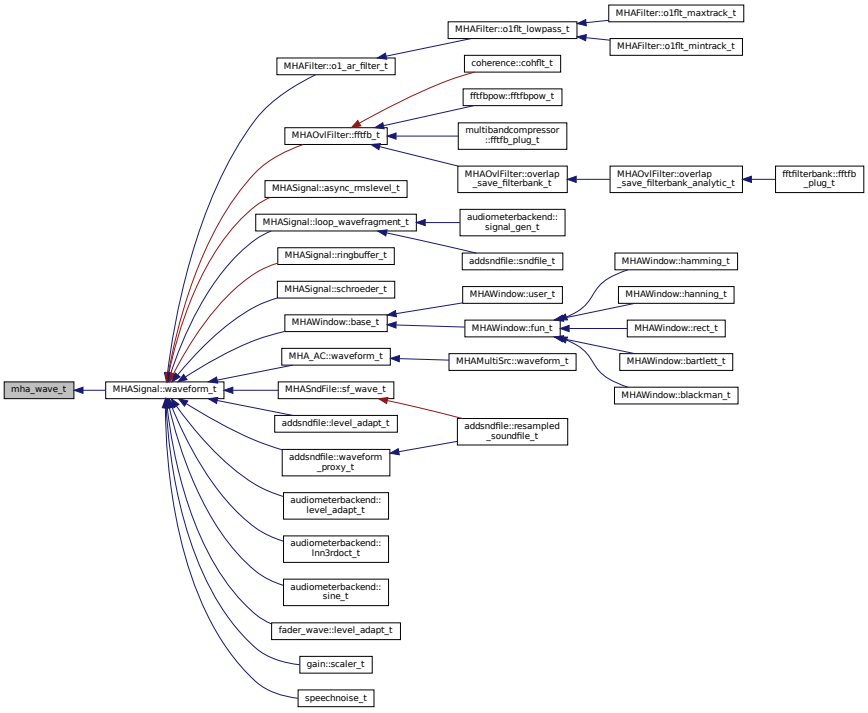
The documentation for this struct was generated from the following file:

- **mha_profiling.h**

5.253 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

5.253.1 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.

In a calibrated openMHA, audio samples are stored in unit Pascal, see **Central Calibration** (p. 3).

The field channel_info must be an array of num_channels entries or NULL.

5.253.2 Member Data Documentation

5.253.2.1 buf `mha_real_t* mha_wave_t::buf`

signal buffer

5.253.2.2 num_channels `unsigned int mha_wave_t::num_channels`

number of channels

5.253.2.3 num_frames `unsigned int mha_wave_t::num_frames`

number of frames in each channel

5.253.2.4 channel_info `mha_channel_info_t* mha_wave_t::channel_info`

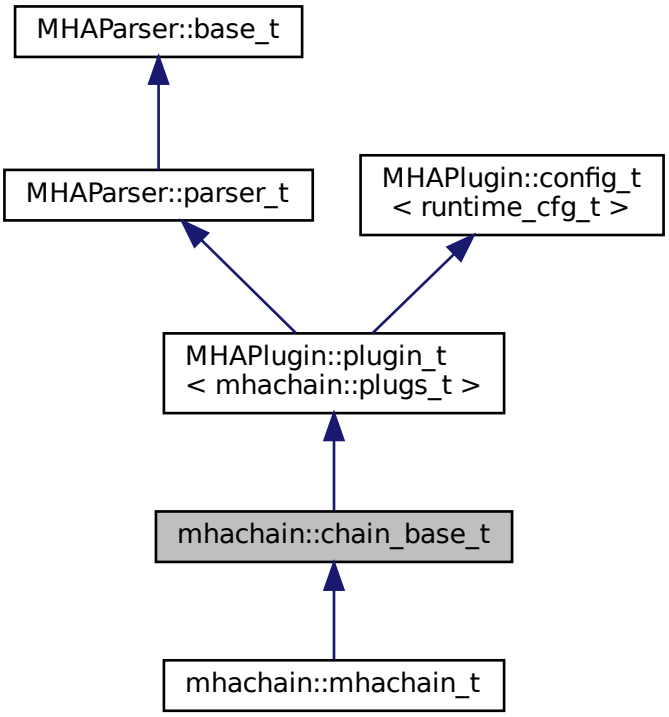
detailed channel description

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

- **mha.hh**

5.254 mhachain::chain_base_t Class Reference

Inheritance diagram for mhachain::chain_base_t:



Public Member Functions

- **chain_base_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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**

Additional Inherited Members

5.254.1 Constructor & Destructor Documentation

5.254.1.1 chain_base_t() mhachain::chain_base_t::chain_base_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.254.2 Member Function Documentation

5.254.2.1 process() [1/4] void mhachain::chain_base_t::process (

```
mha_wave_t * sin,
mha_wave_t ** sout )
```

5.254.2.2 process() [2/4] void mhachain::chain_base_t::process (

```
mha_spec_t * sin,
mha_wave_t ** sout )
```

5.254.2.3 process() [3/4]

```
void mhachain::chain_base_t::process (
    mha_wave_t * sin,
    mha_spec_t ** sout )
```

5.254.2.4 process() [4/4]

```
void mhachain::chain_base_t::process (
    mha_spec_t * sin,
    mha_spec_t ** sout )
```

5.254.2.5 prepare()

```
void mhachain::chain_base_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< mhachain::plugs_t >** (p. 1301).

5.254.2.6 release()

```
void mhachain::chain_base_t::release (
    void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< mhachain::plugs_t >** (p. 1302).

5.254.2.7 update()

```
void mhachain::chain_base_t::update ( ) [private]
```

5.254.3 Member Data Documentation

5.254.3.1 bprofiling

```
MHAParser::bool_t mhachain::chain_base_t::bprofiling [protected]
```

5.254.3.2 algos

```
MHAParser::vstring_t mhachain::chain_base_t::algos [protected]
```

5.254.3.3 old_algos `std::vector<std::string> mhachain::chain_base_t::old_algos [private]`

5.254.3.4 patchbay `MHAEvents::patchbay_t< mhachain::chain_base_t > mhachain::chain_base_t::patchbay [private]`

5.254.3.5 cfin `mhaconfig_t mhachain::chain_base_t::cfin [private]`

5.254.3.6 cfout `mhaconfig_t mhachain::chain_base_t::cfout [private]`

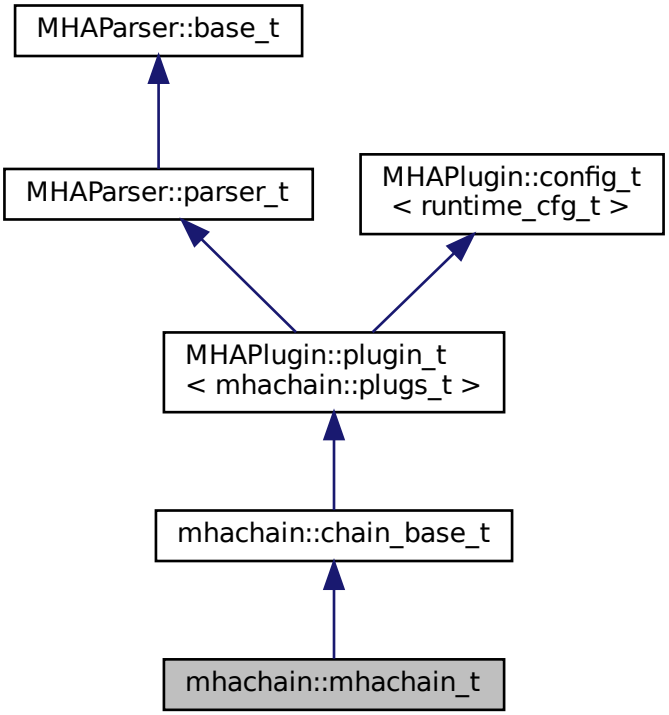
5.254.3.7 b_prepared `bool mhachain::chain_base_t::b_prepared [private]`

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

- **mha_generic_chain.h**
- **mha_generic_chain.cpp**

5.255 mhachain::mhachain_t Class Reference

Inheritance diagram for mhachain::mhachain_t:



Public Member Functions

- **mhachain_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)

Additional Inherited Members

5.255.1 Constructor & Destructor Documentation

5.255.1.1 mhachain_t() `mhachain::mhachain_t::mhachain_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

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

- **mhachain.cpp**

5.256 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, **MHA_AC::algo_comm_t** &iac, 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**
- **MHA_AC::algo_comm_t** & **ac**
- **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**

5.256.1 Constructor & Destructor Documentation

5.256.1.1 plugs_t() mhachain::plugs_t::plugs_t (

```
std::vector< std::string > algos,
 mhaconfig_t cfin,
 mhaconfig_t cfout,
bool do_prepare,
 MHAParser::parser_t & p,
 MHA_AC::algo_comm_t & iac,
bool use_profiling )
```

5.256.1.2 ~plugs_t() mhachain::plugs_t::~plugs_t ( )

5.256.2 Member Function Documentation

5.256.2.1 prepare() void mhachain::plugs_t::prepare (

```
mhaconfig_t & tf )
```

5.256.2.2 release() void mhachain::plugs_t::release (

```
void  )
```

5.256.2.3 process() void mhachain::plugs_t::process (

```
mha_wave_t * win,
mha_spec_t * sin,
mha_wave_t ** wout,
mha_spec_t ** sout )
```

5.256.2.4 prepared() bool mhachain::plugs_t::prepared ( ) const [inline]

5.256.2.5 alloc_plugs() void mhachain::plugs_t::alloc_plugs (std::vector< std::string > *algos*) [private]

5.256.2.6 cleanup_plugs() void mhachain::plugs_t::cleanup_plugs ( ) [private]

5.256.2.7 update_proc_load() void mhachain::plugs_t::update_proc_load ( ) [private]

5.256.3 Member Data Documentation

5.256.3.1 b_prepared bool mhachain::plugs_t::b_prepared [private]

5.256.3.2 algos std::vector< **PluginLoader::mhapluginloader_t*** > mhachain::plugs_t::algos [private]

5.256.3.3 parser **MHAParser::parser_t**& mhachain::plugs_t::parser [private]

5.256.3.4 ac **MHA_AC::algo_comm_t**& mhachain::plugs_t::ac [private]

5.256.3.5 profiling **MHAParser::parser_t** mhachain::plugs_t::profiling [private]

5.256.3.6 prof_algos

MHAParser::vstring_mon_t mhachain::plugs_t::prof_algos [private]

5.256.3.7 prof_init

MHAParser::vfloat_mon_t mhachain::plugs_t::prof_init [private]

5.256.3.8 prof_prepare

MHAParser::vfloat_mon_t mhachain::plugs_t::prof_prepare [private]

5.256.3.9 prof_release

MHAParser::vfloat_mon_t mhachain::plugs_t::prof_release [private]

5.256.3.10 prof_process

MHAParser::vfloat_mon_t mhachain::plugs_t::prof_process [private]

5.256.3.11 prof_process_tt

MHAParser::float_mon_t mhachain::plugs_t::prof_process↩_tt [private]

5.256.3.12 prof_process_load

MHAParser::vfloat_mon_t mhachain::plugs_t::prof_↩process_load [private]

5.256.3.13 proc_cnt

unsigned int mhachain::plugs_t::proc_cnt [private]

5.256.3.14 prof_cfg

`mhaconfig_t mhachain::plugs_t::prof_cfg [private]`

5.256.3.15 prof_load_con

`MHAEvents::connector_t< mhachain::plugs_t> mhachain::plugs_t::prof_load_con [private]`

5.256.3.16 prof_tt_con

`MHAEvents::connector_t< mhachain::plugs_t> mhachain::plugs_t::prof_tt_con [private]`

5.256.3.17 b_use_profiling

`bool mhachain::plugs_t::b_use_profiling [private]`

5.256.3.18 tictoc

`mha_platform_tictoc_t mhachain::plugs_t::tictoc [private]`

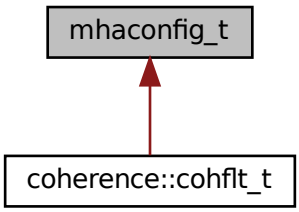
The documentation for this class was generated from the following files:

- **mha_generic_chain.h**
- **mha_generic_chain.cpp**

5.257 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.

5.257.1 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`).

5.257.2 Member Data Documentation

5.257.2.1 channels `unsigned int mhaconfig_t::channels`

Number of audio channels.

5.257.2.2 domain `unsigned int mhaconfig_t::domain`

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.257.2.3 fragsize `unsigned int mhaconfig_t::fragsize`

Fragment size of waveform data.

5.257.2.4 wndlen `unsigned int mhaconfig_t::wndlen`

Window length of spectral data.

5.257.2.5 fftlen `unsigned int mhaconfig_t::fftlen`

FFT length of spectral data.

5.257.2.6 srate `mha_real_t mhaconfig_t::srate`

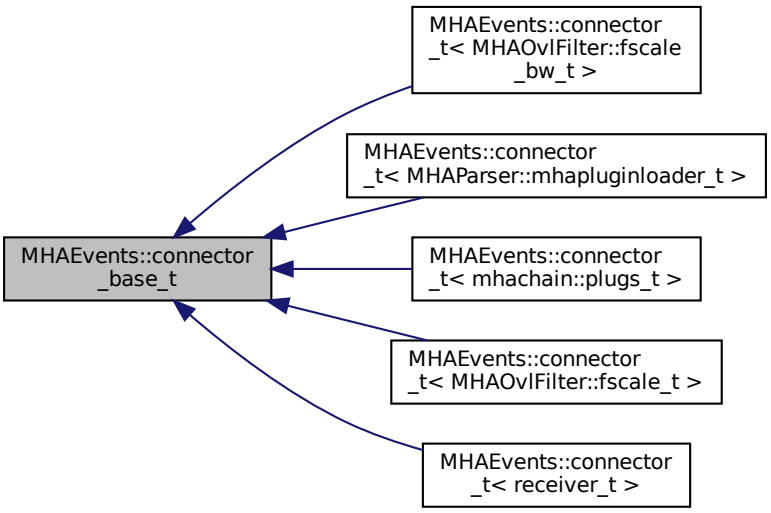
Sampling rate in Hz.

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

- **mha.hh**

5.258 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**

5.258.1 Constructor & Destructor Documentation

5.258.1.1 connector_base_t() `MHAEvents::connector_base_t::connector_base_t ( )`

5.258.1.2 ~connector_base_t() `MHAEvents::connector_base_t::~connector_base_t ( ) [virtual]`

5.258.2 Member Function Documentation

5.258.2.1 emit_event() `[1/3]` `void MHAEvents::connector_base_t::emit_event ( ) [virtual]`

Reimplemented in **MHAEvents::connector_t**< **receiver_t** > (p. 1002), **MHAEvents::connector_t**< **MHAOvlFilter::fscale_bw_t** > (p. 1002), **MHAEvents::connector_t**< **MHAParser::mhapluginloader_t** > (p. 1002), **MHAEvents::connector_t**< **mhachain::plugs_t** > (p. 1002), and **MHAEvents::connector_t**< **MHAOvlFilter::fscale_t** > (p. 1002).

5.258.2.2 emit_event() [2/3] void MHAEvents::connector_base_t::emit_event (const std::string &) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 1003), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 1003), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 1003), **MHAEvents::connector_t< mhachain::plugs_t >** (p. 1003), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 1003).

5.258.2.3 emit_event() [3/3] void MHAEvents::connector_base_t::emit_event (const std::string & , unsigned int , unsigned int) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 1003), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 1003), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 1003), **MHAEvents::connector_t< mhachain::plugs_t >** (p. 1003), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 1003).

5.258.2.4 emitter_die() void MHAEvents::connector_base_t::emitter_die ( )

5.258.3 Member Data Documentation

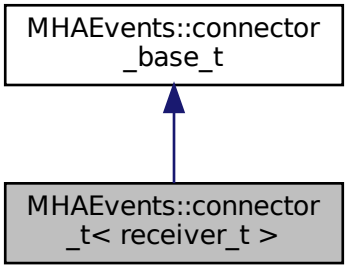
5.258.3.1 emitter_is_alive bool MHAEvents::connector_base_t::emitter_is_alive [protected]

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

- **mha_event_emitter.h**
- **mha_events.cpp**

5.259 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

5.259.1 Constructor & Destructor Documentation

5.259.1.1 connector_t() [1/3] template<class receiver_t >

```
MHAEvents::connector_t< receiver_t >::  connector_t (
            emitter_t * e,
           receiver_t * r,
           void(receiver_t::*)() rfun )
```

5.259.1.2 connector_t() [2/3] template<class receiver_t >

```
MHAEvents::connector_t< receiver_t >::  connector_t (
            emitter_t * e,
           receiver_t * r,
           void(receiver_t::*)(const std::string &) rfun )
```

5.259.1.3 connector_t() [3/3] 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 )
```

5.259.1.4 ~connector_t() template<class receiver_t >

```
MHAEvents::connector_t< receiver_t >::~ connector_t
```

5.259.2 Member Function Documentation

5.259.2.1 emit_event() [1/3] template<class receiver_t >

void **MHAEvents::connector_t**< receiver_t >::emit_event [private], [virtual]

Reimplemented from **MHAEvents::connector_base_t** (p. 999).

5.259.2.2 emit_event() [2/3] 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. 999).

5.259.2.3 emit_event() [3/3] 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. 1000).

5.259.3 Member Data Documentation

5.259.3.1 emitter template<class receiver_t >

emitter_t* **MHAEvents::connector_t**< receiver_t >::emitter [private]

5.259.3.2 receiver template<class receiver_t >

receiver_t* **MHAEvents::connector_t**< receiver_t >::receiver [private]

5.259.3.3 eventhandler template<class receiver_t >

void(receiver_t::* **MHAEvents::connector_t**< receiver_t >::eventhandler) () [private]

5.259.3.4 eventhandler_s `template<class receiver_t >`

`void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_s) (const std::string &) [private]`

5.259.3.5 eventhandler_suu `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**

5.260 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**

5.260.1 Detailed Description

Class for emitting openMHA events.

Use the template claas **MHAEvents::patchbay_t** (p. 1006) for connecting to an emitter.

5.260.2 Constructor & Destructor Documentation

5.260.2.1 ~emitter_t() `MHAEvents::emitter_t::~emitter_t ( )`

5.260.3 Member Function Documentation

5.260.3.1 operator()() [1/3] `void MHAEvents::emitter_t::operator() ( )`

Emit an event without parameter.

5.260.3.2 operator()() [2/3]

```
void MHAEvents::emitter_t::operator() (
    const std::string & arg )
```

Emit an event with string parameter.

5.260.3.3 operator()() [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.

5.260.3.4 connect()

```
void MHAEvents::emitter_t::connect (
    connector_base_t * c )
```

5.260.3.5 disconnect()

```
void MHAEvents::emitter_t::disconnect (
    connector_base_t * c )
```

5.260.4 Member Data Documentation

5.260.4.1 connections `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**

5.261 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**

5.261.1 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. 1007) 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.

5.261.2 Constructor & Destructor Documentation

5.261.2.1 ~patchbay_t() template<class receiver_t >

`MHAEvents::patchbay_t< receiver_t >::~ patchbay_t`

5.261.3 Member Function Documentation

5.261.3.1 connect() [1/3] 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

5.261.3.2 connect() [2/3] 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

5.261.3.3 connect() [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 )
```

5.261.4 Member Data Documentation

5.261.4.1 cons 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**

5.262 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**

5.262.1 Constructor & Destructor Documentation

5.262.1.1 adapt_filter_param_t() `MHAFilter::adapt_filter_param_t::adapt_filter_param_t (`

```
            mha_real_t imu,
           bool ierr_in )
```

5.262.2 Member Data Documentation

5.262.2.1 mu `mha_real_t MHAFilter::adapt_filter_param_t::mu`

5.262.2.2 err_in `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**

5.263 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**

5.263.1 Constructor & Destructor Documentation

5.263.1.1 adapt_filter_state_t() MHAFilter::adapt_filter_state_t::adapt_filter_↩ state_t (

```
int ntaps,
int nchannels )
```

5.263.2 Member Function Documentation

5.263.2.1 filter() 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 )
```

5.263.3 Member Data Documentation

5.263.3.1 ntaps `int MHAFilter::adapt_filter_state_t::ntaps [private]`

5.263.3.2 nchannels `int MHAFilter::adapt_filter_state_t::nchannels [private]`

5.263.3.3 W **`MHASignal::waveform_t`** `MHAFilter::adapt_filter_state_t::W [private]`

5.263.3.4 X **`MHASignal::waveform_t`** `MHAFilter::adapt_filter_state_t::X [private]`

5.263.3.5 od **`MHASignal::waveform_t`** `MHAFilter::adapt_filter_state_t::od [private]`

5.263.3.6 oy **`MHASignal::waveform_t`** `MHAFilter::adapt_filter_state_t::oy [private]`

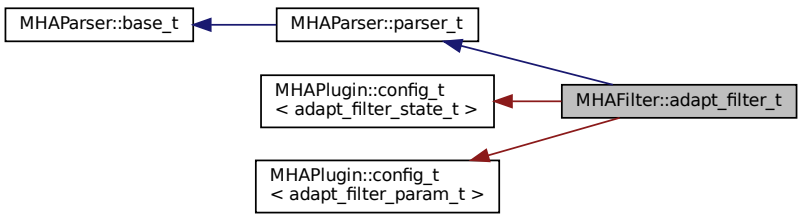
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.264 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

5.264.1 Detailed Description

Adaptive filter.

5.264.2 Constructor & Destructor Documentation

5.264.2.1 adapt_filter_t() MHAFilter::adapt_filter_t::adapt_filter_t (std::string *help*)

5.264.3 Member Function Documentation

5.264.3.1 filter() void MHAFilter::adapt_filter_t::filter (**mha_wave_t** *y*, **mha_wave_t** *e*, **mha_wave_t** *x*, **mha_wave_t** *d*)

5.264.3.2 set_channelcnt() void MHAFilter::adapt_filter_t::set_channelcnt (unsigned int *nch*)

5.264.3.3 update_mu() void MHAFilter::adapt_filter_t::update_mu ( ) [private]

5.264.3.4 update_ntaps() void MHAFilter::adapt_filter_t::update_ntaps ( ) [private]

5.264.4 Member Data Documentation

5.264.4.1 mu **MHAParser::float_t** MHAFilter::adapt_filter_t::mu [private]

5.264.4.2 ntaps

`MHAParser::int_t MHAFilter::adapt_filter_t::ntaps [private]`

5.264.4.3 err_in

`MHAParser::bool_t MHAFilter::adapt_filter_t::err_in [private]`

5.264.4.4 connector

`MHAEvents::patchbay_t< adapt_filter_t> MHAFilter::adapt_filter_t::connector [private]`

5.264.4.5 nchannels

`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**

5.265 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**

5.265.1 Detailed Description

A class that does polyphase resampling and takes into account block processing.

5.265.2 Constructor & Destructor Documentation

5.265.2.1 blockprocessing_polyphase_resampling_t() 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 a source_srate
target_srate	Target sampling rate / Hz
target_fragsize	Fragment size of produced audio blocks / frames a target_srate
nyquist_ratio	Low pass filter cutoff frequency relative to the nyqu frequency of the smaller of the two sampling rates Example values: 0.8, 0.9
irslen	Impulse response length used for low pass filterin
nchannels	Number of audio channels

Parameters

add_delay	To avoid underruns, a delay is generally necessary round trip block size adaptations. It is only necess to add this delay to one of the two resampling chai Set this parameter to true for the first resampling c of a round trip pair. It will add the necessary delay, calculate the size of the ring buffer appropriately, V set to false, only the ringbuffer size will be set sufficiently.

5.265.2.2 ~blockprocessing_polyphase_resampling_t() `virtual MHAFilter::blockprocessing↩ _polyphase_resampling_t::~blockprocessing_polyphase_resampling_t ( ) [inline], [virtual]`

5.265.3 Member Function Documentation

5.265.3.1 write() `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. 906)*	Raises exception if there is not enough room, if t number of channels does not match, or if the nur frames is not equal to the number specified in the constructor

5.265.3.2 read() `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. 906)*	Raises exception if there is not enough input sig the number of channels of frames does not matc

5.265.3.3 can_read() `bool MHAFilter::blockprocessing_polyphase_resampling_t::can_read ( ) const [inline]`

Checks if the resampling ring buffer can produce another output signal block.

5.265.4 Member Data Documentation

5.265.4.1 resampling `polyphase_resampling_t* MHAFilter::blockprocessing_polyphase_resampling_t::resampling [private]`

5.265.4.2 fragsize_in `unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_in [private]`

5.265.4.3 fragsize_out `unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_out [private]`

5.265.4.4 num_channels `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**

5.266 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**

5.266.1 Detailed Description

Complex bandpass filter.

5.266.2 Constructor & Destructor Documentation

5.266.2.1 complex_bandpass_t() 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

5.266.3 Member Function Documentation

5.266.3.1 creator_A() 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]
```

5.266.3.2 creator_B()

```
std::vector< mha_complex_t > MHAFilter::complex_bandpass_t::creator_B (
            std::vector< mha_complex_t > A,
            unsigned int order ) [static]
```

5.266.3.3 set_state() [1/3]

```
void MHAFilter::complex_bandpass_t::set_state (
            mha_real_t val )
```

5.266.3.4 set_state() [2/3]

```
void MHAFilter::complex_bandpass_t::set_state (
            std::vector< mha_real_t > val )
```

5.266.3.5 set_state() [3/3]

```
void MHAFilter::complex_bandpass_t::set_state (
            mha_complex_t val )
```

5.266.3.6 set_weights()

```
void MHAFilter::complex_bandpass_t::set_weights (
            std::vector< mha_complex_t > new_B )
```

Allow to modify the input weights at a later stage.

5.266.3.7 get_weights()

```
std::vector< mha_complex_t> MHAFilter::complex_bandpass_t::get_weights ( ) const [inline]
```

5.266.3.8 filter() [1/4]

```
void MHAFilter::complex_bandpass_t::filter (
            const mha_wave_t & X,
            mha_spec_t & Y ) [inline]
```

Filter method for real value input.

5.266.3.9 filter() [2/4] 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.

5.266.3.10 filter() [3/4] void MHAFilter::complex_bandpass_t::filter (const **mha_spec_t** & *X*, **mha_spec_t** & *Y*) [inline]

Filter method for complex value input.

5.266.3.11 filter() [4/4] 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.

5.266.3.12 inspect() std::string MHAFilter::complex_bandpass_t::inspect ( ) const [inline]

5.266.4 Member Data Documentation

5.266.4.1 A_ std::vector< **mha_complex_t**> MHAFilter::complex_bandpass_t::A_ [private]

5.266.4.2 B_ std::vector< **mha_complex_t**> MHAFilter::complex_bandpass_t::B_ [private]

5.266.4.3 Yn `std::vector< mha_complex_t> MHAFilter::complex_bandpass_t::Yn [private]`

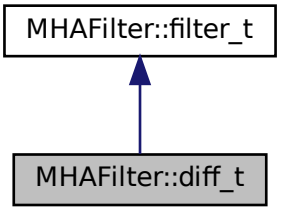
The documentation for this class was generated from the following files:

- **complex_filter.h**
- **complex_filter.cpp**

5.267 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

5.267.1 Detailed Description

Differentiator class (non-normalized)

5.267.2 Constructor & Destructor Documentation

5.267.2.1 diff_t() `MHAFilter::diff_t::diff_t (`
`unsigned int ch )`

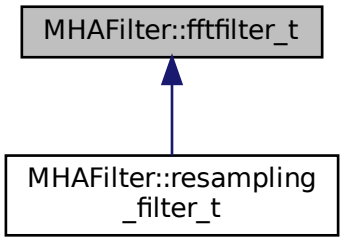
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.268 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**

5.268.1 Detailed Description

FFT based FIR filter implementation.

The maximal number of coefficients can be FFT length - fragsize + 1.

5.268.2 Constructor & Destructor Documentation

5.268.2.1 fftfilter_t() `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.

5.268.2.2 ~fftfilter_t() MHAFilter::fftfilter_t::~fftfilter_t ( )

5.268.3 Member Function Documentation

5.268.3.1 update_coeffs() 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).

5.268.3.2 filter() [1/3] 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

pwIn	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.

5.268.3.3 filter() [2/3]

```
void MHAFilter::fftfilter_t::filter (
    const mha_wave_t * pwIn,
    mha_wave_t ** ppwOut )
```

Apply filter to waveform fragment, without changing the coefficients.

Parameters

pwIn	Input signal pointer.

Return values

ppwOut	Pointer to output signal pointer, will be set to a valid signal

5.268.3.4 filter() [3/3]

```
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

pwIn	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.

5.268.4 Member Data Documentation

5.268.4.1 fragsize

`unsigned int MHAFilter::fftfilter_t::fragsize [private]`

5.268.4.2 channels

`unsigned int MHAFilter::fftfilter_t::channels [private]`

5.268.4.3 fftlen

`unsigned int MHAFilter::fftfilter_t::fftlen [private]`

5.268.4.4 wInput_fft

`MHASignal::waveform_t` `MHAFilter::fftfilter_t::wInput_fft [private]`

5.268.4.5 wInput

`mha_wave_t` `MHAFilter::fftfilter_t::wInput [private]`

5.268.4.6 wOutput_fft

`MHASignal::waveform_t` `MHAFilter::fftfilter_t::wOutput_fft [private]`

5.268.4.7 wOutput

`mha_wave_t MHAFilter::fftfilter_t::wOutput [private]`

5.268.4.8 sInput

`MHASignal::spectrum_t MHAFilter::fftfilter_t::sInput [private]`

5.268.4.9 sWeights

`MHASignal::spectrum_t MHAFilter::fftfilter_t::sWeights [private]`

5.268.4.10 wIRS_fft

`MHASignal::waveform_t MHAFilter::fftfilter_t::wIRS_fft [private]`

5.268.4.11 fft

`mha_fft_t MHAFilter::fftfilter_t::fft [private]`

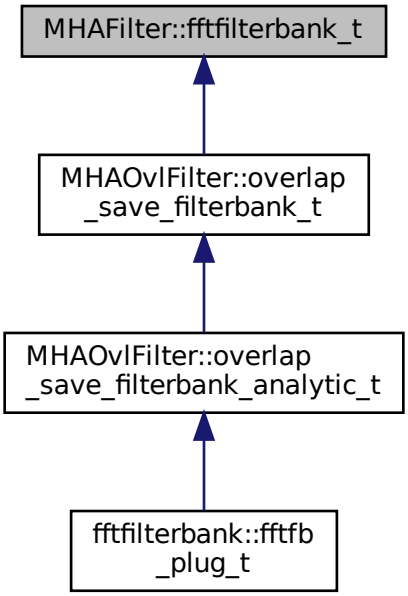
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.269 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**

5.269.1 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.

5.269.2 Constructor & Destructor Documentation

5.269.2.1 fftfilterbank_t()

```
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.

5.269.2.2 ~fftfilterbank_t() `MHAFilter::fftfilterbank_t::~fftfilterbank_t ( )`

5.269.3 Member Function Documentation

5.269.3.1 update_coeffs() `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.

5.269.3.2 filter() [1/2] `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 coeffi-cients

5.269.3.3 filter() [2/2] 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

5.269.3.4 get_irs() const mha_wave_t* MHAFilter::fftfilterbank_t::get_irs ( ) const [inline]

Return the current IRS.

5.269.4 Member Data Documentation

5.269.4.1 fragsize unsigned int MHAFilter::fftfilterbank_t::fragsize [private]

5.269.4.2 inputchannels

`unsigned int MHAFilter::fftfilterbank_t::inputchannels [private]`

5.269.4.3 firchannels

`unsigned int MHAFilter::fftfilterbank_t::firchannels [private]`

5.269.4.4 outputchannels

`unsigned int MHAFilter::fftfilterbank_t::outputchannels [private]`

5.269.4.5 fftlen

`unsigned int MHAFilter::fftfilterbank_t::fftlen [private]`

5.269.4.6 hw

MHASignal::waveform_t `MHAFilter::fftfilterbank_t::hw [private]`

5.269.4.7 Hs

MHASignal::spectrum_t `MHAFilter::fftfilterbank_t::Hs [private]`

5.269.4.8 xw

MHASignal::waveform_t `MHAFilter::fftfilterbank_t::xw [private]`

5.269.4.9 Xs

MHASignal::spectrum_t `MHAFilter::fftfilterbank_t::Xs [private]`

5.269.4.10 yw

MHASignal::waveform_t `MHAFilter::fftfilterbank_t::yw [private]`

5.269.4.11 Ys `MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Ys [private]`

5.269.4.12 yw_temp `MHASignal::waveform_t MHAFilter::fftfilterbank_t::yw_temp [private]`

5.269.4.13 tail `MHASignal::waveform_t MHAFilter::fftfilterbank_t::tail [private]`

5.269.4.14 fft `mha_fft_t MHAFilter::fftfilterbank_t::fft [private]`

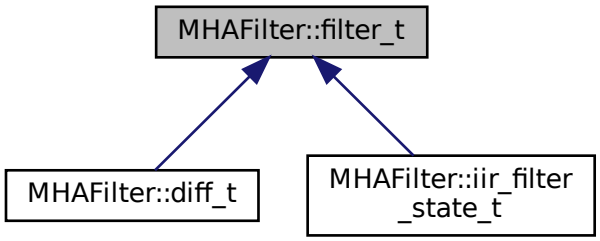
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.270 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)

  Copy constructor.
- **filter_t** & **operator=** (const **MHAFilter::filter_t** &)=delete

  Assignment operator is not implemented.
- ~**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**

5.270.1 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. 985) structs. The filter coefficients can be directly accessed.

5.270.2 Constructor & Destructor Documentation

5.270.2.1 filter_t() [1/3]

```
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

5.270.2.2 filter_t() [2/3]

```
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.

Parameters

vB	Non-recursive coefficients.

5.270.2.3 filter_t() [3/3] MHAFilter::filter_t::filter_t (const **MHAFilter::filter_t** & *src*)

Copy constructor.

5.270.2.4 ~filter_t() MHAFilter::filter_t::~filter_t ( )

5.270.3 Member Function Documentation

5.270.3.1 operator=() **filter_t**& MHAFilter::filter_t::operator= (const **MHAFilter::filter_t** &) [delete]

Assignment operator is not implemented.

Use copy constructor instead.

5.270.3.2 filter() [1/3] 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

5.270.3.3 filter() [2/3]

```
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

5.270.3.4 filter() [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

5.270.3.5 get_len_A()

```
unsigned int MHAFilter::filter_t::get_len_A ( ) const [inline]
```

Return length of recursive coefficients.

5.270.3.6 get_len_B() `unsigned int MHAFilter::filter_t::get_len_B ( ) const [inline]`

Return length of non-recursive coefficients.

5.270.4 Member Data Documentation

5.270.4.1 A `double* MHAFilter::filter_t::A`

Pointer to recursive coefficients.

5.270.4.2 B `double* MHAFilter::filter_t::B`

Pointer to non-recursive coefficients.

5.270.4.3 len_A `unsigned int MHAFilter::filter_t::len_A [private]`

5.270.4.4 len_B `unsigned int MHAFilter::filter_t::len_B [private]`

5.270.4.5 len `unsigned int MHAFilter::filter_t::len [private]`

5.270.4.6 channels `unsigned int MHAFilter::filter_t::channels [private]`

5.270.4.7 state `double* MHAFilter::filter_t::state [private]`

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.271 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_**

5.271.1 Detailed Description

Class for gammatone filter.

5.271.2 Constructor & Destructor Documentation

5.271.2.1 gamma_flt_t()

```
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.

5.271.2.2 ~gamma_flt_t()

```
MHAFilter::gamma_flt_t::~gamma_flt_t ( )
```

5.271.3 Member Function Documentation

5.271.3.1 operator()() [1/3]

```
void MHAFilter::gamma_flt_t::operator() (
    mha_wave_t & X,
    mha_spec_t & Y ) [inline]
```

Filter method.

5.271.3.2 operator()() [2/3]

```
void MHAFilter::gamma_flt_t::operator() (
    mha_wave_t & X,
    mha_wave_t & Yre,
    mha_wave_t & Yim ) [inline]
```

Filter method.

5.271.3.3 operator()() [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.

5.271.3.4 phase_correction()

```
void MHAFilter::gamma_flt_t::phase_correction (
    unsigned int desired_delay,
    unsigned int inchannels )
```

5.271.3.5 set_weights() [1/2]

```
void MHAFilter::gamma_flt_t::set_weights (
    std::vector< mha_complex_t > new_B )
```

5.271.3.6 set_weights() [2/2]

```
void MHAFilter::gamma_flt_t::set_weights (
    unsigned int stage,
    std::vector< mha_complex_t > new_B )
```

5.271.3.7 get_weights() [1/2]

```
std::vector< mha_complex_t> MHAFilter::gamma_flt_t↩
::get_weights ( ) const [inline]
```

5.271.3.8 get_weights() [2/2]

```
std::vector< mha_complex_t> MHAFilter::gamma_flt_t↩
::get_weights (
            unsigned int stage ) const  [inline]
```

5.271.3.9 get_resynthesis_gain()

```
std::vector< mha_real_t> MHAFilter::gamma_flt_t↩
::get_resynthesis_gain ( ) const  [inline]
```

5.271.3.10 reset_state()

```
void MHAFilter::gamma_flt_t::reset_state ( )
```

5.271.3.11 get_A()

```
const std::vector< mha_complex_t>& MHAFilter::gamma_flt_t↩
::get_A ( )  [inline]
```

5.271.3.12 inspect()

```
std::string MHAFilter::gamma_flt_t::inspect ( ) const  [inline]
```

5.271.4 Member Data Documentation

5.271.4.1 A

```
std::vector< mha_complex_t> MHAFilter::gamma_flt_t::A  [private]
```

5.271.4.2 GF

```
std::vector< complex_bandpass_t> MHAFilter::gamma_flt_t::GF  [private]
```

5.271.4.3 delay

```
MHASignal::delay_t* MHAFilter::gamma_flt_t::delay  [private]
```

5.271.4.4 envelope_delay

`std::vector<int> MHAFilter::gamma_flt_t::envelope_delay [private]`

5.271.4.5 resynthesis_gain

`std::vector< mha_real_t> MHAFilter::gamma_flt_t::resynthesis_gain [private]`

5.271.4.6 cf_

`std::vector< mha_real_t> MHAFilter::gamma_flt_t::cf_ [private]`

5.271.4.7 bw_

`std::vector< mha_real_t> MHAFilter::gamma_flt_t::bw_ [private]`

5.271.4.8 srate_

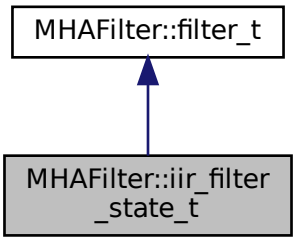
`mha_real_t MHAFilter::gamma_flt_t::srate_ [private]`

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

- **complex_filter.h**
- **complex_filter.cpp**

5.272 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

5.272.1 Constructor & Destructor Documentation

5.272.1.1 iir_filter_state_t() `MHAFilter::iir_filter_state_t::iir_filter_state_t (`

```
unsigned int channels,
std::vector< float > cf_A,
std::vector< float > cf_B )
```

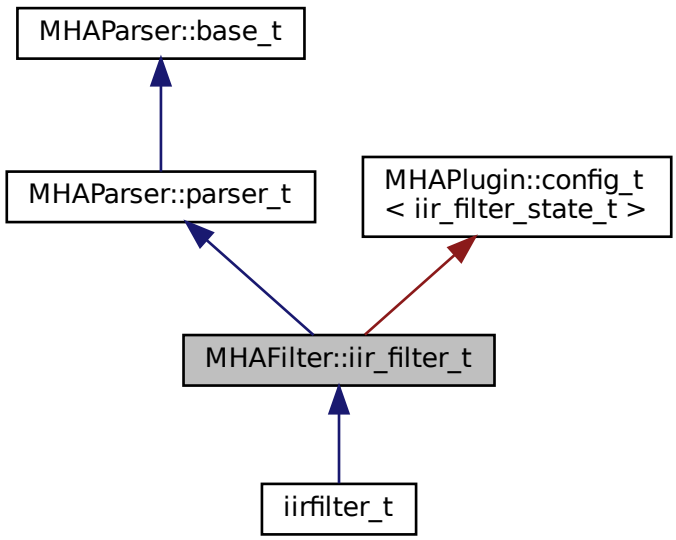
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.273 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

5.273.1 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. 1246), 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.

5.273.2 Constructor & Destructor Documentation

5.273.2.1 iir_filter_t()

```
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.

5.273.3 Member Function Documentation

5.273.3.1 filter() [1/2]

```
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.

Parameters

x	Pointer to input signal holder. Number of channels has to be the same as given to the constructor, or to the **resize** (p. 1048) method.

5.273.3.2 filter() [2/2]

```
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. 1048) method.

Returns

the filtered result sample.

5.273.3.3 resize()

```
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.

5.273.3.4 update_filter()

```
void MHAFilter::iir_filter_t::update_filter ( ) [private]
```

5.273.4 Member Data Documentation

5.273.4.1 A `MHAParser::vfloat_t MHAFilter::iir_filter_t::A [private]`

5.273.4.2 B `MHAParser::vfloat_t MHAFilter::iir_filter_t::B [private]`

5.273.4.3 connector `MHAEvents::patchbay_t< iir_filter_t> MHAFilter::iir_filter_t::connector [private]`

5.273.4.4 nchannels `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**

5.274 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**

5.274.1 Detailed Description

First order recursive filter.

5.274.2 Constructor & Destructor Documentation

5.274.2.1 iir_ord1_real_t() [1/2]

```
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)

5.274.2.2 iir_ord1_real_t() [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)

5.274.3 Member Function Documentation

5.274.3.1 set_state() [1/3]

```
void MHAFilter::iir_ord1_real_t::set_state (
    mha_real_t val )
```

5.274.3.2 set_state() [2/3]

```
void MHAFilter::iir_ord1_real_t::set_state (
    std::vector< mha_real_t > val )
```

5.274.3.3 set_state() [3/3]

```
void MHAFilter::iir_ord1_real_t::set_state (
    mha_complex_t val )
```

5.274.3.4 operator()() [1/5]

```
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.

5.274.3.5 operator()() [2/5]

```
mha_complex_t MHAFilter::iir_ord1_real_t::operator() (
    unsigned int ch,
    mha_complex_t x ) [inline]
```

Filter method for complex input, one element.

5.274.3.6 operator()() [3/5]

```
void MHAFilter::iir_ord1_real_t::operator() (
    const mha_wave_t & X,
    mha_wave_t & Y ) [inline]
```

Filter method for real value input.

5.274.3.7 operator()() [4/5]

```
void MHAFilter::iir_ord1_real_t::operator() (
    const mha_spec_t & X,
    mha_spec_t & Y ) [inline]
```

Filter method for complex value input.

5.274.3.8 operator()() [5/5] 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.

5.274.4 Member Data Documentation

5.274.4.1 A_ std::vector< mha_real_t> MHAFilter::iir_ord1_real_t::A_ [private]

5.274.4.2 B_ std::vector< mha_real_t> MHAFilter::iir_ord1_real_t::B_ [private]

5.274.4.3 Yn std::vector< mha_complex_t> MHAFilter::iir_ord1_real_t::Yn [private]

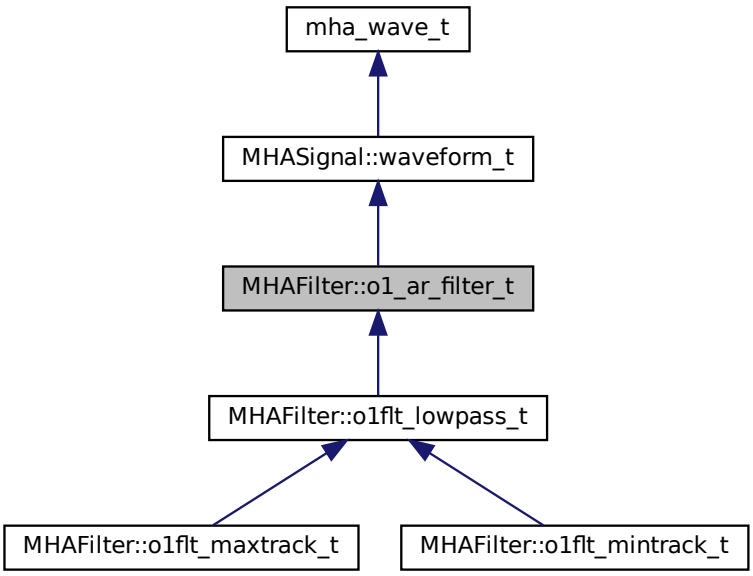
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.275 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. 985) 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

5.275.1 Detailed Description

First order attack-release lowpass filter.

This filter is the base of first order lowpass filter, maximum tracker and minimum tracker.

5.275.2 Constructor & Destructor Documentation

5.275.2.1 o1_ar_filter_t()

```
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. 1414).

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)

5.275.3 Member Function Documentation

5.275.3.1 set_tau_attack() 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

5.275.3.2 set_tau_release() 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

5.275.3.3 operator()() [1/2] **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

5.275.3.4 operator()() [2/2] 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. 985) data.

Parameters

in	Input signal
out	Output signal

The number of channels must match the number of filter bands.

5.275.4 Member Data Documentation

5.275.4.1 c1_a **MHASignal::waveform_t** MHAFilter::o1_ar_filter_t::c1_a [protected]

5.275.4.2 c2_a **MHASignal::waveform_t** MHAFilter::o1_ar_filter_t::c2_a [protected]

5.275.4.3 c1_r **MHASignal::waveform_t** MHAFilter::o1_ar_filter_t::c1_r [protected]

5.275.4.4 c2_r **MHASignal::waveform_t** MHAFilter::o1_ar_filter_t::c2_r [protected]

5.275.4.5 fs **mha_real_t** MHAFilter::o1_ar_filter_t::fs [protected]

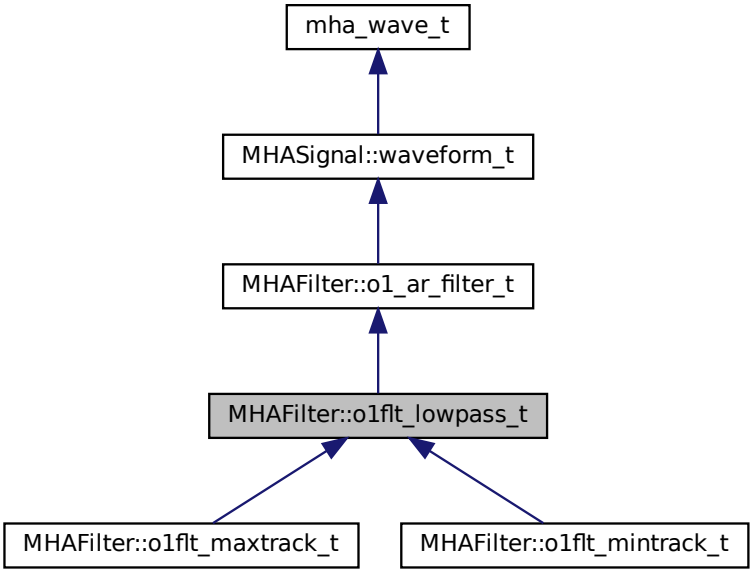
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.276 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.
- **o1flt_lowpass_t** (const std::vector< **mha_real_t** > &tau, **mha_real_t** **fs**, const std::vector< **mha_real_t** > &startval)
  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

5.276.1 Detailed Description

First order low pass filter.

5.276.2 Constructor & Destructor Documentation

5.276.2.1 o1flt_lowpass_t() [1/2] 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

5.276.2.2 o1flt_lowpass_t() [2/2] MHAFilter::o1flt_lowpass_t::o1flt_lowpass_t (

```
const std::vector< mha_real_t > & tau,
mha_real_t fs,
const std::vector< mha_real_t > & startval )
```

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

5.276.3 Member Function Documentation

5.276.3.1 set_tau() [1/2] void MHAFilter::o1flt_lowpass_t::set_tau (unsigned int *ch*, **mha_real_t** *tau*)

change the time constant in one channel

5.276.3.2 set_tau() [2/2] void MHAFilter::o1flt_lowpass_t::set_tau (**mha_real_t** *tau*)

set time constant in all channels to tau

5.276.3.3 get_c1() **mha_real_t** MHAFilter::o1flt_lowpass_t::get_c1 (unsigned int *ch*) const [inline]

5.276.3.4 get_last_output() **mha_real_t** MHAFilter::o1flt_lowpass_t::get_last_output (unsigned int *ch*) const [inline]

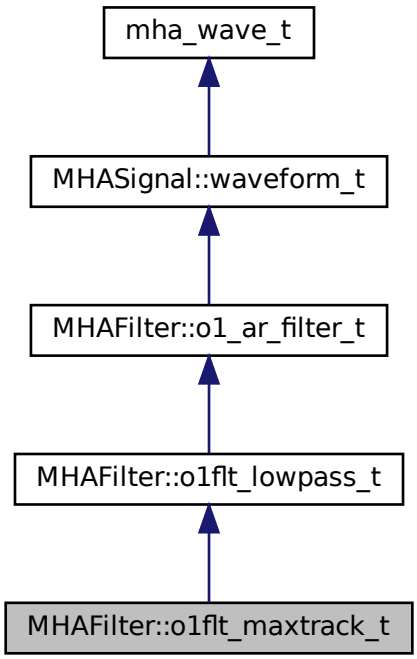
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.277 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.
- **o1flt_maxtrack_t** (const std::vector< **mha_real_t** > &tau, **mha_real_t** **fs**, const std::vector< **mha_real_t** > &startval)
- 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

5.277.1 Detailed Description

First order maximum tracker.

5.277.2 Constructor & Destructor Documentation

5.277.2.1 o1flt_maxtrack_t() [1/2]

```
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

5.277.2.2 o1flt_maxtrack_t() [2/2]

```
MHAFilter::o1flt_maxtrack_t::o1flt_maxtrack_t (
    const std::vector< mha_real_t > & tau,
    mha_real_t fs,
    const std::vector< mha_real_t > & startval )
```

5.277.3 Member Function Documentation

5.277.3.1 set_tau() [1/2]

```
void MHAFilter::o1flt_maxtrack_t::set_tau (
    unsigned int ch,
    mha_real_t tau )
```

change the time constant in one channel

5.277.3.2 set_tau() [2/2] void MHAFilter::o1flt_maxtrack_t::set_tau (mha_real_t tau)

set time constant in all channels to tau

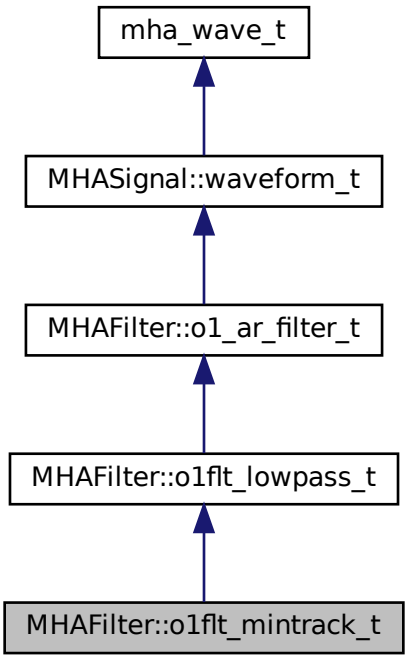
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.278 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)
- **o1flt_mintrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, const std::vector< **mha_real_t** > &)
- 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

5.278.1 Detailed Description

First order minimum tracker.

5.278.2 Constructor & Destructor Documentation

5.278.2.1 o1flt_mintrack_t() [1/2]

```
MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (
    const std::vector< mha_real_t > & tau,
    mha_real_t fs_,
    mha_real_t startval = 0 )
```

5.278.2.2 o1flt_mintrack_t() [2/2]

```
MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (
    const std::vector< mha_real_t > & tau,
    mha_real_t fs,
    const std::vector< mha_real_t > & startval )
```

5.278.3 Member Function Documentation

5.278.3.1 set_tau() [1/2]

```
void MHAFilter::o1flt_mintrack_t::set_tau (
    unsigned int ch,
    mha_real_t tau )
```

change the time constant in one channel

5.278.3.2 set_tau() [2/2]

```
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**

5.279 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.

5.279.1 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.

5.279.2 Constructor & Destructor Documentation

5.279.2.1 partitioned_convolution_t() `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.

5.279.2.2 ~partitioned_convolution_t() MHAFilter::partitioned_convolution_t::~partitioned_convolution_t ( )

Free fftw resource allocated in constructor.

5.279.3 Member Function Documentation

5.279.3.1 process() mha_wave_t * MHAFilter::partitioned_convolution_t::process (const mha_wave_t * s_in)

processing

5.279.4 Member Data Documentation

5.279.4.1 fragsize unsigned int MHAFilter::partitioned_convolution_t::fragsize

Audio fragment size, always equal to partition size.

5.279.4.2 nchannels_in unsigned int MHAFilter::partitioned_convolution_t::nchannels_in

Number of audio input channels.

5.279.4.3 nchannels_out unsigned int MHAFilter::partitioned_convolution_t::nchannels_out

Number of audio output channels.

5.279.4.4 output_partitions

`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.

5.279.4.5 filter_partitions

`unsigned int MHAFilter::partitioned_convolution_t::filter_partitions`

The total number of non-zero impulse response partitions.

5.279.4.6 input_signal_wave

`MHASignal::waveform_t` `MHAFilter::partitioned_convolution_t::input_signal_wave`

Buffer for input signal.

Has nchannels_in channels and fragsize∗2 frames

5.279.4.7 current_input_signal_buffer_half_index

`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.

5.279.4.8 input_signal_spec

`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).

5.279.4.9 frequency_response

`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).

5.279.4.10 bookkeeping `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.

5.279.4.11 output_signal_spec `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.

5.279.4.12 current_output_partition_index `unsigned int MHAFilter::partitioned_convolution_t::current_output_partition_index`

A counter modulo output_partitions, indexing the "current" output partition.

5.279.4.13 output_signal_wave `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

5.279.4.14 fft `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**

5.280 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.

5.280.1 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.

5.280.2 Constructor & Destructor Documentation

5.280.2.1 index_t() [1/2]

```
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.

Parameters

dly	The delay (in blocks) of this partition

5.280.2.2 index_t() [2/2] `MHAFilter::partitioned_convolution_t::index_t::index_t ( ) [inline]`

Default constructor for STL compatibility.

5.280.3 Member Data Documentation

5.280.3.1 source_channel_index `unsigned int MHAFilter::partitioned_convolution_t::index_t::source_channel_index`

The input channel index to apply the current partition to.

5.280.3.2 target_channel_index `unsigned int MHAFilter::partitioned_convolution_t::index_t::target_channel_index`

The index of the output channel to which the filter result should go.

5.280.3.3 delay `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**

5.281 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.

5.281.1 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.

5.281.2 Constructor & Destructor Documentation

5.281.2.1 polyphase_resampling_t() 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

5.281.3 Member Function Documentation

5.281.3.1 write() `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. 906)*	Raises exception if there is not enough room or if number of channels does not match.

5.281.3.2 read() `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. 906)*	Raises exception if there is not enough input sign the number of channels is too high.

5.281.3.3 readable_frames() `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. 1074) ensures that enough past samples are present to compute the next target sample.

5.281.4 Member Data Documentation

5.281.4.1 upsampling_factor `unsigned MHAFilter::polyphase_resampling_t::upsampling_factor [private]`

Integer upsampling factor.

Interpolation rate divided by source rate.

5.281.4.2 downsampling_factor `unsigned MHAFilter::polyphase_resampling_t::downsampling_factor [private]`

Integer downsampling factor.

Interpolation rate divided by target rate.

5.281.4.3 now_index `unsigned MHAFilter::polyphase_resampling_t::now_index [private]`

Index of "now" in the interpolated sampling rate.

5.281.4.4 underflow `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. 1074) are present before calling **read** (p. 1074)

5.281.4.5 impulse_response **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. 1451) 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. 1451) can be altered with $*=$, which our constructor does.

5.281.4.6 ringbuffer `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.

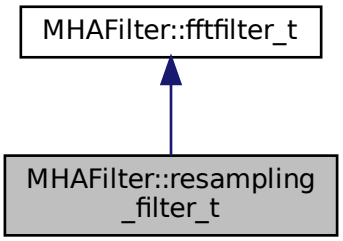
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.282 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**

5.282.1 Detailed Description

Hann shaped low pass filter for resampling.

This class uses FFT filter at upsampled rate.

5.282.2 Constructor & Destructor Documentation

5.282.2.1 resampling_filter_t() `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)

5.282.3 Member Function Documentation

5.282.3.1 fragsize_validator() unsigned int MHAFilter::resampling_filter_t::fragsize_validator (
unsigned int *fftlen*,
unsigned int *irslen*) [static]

5.282.4 Member Data Documentation

5.282.4.1 fragsize 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**

5.283 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**

5.283.1 Detailed Description

Smooth spectral gains, create a windowed impulse response.

Spectral gains are smoothed by multiplying 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 symmetrically arround zero, i.e. to the first and last samples of the inverse Fourier transform of the input spectrum. The **spec2fir()** (p. 1081) function creates a causal impulse response by circularly shifting the impulse response by half of the window length.

The signal dimensions of the arguments of **smoothspec()** (p. 1080) must correspond to the FFT length and number of channels provided in the constructor. The function **spec2fir()** (p. 1081) can fill signal structures with more than window length frames.

5.283.2 Constructor & Destructor Documentation

5.283.2.1 smoothspec_t()

```
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

5.283.2.2 ~smoothspec_t() `MHAFilter::smoothspec_t::~smoothspec_t ( )`

5.283.3 Member Function Documentation

5.283.3.1 smoothspec() [1/2]

```
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

5.283.3.2 smoothspec() [2/2]

```
void MHAFilter::smoothspec_t::smoothspec (
    mha_spec_t & spec ) [inline]
```

Create a smoothed spectrum (in place)

Parameters

spec	Spectrum to be smoothed.

5.283.3.3 spec2fir()

```
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

5.283.3.4 internal_fir()

```
void MHAFilter::smoothspec_t::internal_fir (
    const mha_spec_t & s_in ) [private]
```

5.283.4 Member Data Documentation

5.283.4.1 fftlen

```
unsigned int MHAFilter::smoothspec_t::fftlen [private]
```

5.283.4.2 nchannels `unsigned int MHAFilter::smoothspec_t::nchannels [private]`

5.283.4.3 window **MHAWindow::base_t** `MHAFilter::smoothspec_t::window [private]`

5.283.4.4 tmp_wave **MHASignal::waveform_t** `MHAFilter::smoothspec_t::tmp_wave [private]`

5.283.4.5 tmp_spec **MHASignal::spectrum_t** `MHAFilter::smoothspec_t::tmp_spec [private]`

5.283.4.6 minphase **MHASignal::minphase_t*** `MHAFilter::smoothspec_t::minphase [private]`

5.283.4.7 _linphase_asym `bool MHAFilter::smoothspec_t::_linphase_asym [private]`

5.283.4.8 fft **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**

5.284 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**

5.284.1 Constructor & Destructor Documentation

5.284.1.1 thirdoctave_analyzer_t()

```
MHAFilter::thirdoctave_analyzer_t::thirdoctave_analyzer_t (
            mhaconfig_t cfg )
```

5.284.2 Member Function Documentation

5.284.2.1 process()

```
mha_wave_t * MHAFilter::thirdoctave_analyzer_t::process (
            mha_wave_t * sIn )
```

5.284.2.2 nbands()

```
unsigned int MHAFilter::thirdoctave_analyzer_t::nbands ( )
```

5.284.2.3 nchannels()

```
unsigned int MHAFilter::thirdoctave_analyzer_t::nchannels (
)
```

5.284.2.4 get_cf_hz() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_↩ t::get_cf_hz ( )

5.284.2.5 cf_generator() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer↩ _t::cf_generator (**mhaconfig_t** *cfg*) [static]

5.284.2.6 bw_generator() std::vector< **mha_real_t** > MHAFilter::thirdoctave_↩ analyzer_t::bw_generator (**mhaconfig_t** *cfg*) [static]

5.284.2.7 dup() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_t::dup (std::vector< **mha_real_t** > *vec,* **mhaconfig_t** *cfg*) [static]

5.284.3 Member Data Documentation

5.284.3.1 cfg_ **mhaconfig_t** MHAFilter::thirdoctave_analyzer_t::cfg_ [private]

5.284.3.2 cf std::vector< **mha_real_t**> MHAFilter::thirdoctave_analyzer_t::cf [private]

5.284.3.3 fb **MHAFilter**::**gamma_flt_t** MHAFilter::thirdoctave_analyzer_t::fb [private]

5.284.3.4 out_chunk `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk [private]`

5.284.3.5 out_chunk_im `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**

5.285 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.

5.285.1 Detailed Description

a structure containing a source channel number, a target channel number, and an impulse response.

5.285.2 Constructor & Destructor Documentation

5.285.2.1 transfer_function_t() [1/2] MHAFilter::transfer_function_t::transfer_↩ function_t ( ) [inline]

Default constructor for STL conformity.

Not used.

5.285.2.2 transfer_function_t() [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

5.285.3 Member Function Documentation

5.285.3.1 partitions() 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

5.285.3.2 non_empty_partitions() 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.

5.285.3.3 isempty() 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.

5.285.4 Member Data Documentation

5.285.4.1 source_channel_index `unsigned int MHAFilter::transfer_function_t::source↩ _channel_index`

Source audio channel index for this transfer function.

5.285.4.2 target_channel_index `unsigned int MHAFilter::transfer_function_t::target↩ _channel_index`

Target audio channel index for this transfer function.

5.285.4.3 impulse_response `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**

5.286 MHAFilter::transfer_matrix_t Struct Reference

A sparse matrix of transfer function partitionss.

Inherits std::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. 1089) 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. 1089) method on every matrix member.*

5.286.1 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.

5.286.2 Member Function Documentation

5.286.2.1 partitions()

```
std::valarray<unsigned int> MHAFilter::transfer_matrix_t::partitions (
            unsigned fragsize ) const  [inline]
```

Returns an array of the results of calling the **partitions()** (p. 1089) method on every matrix member.

5.286.2.2 non_empty_partitions()

```
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. 1089) method on every matrix member.

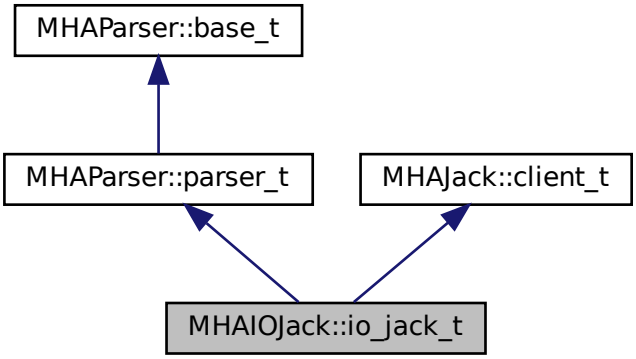
The documentation for this struct was generated from the following file:

- **mha_filter.hh**

5.287 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

5.287.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

5.287.2 Constructor & Destructor Documentation

5.287.2.1 io_jack_t()

```
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 )
```

5.287.3 Member Function Documentation

5.287.3.1 prepare() `void io_jack_t::prepare (`

```
int nch_in,
int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

5.287.3.2 release() `void io_jack_t::release (`

```
void )
```

5.287.3.3 reconnect_inports() `void io_jack_t::reconnect_inports ( ) [private]`

Connect the input ports when connection variable is accessed.

5.287.3.4 reconnect_outports() `void io_jack_t::reconnect_outports ( ) [private]`

Connect the output ports when connection variable is accessed.

5.287.3.5 get_physical_input_ports() `void io_jack_t::get_physical_input_ports ( ) [private]`

5.287.3.6 get_physical_output_ports() `void io_jack_t::get_physical_output_ports ( ) [private]`

5.287.3.7 get_all_input_ports() void io_jack_t::get_all_input_ports ( ) [private]

5.287.3.8 get_all_output_ports() void io_jack_t::get_all_output_ports ( ) [private]

5.287.3.9 get_delays_in() void io_jack_t::get_delays_in ( ) [private]

5.287.3.10 get_delays_out() void io_jack_t::get_delays_out ( ) [private]

5.287.3.11 read_get_cpu_load() void io_jack_t::read_get_cpu_load ( ) [private]

5.287.3.12 read_get_xruns() void io_jack_t::read_get_xruns ( ) [private]

5.287.3.13 read_get_scheduler() void io_jack_t::read_get_scheduler ( ) [private]

5.287.4 Member Data Documentation

5.287.4.1 fw_fragsize unsigned int MHAIOJack::io_jack_t::fw_fragsize [private]

5.287.4.2 fw_samplerate float MHAIOJack::io_jack_t::fw_samplerate [private]

5.287.4.3 servername **MHAParser::string_t** MHAIOJack::io_jack_t::servername [private]

5.287.4.4 clientname **MHAParser::string_t** MHAIOJack::io_jack_t::clientname [private]

5.287.4.5 connections_in **MHAParser::vstring_t** MHAIOJack::io_jack_t::connections↩_in [private]

5.287.4.6 delays_in **MHAParser::vint_mon_t** MHAIOJack::io_jack_t::delays_in [private]

5.287.4.7 connections_out **MHAParser::vstring_t** MHAIOJack::io_jack_t::connections↩_out [private]

5.287.4.8 delays_out **MHAParser::vint_mon_t** MHAIOJack::io_jack_t::delays_out [private]

5.287.4.9 portnames_in **MHAParser::vstring_t** MHAIOJack::io_jack_t::portnames_in [private]

5.287.4.10 portnames_out **MHAParser::vstring_t** MHAIOJack::io_jack_t::portnames_↩out [private]

5.287.4.11 ports_in_physical **MHAParser::vstring_mon_t** MHAIOJack::io_jack_t::ports↩_in_physical [private]

5.287.4.12 ports_out_physical

MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_physical [private]

5.287.4.13 ports_in_all

MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_all [private]

5.287.4.14 ports_out_all

MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_all [private]

5.287.4.15 ports_parser

MHAParser::parser_t MHAIOJack::io_jack_t::ports_parser [private]

5.287.4.16 state_cpuload

MHAParser::float_mon_t MHAIOJack::io_jack_t::state_cpuload [private]

5.287.4.17 state_xruns

MHAParser::int_mon_t MHAIOJack::io_jack_t::state_xruns [private]

5.287.4.18 state_priority

MHAParser::int_mon_t MHAIOJack::io_jack_t::state_priority [private]

5.287.4.19 state_scheduler

MHAParser::string_mon_t MHAIOJack::io_jack_t::state_scheduler [private]

5.287.4.20 state_parser `MHAParser::parser_t MHAIOJack::io_jack_t::state_parser [private]`

5.287.4.21 patchbay `MHAEvents::patchbay_t< io_jack_t> MHAIOJack::io_jack_t::patchbay [private]`

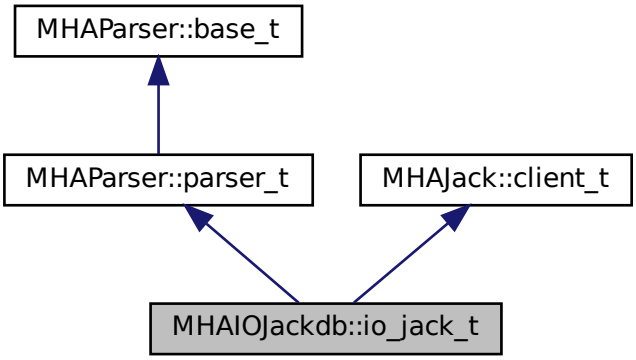
The documentation for this class was generated from the following file:

- **MHAIOJack.cpp**

5.288 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

5.288.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

5.288.2 Constructor & Destructor Documentation

5.288.2.1 io_jack_t()

```
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 )
```

5.288.3 Member Function Documentation

5.288.3.1 prepare()

```
void io_jack_t::prepare (
 int nch_in,
 int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

5.288.3.2 release() void io_jack_t::release (void)

5.288.3.3 fail_on_async_jackerror() bool MHAIOJackdb::io_jack_t::fail_on_async_jackerror ( ) const [inline]

5.288.3.4 IOProcessEvent_inner() [1/2] int io_jack_t::IOProcessEvent_inner (void * *handle*, **mha_wave_t** * *sIn*, **mha_wave_t** ** *sOut*) [static], [private]

5.288.3.5 IOProcessEvent_inner() [2/2] int io_jack_t::IOProcessEvent_inner (**mha_wave_t** * *sIn*, **mha_wave_t** ** *sOut*) [private]

5.288.3.6 reconnect_inports() void io_jack_t::reconnect_inports ( ) [private]

Connect the input ports when connection variable is accessed.

5.288.3.7 reconnect_outports() void io_jack_t::reconnect_outports ( ) [private]

Connect the output ports when connection variable is accessed.

5.288.3.8 get_physical_input_ports() void io_jack_t::get_physical_input_ports ( ) [private]

5.288.3.9 get_physical_output_ports() void io_jack_t::get_physical_output_ports ( ) [private]

5.288.3.10 get_all_input_ports() void io_jack_t::get_all_input_ports ( ) [private]

5.288.3.11 get_all_output_ports() void io_jack_t::get_all_output_ports ( ) [private]

5.288.3.12 read_get_cpu_load() void io_jack_t::read_get_cpu_load ( ) [private]

5.288.3.13 read_get_xruns() void io_jack_t::read_get_xruns ( ) [private]

5.288.3.14 read_get_scheduler() void io_jack_t::read_get_scheduler ( ) [private]

5.288.3.15 set_use_jack_transport() void io_jack_t::set_use_jack_transport ( ) [private]

5.288.3.16 set_locate() void io_jack_t::set_locate ( ) [private]

5.288.4 Member Data Documentation

5.288.4.1 proc_event

`IOProcessEvent_t` MHAIOJackdb::io_jack_t::proc_event [private]

5.288.4.2 proc_handle

void* MHAIOJackdb::io_jack_t::proc_handle [private]

5.288.4.3 mha_fragsize

unsigned int MHAIOJackdb::io_jack_t::mha_fragsize [private]

5.288.4.4 mha_samplerate

float MHAIOJackdb::io_jack_t::mha_samplerate [private]

5.288.4.5 fragsize_ratio

unsigned int MHAIOJackdb::io_jack_t::fragsize_ratio [private]

5.288.4.6 servername

`MHAParser::string_t` MHAIOJackdb::io_jack_t::servername [private]

5.288.4.7 clientname

`MHAParser::string_t` MHAIOJackdb::io_jack_t::clientname [private]

5.288.4.8 connections_in

`MHAParser::vstring_t` MHAIOJackdb::io_jack_t::connections↩
_in [private]

5.288.4.9 connections_out

`MHAParser::vstring_t` MHAIOJackdb::io_jack_t::connections↩
_out [private]

5.288.4.10 portnames_in

MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_in [private]

5.288.4.11 portnames_out

MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_out [private]

5.288.4.12 fail_on_async_jackerr

MHAParser::bool_t MHAIOJackdb::io_jack_t::fail_on_async_jackerr [private]

5.288.4.13 use_jack_transport

MHAParser::bool_t MHAIOJackdb::io_jack_t::use_jack_transport [private]

5.288.4.14 locate

MHAParser::float_t MHAIOJackdb::io_jack_t::locate [private]

5.288.4.15 server_srate

MHAParser::float_mon_t MHAIOJackdb::io_jack_t::server_srate [private]

5.288.4.16 server_fragsize

MHAParser::int_mon_t MHAIOJackdb::io_jack_t::server_fragsize [private]

5.288.4.17 ports_in_physical

MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_in_physical [private]

5.288.4.18 ports_out_physical

MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t↩
::ports_out_physical [private]

5.288.4.19 ports_in_all

MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_↩
in_all [private]

5.288.4.20 ports_out_all

MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_↩
out_all [private]

5.288.4.21 ports_parser

MHAParser::parser_t MHAIOJackdb::io_jack_t::ports_parser
[private]

5.288.4.22 state_cpuload

MHAParser::float_mon_t MHAIOJackdb::io_jack_t::state_↩
cpuload [private]

5.288.4.23 state_xruns

MHAParser::int_mon_t MHAIOJackdb::io_jack_t::state_xruns
[private]

5.288.4.24 state_priority

MHAParser::int_mon_t MHAIOJackdb::io_jack_t::state_↩
priority [private]

5.288.4.25 state_scheduler

MHAParser::string_mon_t MHAIOJackdb::io_jack_t::state↩
_scheduler [private]

5.288.4.26 state_parser

`MHAParser::parser_t MHAIOJackdb::io_jack_t::state_parser [private]`

5.288.4.27 pwinner_out

`MHASignal::waveform_t* MHAIOJackdb::io_jack_t::pwinner_out [private]`

5.288.4.28 patchbay

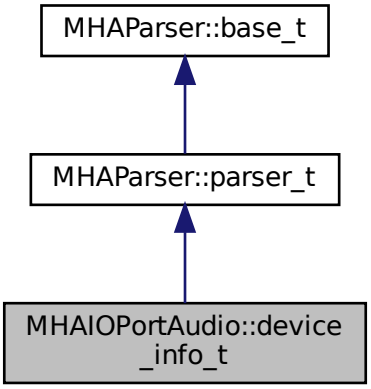
`MHAEvents::patchbay_t< io_jack_t> MHAIOJackdb::io_jack_t::patchbay [private]`

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

- **MHAIOJackdb.cpp**

5.289 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

5.289.1 Constructor & Destructor Documentation

5.289.1.1 device_info_t() `MHAIOPortAudio::device_info_t::device_info_t ( ) [inline]`

5.289.2 Member Function Documentation

5.289.2.1 fill_info() `void MHAIOPortAudio::device_info_t::fill_info ( ) [inline]`

5.289.3 Member Data Documentation

5.289.3.1 numDevices **`MHAParser::int_mon_t`** `MHAIOPortAudio::device_info_t::numDevices`

5.289.3.2 structVersion **MHAParser::vint_mon_t** MHAIOPortAudio::device_info_t::structVersion

5.289.3.3 name **MHAParser::vstring_mon_t** MHAIOPortAudio::device_info_t::name

5.289.3.4 hostApi **MHAParser::vint_mon_t** MHAIOPortAudio::device_info_t::hostApi

5.289.3.5 maxInputChannels **MHAParser::vint_mon_t** MHAIOPortAudio::device_info_t::maxInputChannels

5.289.3.6 maxOutputChannels **MHAParser::vint_mon_t** MHAIOPortAudio::device_info_t::maxOutputChannels

5.289.3.7 defaultLowInputLatency **MHAParser::vfloat_mon_t** MHAIOPortAudio::device_info_t::defaultLowInputLatency

5.289.3.8 defaultLowOutputLatency **MHAParser::vfloat_mon_t** MHAIOPortAudio::device_info_t::defaultLowOutputLatency

5.289.3.9 defaultHighInputLatency **MHAParser::vfloat_mon_t** MHAIOPortAudio::device_info_t::defaultHighInputLatency

5.289.3.10 defaultHighOutputLatency

MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighOutputLatency

5.289.3.11 defaultSampleRate

MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultSampleRate

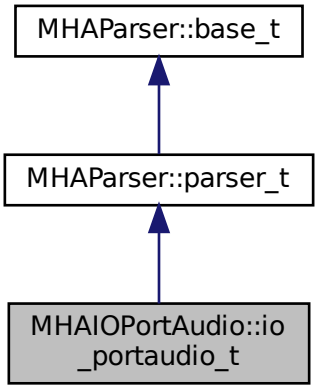
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.290 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**, **IOStoppedEvent_t** **stop_event**, void * **stop_handle**)
- void **device_name_in_updated** ()
- void **device_name_out_updated** ()
- void **device_index_in_updated** ()

- void **device_index_out_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**
- **stream_info_t stream_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_in**
- **MHAParser::int_t device_index_in**
- **MHAParser::string_t device_name_out**
- **MHAParser::int_t device_index_out**
- **MHAParser::float_t suggestedInputLatency**
- **MHAParser::float_t suggestedOutputLatency**
- **MHAEvents::patchbay_t**< **io_portaudio_t** > **patchbay**

Additional Inherited Members

5.290.1 Detailed Description

Main class for Portaudio sound IO.

5.290.2 Constructor & Destructor Documentation

5.290.2.1 io_portaudio_t()

```
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]
```

5.290.2.2 ~io_portaudio_t()

```
MHAIOPortAudio::io_portaudio_t::~io_portaudio_t ( ) [inline]
```

5.290.3 Member Function Documentation

5.290.3.1 device_name_in_updated()

```
void MHAIOPortAudio::io_portaudio_t::device_name_in_updated ( ) [inline]
```

5.290.3.2 device_name_out_updated()

```
void MHAIOPortAudio::io_portaudio_t::device_name_out_updated ( ) [inline]
```

5.290.3.3 device_index_in_updated()

```
void MHAIOPortAudio::io_portaudio_t::device_index_in_updated ( ) [inline]
```

5.290.3.4 device_index_out_updated()

```
void MHAIOPortAudio::io_portaudio_t::device_index_out_updated ( ) [inline]
```

5.290.3.5 cmd_prepare()

```
void MHAIOPortAudio::io_portaudio_t::cmd_prepare (
    int nchannels_in,
    int nchannels_out )
```

5.290.3.6 cmd_start()

```
void MHAIOPortAudio::io_portaudio_t::cmd_start ( )
```

5.290.3.7 cmd_stop()

```
void MHAIOPortAudio::io_portaudio_t::cmd_stop ( )
```

5.290.3.8 cmd_release()

```
void MHAIOPortAudio::io_portaudio_t::cmd_release ( )
```

5.290.3.9 portaudio_callback()

```
int MHAIOPortAudio::io_portaudio_t::portaudio_↩
callback (
    const void * input,
    void * output,
    unsigned long frame_count,
    const PaStreamCallbackTimeInfo * time_info,
    PaStreamCallbackFlags status_flags )
```

5.290.4 Member Data Documentation

5.290.4.1 device_info

```
device_info_t MHAIOPortAudio::io_portaudio_t::device_info
[private]
```

5.290.4.2 stream_info

```
stream_info_t MHAIOPortAudio::io_portaudio_t::stream_info
[private]
```

5.290.4.3 s_in

MHASignal::waveform_t* MHAIOPortAudio::io_portaudio_t::s_in [private]

5.290.4.4 s_out

mha_wave_t* MHAIOPortAudio::io_portaudio_t::s_out [private]

5.290.4.5 samplerate

float MHAIOPortAudio::io_portaudio_t::samplerate [private]

5.290.4.6 nchannels_out

unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_↩ out [private]

5.290.4.7 nchannels_in

unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_in [private]

5.290.4.8 fragsize

unsigned int MHAIOPortAudio::io_portaudio_t::fragsize [private]

5.290.4.9 proc_event

IOProcessEvent_t MHAIOPortAudio::io_portaudio_t::proc_event [private]

5.290.4.10 proc_handle

void* MHAIOPortAudio::io_portaudio_t::proc_handle [private]

5.290.4.11 start_event

IOStartedEvent_t MHAIOPortAudio::io_portaudio_t::start_↩ event [private]

5.290.4.12 start_handle

`void* MHAIOPortAudio::io_portaudio_t::start_handle [private]`

5.290.4.13 stop_event

`IOStoppedEvent_t` `MHAIOPortAudio::io_portaudio_t::stop_event [private]`

5.290.4.14 stop_handle

`void* MHAIOPortAudio::io_portaudio_t::stop_handle [private]`

5.290.4.15 portaudio_stream

`PaStream* MHAIOPortAudio::io_portaudio_t::portaudio_stream [private]`

5.290.4.16 device_name_in

`MHAParser::string_t` `MHAIOPortAudio::io_portaudio_t::device_name_in [private]`

5.290.4.17 device_index_in

`MHAParser::int_t` `MHAIOPortAudio::io_portaudio_t::device_index_in [private]`

5.290.4.18 device_name_out

`MHAParser::string_t` `MHAIOPortAudio::io_portaudio_t::device_name_out [private]`

5.290.4.19 device_index_out

`MHAParser::int_t` `MHAIOPortAudio::io_portaudio_t::device_index_out [private]`

5.290.4.20 suggestedInputLatency

`MHAParser::float_t MHAIOPortAudio::io_portaudio_t::suggestedInputLatency [private]`

5.290.4.21 suggestedOutputLatency

`MHAParser::float_t MHAIOPortAudio::io_portaudio_t::suggestedOutputLatency [private]`

5.290.4.22 patchbay

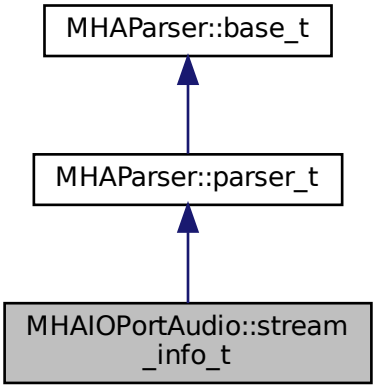
`MHAEvents::patchbay_t< io_portaudio_t> MHAIOPortAudio::io_portaudio_t::patchbay [private]`

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

- **MHAIOPortAudio.cpp**

5.291 MHAIOPortAudio::stream_info_t Class Reference

Inheritance diagram for MHAIOPortAudio::stream_info_t:



Public Member Functions

- **stream_info_t** ()
- void **fill_info** (PaStream ∗stream_)

Public Attributes

- **MHAParser::float_mon_t paInputLatency**
- **MHAParser::float_mon_t paOutputLatency**
- **MHAParser::float_mon_t paSampleRate**

Additional Inherited Members

5.291.1 Constructor & Destructor Documentation

5.291.1.1 stream_info_t() `MHAIOPortAudio::stream_info_t::stream_info_t ( ) [inline]`

5.291.2 Member Function Documentation

5.291.2.1 fill_info() `void MHAIOPortAudio::stream_info_t::fill_info ( PaStream * stream_ ) [inline]`

5.291.3 Member Data Documentation

5.291.3.1 paInputLatency **MHAParser::float_mon_t** MHAIOPortAudio::stream_info_t::paInputLatency

5.291.3.2 paOutputLatency **MHAParser::float_mon_t** MHAIOPortAudio::stream_info_t::paOutputLatency

5.291.3.3 paSampleRate `MHAParser::float_mon_t MHAIOPortAudio::stream_info_t::paSampleRate`

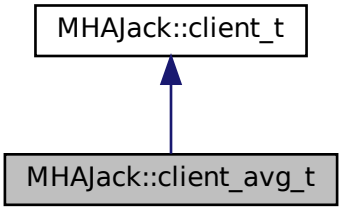
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.292 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

5.292.1 Detailed Description

Generic JACK client for averaging a system response across time.

5.292.2 Constructor & Destructor Documentation

5.292.2.1 client_avg_t()

```
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

5.292.3 Member Function Documentation

5.292.3.1 io()

```
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

5.292.3.2 proc() [1/2]

```
int MHAJack::client_avg_t::proc (
    void * handle,
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [static], [private]
```

5.292.3.3 IOStoppedEvent() [1/2]

```
void MHAJack::client_avg_t::IOStoppedEvent (
    void * handle,
    int proc_err,
    int io_err ) [static], [private]
```

5.292.3.4 proc() [2/2]

```
void MHAJack::client_avg_t::proc (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [private]
```

5.292.3.5 IOStoppedEvent() [2/2] void MHAJack::client_avg_t::IOStoppedEvent ( ) [private]

5.292.4 Member Data Documentation

5.292.4.1 b_stopped bool MHAJack::client_avg_t::b_stopped [private]

5.292.4.2 pos unsigned int MHAJack::client_avg_t::pos [private]

5.292.4.3 sn_in mha_wave_t* MHAJack::client_avg_t::sn_in [private]

5.292.4.4 sn_out mha_wave_t* MHAJack::client_avg_t::sn_out [private]

5.292.4.5 name std::string MHAJack::client_avg_t::name [private]

5.292.4.6 frag_out MHASignal::waveform_t* MHAJack::client_avg_t::frag_out [private]

5.292.4.7 nrep const unsigned int MHAJack::client_avg_t::nrep [private]

5.292.4.8 n `unsigned int MHAJack::client_avg_t::n [private]`

5.292.4.9 b_ready `bool MHAJack::client_avg_t::b_ready [private]`

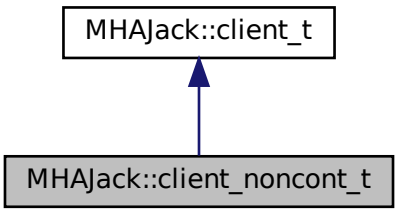
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

5.293 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

5.293.1 Detailed Description

Generic client for synchronous playback and recording of waveform fragments.

5.293.2 Constructor & Destructor Documentation

5.293.2.1 client_noncont_t()

```
MHAJack::client_noncont_t::client_noncont_t (
    const std::string & name,
    bool use_jack_transport = false )
```

5.293.3 Member Function Documentation

5.293.3.1 io()

```
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 )
```

5.293.3.2 proc() [1/2]

```
int MHAJack::client_noncont_t::proc (
    void * handle,
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [static], [private]
```

5.293.3.3 IOStoppedEvent() [1/2]

```
void MHAJack::client_noncont_t::IOStoppedEvent (
    void * handle,
    int proc_err,
    int io_err ) [static], [private]
```

5.293.3.4 proc() [2/2]

```
void MHAJack::client_noncont_t::proc (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [private]
```

5.293.3.5 IOStoppedEvent() [2/2]

```
void MHAJack::client_noncont_t::IOStoppedEvent (
) [private]
```

5.293.4 Member Data Documentation

5.293.4.1 b_stopped

```
bool MHAJack::client_noncont_t::b_stopped [private]
```

5.293.4.2 pos

```
unsigned int MHAJack::client_noncont_t::pos [private]
```

5.293.4.3 sn_in

```
mha_wave_t* MHAJack::client_noncont_t::sn_in [private]
```

5.293.4.4 sn_out

`mha_wave_t* MHAJack::client_noncont_t::sn_out [private]`

5.293.4.5 name

`std::string MHAJack::client_noncont_t::name [private]`

5.293.4.6 frag_out

`MHASignal::waveform_t* MHAJack::client_noncont_t::frag_out [private]`

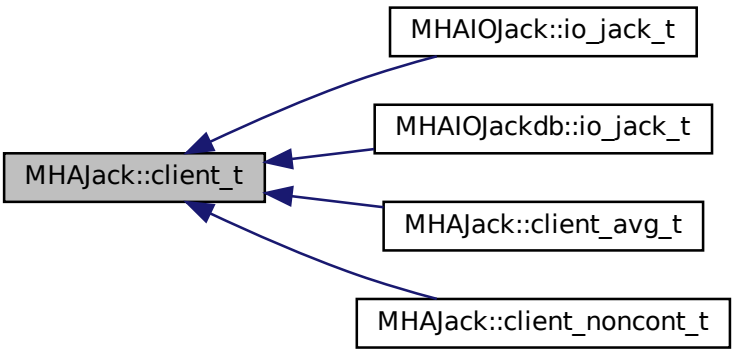
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

5.294 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, **IOStarted↩ Event_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)
- bool **is_prepared** () const

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**

5.294.1 Detailed Description

Generic asynchronous JACK client.

5.294.2 Constructor & Destructor Documentation

5.294.2.1 client_t()

```
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 )
```

5.294.3 Member Function Documentation

5.294.3.1 prepare() [1/2]

```
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

5.294.3.2 prepare() [2/2]

```
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

5.294.3.3 release() `void MHAJack::client_t::release ( void )`

Remove JACK client and deallocate internal ports and buffers.

5.294.3.4 start() `void MHAJack::client_t::start ( bool fail_on_async_jack_error = true )`

5.294.3.5 stop() `void MHAJack::client_t::stop ( )`

5.294.3.6 connect_input() `void MHAJack::client_t::connect_input ( const std::vector< std::string > & con )`

Connect the input ports when connection variable is accessed.

5.294.3.7 connect_output() `void MHAJack::client_t::connect_output ( const std::vector< std::string > & con )`

Connect the output ports when connection variable is accessed.

5.294.3.8 get_fragsize() unsigned int MHAJack::client_t::get_fragsize ( ) const [inline]

5.294.3.9 get_srate() float MHAJack::client_t::get_srate ( ) const [inline]

5.294.3.10 get_xruns() unsigned long MHAJack::client_t::get_xruns ( ) [inline]

5.294.3.11 get_xruns_reset() unsigned long MHAJack::client_t::get_xruns_reset ( )

5.294.3.12 str_error() std::string MHAJack::client_t::str_error (int *err*)

5.294.3.13 get_ports() 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.)

5.294.3.14 get_my_input_ports() std::vector< std::string > MHAJack::client_t↩ ::get_my_input_ports ( )

5.294.3.15 get_my_output_ports()

```
std::vector< std::string > MHAJack::client_t↩
::get_my_output_ports ( )
```

5.294.3.16 set_input_portnames()

```
void MHAJack::client_t::set_input_portnames (
        const std::vector< std::string > & names )
```

5.294.3.17 set_output_portnames()

```
void MHAJack::client_t::set_output_portnames (
        const std::vector< std::string > & names )
```

5.294.3.18 get_cpu_load()

```
float MHAJack::client_t::get_cpu_load ( )
```

5.294.3.19 set_use_jack_transport()

```
void MHAJack::client_t::set_use_jack_transport
(
        bool ut ) [inline]
```

5.294.3.20 is_prepared()

```
bool MHAJack::client_t::is_prepared ( ) const [inline]
```

5.294.3.21 prepare_impl()

```
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

5.294.3.22 internal_start() `void MHAJack::client_t::internal_start ( ) [private]`

5.294.3.23 internal_stop() `void MHAJack::client_t::internal_stop ( ) [private]`

5.294.3.24 stopped()

```
void MHAJack::client_t::stopped (
int proc_err,
int io_err ) [private]
```

5.294.3.25 jack_proc_cb() [1/2]

```
int MHAJack::client_t::jack_proc_cb (
jack_nframes_t n,
void * h ) [static], [private]
```

5.294.3.26 jack_proc_cb() [2/2]

```
int MHAJack::client_t::jack_proc_cb (
jack_nframes_t n ) [private]
```

This is the main processing callback.

Here happens double buffering and downsampling.

5.294.3.27 jack_xrun_cb() [1/2]

```
int MHAJack::client_t::jack_xrun_cb (
void * h ) [static], [private]
```

5.294.3.28 jack_xrun_cb() [2/2] int MHAJack::client_t::jack_xrun_cb ( ) [inline], [private]

5.294.4 Member Data Documentation

5.294.4.1 num_xruns unsigned long MHAJack::client_t::num_xruns [private]

5.294.4.2 fragsize unsigned int MHAJack::client_t::fragsize [private]

5.294.4.3 samplerate float MHAJack::client_t::samplerate [private]

5.294.4.4 nchannels_in unsigned int MHAJack::client_t::nchannels_in [private]

5.294.4.5 nchannels_out unsigned int MHAJack::client_t::nchannels_out [private]

5.294.4.6 proc_event IOProcessEvent_t MHAJack::client_t::proc_event [private]

5.294.4.7 proc_handle void* MHAJack::client_t::proc_handle [private]

5.294.4.8 start_event

`IOStartedEvent_t MHAJack::client_t::start_event [private]`

5.294.4.9 start_handle

`void* MHAJack::client_t::start_handle [private]`

5.294.4.10 stop_event

`IOStoppedEvent_t MHAJack::client_t::stop_event [private]`

5.294.4.11 stop_handle

`void* MHAJack::client_t::stop_handle [private]`

5.294.4.12 s_in

`MHASignal::waveform_t* MHAJack::client_t::s_in [private]`

5.294.4.13 s_out

`mha_wave_t* MHAJack::client_t::s_out [private]`

5.294.4.14 inch

`MHAJack::port_t** MHAJack::client_t::inch [private]`

5.294.4.15 outch

`MHAJack::port_t** MHAJack::client_t::outch [private]`

5.294.4.16 jc

`jack_client_t* MHAJack::client_t::jc [protected]`

5.294.4.17 flags `unsigned int MHAJack::client_t::flags [private]`

5.294.4.18 b_prepared `bool MHAJack::client_t::b_prepared [private]`

5.294.4.19 use_jack_transport `bool MHAJack::client_t::use_jack_transport [private]`

5.294.4.20 jstate_prev `jack_transport_state_t MHAJack::client_t::jstate_prev [private]`

5.294.4.21 input_portnames `std::vector<std::string> MHAJack::client_t::input_portnames [private]`

5.294.4.22 output_portnames `std::vector<std::string> MHAJack::client_t::output_portnames [private]`

5.294.4.23 fail_on_async_jackerror `bool MHAJack::client_t::fail_on_async_jackerror [private]`

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

- **mhajack.h**
- **mhajack.cpp**

5.295 MHAJack::port_t Class Reference

Class for one channel/port.

Public Types

- enum **dir_t** { **input** , **output** }

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**

5.295.1 Detailed Description

Class for one channel/port.

This class represents one JACK port. Double buffering for asynchronous process callbacks is managed by this class.

5.295.2 Member Enumeration Documentation

5.295.2.1 dir_t enum MHAJack::port_t::dir_t

Enumerator

input	
output	

5.295.3 Constructor & Destructor Documentation

5.295.3.1 port_t() [1/2] 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).

5.295.3.2 port_t() [2/2] 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.

5.295.3.3 ~port_t() MHAJack::port_t::~port_t ( )

5.295.4 Member Function Documentation

5.295.4.1 read() 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.

5.295.4.2 write() 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.

5.295.4.3 mute() 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).

5.295.4.4 connect_to() `void MHAJack::port_t::connect_to ( const char * pn )`

Parameters

pn	Port name to connect to

5.295.4.5 get_short_name() `const char * MHAJack::port_t::get_short_name ( )`

Return the port name.

5.295.5 Member Data Documentation

5.295.5.1 dir_type `dir_t MHAJack::port_t::dir_type [private]`

5.295.5.2 port `jack_port_t* MHAJack::port_t::port [private]`

5.295.5.3 iob `jack_default_audio_sample_t* MHAJack::port_t::iob [private]`

5.295.5.4 jc `jack_client_t* MHAJack::port_t::jc [private]`

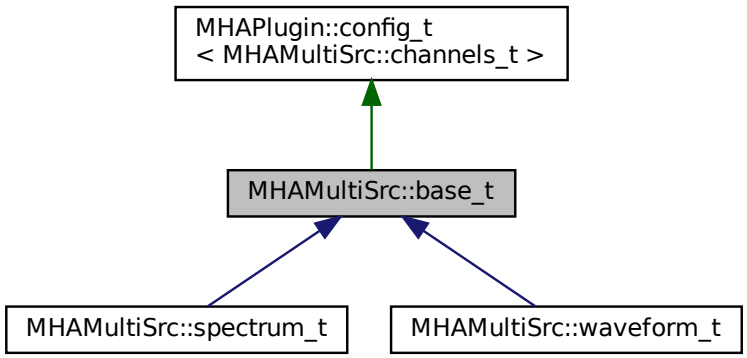
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

5.296 MHAMultiSrc::base_t Class Reference

Base class for source selection.

Inheritance diagram for MHAMultiSrc::base_t:



Public Member Functions

- **base_t** (**MHA_AC::algo_comm_t** &iac)
- void **select_source** (const std::vector< std::string > &src, int in_channels)

  Change the selection of input sources.

Protected Attributes

- **MHA_AC::algo_comm_t** & **ac**

Additional Inherited Members

5.296.1 Detailed Description

Base class for source selection.

See also

MHAMultiSrc::channel_t (p. 1139)
MHAMultiSrc::channels_t (p. 1139)

5.296.2 Constructor & Destructor Documentation

5.296.2.1 base_t() `MHAMultiSrc::base_t::base_t (`

`MHA_AC::algo_comm_t` `& iac )`

5.296.3 Member Function Documentation

5.296.3.1 select_source() `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)

5.296.4 Member Data Documentation

5.296.4.1 ac **`MHA_AC::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**

5.297 MHAMultiSrc::channel_t Class Reference

Public Attributes

- std::string **name**
- int **channel**

5.297.1 Member Data Documentation

5.297.1.1 name `std::string MHAMultiSrc::channel_t::name`

5.297.1.2 channel `int MHAMultiSrc::channel_t::channel`

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

- **mha_multisrc.h**

5.298 MHAMultiSrc::channels_t Class Reference

Inherits std::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.

5.298.1 Constructor & Destructor Documentation

5.298.1.1 channels_t()

```
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. 1139) is filled.

Parameters

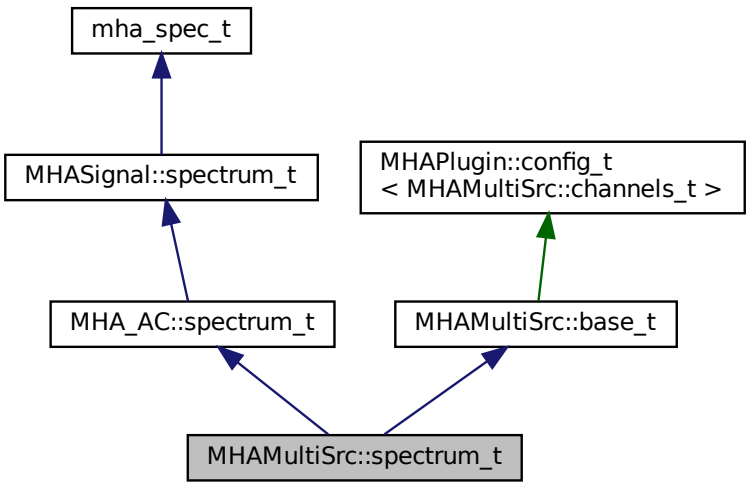
route	vector of source channel ids
in_channels	number of channels in the processed input signal

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.299 MHAMultiSrc::spectrum_t Class Reference

Inheritance diagram for MHAMultiSrc::spectrum_t:



Public Member Functions

- **spectrum_t** (**MHA_AC::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

5.299.1 Constructor & Destructor Documentation

5.299.1.1 spectrum_t()

```
MHAMultiSrc::spectrum_t::spectrum_t (
    MHA_AC::algo_comm_t & iac,
    std::string name,
    unsigned int frames,
    unsigned int channels )
```

5.299.2 Member Function Documentation

5.299.2.1 update()

```
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

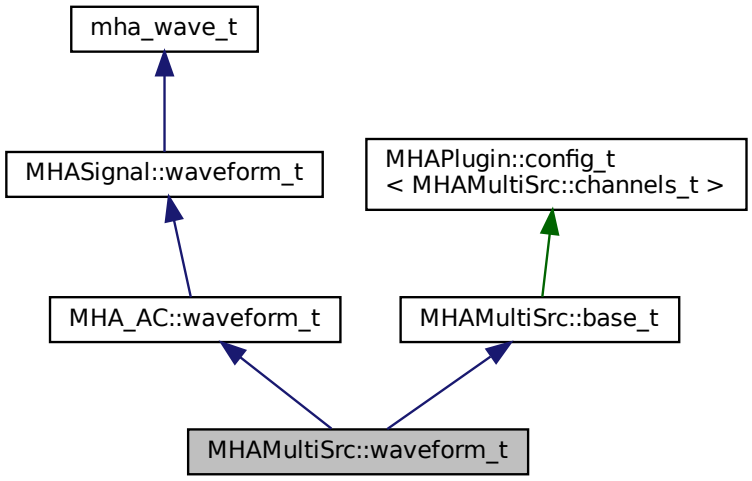
Return pointer to spectrum structure

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.300 MHAMultiSrc::waveform_t Class Reference

Inheritance diagram for MHAMultiSrc::waveform_t:



Public Member Functions

- **waveform_t** (**MHA_AC::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

5.300.1 Constructor & Destructor Documentation

5.300.1.1 waveform_t()

```
MHAMultiSrc::waveform_t::waveform_t (
    MHA_AC::algo_comm_t & iac,
    std::string name,
    unsigned int frames,
    unsigned int channels )
```

5.300.2 Member Function Documentation

5.300.2.1 update() `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**

5.301 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**

5.301.1 Member Data Documentation

5.301.1.1 cf_l **mha_real_t** MHAOvlFilter::band_descriptor_t::cf_l

5.301.1.2 ef_l **mha_real_t** MHAOvlFilter::band_descriptor_t::ef_l

5.301.1.3 cf **mha_real_t** MHAOvlFilter::band_descriptor_t::cf

5.301.1.4 ef_h **mha_real_t** MHAOvlFilter::band_descriptor_t::ef_h

5.301.1.5 cf_h **mha_real_t** MHAOvlFilter::band_descriptor_t::cf_h

5.301.1.6 low_side_flat bool MHAOvlFilter::band_descriptor_t::low_side_flat

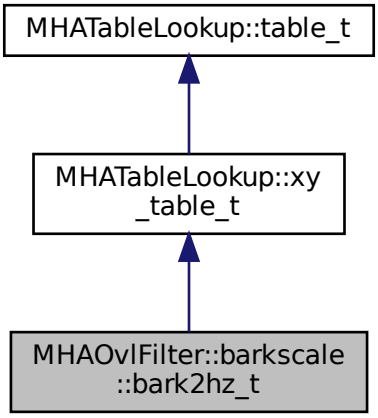
5.301.1.7 high_side_flat bool MHAOvlFilter::band_descriptor_t::high_side_flat

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

- **mha_fftfb.hh**

5.302 MHAOvlFilter::barkscale::bark2hz_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::bark2hz_t:



Public Member Functions

- **bark2hz_t** ()
- ∼**bark2hz_t** ()

Additional Inherited Members

5.302.1 Constructor & Destructor Documentation

5.302.1.1 bark2hz_t() `MHAOvlFilter::barkscale::bark2hz_t::bark2hz_t ( )`

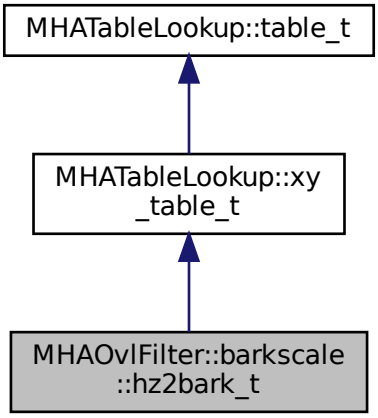
5.302.1.2 ∼bark2hz_t() `MHAOvlFilter::barkscale::bark2hz_t::∼bark2hz_t ( )`

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

- **mha_fftfb.cpp**

5.303 MHAOvlFilter::barkscale::hz2bark_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::hz2bark_t:



Public Member Functions

- **hz2bark_t** ()
- ~**hz2bark_t** ()

Additional Inherited Members

5.303.1 Constructor & Destructor Documentation

5.303.1.1 hz2bark_t() `MHAOvlFilter::barkscale::hz2bark_t::hz2bark_t ( )`

5.303.1.2 ~hz2bark_t() `MHAOvlFilter::barkscale::hz2bark_t::~hz2bark_t ( )`

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

- **mha_fftfb.cpp**

5.304 MHAOvlFilter::fftfb_ac_info_t Class Reference

Public Member Functions

- **fftfb_ac_info_t** (const **MHAOvlFilter::fftfb_t** &fb, **MHA_AC::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

5.304.1 Constructor & Destructor Documentation

5.304.1.1 fftfb_ac_info_t() `MHAOvlFilter::fftfb_ac_info_t::fftfb_ac_info_t (`

```
const MHAOvlFilter::fftfb_t & fb,
MHA_AC::algo_comm_t & ac,
const std::string & prefix )
```

5.304.2 Member Function Documentation

5.304.2.1 insert() `void MHAOvlFilter::fftfb_ac_info_t::insert ( )`

5.304.3 Member Data Documentation

5.304.3.1 cfv `MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cfv [private]`

vector of nominal center frequencies / Hz

5.304.3.2 efv `MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::efv [private]`

vector of edge frequencies / Hz

5.304.3.3 bwv `MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::bwv [private]`

vector of band-weigths (sum of squared fft-bin-weigths)/num_frames

5.304.3.4 cLTASS `MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cLTASS [private]`

vector of LTASS correction

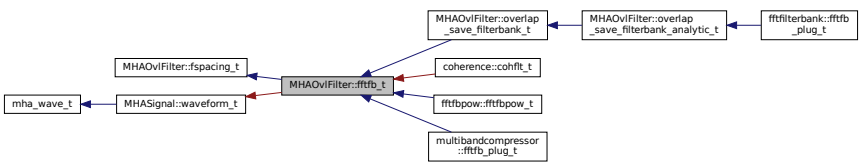
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.305 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

5.305.1 Detailed Description

FFT based overlapping filter bank.

5.305.2 Constructor & Destructor Documentation

5.305.2.1 fftfb_t() 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

5.305.2.2 ~fftfb_t() MHAOvlFilter::fftfb_t::~fftfb_t ( )

5.305.3 Member Function Documentation

5.305.3.1 apply_gains() void MHAOvlFilter::fftfb_t::apply_gains (

```
mha_spec_t * s_out,
const mha_spec_t * s_in,
const mha_wave_t * gains )
```

5.305.3.2 get_fbpower() void MHAOvlFilter::fftfb_t::get_fbpower (

```
mha_wave_t * fbpow,
const mha_spec_t * s_in )
```

5.305.3.3 get_fbpower_db() void MHAOvlFilter::fftfb_t::get_fbpower_db (

```
mha_wave_t * fbpow,
const mha_spec_t * s_in )
```

5.305.3.4 get_ltass_gain_db()

```
std::vector< float > MHAOvlFilter::fftfb_t::get_ltass_gain_db ( ) const
```

5.305.3.5 bin1()

```
unsigned int MHAOvlFilter::fftfb_t::bin1 (
        unsigned int band ) const  [inline]
```

Return index of first non-zero filter shape window.

5.305.3.6 bin2()

```
unsigned int MHAOvlFilter::fftfb_t::bin2 (
        unsigned int band ) const  [inline]
```

Return index of first zero filter shape window above center frequency.

5.305.3.7 get_fftlen()

```
unsigned int MHAOvlFilter::fftfb_t::get_fftlen ( ) const  [inline]
```

Return fft length.

5.305.3.8 w()

```
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

5.305.4 Member Data Documentation

5.305.4.1 vbin1 `unsigned int* MHAOvlFilter::fftfb_t::vbin1 [private]`

5.305.4.2 vbin2 `unsigned int* MHAOvlFilter::fftfb_t::vbin2 [private]`

5.305.4.3 shape `mha_real_t(* MHAOvlFilter::fftfb_t::shape) ( mha_real_t) [private]`

5.305.4.4 fftlen `unsigned int MHAOvlFilter::fftfb_t::fftlen [private]`

5.305.4.5 samplingrate `mha_real_t MHAOvlFilter::fftfb_t::samplingrate [private]`

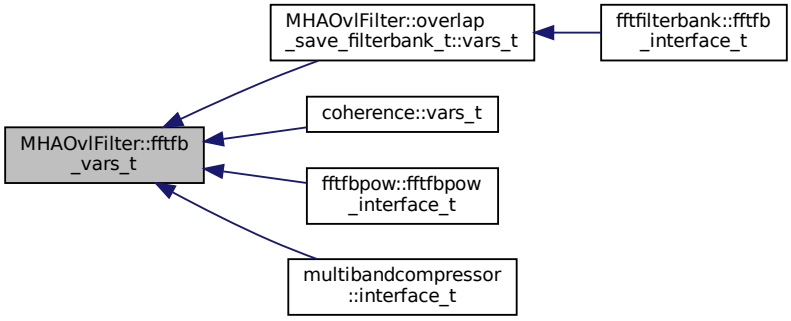
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.306 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::bool_t flag_allow_empty_bands**
  Allow that frequency bands contain only zeros.
- **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**

5.306.1 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. 1152) 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.

5.306.2 Constructor & Destructor Documentation

5.306.2.1 fftfb_vars_t() `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.

5.306.3 Member Data Documentation

5.306.3.1 fscale `scale_var_t MHAOvlFilter::fftfb_vars_t::fscale`

Frequency scale type (lin/bark/log/erb).

5.306.3.2 ovltype `scale_var_t MHAOvlFilter::fftfb_vars_t::ovltype`

Filter shape (rect/lin/hann).

5.306.3.3 plateau `MHAParser::float_t MHAOvlFilter::fftfb_vars_t::plateau`

relative plateau width.

5.306.3.4 ftype **MHAParser::kw_t** MHAOvlFilter::fftfb_vars_t::ftype

Flag to decide wether edge or center frequencies are used.

5.306.3.5 f **fscale_t** MHAOvlFilter::fftfb_vars_t::f

Frequency.

5.306.3.6 normalize **MHAParser::bool_t** MHAOvlFilter::fftfb_vars_t::normalize

Normalize sum of channels.

5.306.3.7 fail_on_nonmonotonic **MHAParser::bool_t** MHAOvlFilter::fftfb_vars_t↩ ::fail_on_nonmonotonic

Fail if frequency entries are non-monotonic (otherwise sort)

5.306.3.8 fail_on_unique_bins **MHAParser::bool_t** MHAOvlFilter::fftfb_vars_t::fail↩ _on_unique_bins

Fail if center frequencies share the same FFT bin.

5.306.3.9 flag_allow_empty_bands **MHAParser::bool_t** MHAOvlFilter::fftfb_vars_t↩ ::flag_allow_empty_bands

Allow that frequency bands contain only zeros.

5.306.3.10 cf `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cf`

Final center frequencies in Hz.

5.306.3.11 ef `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::ef`

Final edge frequencies in Hz.

5.306.3.12 cLTASS `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cLTASS`

Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)

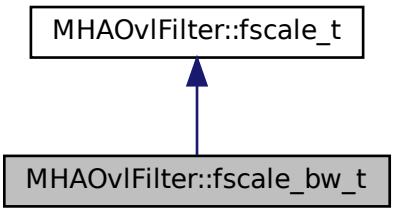
5.306.3.13 shapes `MHAParser::mfloat_mon_t MHAOvlFilter::fftfb_vars_t::shapes`

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.307 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

5.307.1 Constructor & Destructor Documentation

5.307.1.1 fscale_bw_t() `MHAOvlFilter::fscale_bw_t::fscale_bw_t ( MHAParser::parser_t & parent )`

5.307.2 Member Function Documentation

5.307.2.1 get_bw_hz() `std::vector< mha_real_t > MHAOvlFilter::fscale_bw_t::get_bw_hz ( ) const`

5.307.2.2 update_hz() `void MHAOvlFilter::fscale_bw_t::update_hz ( ) [private]`

5.307.3 Member Data Documentation

5.307.3.1 bw `MHAParser::vfloat_t MHAOvlFilter::fscale_bw_t::bw [protected]`

5.307.3.2 bw_hz `MHAParser::vfloat_mon_t MHAOvlFilter::fscale_bw_t::bw_hz [protected]`

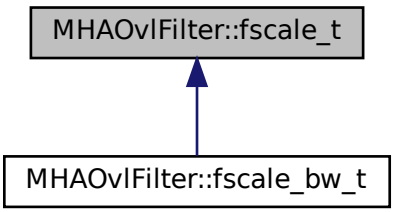
5.307.3.3 updater `MHAEvents::connector_t< fscale_bw_t> MHAOvlFilter::fscale_bw_t::updater [private]`

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.308 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**

5.308.1 Constructor & Destructor Documentation

5.308.1.1 fscale_t() `MHAOvlFilter::fscale_t::fscale_t ( MHAParser::parser_t & parent )`

5.308.2 Member Function Documentation

5.308.2.1 get_f_hz() `std::vector< mha_real_t > MHAOvlFilter::fscale_t::get_f_hz ( ) const`

5.308.2.2 update_hz() `void MHAOvlFilter::fscale_t::update_hz ( ) [private]`

5.308.3 Member Data Documentation

5.308.3.1 unit `scale_var_t MHAOvlFilter::fscale_t::unit`

5.308.3.2 f `MHAParser::vfloat_t MHAOvlFilter::fscale_t::f`

5.308.3.3 f_hz `MHAParser::vfloat_mon_t MHAOvlFilter::fscale_t::f_hz`

5.308.3.4 updater `MHAEvents::connector_t< fscale_t> MHAOvlFilter::fscale_t::updater [private]`

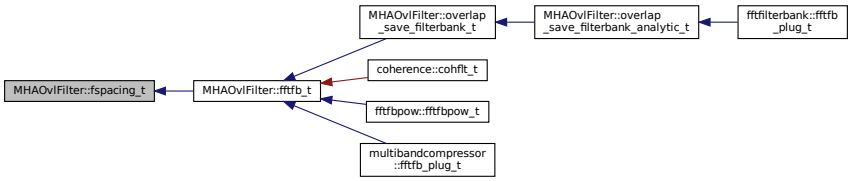
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.309 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_**

5.309.1 Detailed Description

Class for frequency spacing, used by filterbank shape generator class.

5.309.2 Constructor & Destructor Documentation

5.309.2.1 fspacing_t()

```
MHAOvlFilter::fspacing_t::fspacing_t (
    const MHAOvlFilter::fftfb_vars_t & par,
    unsigned int nfft,
    mha_real_t fs )
```

5.309.3 Member Function Documentation

5.309.3.1 get_cf_fftbin() std::vector< unsigned int > MHAOvlFilter::fspacing_t↩ ::get_cf_fftbin ( ) const

5.309.3.2 get_cf_hz() std::vector< **mha_real_t** > MHAOvlFilter::fspacing_t::get_cf↩ _hz ( ) const

5.309.3.3 get_ef_hz() std::vector< **mha_real_t** > MHAOvlFilter::fspacing_t::get_ef↩ _hz ( ) const

5.309.3.4 nbands() unsigned int MHAOvlFilter::fspacing_t::nbands ( ) const [inline]

Return number of bands in filter bank.

5.309.3.5 fail_on_nonmonotonic_cf() void MHAOvlFilter::fspacing_t::fail_on_↩ nonmonotonic_cf ( ) [protected]

5.309.3.6 fail_on_unique_fftbins() void MHAOvlFilter::fspacing_t::fail_on_unique_↩ fftbins ( ) [protected]

5.309.3.7 ef2bands() void MHAOvlFilter::fspacing_t::ef2bands (std::vector< **mha_real_t** > *vef*) [private]

5.309.3.8 cf2bands() void MHAOvlFilter::fspacing_t::cf2bands (std::vector< **mha_real_t** > *vcf*) [private]

5.309.3.9 equidist2bands() void MHAOvlFilter::fspacing_t::equidist2bands (std::vector< **mha_real_t** > *vcf*) [private]

5.309.4 Member Data Documentation

5.309.4.1 bands std::vector< **MHAOvlFilter::band_descriptor_t**> MHAOvlFilter::fspacing_t::bands [protected]

5.309.4.2 symmetry_scale **mha_real_t**(* MHAOvlFilter::fspacing_t::symmetry_scale) (**mha_real_t**) [protected]

5.309.4.3 nfft_ unsigned int MHAOvlFilter::fspacing_t::nfft_ [private]

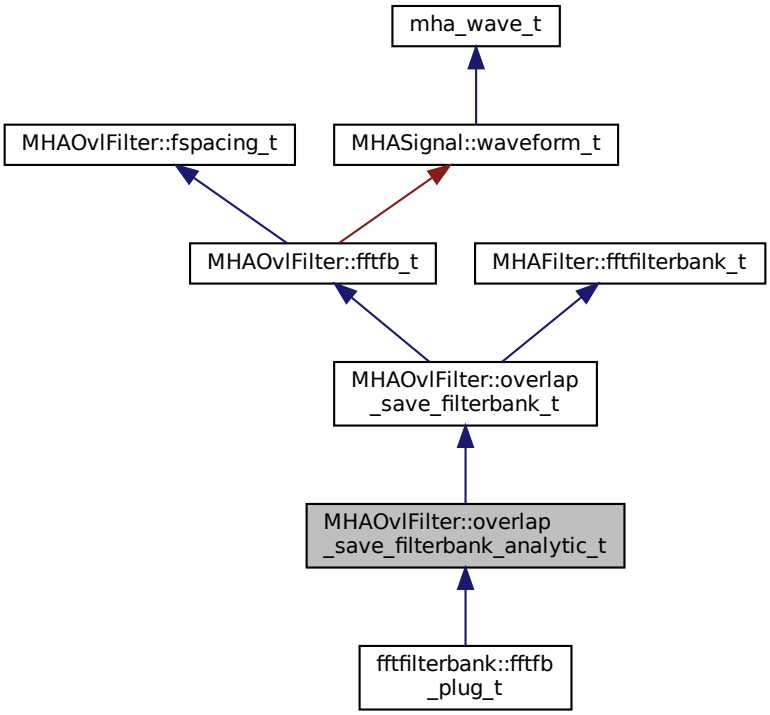
5.309.4.4 fs_ **mha_real_t** MHAOvlFilter::fspacing_t::fs_ [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.310 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** ∗∗fltIm)

Private Attributes

- **MHAFilter::fftfilterbank_t imagfb**

Additional Inherited Members

5.310.1 Constructor & Destructor Documentation

5.310.1.1 overlap_save_filterbank_analytic_t()

```
MHAOvlFilter::overlap_save_filterbank_analytic_t::overlap_save_filterbank_analytic_t (
            MHAOvlFilter::overlap_save_filterbank_t::vars_t & fbpar,
            mhaconfig_t channelconfig_in )
```

5.310.2 Member Function Documentation

5.310.2.1 filter_analytic()

```
void MHAOvlFilter::overlap_save_filterbank_analytic_t::filter_analytic (
            const  mha_wave_t * sIn,
            mha_wave_t ** fltRe,
            mha_wave_t ** fltIm )
```

5.310.3 Member Data Documentation

5.310.3.1 imagfb

```
MHAFilter::fftfilterbank_t MHAOvlFilter::overlap_save_filterbank_analytic_t::imagfb [private]
```

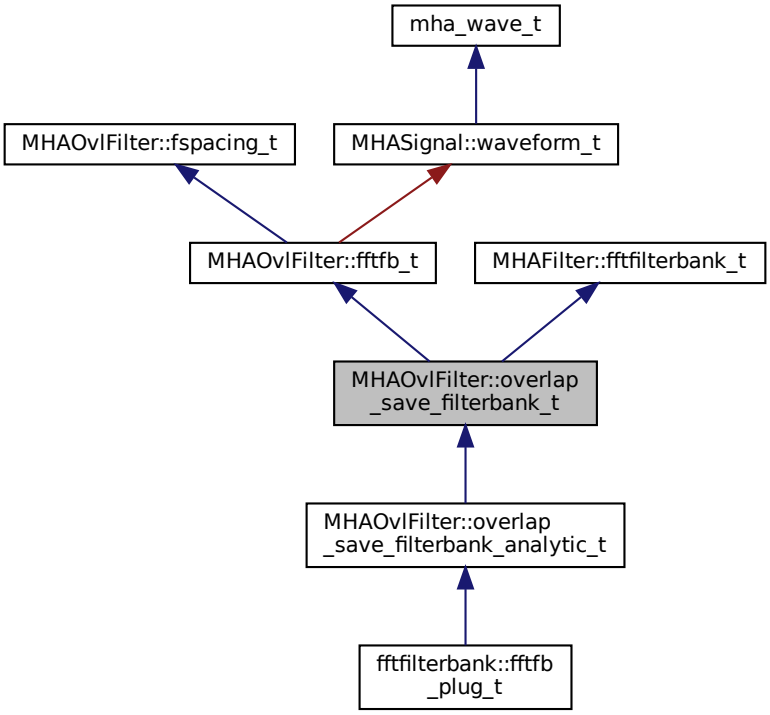
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.311 MHAOvlFilter::overlap_save_filterbank_t Class Reference

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1148).

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

5.311.1 Detailed Description

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1148).

5.311.2 Constructor & Destructor Documentation

5.311.2.1 overlap_save_filterbank_t()

```
MHAOvlFilter::overlap_save_filterbank_t::overlap_save_filterbank_t (
            MHAOvlFilter::overlap_save_filterbank_t::vars_t & fbpar,
            mhaconfig_t channelconfig_in )
```

5.311.3 Member Function Documentation

5.311.3.1 get_channelconfig()

```
mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::get_channelconfig ( ) const  [inline]
```

5.311.4 Member Data Documentation

5.311.4.1 channelconfig_out_

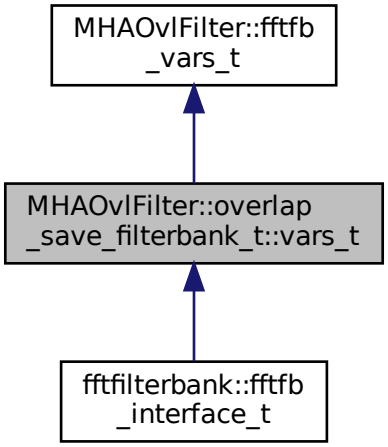
```
mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::channelconfig_out_  [private]
```

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.312 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**

5.312.1 Constructor & Destructor Documentation

5.312.1.1 vars_t() `MHAOvlFilter::overlap_save_filterbank_t::vars_t::vars_t (` **`MHAParser::parser_t`** `& p )`

5.312.2 Member Data Documentation

5.312.2.1 fftlen

MHAParser::int_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::fftlen

5.312.2.2 phasemodel

MHAParser::kw_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::phasemodel

5.312.2.3 irswnd

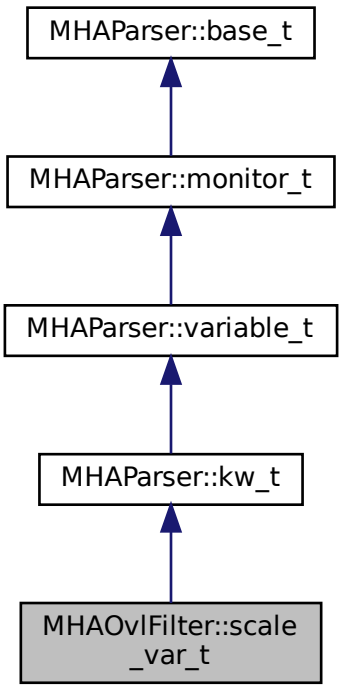
MHAParser::window_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::irswnd

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.313 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

5.313.1 Constructor & Destructor Documentation

5.313.1.1 scale_var_t() `MHAOvlFilter::scale_var_t::scale_var_t ( const std::string & help )`

5.313.2 Member Function Documentation

5.313.2.1 add_fun() `void MHAOvlFilter::scale_var_t::add_fun ( const std::string & name, scale_fun_t * fun )`

5.313.2.2 get_name() `std::string MHAOvlFilter::scale_var_t::get_name ( ) const [inline]`

5.313.2.3 get_fun() `scale_fun_t* MHAOvlFilter::scale_var_t::get_fun ( ) const [inline]`

5.313.2.4 hz2unit() `mha_real_t MHAOvlFilter::scale_var_t::hz2unit ( mha_real_t x ) const`

5.313.2.5 unit2hz() `mha_real_t MHAOvlFilter::scale_var_t::unit2hz ( mha_real_t x ) const`

5.313.3 Member Data Documentation

5.313.3.1 names `std::vector<std::string> MHAOvlFilter::scale_var_t::names [private]`

5.313.3.2 funs `std::vector< scale_fun_t*> MHAOvlFilter::scale_var_t::funs [private]`

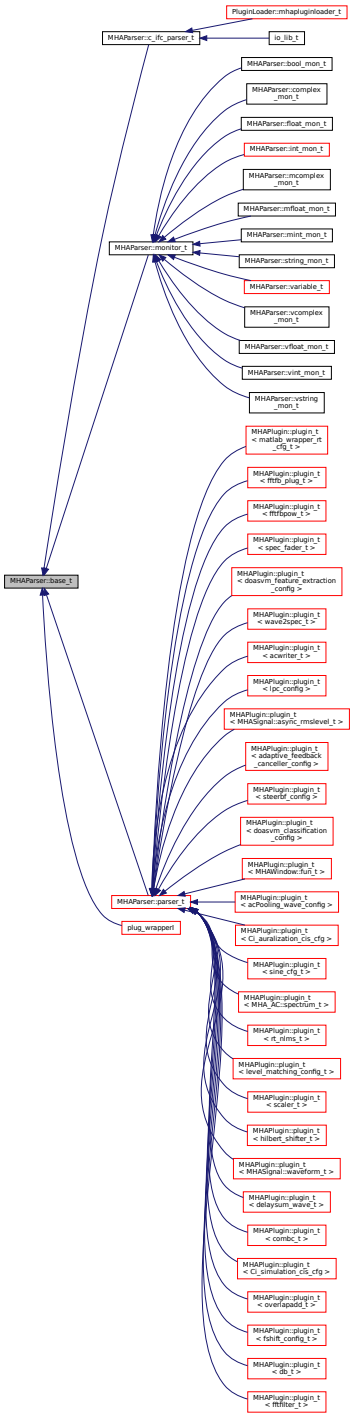
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.314 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** &)

  *Copy constructor for **base_t** (p. 1172).*
- **base_t** & **operator=** (const **base_t** &)=default
- **base_t** (**base_t** &&)=delete
- **base_t** & **operator=** (**base_t** &&)=delete
- 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**

5.314.1 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. 1177) method. Parser proxy derivatives which overwrite any of the other **parse()** (p. 1177) methods to be the key method must make sure that the original **parse()** (p. 1177) method utilizes the new key method.

5.314.2 Member Typedef Documentation

5.314.2.1 repl_list_t `typedef std::vector< replace_t> MHAParser::base_t::repl_list_t [private]`

5.314.3 Constructor & Destructor Documentation

5.314.3.1 base_t() `[1/3]` `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.

5.314.3.2 base_t() [2/3] MHAParser::base_t::base_t (const base_t & src)

Copy constructor for **base_t** (p. 1172).

Copies help text and id string, but does not insert the new node into the parser tree structure.

Parameters

src	Source parser

Deprecated Copying parser nodes makes little sense, avoid wherever possible

5.314.3.3 base_t() [3/3] MHAParser::base_t::base_t (base_t &&) [delete]

5.314.3.4 ~base_t() MHAParser::base_t::~base_t ( ) [virtual]

5.314.4 Member Function Documentation

5.314.4.1 operator=() [1/2] base_t& MHAParser::base_t::operator= (const base_t &) [default]

5.314.4.2 operator=() [2/2] base_t& MHAParser::base_t::operator= (base_t &&) [delete]

5.314.4.3 parse() [1/3] 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. 906)*	If the command cannot be executed successfully. The reason for failure is given in the message string of the exception.

Reimplemented in **plug_wrapperl** (p. 1508), **PluginLoader::mhapluginloader_t** (p. 1526), **plug_wrapper** (p. 1506), **io_wrapper** (p. 753), and **altplugs_t** (p. 325).

5.314.4.4 parse() [2/3] 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. 1177).

Parameters

cmd	Command to be parsed
retv	Buffer for the result
len	Length of buffer

Reimplemented in **altplugs_t** (p. 325).

5.314.4.5 parse() [3/3]

```
void MHAParser::base_t::parse (
    const std::vector< std::string > & cs,
    std::vector< std::string > & retv ) [virtual]
```

5.314.4.6 op_subparse()

```
std::string MHAParser::base_t::op_subparse (
    expression_t & ) [virtual]
```

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1194), and **MHAParser::parser_t** (p. 1250).

5.314.4.7 op_setval()

```
std::string MHAParser::base_t::op_setval (
    expression_t & ) [virtual]
```

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1194), **MHAParser::mcomplex_t** (p. 1226), **MHAParser::mfloat_t** (p. 1231), **MHAParser::mint_t** (p. 1243), **MHAParser::vcomplex_t** (p. 1269), **MHAParser::vfloat_t** (p. 1274), **MHAParser::vint_t** (p. 1279), **MHAParser::complex_t** (p. 1201), **MHAParser::float_t** (p. 1208), **MHAParser::int_t** (p. 1214), **MHAParser::bool_t** (p. 1191), **MHAParser::vstring_t** (p. 1283), **MHAParser::string_t** (p. 1262), **MHAParser::kw_t** (p. 1221), **MHAParser::variable_t** (p. 1264), and **MHAParser::parser_t** (p. 1251).

5.314.4.8 op_query()

```
std::string MHAParser::base_t::op_query (
    expression_t & ) [virtual]
```

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1194), **MHAParser::monitor_t** (p. 1245), and **MHAParser::parser_t** (p. 1251).

5.314.4.9 query_dump()

```
std::string MHAParser::base_t::query_dump (
    const std::string & ) [virtual]
```

Reimplemented in **MHAParser::monitor_t** (p. 1246), and **MHAParser::parser_t** (p. 1251).

5.314.4.10 query_entries() `std::string MHAParser::base_t::query_entries ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 1251).

5.314.4.11 query_perm() `std::string MHAParser::base_t::query_perm ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::variable_t** (p. 1264), and **MHAParser::monitor_t** (p. 1246).

5.314.4.12 query_range() `std::string MHAParser::base_t::query_range ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::kw_t** (p. 1221), and **MHAParser::range_var_t** (p. 1255).

5.314.4.13 query_type() `std::string MHAParser::base_t::query_type ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 1224), **MHAParser::vcomplex_mon_t** (p. 1267), **MHAParser::complex_mon_t** (p. 1199), **MHAParser::float_mon_t** (p. 1206), **MHAParser::mfloat_mon_t** (p. 1229), **MHAParser::vfloat_mon_t** (p. 1271), **MHAParser::mint_mon_t** (p. 1241), **MHAParser::vint_mon_t** (p. 1276), **MHAParser::vstring_mon_t** (p. 1281), **MHAParser::string_mon_t** (p. 1259), **MHAParser::bool_mon_t** (p. 1189), **MHAParser::int_mon_t** (p. 1211), **MHAParser::mcomplex_t** (p. 1226), **MHAParser::mfloat_t** (p. 1231), **MHAParser::mint_t** (p. 1243), **MHAParser::vcomplex_t** (p. 1269), **MHAParser::vfloat_t** (p. 1274), **MHAParser::vint_t** (p. 1279), **MHAParser::complex_t** (p. 1201), **MHAParser::float_t** (p. 1209), **MHAParser::int_t** (p. 1214), **MHAParser::bool_t** (p. 1191), **MHAParser::vstring_t** (p. 1283), **MHAParser::string_t** (p. 1262), **MHAParser::kw_t** (p. 1222), and **MHAParser::parser_t** (p. 1251).

5.314.4.14 query_val() `std::string MHAParser::base_t::query_val ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 1224), **MHAParser::vcomplex_mon_t** (p. 1266), **MHAParser::complex_mon_t** (p. 1199), **MHAParser::float_mon_t** (p. 1206), **MHAParser::mfloat_mon_t** (p. 1228), **MHAParser::vfloat_mon_t** (p. 1271), **MHAParser::mint_mon_t** (p. 1240), **MHAParser::vint_mon_t** (p. 1276), **MHAParser::vstring_mon_t** (p. 1281), **MHAParser::string_mon_t** (p. 1259), **MHAParser::bool_mon_t** (p. 1189), **MHAParser::int_mon_t** (p. 1211), **MHAParser::mcomplex_t** (p. 1227), **MHAParser::mfloat_t** (p. 1232), **MHAParser::mint_t** (p. 1244), **MHAParser::vcomplex_t** (p. 1269), **MHAParser::vfloat_t** (p. 1274), **MHAParser::vint_t** (p. 1279), **MHAParser::complex_t** (p. 1201), **MHAParser::float_t** (p. 1209), **MHAParser::int_t** (p. 1214), **MHAParser::bool_t** (p. 1191), **MHAParser::vstring_t** (p. 1283), **MHAParser::string_t** (p. 1262), **MHAParser::kw_t** (p. 1222), and **MHAParser::parser_t** (p. 1252).

5.314.4.15 query_readfile() `std::string MHAParser::base_t::query_readfile ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 1251).

5.314.4.16 query_savefile() `std::string MHAParser::base_t::query_savefile ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 1252).

5.314.4.17 query_savefile_compact() `std::string MHAParser::base_t::query_savefile_compact ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 1252).

5.314.4.18 query_savemons() `std::string MHAParser::base_t::query_savemons ( const std::string & ) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 1252).

5.314.4.19 query_listids() std::string MHAParser::base_t::query_listids (const std::string &) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1252).

5.314.4.20 query_version() std::string MHAParser::base_t::query_version (const std::string &)

5.314.4.21 query_id() std::string MHAParser::base_t::query_id (const std::string &)

5.314.4.22 query_subst() std::string MHAParser::base_t::query_subst (const std::string &)

5.314.4.23 query_addsubst() std::string MHAParser::base_t::query_addsubst (const std::string & *s*)

5.314.4.24 query_help() std::string MHAParser::base_t::query_help (const std::string &)

5.314.4.25 query_cmds() std::string MHAParser::base_t::query_cmds (const std::string &)

5.314.4.26 set_node_id() `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.

5.314.4.27 set_help() `void MHAParser::base_t::set_help ( const std::string & s )`

Set the help comment of a variable or parser.

Parameters

s	New help comment.

5.314.4.28 add_parent_on_insert() `void MHAParser::base_t::add_parent_on_insert ( parser_t * p, std::string n )`

5.314.4.29 rm_parent_on_remove() `void MHAParser::base_t::rm_parent_on_remove ( parser_t * )`

5.314.4.30 fullname() `const std::string & MHAParser::base_t::fullname ( ) const`

Return the full dot-separated path name of this parser node in the openMHA configuration tree.

5.314.4.31 activate_query() void MHAParser::base_t::activate_query (const std::string & *n*, **query_t** *a*) [protected]

5.314.4.32 notify() void MHAParser::base_t::notify ( ) [protected]

5.314.4.33 add_replace_pair() void MHAParser::base_t::add_replace_pair (const std::string & *a*, const std::string & *b*) [private]

5.314.4.34 oplist() std::string MHAParser::base_t::oplist ( ) [private]

5.314.5 Member Data Documentation

5.314.5.1 writeaccess **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.

5.314.5.2 valuechanged **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.

5.314.5.3 readaccess `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.

5.314.5.4 prereadaccess `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.

5.314.5.5 queries `query_map_t MHAParser::base_t::queries [protected]`

5.314.5.6 data_is_initialized `bool MHAParser::base_t::data_is_initialized [protected]`

5.314.5.7 help `std::string MHAParser::base_t::help [private]`

5.314.5.8 id_str `std::string MHAParser::base_t::id_str [private]`

5.314.5.9 operators `opact_map_t MHAParser::base_t::operators [private]`

5.314.5.10 repl_list `repl_list_t MHAParser::base_t::repl_list [private]`

5.314.5.11 nested_lock `bool MHAParser::base_t::nested_lock [private]`

5.314.5.12 parent `parser_t* MHAParser::base_t::parent [private]`

5.314.5.13 thefullname `std::string MHAParser::base_t::thefullname [private]`

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.315 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**

5.315.1 Constructor & Destructor Documentation

5.315.1.1 replace_t() `MHAParser::base_t::replace_t::replace_t (`

```
const std::string & ia,
const std::string & ib )
```

5.315.2 Member Function Documentation

5.315.2.1 replace() `void MHAParser::base_t::replace_t::replace ( std::string & s )`

5.315.2.2 get_a() `const std::string& MHAParser::base_t::replace_t::get_a ( ) const [inline]`

5.315.2.3 get_b() `const std::string& MHAParser::base_t::replace_t::get_b ( ) const [inline]`

5.315.3 Member Data Documentation

5.315.3.1 a `std::string MHAParser::base_t::replace_t::a [private]`

5.315.3.2 b `std::string MHAParser::base_t::replace_t::b [private]`

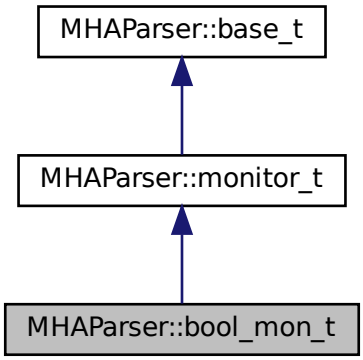
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.316 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

5.316.1 Detailed Description

Monitor with string value.

5.316.2 Constructor & Destructor Documentation

5.316.2.1 bool_mon_t() `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.

5.316.3 Member Function Documentation

5.316.3.1 query_val() `std::string MHAParser::bool_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.316.3.2 query_type() `std::string MHAParser::bool_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.316.4 Member Data Documentation

5.316.4.1 data `bool MHAParser::bool_mon_t::data`

Data field.

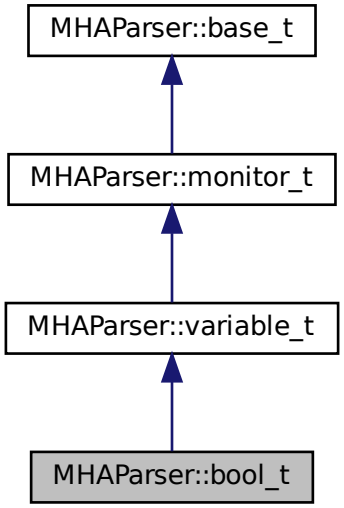
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.317 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

5.317.1 Detailed Description

Variable with a boolean value ("yes"/"no")

5.317.2 Constructor & Destructor Documentation

5.317.2.1 bool_t() `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".

5.317.3 Member Function Documentation

5.317.3.1 op_setval() `std::string MHAParser::bool_t::op_setval (`

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.317.3.2 query_type() `std::string MHAParser::bool_t::query_type (`

```
const std::string & ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.317.3.3 query_val() `std::string MHAParser::bool_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.317.4 Member Data Documentation

5.317.4.1 data `bool MHAParser::bool_t::data`

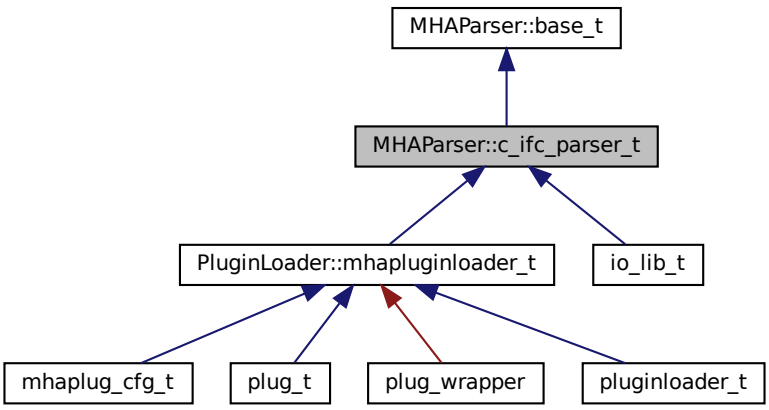
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.318 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

5.318.1 Constructor & Destructor Documentation

5.318.1.1 c_ifc_parser_t() `MHAParser::c_ifc_parser_t::c_ifc_parser_t ( const std::string & modulename_ )`

5.318.1.2 ~c_ifc_parser_t() `MHAParser::c_ifc_parser_t::~c_ifc_parser_t ( )`

5.318.2 Member Function Documentation

5.318.2.1 set_parse_cb()

```
void MHAParser::c_ifc_parser_t::set_parse_cb (
    MHAParser::c_parse_cmd_t cb,
    MHAParser::c_parse_err_t strerr,
    void * d )
```

5.318.2.2 op_subparse()

```
std::string MHAParser::c_ifc_parser_t::op_subparse (
    MHAParser::expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1179).

5.318.2.3 op_setval()

```
std::string MHAParser::c_ifc_parser_t::op_setval (
    MHAParser::expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1179).

5.318.2.4 op_query()

```
std::string MHAParser::c_ifc_parser_t::op_query (
    MHAParser::expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1179).

5.318.2.5 test_error()

```
void MHAParser::c_ifc_parser_t::test_error ( ) [private]
```

5.318.3 Member Data Documentation

5.318.3.1 modulename

```
std::string MHAParser::c_ifc_parser_t::modulename [private]
```

5.318.3.2 c_parse_cmd

`c_parse_cmd_t` MHAParser::c_ifc_parser_t::c_parse_cmd [private]

5.318.3.3 c_parse_err

`c_parse_err_t` MHAParser::c_ifc_parser_t::c_parse_err [private]

5.318.3.4 liberr

int MHAParser::c_ifc_parser_t::liberr [private]

5.318.3.5 libdata

void* MHAParser::c_ifc_parser_t::libdata [private]

5.318.3.6 ret_size

unsigned int MHAParser::c_ifc_parser_t::ret_size [private]

5.318.3.7 retv

char* MHAParser::c_ifc_parser_t::retv [private]

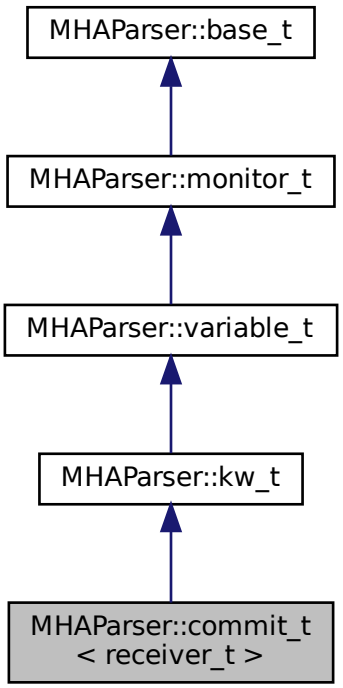
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.319 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

5.319.1 Detailed Description

template<class receiver_t>
class MHAParser::commit_t< receiver_t >

Parser variable with event-emission functionality.

The **commit_t** (p. 1196) variable can register an event receiver in its constructor, which is called whenever the variable is set to "commit".

5.319.2 Constructor & Destructor Documentation

5.319.2.1 commit_t() 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" )
```

5.319.3 Member Data Documentation

5.319.3.1 extern_connector template<class receiver_t >

```
MHAEvents::connector_t<receiver_t>  MHAParser::commit_t< receiver_t >::extern_↩
connector  [private]
```

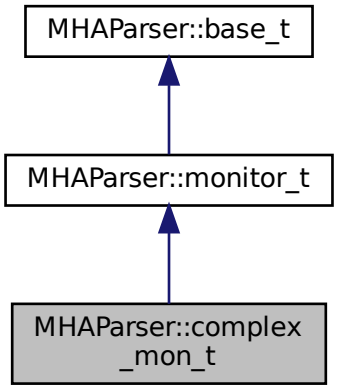
The documentation for this class was generated from the following file:

- **mha_parser.hh**

5.320 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

5.320.1 Detailed Description

Monitor with complex value.

5.320.2 Constructor & Destructor Documentation

5.320.2.1 complex_mon_t() `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.

5.320.3 Member Function Documentation

5.320.3.1 query_val() `std::string MHAParser::complex_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.320.3.2 query_type() `std::string MHAParser::complex_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.320.4 Member Data Documentation

5.320.4.1 data `mha_complex_t MHAParser::complex_mon_t::data`

Data field.

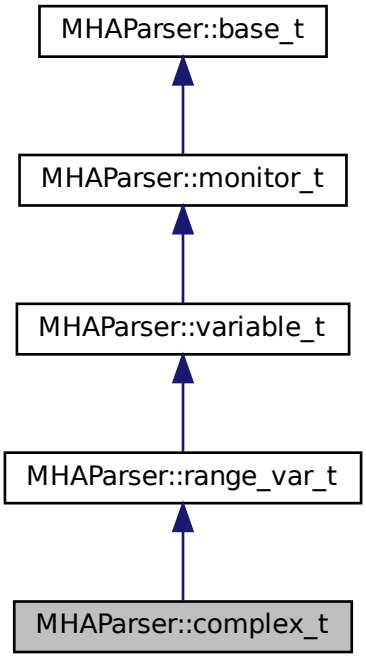
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.321 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

5.321.1 Detailed Description

Variable with complex value.

5.321.2 Constructor & Destructor Documentation

5.321.2.1 complex_t()

```
MHAParser::complex_t::complex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

5.321.3 Member Function Documentation

5.321.3.1 op_setval()

```
std::string MHAParser::complex_t::op_setval (
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.321.3.2 query_type()

```
std::string MHAParser::complex_t::query_type (
    const std::string & ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.321.3.3 query_val()

```
std::string MHAParser::complex_t::query_val (
    const std::string & s ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.321.4 Member Data Documentation

5.321.4.1 data `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**

5.322 MHAParser::entry_t Class Reference

Public Member Functions

- **entry_t** (const std::string &, **base_t** ∗)

Public Attributes

- std::string **name**
- **base_t** ∗ **entry**

5.322.1 Constructor & Destructor Documentation

5.322.1.1 entry_t() `MHAParser::entry_t::entry_t (`

```
const std::string & n,
 base_t * e )
```

5.322.2 Member Data Documentation

5.322.2.1 name `std::string MHAParser::entry_t::name`

5.322.2.2 entry `base_t* MHAParser::entry_t::entry`

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.323 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**

5.323.1 Constructor & Destructor Documentation

Parameters

5.323.1.1 expression_t() [1/2] 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)

5.323.1.2 expression_t() [2/2] expression_t::expression_t ( )

5.323.2 Member Data Documentation

5.323.2.1 lval std::string MHAParser::expression_t::lval

5.323.2.2 rval std::string MHAParser::expression_t::rval

5.323.2.3 op std::string MHAParser::expression_t::op

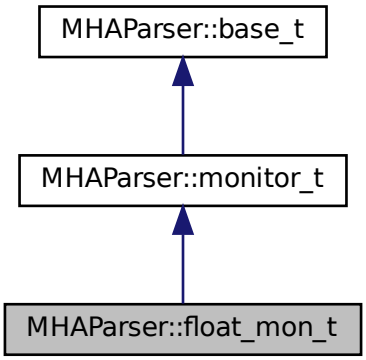
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.324 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

5.324.1 Detailed Description

Monitor with float value.

5.324.2 Constructor & Destructor Documentation

5.324.2.1 float_mon_t() `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.

5.324.3 Member Function Documentation

5.324.3.1 query_val() `std::string MHAParser::float_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.324.3.2 query_type() `std::string MHAParser::float_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.324.4 Member Data Documentation

5.324.4.1 data `float MHAParser::float_mon_t::data`

Data field.

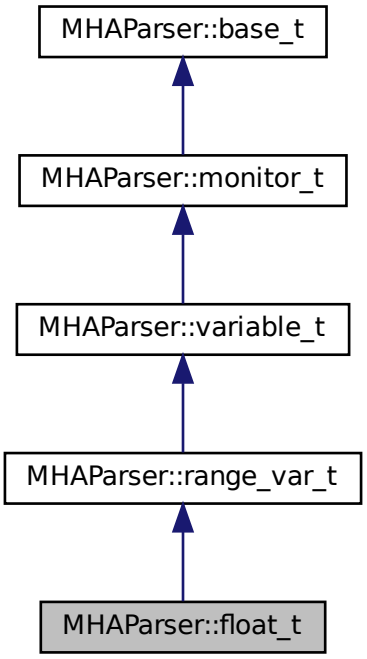
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.325 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

5.325.1 Detailed Description

Variable with float value.

5.325.2 Constructor & Destructor Documentation

5.325.2.1 float_t()

```
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.

5.325.3 Member Function Documentation

5.325.3.1 op_setval() std::string MHAParser::float_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.325.3.2 query_type() std::string MHAParser::float_t::query_type (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

5.325.3.3 query_val() std::string MHAParser::float_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

5.325.4 Member Data Documentation

5.325.4.1 data float MHAParser::float_t::data

Data field.

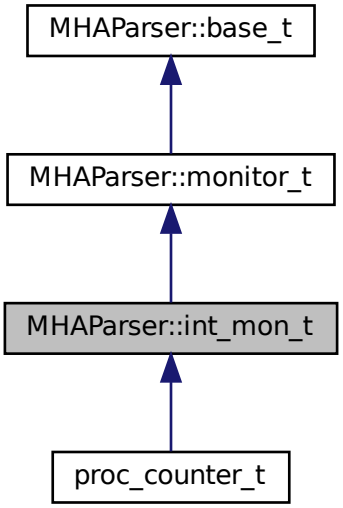
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.326 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

5.326.1 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.

5.326.2 Constructor & Destructor Documentation

5.326.2.1 int_mon_t() `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.

5.326.3 Member Function Documentation

5.326.3.1 query_val() `std::string MHAParser::int_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.326.3.2 query_type() `std::string MHAParser::int_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.326.4 Member Data Documentation

5.326.4.1 data `int MHAParser::int_mon_t::data`

Data field.

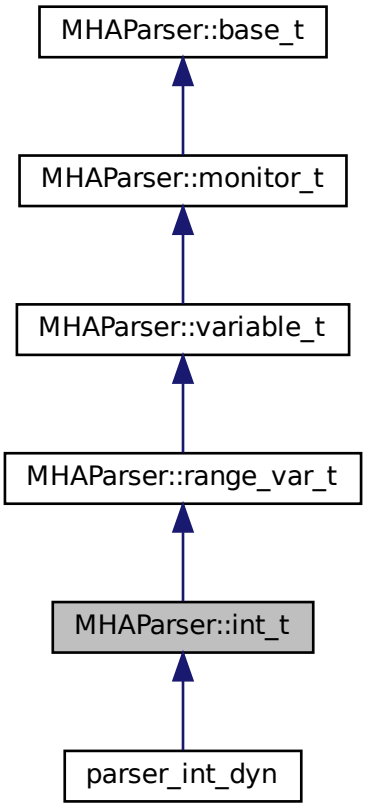
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.327 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

5.327.1 Detailed Description

Variable with integer value.

5.327.2 Constructor & Destructor Documentation

5.327.2.1 int_t()

```
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.

Parameters

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]).

5.327.3 Member Function Documentation

5.327.3.1 op_setval() `std::string MHAParser::int_t::op_setval ( expression_t & x ) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.327.3.2 query_type() `std::string MHAParser::int_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.327.3.3 query_val() `std::string MHAParser::int_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.327.4 Member Data Documentation

5.327.4.1 data `int MHAParser::int_t::data`

Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.328 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**

5.328.1 Detailed Description

Keyword list class.

The stucture **keyword_list_t** (p. 1215) defines a keyword list (vector of strings) with an index into the list. Used as **MHAParser::kw_t** (p. 1219), it can be used to access a set of valid keywords through the parser (i.e. one of "pear apple banana").

5.328.2 Member Typedef Documentation

5.328.2.1 size_t `typedef std::vector<std::string>::size_type` **MHAParser::keyword_list_t::size_t**

5.328.3 Constructor & Destructor Documentation

5.328.3.1 keyword_list_t() `MHAParser::keyword_list_t::keyword_list_t ( )`

Constructor.

5.328.4 Member Function Documentation

5.328.4.1 set_value() `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.

5.328.4.2 set_entries() `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.

5.328.4.3 get_value() `const std::string & MHAParser::keyword_list_t::get_value ( ) const`

Return selected value.

5.328.4.4 get_entries() `const std::vector< std::string > & MHAParser::keyword_list_t::get_entries ( ) const`

Return keyword list.

5.328.4.5 get_index() const **MHAParser::keyword_list_t::size_t** & MHAParser::keyword↩ _list_t::get_index ( ) const

Return index of selected value.

5.328.4.6 set_index() void MHAParser::keyword_list_t::set_index (unsigned int *idx*)

5.328.4.7 validate() void MHAParser::keyword_list_t::validate ( ) const

Check if index of selected value is valid.

5.328.4.8 add_entry() void MHAParser::keyword_list_t::add_entry (const std::string & *en*) [inline]

5.328.5 Member Data Documentation

5.328.5.1 index **size_t** MHAParser::keyword_list_t::index [private]

Index into list.

5.328.5.2 entries std::vector<std::string> MHAParser::keyword_list_t::entries [private]

List of valid entries.

5.328.5.3 empty_string `std::string MHAParser::keyword_list_t::empty_string [private]`

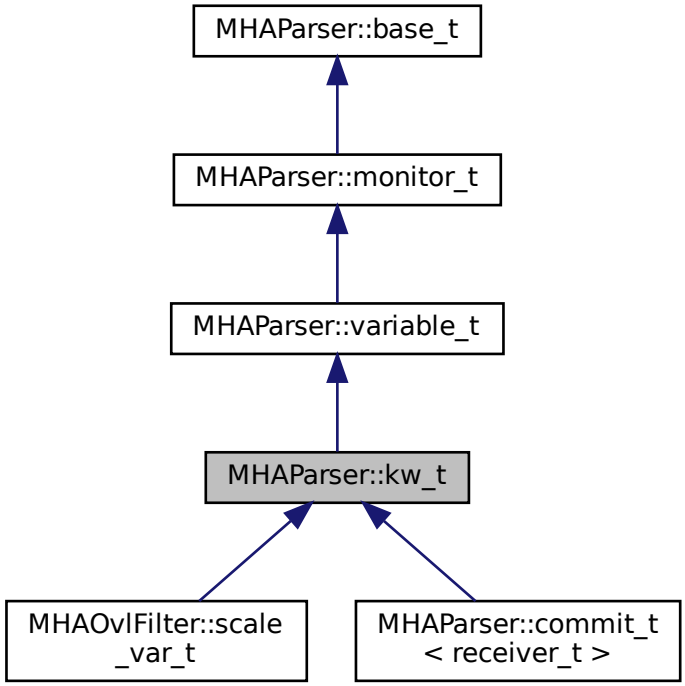
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.329 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

5.329.1 Detailed Description

Variable with keyword list value.

5.329.2 Constructor & Destructor Documentation

5.329.2.1 kw_t() [1/2]

```
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 "[", "]".

5.329.2.2 kw_t() [2/2] MHAParser::kw_t::kw_t (const **kw_t** & *src*)

Copy constructor.

5.329.3 Member Function Documentation

5.329.3.1 set_range() 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 "[", "]".

5.329.3.2 isval() bool MHAParser::kw_t::isval (const std::string & *testval*) const

Test if the given value is selected.

5.329.3.3 validate() void MHAParser::kw_t::validate (const **keyword_list_t** & *s*) [protected]

5.329.3.4 op_setval() std::string MHAParser::kw_t::op_setval (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.329.3.5 query_range() `std::string MHAParser::kw_t::query_range ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.329.3.6 query_val() `std::string MHAParser::kw_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.329.3.7 query_type() `std::string MHAParser::kw_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.329.4 Member Data Documentation

5.329.4.1 data `keyword_list_t MHAParser::kw_t::data`

Variable data in its native type.

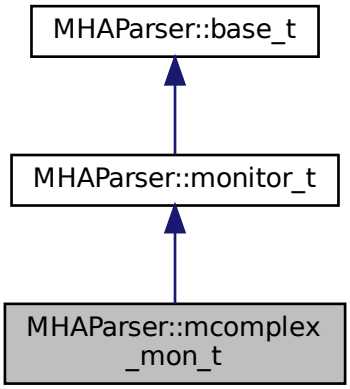
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.330 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

5.330.1 Detailed Description

Matrix of complex numbers monitor.

5.330.2 Constructor & Destructor Documentation

5.330.2.1 mcomplex_mon_t() `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.

5.330.3 Member Function Documentation

5.330.3.1 query_val() `std::string MHAParser::mcomplex_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.330.3.2 query_type() `std::string MHAParser::mcomplex_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.330.4 Member Data Documentation

5.330.4.1 data `std::vector< std::vector< mha_complex_t> > MHAParser::mcomplex_t::data`

Data field.

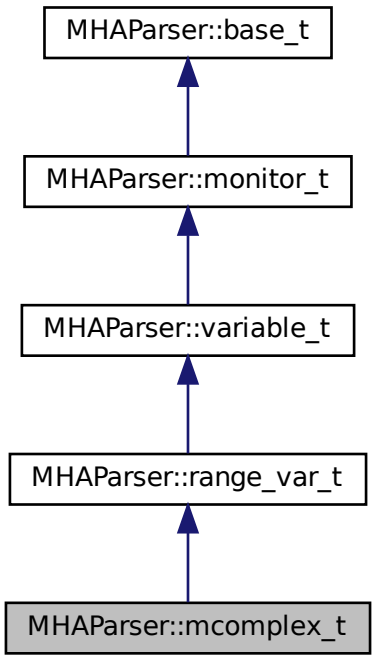
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.331 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

5.331.1 Detailed Description

Matrix variable with complex value.

5.331.2 Constructor & Destructor Documentation

5.331.2.1 mcomplex_t()

```
MHAParser::mcomplex_t::mcomplex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

5.331.3 Member Function Documentation

5.331.3.1 op_setval()

```
std::string MHAParser::mcomplex_t::op_setval (
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.331.3.2 query_type() `std::string MHAParser::mcomplex_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.331.3.3 query_val() `std::string MHAParser::mcomplex_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.331.4 Member Data Documentation

5.331.4.1 data `std::vector<std::vector< mha_complex_t> > MHAParser::mcomplex_t::data`

Data field.

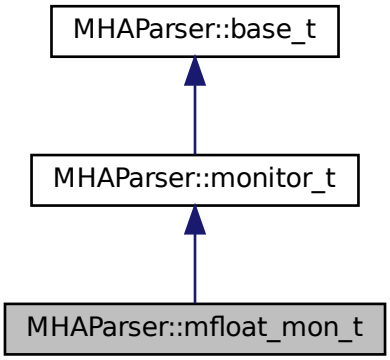
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.332 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

5.332.1 Detailed Description

Matrix of floats monitor.

5.332.2 Constructor & Destructor Documentation

5.332.2.1 mfloat_mon_t() `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.

5.332.3 Member Function Documentation

5.332.3.1 query_val() `std::string MHAParser::mfloat_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.332.3.2 query_type() `std::string MHAParser::mfloat_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.332.4 Member Data Documentation

5.332.4.1 data `std::vector< std::vector<float> > MHAParser::mfloat_mon_t::data`

Data field.

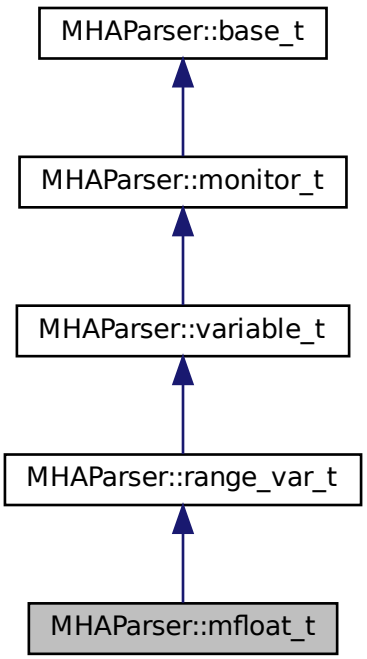
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.333 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

5.333.1 Detailed Description

Matrix variable with float value.

5.333.2 Constructor & Destructor Documentation

5.333.2.1 mfloat_t() `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.

5.333.3 Member Function Documentation

5.333.3.1 op_setval() `std::string MHAParser::mfloat_t::op_setval (`

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.333.3.2 query_type() `std::string MHAParser::mfloat_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.333.3.3 query_val() `std::string MHAParser::mfloat_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.333.4 Member Data Documentation

5.333.4.1 data `std::vector<std::vector<float> > MHAParser::mfloat_t::data`

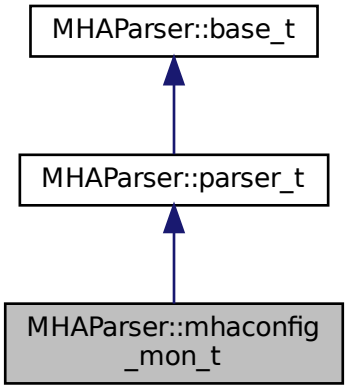
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.334 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

5.334.1 Constructor & Destructor Documentation

5.334.1.1 mhaconfig_mon_t() `MHAParser::mhaconfig_mon_t::mhaconfig_mon_t ( const std::string & help = "" )`

5.334.2 Member Function Documentation

5.334.2.1 update() `void MHAParser::mhaconfig_mon_t::update ( const mhaconfig_t & cf )`

5.334.3 Member Data Documentation

5.334.3.1 channels **MHAParser::int_mon_t** MHAParser::mhaconfig_mon_t::channels [private]

Number of audio channels.

5.334.3.2 domain **MHAParser::string_mon_t** MHAParser::mhaconfig_mon_t::domain [private]

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.334.3.3 fragsize **MHAParser::int_mon_t** MHAParser::mhaconfig_mon_t::fragsize [private]

Fragment size of waveform data.

5.334.3.4 wndlen **MHAParser::int_mon_t** MHAParser::mhaconfig_mon_t::wndlen [private]

Window length of spectral data.

5.334.3.5 fftlen **MHAParser::int_mon_t** MHAParser::mhaconfig_mon_t::fftlen [private]

FFT length of spectral data.

5.334.3.6 srate **MHAParser::float_mon_t** MHAParser::mhaconfig_mon_t::srate [private]

Sampling rate in Hz.

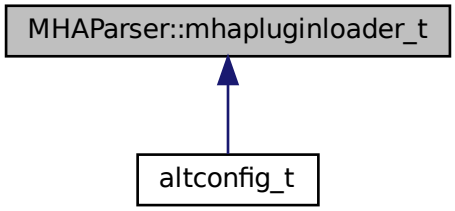
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.335 MHAParser::mhapluginloader_t Class Reference

Class to create a plugin loader in a parser, including the load logic.

Inheritance diagram for MHAParser::mhapluginloader_t:



Public Member Functions

- **mhapluginloader_t** (**MHAParser::parser_t** &parent, **MHA_AC::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**
- **MHA_AC::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**

5.335.1 Detailed Description

Class to create a plugin loader in a parser, including the load logic.

5.335.2 Constructor & Destructor Documentation

5.335.2.1 mhapluginloader_t()

```
MHAParser::mhapluginloader_t::mhapluginloader_t (
    MHAParser::parser_t & parent,
    MHA_AC::algo_comm_t & ac,
    const std::string & plugname_name = "plugin_name",
    const std::string & prefix = "" )
```

5.335.2.2 ~mhapluginloader_t()

```
MHAParser::mhapluginloader_t::~mhapluginloader_t ( )
```

5.335.3 Member Function Documentation

5.335.3.1 prepare()

```
void MHAParser::mhapluginloader_t::prepare (
    mhaconfig_t & cf )
```

5.335.3.2 release()

```
void MHAParser::mhapluginloader_t::release ( )
```

5.335.3.3 process() [1/4]

```
void MHAParser::mhapluginloader_t::process (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [inline]
```

5.335.3.4 process() [2/4]

```
void MHAParser::mhapluginloader_t::process (
    mha_spec_t * sIn,
    mha_spec_t ** sOut ) [inline]
```

5.335.3.5 process() [3/4]

```
void MHAParser::mhapluginloader_t::process (
    mha_wave_t * sIn,
    mha_spec_t ** sOut ) [inline]
```

5.335.3.6 process() [4/4]

```
void MHAParser::mhapluginloader_t::process (
    mha_spec_t * sIn,
    mha_wave_t ** sOut ) [inline]
```

5.335.3.7 get_cfin()

```
mhaconfig_t MHAParser::mhapluginloader_t::get_cfin ( ) const
[inline]
```

5.335.3.8 get_cfout() **mhaconfig_t** MHAParser::mhapluginloader_t::get_cfout ( ) const [inline]

5.335.3.9 get_last_name() const std::string& MHAParser::mhapluginloader_t::get_↩ last_name ( ) const [inline]

5.335.3.10 load_plug() void MHAParser::mhapluginloader_t::load_plug ( ) [private]

5.335.4 Member Data Documentation

5.335.4.1 plug **PluginLoader**::**mhapluginloader_t*** MHAParser::mhapluginloader_t::plug [protected]

5.335.4.2 parent_ **MHAParser**::**parser_t**& MHAParser::mhapluginloader_t::parent_↩ [private]

5.335.4.3 plugname **MHAParser**::**string_t** MHAParser::mhapluginloader_t::plugname [private]

5.335.4.4 prefix_ std::string MHAParser::mhapluginloader_t::prefix_ [private]

5.335.4.5 connector **MHAEvents**::**connector_t**< **mhapluginloader_t**> MHAParser::mhapluginloader↩ _t::connector [private]

5.335.4.6 ac_ `MHA_AC::algo_comm_t& MHAParser::mhapluginloader_t::ac_ [private]`

5.335.4.7 last_name `std::string MHAParser::mhapluginloader_t::last_name [private]`

5.335.4.8 plugname_name_ `std::string MHAParser::mhapluginloader_t::plugname_↩ name_ [private]`

5.335.4.9 cf_in_ `mhaconfig_t MHAParser::mhapluginloader_t::cf_in_ [private]`

5.335.4.10 cf_out_ `mhaconfig_t MHAParser::mhapluginloader_t::cf_out_ [private]`

5.335.4.11 bookkeeping `double MHAParser::mhapluginloader_t::bookkeeping [static], [private]`

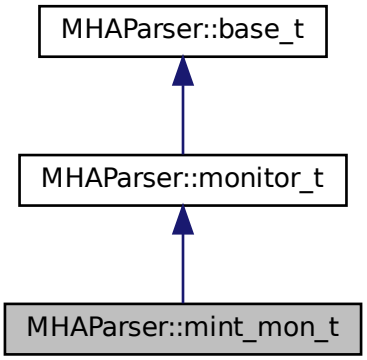
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.336 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

5.336.1 Detailed Description

Matrix of ints monitor.

5.336.2 Constructor & Destructor Documentation

5.336.2.1 mint_mon_t() `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.

5.336.3 Member Function Documentation

5.336.3.1 query_val() `std::string MHAParser::mint_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.336.3.2 query_type() `std::string MHAParser::mint_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.336.4 Member Data Documentation

5.336.4.1 data `std::vector< std::vector<int> > MHAParser::mint_mon_t::data`

Data field.

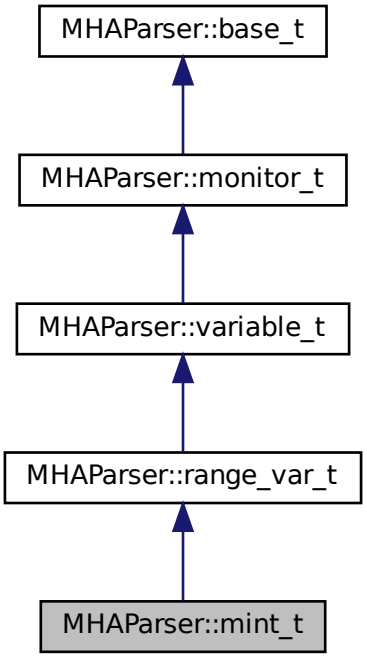
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.337 MHAParser::mint_t Class Reference

Matrix variable with int value.

Inheritance diagram for MHAParser::mint_t:



Public Member Functions

- **mint_t** (const std::string &, const std::string &, const std::string &="")
  Create a int matrix parser variable.

Public Attributes

- std::vector< 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

5.337.1 Detailed Description

Matrix variable with int value.

5.337.2 Constructor & Destructor Documentation

5.337.2.1 mint_t() `MHAParser::mint_t::mint_t (`

```
const std::string & h,
const std::string & v,
const std::string & rg = "" )
```

Create a int 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.

5.337.3 Member Function Documentation

5.337.3.1 op_setval() `std::string MHAParser::mint_t::op_setval (`

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.337.3.2 query_type() `std::string MHAParser::mint_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.337.3.3 query_val() `std::string MHAParser::mint_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.337.4 Member Data Documentation

5.337.4.1 data `std::vector<std::vector<int> > MHAParser::mint_t::data`

Data field.

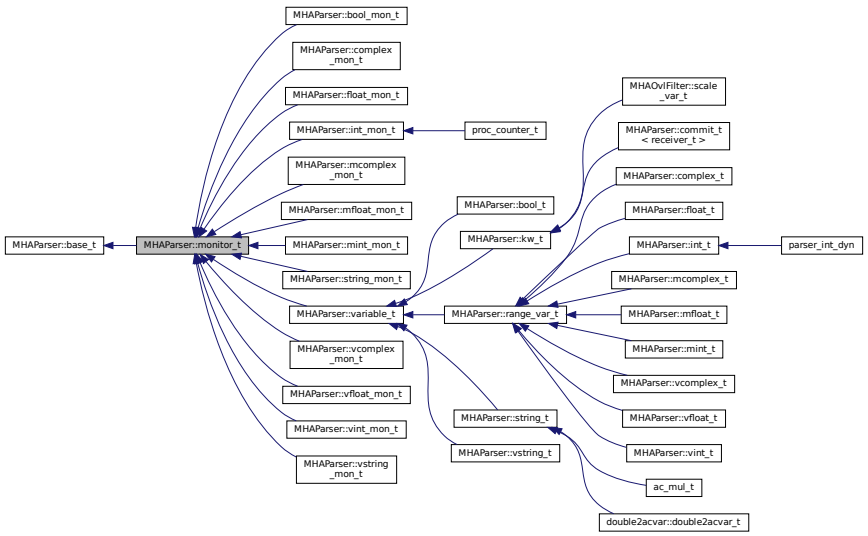
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.338 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** &)
- **monitor_t** & **operator=** (const **monitor_t** &)=default
- std::string **op_query** (**expression_t** &)
- std::string **query_dump** (const std::string &)
- std::string **query_perm** (const std::string &)

Additional Inherited Members

5.338.1 Detailed Description

Base class for monitors and variable nodes.

5.338.2 Constructor & Destructor Documentation

5.338.2.1 monitor_t() [1/2] MHAParser::monitor_t::monitor_t (const std::string & *h*)

5.338.2.2 monitor_t() [2/2] MHAParser::monitor_t::monitor_t (const **monitor_t** & *src*)

5.338.3 Member Function Documentation

5.338.3.1 operator=() **monitor_t**& MHAParser::monitor_t::operator= (const **monitor_t** &) [default]

5.338.3.2 op_query() std::string MHAParser::monitor_t::op_query (expression_t & x) [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.338.3.3 query_dump() std::string MHAParser::monitor_t::query_dump (const std::string &) [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.338.3.4 query_perm() std::string MHAParser::monitor_t::query_perm (const std::string &) [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

Reimplemented in **MHAParser::variable_t** (p. 1264).

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.339 MHAParser::parser_t Class Reference

Parser node class.

Inherits **MHAParser::base_t**.

Inherited by **MHAPlugin::plugin_t< matlab_wrapper_rt_cfg_t >**, **MHAPlugin::plugin_t< fftfb_plug_t >**, **MHAPlugin::plugin_t< fftfbpow_t >**, **MHAPlugin::plugin_t< spec_fader_t >**, **MHAPlugin::plugin_t< doasvm_feature_extraction_config >**, **MHAPlugin::plugin_t< wave2spec_t >**, **MHAPlugin::plugin_t< acwriter_t >**, **MHAPlugin::plugin_t< lpc_config >**, **MHAPlugin::plugin_t< MHASignal::async_rmslevel_t >**, **MHAPlugin::plugin_t< adaptive_feedback_canceller_config >**, **MHAPlugin::plugin_t< steerbf_config >**, **MHAPlugin::plugin_t< doasvm_classification_config >**, **MHAPlugin::plugin_t< MHAWindow::fun_t >**, **MHAPlugin::plugin_t< acPooling_wave_config >**, **MHAPlugin::plugin_t< Ci_auralization_cis_cfg >**, **MHAPlugin::plugin_t< sine_cfg_t >**, **MHAPlugin::plugin_t< MHA_AC::spectrum_t >**, **MHAPlugin::plugin_t< rt_nlms_t >**, **MHAPlugin::plugin_t< level_matching_config_t >**, **MHAPlugin::plugin_t< scaler_t >**, **MHAPlugin::plugin_t< hilbert_shifter_t >**, **MHAPlugin::plugin_t<**

MHASignal::waveform_t >, **MHAPlugin::plugin_t**< **delaysum_wave_t** >, **MHAPlugin::plugin_t**< **combc_t** >, **MHAPlugin::plugin_t**< **Ci_simulation_cis_cfg** >, **MHAPlugin::plugin_t**< **overlapadd_t** >, **MHAPlugin::plugin_t**< **fshift_config_t** >, **MHAPlugin::plugin_t**< **db_t** >, **MHAPlugin::plugin_t**< **fftfilter_t** >, **MHAPlugin::plugin_t**< **dc_t** >, **MHAPlugin::plugin_t**< **int** >, **MHAPlugin::plugin_t**< **cfg_t** >, **MHAPlugin::plugin_t**< **Ci_auralization_ace_cfg** >, **MHAPlugin::plugin_t**< **smooth_cepstrum_t** >, **MHAPlugin::plugin_t**< **UNIT** >, **MHAPlugin::plugin_t**< **MHAFilter::partitioned_convolution_t** >, **MHAPlugin::plugin_t**< **route::process_t** >, **MHAPlugin::plugin_t**< **lpc_bl_predictor_config** >, **MHAPlugin::plugin_t**< **dbasync_t** >, **MHAPlugin::plugin_t**< **cpuload_cfg_t** >, **MHAPlugin::plugin_t**< **noise_psd_estimator_t** >, **MHAPlugin::plugin_t**< **gtfb_simple_rt_t** >, **MHAPlugin::plugin_t**< **gtfb_simd_cfg_t** >, **MHAPlugin::plugin_t**< **ac2wave_t** >, **MHAPlugin::plugin_t**< **Ci_simulation_ace_cfg** >, **MHAPlugin::plugin_t**< **MHA_AC::waveform_t** >, **MHAPlugin::plugin_t**< **mhachain::plugs_t** >, **MHAPlugin::plugin_t**< **gsc_adaptive_stage** >, **MHAPlugin::plugin_t**< **char** >, **MHAPlugin::plugin_t**< **smoothspec_wrap_t** >, **MHAPlugin::plugin_t**< **prediction_error_config** >, **MHAPlugin::plugin_t**< **lpc_burglattice_config** >, **MHAPlugin::plugin_t**< **gtfb_analyzer_cfg_t** >, **MHAPlugin::plugin_t**< **adm_rtconfig_t** >, **MHAPlugin::plugin_t**< **example5_t** >, **MHAPlugin::plugin_t**< **cohflt_t** >, **MHAPlugin::plugin_t**< **acConcat_wave_config** >, **MHAPlugin::plugin_t**< **ac2xdf_rt_t** >, **MHAPlugin::plugin_t**< **Set_rms_cfg** >, **MHAPlugin::plugin_t**< **wavwriter_t** >, **MHAPlugin::plugin_t**< **spec2wave_t** >, **MHAPlugin::plugin_t**< **rohConfig** >, **MHAPlugin::plugin_t**< **delaysum_t** >, **MHAPlugin::plugin_t**< **MHASignal::delay_t** >, **MHAPlugin::plugin_t**< **acSteer_config** >, **MHAPlugin::plugin_t**< **resampling_t** >, **MHAPlugin::plugin_t**< **trigger2lsl_rt_t** >, **MHAPlugin::plugin_t**< **plingploing_t** >, **MHAPlugin::plugin_t**< **analysepath_t** >, **MHAPlugin::plugin_t**< **acTransform_wave_config** >, **MHAPlugin::plugin_t**< **Get_rms_cfg** >, **AuditoryProfile::parser_t**, **AuditoryProfile::parser_t::ear_t**, **AuditoryProfile::parser_t::fmap_t**, **DynComp::dc_afterburn_vars_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**, **MHAIOPortAudio::stream_info_t**, **MHAParser::mhaconfig_mon_t**, **MHAParser::window_t**, **MHAPlugin::plugin_t**< **runtime_cfg_t** >, **MHAPlugin_Split::split_t**, **alsa_dev_par_parser_t**, **fw_t**, **io_alsa_t**, **io_asterisk_parser_t**, **io_dummy_t**, **io_file_t**, **io_parser_t**, **io_tcp_parser_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 &)
- bool **has_entry** (const std::string &)

Private Attributes

- **entry_map_t entries**
- std::string **id_string**
  identification string
- std::string **last_errormsg**

Additional Inherited Members

5.339.1 Detailed Description

Parser node class.

A **parser_t** (p. 1246) instance is a node in the configuration tree. A parser node can contain any number of other **parser_t** (p. 1246) instances or configuration language variables. These items are inserted into a parser node using the **parser_t::insert_item** (p. 1249) method.

5.339.2 Constructor & Destructor Documentation

5.339.2.1 parser_t() 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.

5.339.2.2 ~parser_t() MHAParser::parser_t::~parser_t ( )

5.339.3 Member Function Documentation

5.339.3.1 insert_item() 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.

5.339.3.2 remove_item() [1/2] 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.

5.339.3.3 force_remove_item() 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.

5.339.3.4 remove_item() [2/2] 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.

5.339.3.5 op_subparse() std::string MHAParser::parser_t::op_subparse (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.339.3.6 op_setval() std::string MHAParser::parser_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.339.3.7 op_query() std::string MHAParser::parser_t::op_query (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.339.3.8 query_type() std::string MHAParser::parser_t::query_type (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

5.339.3.9 query_dump() std::string MHAParser::parser_t::query_dump (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.339.3.10 query_entries() std::string MHAParser::parser_t::query_entries (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1179).

5.339.3.11 query_readfile()

`std::string MHAParser::parser_t::query_readfile ( const std::string & fname ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1181).

5.339.3.12 query_savefile()

`std::string MHAParser::parser_t::query_savefile ( const std::string & fname ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1181).

5.339.3.13 query_savefile_compact()

`std::string MHAParser::parser_t::query_savefile_compact ( const std::string & fname ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1181).

5.339.3.14 query_savemons()

`std::string MHAParser::parser_t::query_savemons ( const std::string & fname ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1181).

5.339.3.15 query_val()

`std::string MHAParser::parser_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.339.3.16 query_listids()

`std::string MHAParser::parser_t::query_listids ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1181).

5.339.3.17 set_id_string() void MHAParser::parser_t::set_id_string (const std::string & *s*) [protected]

5.339.3.18 has_entry() bool MHAParser::parser_t::has_entry (const std::string & *s*) [protected]

5.339.4 Member Data Documentation

5.339.4.1 entries **entry_map_t** MHAParser::parser_t::entries [private]

5.339.4.2 id_string std::string MHAParser::parser_t::id_string [private]

identification string

5.339.4.3 last_errormsg std::string MHAParser::parser_t::last_errormsg [private]

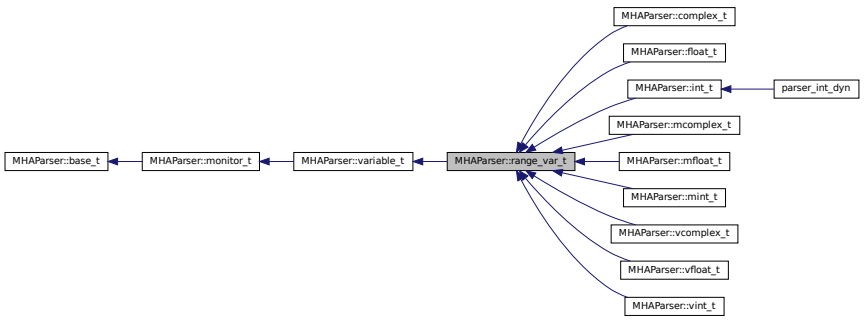
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.340 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< int > > &)
- 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

5.340.1 Detailed Description

Base class for all variables with a numeric value range.

5.340.2 Constructor & Destructor Documentation

5.340.2.1 range_var_t() [1/2] MHAParser::range_var_t::range_var_t (

```
const std::string & h,
const std::string & r = "" )
```

5.340.2.2 range_var_t() [2/2] MHAParser::range_var_t::range_var_t (

```
const  range_var_t & src )
```

5.340.3 Member Function Documentation

5.340.3.1 query_range() std::string MHAParser::range_var_t::query_range (

```
const std::string &  ) [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.340.3.2 set_range() 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)

5.340.3.3 validate() [1/9] void MHAParser::range_var_t::validate (

```
const int & v )
```

5.340.3.4 validate() [2/9]

```
void MHAParser::range_var_t::validate (
    const float & v )
```

5.340.3.5 validate() [3/9]

```
void MHAParser::range_var_t::validate (
    const mha_complex_t & v )
```

5.340.3.6 validate() [4/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< int > & v )
```

5.340.3.7 validate() [5/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< float > & v )
```

5.340.3.8 validate() [6/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< mha_complex_t > & v )
```

5.340.3.9 validate() [7/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< std::vector< int > > & v )
```

5.340.3.10 validate() [8/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< std::vector< float > > & v )
```

5.340.3.11 validate() [9/9]

```
void MHAParser::range_var_t::validate (
    const std::vector< std::vector< mha_complex_t > > & v )
```

5.340.4 Member Data Documentation

5.340.4.1 low_limit `float MHAParser::range_var_t::low_limit [protected]`

Lower limit of range.

5.340.4.2 up_limit `float MHAParser::range_var_t::up_limit [protected]`

Upper limit of range.

5.340.4.3 low_incl `bool MHAParser::range_var_t::low_incl [protected]`

Lower limit is included (or excluded) in range.

5.340.4.4 up_incl `bool MHAParser::range_var_t::up_incl [protected]`

Upper limit is included (or excluded) in range.

5.340.4.5 check_low `bool MHAParser::range_var_t::check_low [protected]`

Check lower limit.

5.340.4.6 check_up `bool MHAParser::range_var_t::check_up [protected]`

Check upper limit.

5.340.4.7 check_range `bool MHAParser::range_var_t::check_range [protected]`

Range checking is active.

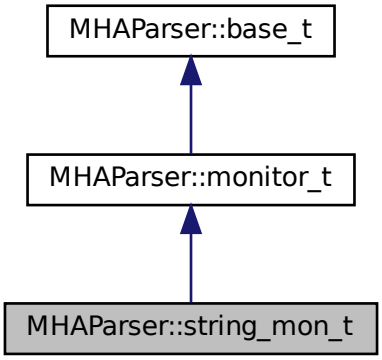
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.341 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

5.341.1 Detailed Description

Monitor with string value.

5.341.2 Constructor & Destructor Documentation

5.341.2.1 string_mon_t() `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.

5.341.3 Member Function Documentation

5.341.3.1 query_val() `std::string MHAParser::string_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.341.3.2 query_type() `std::string MHAParser::string_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.341.4 Member Data Documentation

5.341.4.1 data `std::string MHAParser::string_mon_t::data`

Data field.

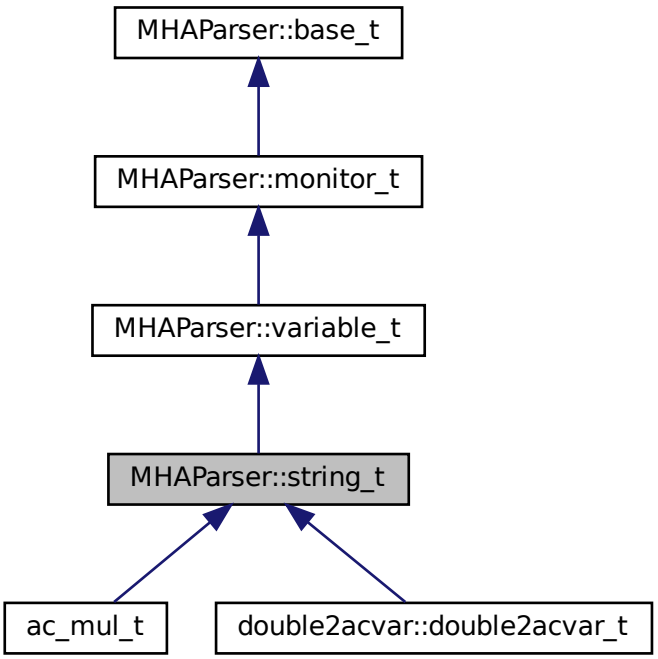
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.342 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

5.342.1 Detailed Description

Variable with a string value.

5.342.2 Constructor & Destructor Documentation

5.342.2.1 string_t() `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

5.342.3 Member Function Documentation

5.342.3.1 op_setval() `std::string MHAParser::string_t::op_setval ( expression_t & x ) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.342.3.2 query_type() `std::string MHAParser::string_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.342.3.3 query_val() `std::string MHAParser::string_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.342.4 Member Data Documentation

5.342.4.1 data `std::string MHAParser::string_t::data`

Data field.

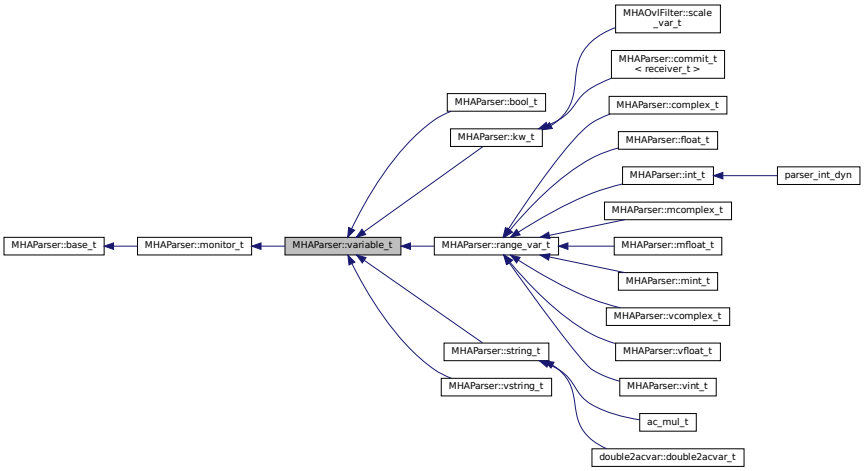
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.343 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

5.343.1 Detailed Description

Base class for variable nodes.

5.343.2 Constructor & Destructor Documentation

5.343.2.1 variable_t() `MHAParser::variable_t::variable_t (`

`const std::string & h )`

5.343.3 Member Function Documentation

5.343.3.1 op_setval() `std::string MHAParser::variable_t::op_setval (`

`expression_t` `& x ) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1179).

Reimplemented in **MHAParser::mcomplex_t** (p. 1226), **MHAParser::mfloat_t** (p. 1231), **MHAParser::mint_t** (p. 1243), **MHAParser::vcomplex_t** (p. 1269), **MHAParser::vfloat_t** (p. 1274), **MHAParser::vint_t** (p. 1279), **MHAParser::complex_t** (p. 1201), **MHAParser::float_t** (p. 1208), **MHAParser::int_t** (p. 1214), **MHAParser::bool_t** (p. 1191), **MHAParser::vstring_t** (p. 1283), **MHAParser::string_t** (p. 1262), and **MHAParser::kw_t** (p. 1221).

5.343.3.2 query_perm() `std::string MHAParser::variable_t::query_perm (`

`const std::string & ) [virtual]`

Reimplemented from **MHAParser::monitor_t** (p. 1246).

5.343.3.3 setlock() `void MHAParser::variable_t::setlock (`

`const bool & b )`

Lock a variable against write access.

Parameters

b	Lock state

5.343.4 Member Data Documentation

5.343.4.1 locked `bool MHAParser::variable_t::locked [private]`

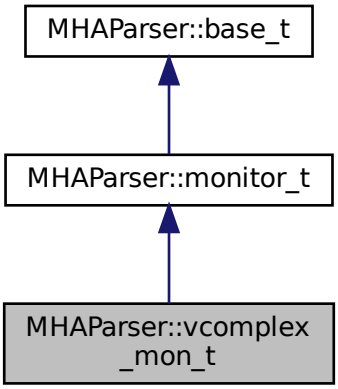
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.344 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

5.344.1 Detailed Description

Monitor with vector of complex values.

5.344.2 Constructor & Destructor Documentation

5.344.2.1 vcomplex_mon_t() `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.

5.344.3 Member Function Documentation

5.344.3.1 query_val() std::string MHAParser::vcomplex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

5.344.3.2 query_type() std::string MHAParser::vcomplex_mon_t::query_type (const std::string &) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1180).

5.344.4 Member Data Documentation

5.344.4.1 data std::vector< **mha_complex_t**> MHAParser::vcomplex_mon_t::data

Data field.

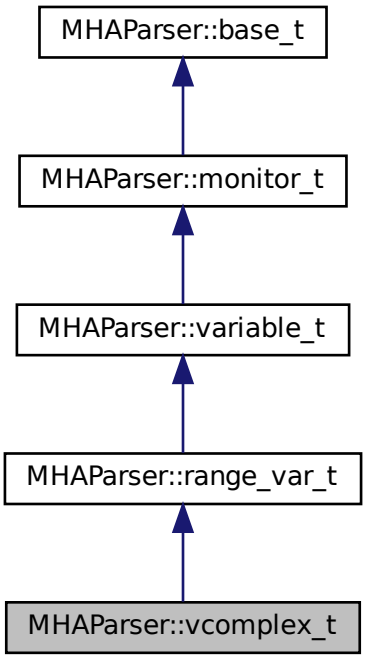
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.345 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

5.345.1 Detailed Description

Vector variable with complex value.

5.345.2 Constructor & Destructor Documentation

5.345.2.1 vcomplex_t() MHAParser::vcomplex_t::vcomplex_t (

```
const std::string & h,
const std::string & v,
const std::string & rg = "" )
```

5.345.3 Member Function Documentation

5.345.3.1 op_setval() std::string MHAParser::vcomplex_t::op_setval (

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.345.3.2 query_type() std::string MHAParser::vcomplex_t::query_type (

```
const std::string & ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.345.3.3 query_val() std::string MHAParser::vcomplex_t::query_val (

```
const std::string & s ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.345.4 Member Data Documentation

5.345.4.1 data `std::vector< mha_complex_t> MHAParser::vcomplex_t::data`

Data field.

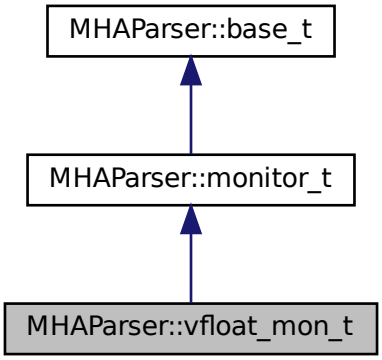
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.346 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

5.346.1 Detailed Description

Vector of floats monitor.

5.346.2 Constructor & Destructor Documentation

5.346.2.1 vfloat_mon_t() `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.

5.346.3 Member Function Documentation

5.346.3.1 query_val() `std::string MHAParser::vfloat_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.346.3.2 query_type() `std::string MHAParser::vfloat_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.346.4 Member Data Documentation

5.346.4.1 data `std::vector<float> MHAParser::vfloat_mon_t::data`

Data field.

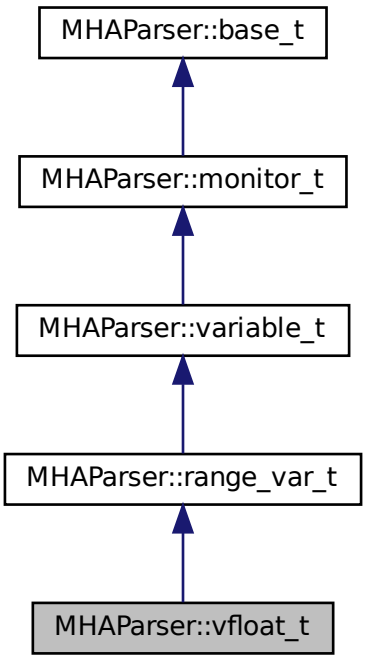
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.347 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

5.347.1 Detailed Description

Vector variable with float value.

5.347.2 Constructor & Destructor Documentation

5.347.2.1 vfloat_t() `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.

Parameters

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.

-

5.347.3 Member Function Documentation

5.347.3.1 op_setval() `std::string MHAParser::vfloat_t::op_setval ( expression_t & x ) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.347.3.2 query_type() `std::string MHAParser::vfloat_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.347.3.3 query_val() `std::string MHAParser::vfloat_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.347.4 Member Data Documentation

5.347.4.1 data `std::vector<float> MHAParser::vfloat_t::data`

Data field.

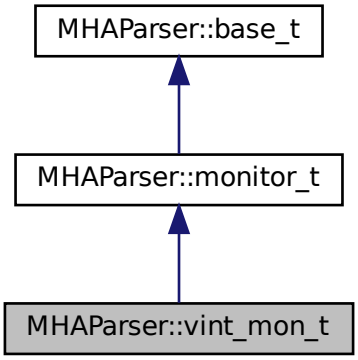
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.348 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

5.348.1 Detailed Description

Vector of ints monitor.

5.348.2 Constructor & Destructor Documentation

5.348.2.1 vint_mon_t() `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.

5.348.3 Member Function Documentation

5.348.3.1 query_val() `std::string MHAParser::vint_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.348.3.2 query_type() `std::string MHAParser::vint_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.348.4 Member Data Documentation

5.348.4.1 data `std::vector<int> MHAParser::vint_mon_t::data`

Data field.

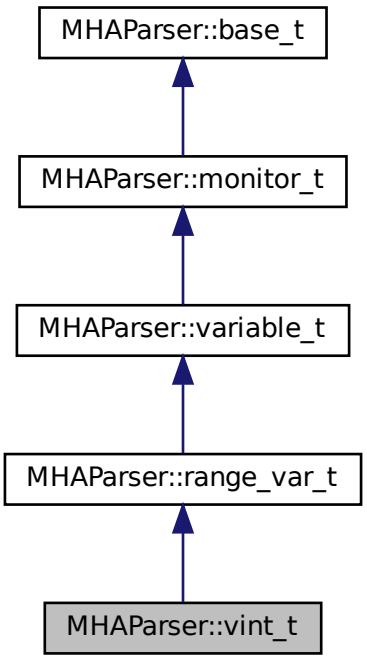
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.349 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

5.349.1 Detailed Description

Variable with vector<int> value.

5.349.2 Constructor & Destructor Documentation

5.349.2.1 vint_t() `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

Parameters

rg	optional: range constraint for all elements

5.349.3 Member Function Documentation

5.349.3.1 op_setval() `std::string MHAParser::vint_t::op_setval ( expression_t & x ) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.349.3.2 query_type() `std::string MHAParser::vint_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.349.3.3 query_val() `std::string MHAParser::vint_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.349.4 Member Data Documentation

5.349.4.1 data `std::vector<int> MHAParser::vint_t::data`

Data field.

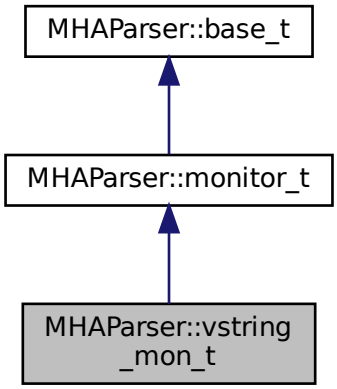
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.350 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

5.350.1 Detailed Description

Vector of monitors with string value.

5.350.2 Constructor & Destructor Documentation

5.350.2.1 vstring_mon_t() `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.

5.350.3 Member Function Documentation

5.350.3.1 query_val() `std::string MHAParser::vstring_mon_t::query_val ( const std::string & s ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.350.3.2 query_type() `std::string MHAParser::vstring_mon_t::query_type ( const std::string & ) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1180).

5.350.4 Member Data Documentation

5.350.4.1 data `std::vector<std::string> MHAParser::vstring_mon_t::data`

Data field.

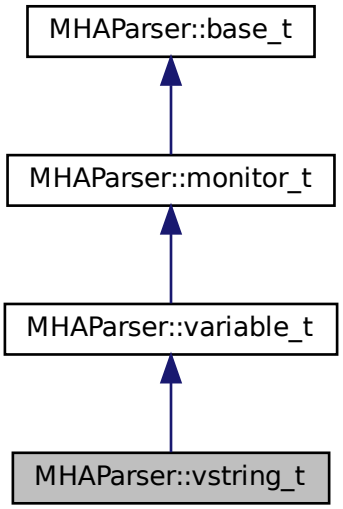
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.351 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

5.351.1 Detailed Description

Vector variable with string values.

5.351.2 Constructor & Destructor Documentation

5.351.2.1 vstring_t() `MHAParser::vstring_t::vstring_t (`

```
const std::string & h,
const std::string & v )
```

5.351.3 Member Function Documentation

5.351.3.1 op_setval() `std::string MHAParser::vstring_t::op_setval (`

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1264).

5.351.3.2 query_type() `std::string MHAParser::vstring_t::query_type (`

```
const std::string & ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.351.3.3 query_val() `std::string MHAParser::vstring_t::query_val (`

```
const std::string & s ) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1180).

5.351.4 Member Data Documentation

5.351.4.1 data `std::vector<std::string> MHAParser::vstring_t::data`

Data field.

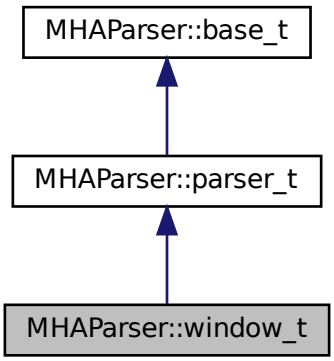
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.352 MHAParser::window_t Class Reference

MHA configuration interface for a window function generator.

Inheritance diagram for MHAParser::window_t:



Public Types

- enum **wtype_t** {
  wnd_rect =0 , **wnd_hann** =1 , **wnd_hamming** =2 , **wnd_blackman** =3 ,
  wnd_bartlett =4 , **wnd_user** =5 }

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.
- void **setlock** (bool b)

Private Attributes

- **MHAParser::kw_t wtype**
- **MHAParser::vfloat_t user**

Additional Inherited Members

5.352.1 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 **MHAWindow::base_t** (p. 1444) based on the values provided by the configuration interface.

The configuration interface is derived from **MHAParser::parser_t** (p. 1246) and can thus be inserted into the configuration tree using the **insert_item()** (p. 1249) method of the parent parser.

If one of the pre-defined window types is used, then the window is generated using the **MHAWindow::fun_t** (p. 1448) class constructor; for the user-defined type the values from the "user" variable are copied.

5.352.2 Member Enumeration Documentation

5.352.2.1 wtype_t enum MHAParser::window_t::wtype_t

Enumerator

wnd_rect	
wnd_hann	
wnd_hamming	
wnd_blackman	
wnd_bartlett	
wnd_user	

5.352.3 Constructor & Destructor Documentation

5.352.3.1 window_t() MHAParser::window_t::window_t (const std::string & *help* = *"Window type configuration."*)

Constructor to create parser class.

5.352.4 Member Function Documentation

5.352.4.1 get_window() [1/5] MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len*) const

Create a window instance, use default parameters.

5.352.4.2 get_window() [2/5]

```
MHAWindow::base_t MHAParser::window_t::get_window (
    unsigned int len,
    float xmin ) const
```

Create a window instance.

5.352.4.3 get_window() [3/5]

```
MHAWindow::base_t MHAParser::window_t::get_window (
    unsigned int len,
    float xmin,
    float xmax ) const
```

Create a window instance.

5.352.4.4 get_window() [4/5]

```
MHAWindow::base_t MHAParser::window_t::get_window (
    unsigned int len,
    float xmin,
    float xmax,
    bool minincluded ) const
```

Create a window instance.

5.352.4.5 get_window() [5/5]

```
MHAWindow::base_t MHAParser::window_t::get_window (
    unsigned int len,
    float xmin,
    float xmax,
    bool minincluded,
    bool maxincluded ) const
```

Create a window instance.

5.352.4.6 get_type()

```
MHAParser::window_t::wtype_t MHAParser::window_t::get_type (
) const
```

Return currently selected window type.

5.352.4.7 setlock() `void MHAParser::window_t::setlock ( bool b ) [inline]`

5.352.5 Member Data Documentation

5.352.5.1 wtype `MHAParser::kw_t MHAParser::window_t::wtype [private]`

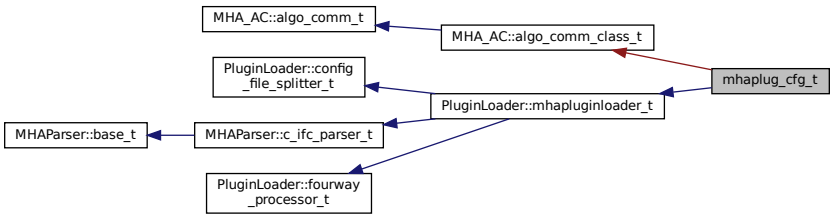
5.352.5.2 user `MHAParser::vfloat_t MHAParser::window_t::user [private]`

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.353 mhaplug_cfg_t Class Reference

Inheritance diagram for mhaplug_cfg_t:



Public Member Functions

- **mhaplug_cfg_t** (**MHA_AC::algo_comm_t** &iac, const std::string & **libname**, bool use↩ _own_ac)
- ∼**mhaplug_cfg_t** () throw ()
- void **prepare** (**mhaconfig_t** &) override
- void **release** () override

Additional Inherited Members

5.353.1 Constructor & Destructor Documentation

5.353.1.1 mhaplug_cfg_t() `mhaplug_cfg_t::mhaplug_cfg_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & libname,
bool use_own_ac )
```

5.353.1.2 ~mhaplug_cfg_t() `mhaplug_cfg_t::~mhaplug_cfg_t ( ) throw ( )  [inline]`

5.353.2 Member Function Documentation

5.353.2.1 prepare() `void mhaplug_cfg_t::prepare (`

```
  mhaconfig_t & signal_dimensions )  [override], [virtual]
```

Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1527).

5.353.2.2 release() `void mhaplug_cfg_t::release (`

```
void  )  [override], [virtual]
```

Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1527).

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

- **altplugs.cpp**

5.354 MHAPlugin::cfg_node_t< runtime_cfg_t > Class Template Reference

A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.

Public Member Functions

- **cfg_node_t** (runtime_cfg_t *runtime_cfg)
  Constructor for a singly linked list node.
- ~**cfg_node_t** ()
  Destructor of the singly linked list node.

Public Attributes

- std::atomic< **cfg_node_t**< runtime_cfg_t > * > **next**
  A pointer to the next node in the singly linked list.
- std::atomic< bool > **not_in_use**
  Initially this data member is set to false by the constructor.
- runtime_cfg_t * **data**
  A native pointer to the runtime configuration managed by this node.

5.354.1 Detailed Description

template<class runtime_cfg_t>
class MHAPlugin::cfg_node_t< runtime_cfg_t >

A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.

The singly linked list is designed for a single producer thread and a single consumer thread, where the producer is also responsible for destroying objects when they are no longer needed because the consumer is the signal processing thread that cannot afford memory allocation or deallocation operations.

5.354.2 Constructor & Destructor Documentation

5.354.2.1 cfg_node_t() `template<class runtime_cfg_t >`

```
MHAPlugin::cfg_node_t< runtime_cfg_t >::  cfg_node_t (
            runtime_cfg_t * runtime_cfg ) [explicit]
```

Constructor for a singly linked list node.

Parameters

runtime_cfg	Pointer to a runtime configuration object that was jus created on the heap. The newly constructed **cfg_node_t** (p. 1289) object takes over object ownership of the pointed-to runtime configuration and will call delete on it in its destructor.

5.354.2.2 ~cfg_node_t() template<class runtime_cfg_t > MHAPlugin::cfg_node_t< runtime_cfg_t >::~ cfg_node_t

Destructor of the singly linked list node.

Will also delete the pointed-to data object (the runtime configuration)

5.354.3 Member Data Documentation

5.354.3.1 next template<class runtime_cfg_t > std::atomic< cfg_node_t<runtime_cfg_t>*> MHAPlugin::cfg_node_t< runtime_cfg_t >::next

A pointer to the next node in the singly linked list.

On construction, this will be a NULL pointer. New objects can be appended to the singly linked list by writing the address to the next node into this data member. Since this pointer is std::atomic, writing to it is a release operation which means all threads seeing the new value of the next pointer will also see all other writes to memory that the thread doing this write has performed, which includes anything the constructor of the new runtime config has written and the assignment of the address of the new runtime config to the data pointer of the next node. This prevents making half-constructed runtime configuration objects visible to the consumer thread.

5.354.3.2 not_in_use template<class runtime_cfg_t > std::atomic<bool> MHAPlugin::cfg_node_t< runtime_cfg_t >::not_in_use

Initially this data member is set to false by the constructor.

It is set to true by the consumer thread when it no longer needs the run time configuration pointed to by this node's data member. This bool is atomic because this node and the runtime object it points to can be deleted by the configuration thread as soon as value true is stored in this bool, which happens right after the processing thread acquires a pointer to the next node in the singly linked list. The atomic ensures all threads agree on this "right after" ordering.

5.354.3.3 data `template<class runtime_cfg_t >`

`runtime_cfg_t* MHAPlugin::cfg_node_t< runtime_cfg_t >::data`

A native pointer to the runtime configuration managed by this node.

The runtime configuration lives on the heap and is owned by this node. It is deleted in this node's destructor. The runtime configuration object must be created by client code with operator new before ownership is transferred to this node by passing a pointer to it as the constructor's parameter. This pointer does not need to be atomic, memory access ordering is ensured by the atomic next pointer and placing accesses to "data" in the correct places relative to accesses to "next".

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

- **mha_plugin.hh**

5.355 MHAPlugin::config_t< runtime_cfg_t > Class Template Reference

Template class for thread safe configuration.

Inherited by **MHAPlugin::plugin_t< matlab_wrapper_rt_cfg_t >**, **MHAPlugin::plugin_t< fftfb_plug_t >**, **MHAPlugin::plugin_t< fftfbpow_t >**, **MHAPlugin::plugin_t< spec_fader_t >**, **MHAPlugin::plugin_t< doasvm_feature_extraction_config >**, **MHAPlugin::plugin_t< wave2spec_t >**, **MHAPlugin::plugin_t< acwriter_t >**, **MHAPlugin::plugin_t< lpc_config >**, **MHAPlugin::plugin_t< MHASignal::async_rmslevel_t >**, **MHAPlugin::plugin_t< adaptive_feedback_canceller_config >**, **MHAPlugin::plugin_t< steerbf_config >**, **MHAPlugin::plugin_t< doasvm_classification_config >**, **MHAPlugin::plugin_t< MHAWindow::fun_t >**, **MHAPlugin::plugin_t< acPooling_wave_config >**, **MHAPlugin::plugin_t< Ci_auralization_cis_cfg >**, **MHAPlugin::plugin_t< sine_cfg_t >**, **MHAPlugin::plugin_t< MHA_AC::spectrum_t >**, **MHAPlugin::plugin_t< rt_nlms_t >**, **MHAPlugin::plugin_t< level_matching_config_t >**, **MHAPlugin::plugin_t< scaler_t >**, **MHAPlugin::plugin_t< hilbert_shifter_t >**, **MHAPlugin::plugin_t< MHASignal::waveform_t >**, **MHAPlugin::plugin_t< delaysum_wave_t >**, **MHAPlugin::plugin_t< combc_t >**, **MHAPlugin::plugin_t< Ci_simulation_cis_cfg >**, **MHAPlugin::plugin_t< overlapadd_t >**, **MHAPlugin::plugin_t< fshift_config_t >**, **MHAPlugin::plugin_t< db_t >**, **MHAPlugin::plugin_t< fftfilter_t >**, **MHAPlugin::plugin_t< dc_t >**, **MHAPlugin::plugin_t< int >**, **MHAPlugin::plugin_t< cfg_t >**, **MHAPlugin::plugin_t< Ci_auralization_ace_cfg >**, **MHAPlugin::plugin_t< smooth_cepstrum_t >**, **MHAPlugin::plugin_t< UNIT >**, **MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >**, **MHAPlugin::plugin_t< route::process_t >**, **MHAPlugin::plugin_t< lpc_bl_predictor_config >**, **MHAPlugin::plugin_t< dbasync_t >**, **MHAPlugin::plugin_t< cpuload_cfg_t >**, **MHAPlugin::plugin_t< noise_psd_estimator_t >**, **MHAPlugin::plugin_t< gtfb_simple_rt_t >**, **MHAPlugin::plugin_t< gtfb_simd_cfg_t >**, **MHAPlugin::plugin_t< ac2wave_t >**, **MHAPlugin::plugin_t< Ci_simulation_ace_cfg >**, **MHAPlugin::plugin_t< MHA_AC::waveform_t >**, **MHAPlugin::plugin_t< mhachain::plugs_t >**, **MHAPlugin::plugin_t< gsc_adaptive_stage >**, **MHAPlugin::plugin_t< char >**, **MHAPlugin::plugin_t< smoothspec_wrap_t >**, **MHAPlugin::plugin_t< prediction_error_config**

>, **MHAPlugin::plugin_t< lpc_burglattice_config >**, **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >**, **MHAPlugin::plugin_t< adm_rtconfig_t >**, **MHAPlugin::plugin_t< example5_t >**, **MHAPlugin::plugin_t< cohflt_t >**, **MHAPlugin::plugin_t< acConcat_wave_config >**, **MHAPlugin::plugin_t< ac2xdf_rt_t >**, **MHAPlugin::plugin_t< Set_rms_cfg >**, **MHAPlugin::plugin_t< wavwriter_t >**, **MHAPlugin::plugin_t< spec2wave_t >**, **MHAPlugin::plugin_t< rohConfig >**, **MHAPlugin::plugin_t< delaysum_t >**, **MHAPlugin::plugin_t< MHASignal::delay_t >**, **MHAPlugin::plugin_t< acSteer_config >**, **MHAPlugin::plugin_t< resampling_t >**, **MHAPlugin::plugin_t< trigger2lsl_rt_t >**, **MHAPlugin::plugin_t< plingploing_t >**, **MHAPlugin::plugin_t< analysepath_t >**, **MHAPlugin::plugin_t< acTransform_wave_config >**, **MHAPlugin::plugin_t< Get_rms_cfg >**, **MHAPlugin::plugin_t< runtime_cfg_t >**, **addsndfile::addsndfile_if_t**[private], **audiometerbackend::audiometer_if_t**[private], **calibrator_t**[private], and **dc_simple::dc_if_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 * **peek_config** () const
  Receive the latest run time configuration without changing the configuration pointer.
- void **push_config** (runtime_cfg_t *ncfg)
  Push a new run time configuration into the configuration fifo.
- void **cleanup_unused_cfg** ()
  *To be called by the **push_config()*** (p. 1296) *for housekeeping.*
- void **remove_all_cfg** ()
  To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in_use flag.

Protected Attributes

- runtime_cfg_t * **cfg**
  Pointer to the runtime configuration currently used by the signal processing thread.

Private Attributes

- std::atomic< **MHAPlugin::cfg_node_t**< runtime_cfg_t > * > **cfg_root**
  Start of a singly linked list of runtime configuration objects.
- **MHAPlugin::cfg_node_t**< runtime_cfg_t > * **cfg_node_current**
  Pointer to the currently used plugin runtime configurations.

5.355.1 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. 125) 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. 1296). In the processing thread, the actual configuration can be received by a call of **poll_config()** (p. 1295).

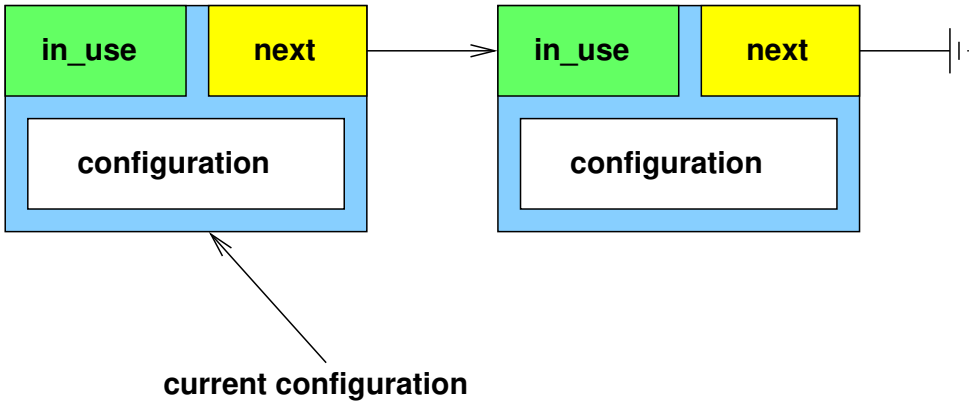


Figure 5 Schematic drawing of runtime configuration update: configuration updated, but not used yet.

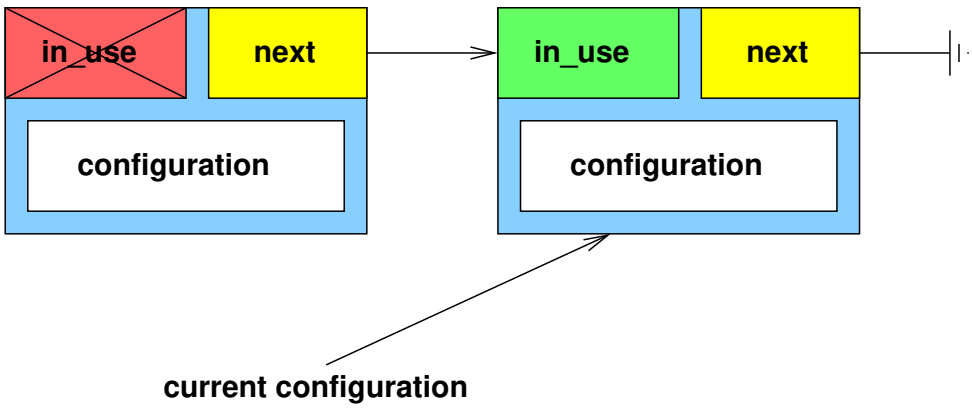


Figure 6 Schematic drawing of runtime configuration update: configuration in use.

To ensure lock-free thread safety, we use C++ atomics and rely on the C++ memory model. We only use store-release and load-acquire operations by using C++ atomics with the default memory ordering. The semantics of these are:

The store-release operation atomically writes to an atomic variable, while the load-acquire operation atomically reads from an atomic variable.

The C++ memory model guarantees that all previous writes to memory performed by the thread doing the store-release are visible to other threads when they see the new value in the shared atomic variable when that value is read by the other thread with a load-acquire operation.

An important precondition of this synchronization scheme is that there is only ever one audio thread and one configuration thread per plugin, i.e. there is only one thread doing the push_↩ config and one thread doing the poll_config for each instance of **config_t** (p. 1292).

For more details on atomics, refer to the C++11 or later documentation, or to these conference talks by Sutter:

- Atomic Weapons, 2012
- Lock-Free Programming, 2014

5.355.2 Constructor & Destructor Documentation

5.355.2.1 config_t() `template<class runtime_cfg_t >` **MHAPlugin::config_t**< runtime_cfg_t >:: **config_t**

5.355.2.2 ~config_t() `template<class runtime_cfg_t >` **MHAPlugin::config_t**< runtime_cfg_t >::~ **config_t**

5.355.3 Member Function Documentation

5.355.3.1 poll_config() 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 only called from the *processing* thread.

Should be called at the start of each process() callback to get the latest runtime configuration.

When this function finds newer run time configurations, it returns the newest and ensures the older run time configurations have their not_in_use flag set to true.

Returns

Pointer to the latest runtime configuration object (same pointer as stored by this function in data member 'cfg').

Exceptions

MHA_Error *(p. 906)*	if the resulting runtime configuration is NULL. This usually means that no push_config has occured.

5.355.3.2 peek_config() template<class runtime_cfg_t >

runtime_cfg_t * **MHAPlugin::config_t**< runtime_cfg_t >::peek_config [protected]

Receive the latest run time configuration without changing the configuration pointer.

This function retrieves the latest run time configuration. Returns a pointer to the latest runtime configuration without updating the data member cfg. For use in the configuration thread when creation of a new runtime configuration object needs access to the previously created runtime configuration object. Should normally not be used because it introduces synchronization requirements between configuration thread and signal processing thread.

5.355.3.3 push_config() 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.

Should be called only by the configuration thread when a new runtime configuration object has been constructed in response to configuration changes, or during execution of the prepare() method to ensure that there is a valid runtime configuration for the signal processing which can start after prepare() returns.

For housekeeping, this method will also delete any runtime configuration objects that have previously been passed to **push_config()** (p. 1296) if they are no longer needed.

Parameters

ncfg	A pointer to the new runtime configuration object. This object must have been created on the heap by the configuration thread with operator new. By passing the pointer to this method, client code gives up ownership. The object will be deleted in a future invocation of push_config, or on destruction of this **config_t** (p. 1292) instance.

5.355.3.4 cleanup_unused_cfg() template<class runtime_cfg_t >

```
void MHAPlugin::config_t< runtime_cfg_t >::cleanup_unused_cfg [protected]
```

To be called by the **push_config()** (p. 1296) for housekeeping.

Will delete any no longer used runtime configuration objects.

5.355.3.5 remove_all_cfg() template<class runtime_cfg_t >

```
void MHAPlugin::config_t< runtime_cfg_t >::remove_all_cfg [protected]
```

To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in_use flag.

5.355.4 Member Data Documentation

5.355.4.1 cfg `template<class runtime_cfg_t >`

`runtime_cfg_t* MHAPlugin::config_t< runtime_cfg_t >::cfg [protected]`

Pointer to the runtime configuration currently used by the signal processing thread.

Should be used to access the current runtime configuration during signal processing. This pointer is updated as a side effect of calling **poll_config()** (p. 1295) on this object.

5.355.4.2 cfg_root `template<class runtime_cfg_t >`

`std::atomic< MHAPlugin::cfg_node_t<runtime_cfg_t> *> MHAPlugin::config_t< runtime_cfg_t >::cfg_root [private]`

Start of a singly linked list of runtime configuration objects.

cfg_root points to the oldest still existing node of that list. After object creation this pointer is updated by the configuration thread and then read by the signal processing thread. To ensure proper order of memory accesses for this transfer between threads, it needs to be atomic, this ensures that the start of the singly linked list of runtime configurations will be properly visible to the signal processing on startup.

5.355.4.3 cfg_node_current `template<class runtime_cfg_t >`

`MHAPlugin::cfg_node_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_node_current [private]`

Pointer to the currently used plugin runtime configurations.

Used as a hint for poll_config where to start looking for the newest node. This optimization allows poll_config to scale better with the number of nodes not yet cleaned up by push_config. Does not need to be atomic because it is only used within the signal processing thread.

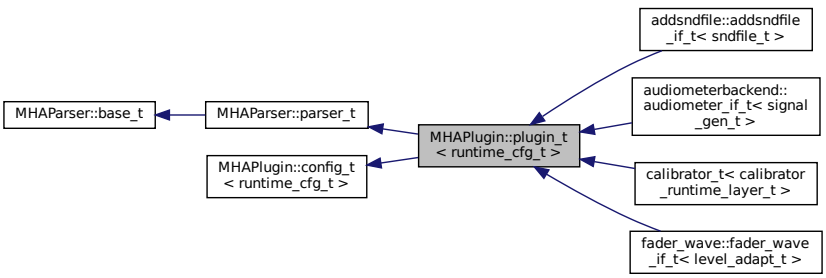
The documentation for this class was generated from the following file:

- **mha_plugin.hh**

5.356 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 &, **MHA_AC::algo_comm_t** &)
  Constructor of plugin template base class.
- virtual ~**plugin_t** ()
  Destructor of plugin template base class.
- 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.
- **MHA_AC::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

5.356.1 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.

This template class provides thread safe configuration handling and standard methods to be compatible to the C++ openMHA plugin wrapper macro **MHAPLUGIN_CALLBACKS** (p. 1835).

The template parameter runtime_cfg_t should be the runtime configuration of the plugin.

See **MHAPlugin::config_t** (p. 1292) for details on the thread safe communication update mechanism.

5.356.2 Constructor & Destructor Documentation

5.356.2.1 plugin_t()

```
template<class runtime_cfg_t >
MHAPlugin::plugin_t< runtime_cfg_t >::  plugin_t (
            const std::string & help,
             MHA_AC::algo_comm_t & iac )
```

Constructor of plugin template base class.

Plugin classes inherit from MHAPlugin::plugin_t<> and will call this base class constructor in their constructor's initialization list.

The constructor of a plugin is called each time a plugin is loaded into MHA: Multiple instances of the same plugin can be loaded and for each plugin instance, a new instance of the plugin class will be created.

Parameters

help	Short help text that provides some general information about the plugin. This text is returned in response to configuration language query command "?help".
iac	AC space handle (will be stored into the member variable ac).

5.356.2.2 ~plugin_t() template<class runtime_cfg_t > MHAPlugin::plugin_t< runtime_cfg_t >::~ plugin_t [virtual]

Destructor of plugin template base class.

Plugin class instances are deleted and the destructor of the instance is called when unloading a plugin instance from the MHA. Typically this happens just before the MHA process terminates, e.g. in response to cmd=quit, but some plugins also have the capability of replacing loaded plugins during runtime of the MHA, in which case the destructor of the active instances is called before the unloading.

In some exceptional cases, e.g. when an error occurs during initializing the MHA, the MHA process may terminate without ever calling the destructors of all existing plugin instances.

5.356.3 Member Function Documentation

5.356.3.1 prepare() template<class runtime_cfg_t > virtual void MHAPlugin::plugin_t< runtime_cfg_t >::prepare (mhaconfig_t &) [pure virtual]

Implemented in **calibrator_t** (p. 367), **save_wave_t** (p. 1598), **save_spec_t** (p. 1596), **fftfbpow::fftfbpow_interface_t** (p. 580), **dc_simple::dc_if_t** (p. 474), **dc::dc_if_t** (p. 454), **ac2wave::ac2wave_if_t** (p. 191), **wave2spec_if_t** (p. 1691), **windnoise::if_t** (p. 1712), **attenuate20_t** (p. 337), **Set_rms** (p. 1601), **matlab_wrapper::matlab_wrapper_t** (p. 821), **Get_rms** (p. 633), **gcfsnet_mono_t** (p. 629), **gcfsnet_bin_t** (p. 624), **example4_t** (p. 564), **example3_t** (p. 560), **example2_t** (p. 556), **example1_t** (p. 554), **droptect_t** (p. 526), **Ci_simulation_cis** (p. 412), **Ci_simulation_ace** (p. 402), **Ci_auralization_cis** (p. 390), **Ci_auralization_ace** (p. 377), **bmfwf_t** (p. 360), **rmslevel::rmslevel_if_t** (p. 1559), **mconv::MConv** (p. 839), **gtfb_simple_t** (p. 675), **wavrec_t** (p. 1701), **trigger2lsl::trigger2lsl_if_t** (p. 1675), **plingploing::if_t** (p. 1496), **altconfig_t** (p. 320), **plugins::hoertech::acrec::acrec_t** (p. 1535), **wave2lsl::wave2lsl_t** (p. 1687), **us_t** (p. 1682), **bbcalib_interface_t** (p. 357), **testplugin::if_t** (p. 1669), **steerbf** (p. 1658), **spec2wave_if_t** (p. 1649), **softclip_t** (p. 1641), **smoothgains_bridge::overlapadd_if_t** (p. 1636), **smooth_cepstrum::smooth_cepstrum_if_t** (p. 1621), **sine_t** (p. 1617), **shadowfilter_end::shadowfilter_end_t** (p. 1613), **shadowfilter_begin::shadowfilter_begin_t** (p. 1609), **route::interface_t** (p. 1587), **rohBeam::rohBeam** (p. 1574), **MHAPlugin_Resampling::resampling_if_t** (p. 1306), **prediction_error** (p. 1546), **overlapadd::overlapadd_if_t** (p. 1487), **osc2ac_t** (p. 1476), **noise_psd_estimator::noise_psd_estimator_if_t** (p. 1467), **noise_t** (p. 1473), **nlms_t** (p. 1463), **multibandcompressor::interface_t** (p. 1458), **mhachain::chain_base_t** (p. 989), **matrixmixer::matmix_t** (p. 836), **lsl2ac::lsl2ac_t** (p. 793), **lpc_burglattice** (p. 782), **lpc_bl_predictor** (p. 776), **lpc** (p. 772), **levelmeter_t** (p. 769), **level_matching::level_matching_t** (p. 766), **identity_t** (p. 681), **gtfb_simd_t** (p. 665), **gtfb_analyzer::gtfb_analyzer_t** (p. 657), **gsc_adaptive_stage::gsc_adaptive_stage_if** (p. 649), **gain::gain_if_t** (p. 620), **fshift_hilbert::frequency_translator_t** (p. 602), **fshift::fshift_t**

(p. 598), **fftfilterbank::fftfb_interface_t** (p. 590), **fftfilter::interface_t** (p. 588), **fader_↩ wave::fader_wave_if_t** (p. 575), **fader_if_t** (p. 573), **example7_t** (p. 571), **example6↩ _t** (p. 568), **plugin_interface_t** (p. 1510), **equalize::freqgains_t** (p. 551), **dropgen_↩ t** (p. 523), **ds_t** (p. 530), **doasvm_feature_extraction** (p. 513), **doasvm_classification** (p. 507), **delaysum::delaysum_wave_if_t** (p. 496), **delaysum_spec::delaysum_spec↩ _if_t** (p. 501), **delay::interface_t** (p. 494), **dbasync_native::db_if_t** (p. 446), **db_if_t** (p. 442), **cpuload::cpuload_if_t** (p. 440), **complex_scale_channel_t** (p. 435), **combc↩ _if_t** (p. 430), **coherence::cohflt_if_t** (p. 421), **audiometerbackend::audiometer_if_↩ t** (p. 339), **analysispath_if_t** (p. 334), **altplugs_t** (p. 324), **adm_if_t** (p. 302), **addsndfile↩ ::addsndfile_if_t** (p. 281), **adaptive_feedback_canceller** (p. 268), **acTransform_wave** (p. 262), **acSteer** (p. 256), **acsave::acsave_t** (p. 246), **acPooling_wave** (p. 239), **acmon↩ ::acmon_t** (p. 234), **acConcat_wave** (p. 225), **ac_proc::interface_t** (p. 221), **ac2xdf↩ ::ac2xdf_if_t** (p. 199), **ac2osc_t** (p. 186), and **ac2lsl::ac2lsl_t** (p. 169).

5.356.3.2 release() `template<class runtime_cfg_t >`

`void` **`MHAPlugin::plugin_t`**`< runtime_cfg_t >::release [virtual]`

Reimplemented in **windnoise::if_t** (p. 1713), **attenuate20_t** (p. 337), **steerbf** (p. 1659), **smooth_cepstrum::smooth_cepstrum_if_t** (p. 1621), **rohBeam::rohBeam** (p. 1574), **prediction_error** (p. 1546), **lpc_burglattice** (p. 782), **lpc_bl_predictor** (p. 776), **lpc** (p. 773), **level_matching::level_matching_t** (p. 766), **gsc_adaptive_stage::gsc_↩ adaptive_stage_if** (p. 649), **gcfsnet_mono_t** (p. 629), **gcfsnet_bin_t** (p. 624), **fshift↩ ::fshift_t** (p. 599), **example7_t** (p. 571), **example4_t** (p. 564), **example3_t** (p. 560), **example2_t** (p. 556), **example1_t** (p. 553), **droptect_t** (p. 527), **doasvm_feature↩ _extraction** (p. 513), **doasvm_classification** (p. 508), **bmfwf_t** (p. 361), **ac↩ Transform_wave** (p. 262), **acSteer** (p. 257), **acPooling_wave** (p. 239), **acConcat_↩ wave** (p. 225), **rmslevel::rmslevel_if_t** (p. 1559), **wavrec_t** (p. 1701), **wave2spec_if_t** (p. 1691), **wave2lsl::wave2lsl_t** (p. 1687), **us_t** (p. 1682), **trigger2lsl::trigger2lsl_↩ if_t** (p. 1675), **bbcalib_interface_t** (p. 358), **calibrator_t** (p. 367), **spec2wave_if_t** (p. 1649), **smoothgains_bridge::overlapadd_if_t** (p. 1636), **sine_t** (p. 1617), **Set_↩ rms** (p. 1602), **route::interface_t** (p. 1587), **MHAPlugin_Resampling::resampling_if_t** (p. 1306), **overlapadd::overlapadd_if_t** (p. 1487), **osc2ac_t** (p. 1476), **nlms_t** (p. 1463), **multibandcompressor::interface_t** (p. 1458), **mhachain::chain_base_t** (p. 989), **mconv↩ ::MConv** (p. 839), **matlab_wrapper::matlab_wrapper_t** (p. 821), **lsl2ac::lsl2ac_t** (p. 793), **identity_t** (p. 681), **gtfb_simple_t** (p. 675), **gtfb_analyzer::gtfb_analyzer_t** (p. 657), **Get↩ _rms** (p. 634), **gain::gain_if_t** (p. 620), **fshift_hilbert::frequency_translator_t** (p. 602), **fftfilterbank::fftfb_interface_t** (p. 590), **fader_wave::fader_wave_if_t** (p. 575), **dropgen↩ _t** (p. 523), **ds_t** (p. 530), **delaysum::delaysum_wave_if_t** (p. 497), **dc_simple::dc_↩ if_t** (p. 474), **dc::dc_if_t** (p. 454), **dbasync_native::db_if_t** (p. 447), **db_if_t** (p. 442), **coherence::cohflt_if_t** (p. 421), **Ci_simulation_cis** (p. 412), **Ci_simulation_ace** (p. 402), **Ci_auralization_cis** (p. 390), **Ci_auralization_ace** (p. 378), **analysispath_if_t** (p. 334), **altplugs_t** (p. 324), **altconfig_t** (p. 320), **adm_if_t** (p. 302), **addsndfile::addsndfile_if_t** (p. 281), **adaptive_feedback_canceller** (p. 268), **acsave::acsave_t** (p. 246), **plugins↩ ::hoertech::acrec::acrec_t** (p. 1536), **acmon::acmon_t** (p. 234), **ac_proc::interface_t** (p. 221), **ac2xdf::ac2xdf_if_t** (p. 200), **ac2wave::ac2wave_if_t** (p. 192), **ac2osc_t** (p. 186), and **ac2lsl::ac2lsl_t** (p. 169).

5.356.3.3 prepare_() template<class runtime_cfg_t >

void **MHAPlugin::plugin_t**< runtime_cfg_t >::prepare_ (**mhaconfig_t** & *cf*)

5.356.3.4 release_() template<class runtime_cfg_t >

void **MHAPlugin::plugin_t**< runtime_cfg_t >::release_

5.356.3.5 is_prepared() 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)

5.356.3.6 input_cfg() template<class runtime_cfg_t >

mhaconfig_t **MHAPlugin::plugin_t**< runtime_cfg_t >::input_cfg ( ) const [inline]

Current input channel configuration.

5.356.3.7 output_cfg() template<class runtime_cfg_t >

mhaconfig_t **MHAPlugin::plugin_t**< runtime_cfg_t >::output_cfg ( ) const [inline]

Current output channel configuration.

5.356.4 Member Data Documentation

5.356.4.1 tftype 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. 996) are stored in this member in the **prepare()** (p. 1301) method.

Note

This member is likely to be removed in later versions, use **input_cfg()** (p. 1303) and **output_cfg()** (p. 1303) instead.

5.356.4.2 ac template<class runtime_cfg_t >

MHA_AC::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.

5.356.4.3 is_prepared_ template<class runtime_cfg_t >

bool **MHAPlugin::plugin_t**< runtime_cfg_t >::is_prepared_ [private]

5.356.4.4 input_cfg_ template<class runtime_cfg_t >

mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg_ [private]

5.356.4.5 output_cfg_ template<class runtime_cfg_t >

mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg_ [private]

5.356.4.6 mhaconfig_in template<class runtime_cfg_t >

MHAParser::mhaconfig_mon_t MHAPlugin::plugin_t< runtime_cfg_t >::mhaconfig_in [private]

5.356.4.7 mhaconfig_out

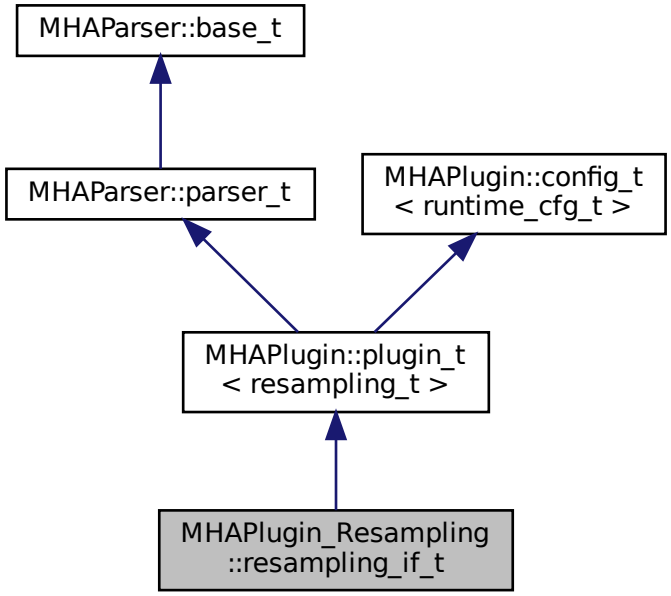
`template<class runtime_cfg_t >`
MHAParser::mhaconfig_mon_t MHAPlugin::plugin_t< runtime_cfg_t >::mhaconfig_out
[private]

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

- **mha_plugin.hh**

5.357 MHAPlugin_Resampling::resampling_if_t Class Reference

Inheritance diagram for MHAPlugin_Resampling::resampling_if_t:



Public Member Functions

- **resampling_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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 **algo**

Additional Inherited Members

5.357.1 Constructor & Destructor Documentation

5.357.1.1 resampling_if_t() MHAPlugin_Resampling::resampling_if_t::resampling_if_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.357.2 Member Function Documentation

5.357.2.1 process() mha_wave_t * MHAPlugin_Resampling::resampling_if_t::process (

```
mha_wave_t * s )
```

5.357.2.2 prepare() void MHAPlugin_Resampling::resampling_if_t::prepare (

```
mhaconfig_t & conf ) [virtual]
```

Implements **MHAPlugin::plugin_t< resampling_t >** (p. 1301).

5.357.2.3 release() void MHAPlugin_Resampling::resampling_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< resampling_t >** (p. 1302).

5.357.3 Member Data Documentation

5.357.3.1 srate **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::srate [private]

5.357.3.2 fragsize **MHAParser::int_t** MHAPlugin_Resampling::resampling_if_t::fragsize [private]

5.357.3.3 nyquist_ratio **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::nyquist_ratio [private]

5.357.3.4 irslen_outer2inner **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::irslen_outer2inner [private]

5.357.3.5 irslen_inner2outer **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::irslen_inner2outer [private]

5.357.3.6 plugloader **MHAParser::mhapluginloader_t** MHAPlugin_Resampling::resampling_if_t::plugloader [private]

5.357.3.7 algo `std::string MHAPlugin_Resampling::resampling_if_t::algo [private]`

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

- **resampling.cpp**

5.358 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**

5.358.1 Constructor & Destructor Documentation

5.358.1.1 resampling_t() `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 )
```

5.358.2 Member Function Documentation

5.358.2.1 process() `mha_wave_t * MHAPlugin_Resampling::resampling_t::process ( mha_wave_t * s )`

5.358.3 Member Data Documentation

5.358.3.1 outer_fragsize `unsigned MHAPlugin_Resampling::resampling_t::outer_fragsize [private]`

5.358.3.2 inner_fragsize `unsigned MHAPlugin_Resampling::resampling_t::inner_fragsize [private]`

5.358.3.3 outer_srate `float MHAPlugin_Resampling::resampling_t::outer_srate [private]`

5.358.3.4 inner_srate `float MHAPlugin_Resampling::resampling_t::inner_srate [private]`

5.358.3.5 nchannels_in `unsigned MHAPlugin_Resampling::resampling_t::nchannels_in [private]`

5.358.3.6 nchannels_out `unsigned MHAPlugin_Resampling::resampling_t::nchannels_out [private]`

5.358.3.7 outer2inner_resampling **MHAFilter::blockprocessing_polyphase_resampling_t** MHAPlugin_Resampling::resampling_t::outer2inner_resampling [private]

5.358.3.8 inner2outer_resampling **MHAFilter::blockprocessing_polyphase_resampling_t** MHAPlugin_Resampling::resampling_t::inner2outer_resampling [private]

5.358.3.9 plugloader **MHAParser::mhapluginloader_t**& MHAPlugin_Resampling::resampling_t::plugloader [private]

5.358.3.10 inner_signal **MHASignal::waveform_t** MHAPlugin_Resampling::resampling_t::inner_signal [private]

5.358.3.11 output_signal **MHASignal::waveform_t** MHAPlugin_Resampling::resampling_t::output_signal [private]

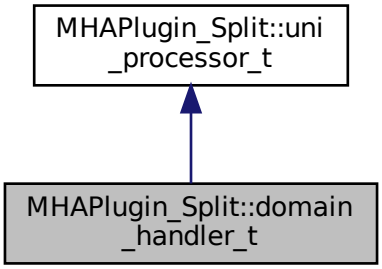
The documentation for this class was generated from the following file:

- **resampling.cpp**

5.359 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.

5.359.1 Detailed Description

Handles domain-specific partial input and output signal.

5.359.2 Constructor & Destructor Documentation

5.359.2.1 domain_handler_t() [1/2] MHAPlugin_Split::domain_handler_t::domain_↩ handler_t (

```
const domain_handler_t & ) [private]
```

Disallow copy constructor.

5.359.2.2 domain_handler_t() [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.

5.359.2.3 ∼domain_handler_t() virtual MHAPlugin_Split::domain_handler_t::∼domain↩ _handler_t ( ) [inline], [virtual]

Deallocation of signal holders.

5.359.3 Member Function Documentation

5.359.3.1 operator=()

```
domain_handler_t& MHAPlugin_Split::domain_handler_t::operator=
(
        const domain_handler_t & ) [private]
```

Disallow assignment operator.

5.359.3.2 set_input_domain()

```
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

5.359.3.3 set_output_domain()

```
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

5.359.3.4 deallocate_domains()

```
void MHAPlugin_Split::domain_handler_t::deallocate↩
_domains ( ) [inline]
```

Deallocate domain indicators and signal holders.

5.359.3.5 put_signal() [1/2] 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. 1316).

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

5.359.3.6 put_signal() [2/2] 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. 1316).

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

5.359.3.7 get_signal() [1/2] 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. 1316) 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

5.359.3.8 get_signal() [2/2] 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. 1316) 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

5.359.3.9 process() `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. 1313) before this method may be called.

Implements **MHAPlugin_Split::uni_processor_t** (p. 1341).

5.359.4 Member Data Documentation

5.359.4.1 wave_in `MHASignal::waveform_t* MHAPlugin_Split::domain_handler_t::wave_in`

Partial wave input signal.

5.359.4.2 wave_out `mha_wave_t** MHAPlugin_Split::domain_handler_t::wave_out`

Partial wave output signal.

5.359.4.3 spec_in `MHASignal::spectrum_t* MHAPlugin_Split::domain_handler_t::spec_in`

Partial spec input signal.

5.359.4.4 spec_out `mha_spec_t** MHAPlugin_Split::domain_handler_t::spec_out`

Partial spec input signal.

5.359.4.5 processor `PluginLoader::fourway_processor_t* MHAPlugin_Split::domain_↩ handler_t::processor`

The domain-specific signal processing methods are implemented here.

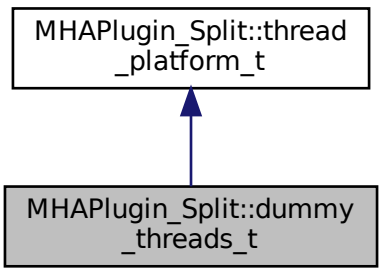
The documentation for this class was generated from the following file:

- **split.cpp**

5.360 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

5.360.1 Detailed Description

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

5.360.2 Constructor & Destructor Documentation

5.360.2.1 dummy_threads_t() `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

5.360.3 Member Function Documentation

5.360.3.1 kick_thread() `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. 1338).

5.360.3.2 catch_thread() `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. 1339).

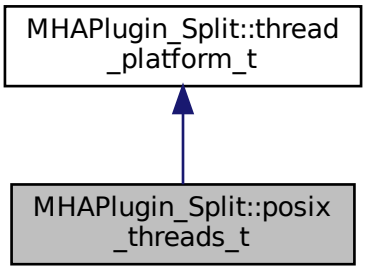
The documentation for this class was generated from the following file:

- **split.cpp**

5.361 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

5.361.1 Detailed Description

Posix threads specification of thread platform.

5.361.2 Constructor & Destructor Documentation

5.361.2.1 posix_threads_t()

```
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.

5.361.2.2 ~posix_threads_t() `MHAPlugin_Split::posix_threads_t::~posix_threads_t ( ) [inline]`

Terminate thread.

5.361.3 Member Function Documentation

5.361.3.1 kick_thread() `void MHAPlugin_Split::posix_threads_t::kick_thread ( ) [inline], [virtual]`

Start signal processing in separate thread.

Implements **MHAPlugin_Split::thread_platform_t** (p. 1338).

5.361.3.2 catch_thread() `void MHAPlugin_Split::posix_threads_t::catch_thread ( ) [inline], [virtual]`

Wait for signal processing to finish.

Implements **MHAPlugin_Split::thread_platform_t** (p. 1339).

5.361.3.3 thread_start() static void* MHAPlugin_Split::posix_threads_t::thread_start (void * *thr*) [inline], [static]

Thread start function.

5.361.3.4 main() void MHAPlugin_Split::posix_threads_t::main ( ) [inline]

Thread main loop. Wait for process/termination trigger, then act.

5.361.3.5 current_thread_scheduler() static std::string MHAPlugin_Split::posix_threads_t::current_thread_scheduler ( ) [inline], [static]

5.361.3.6 current_thread_priority() static int MHAPlugin_Split::posix_threads_t::current_thread_priority ( ) [inline], [static]

5.361.4 Member Data Documentation

5.361.4.1 mutex pthread_mutex_t MHAPlugin_Split::posix_threads_t::mutex [private]

The mutex.

5.361.4.2 kick_condition pthread_cond_t MHAPlugin_Split::posix_threads_t::kick_condition [private]

The condition for signalling the kicking and termination.

5.361.4.3 catch_condition

`pthread_cond_t MHAPlugin_Split::posix_threads_t::catch_condition [private]`

The condition for signalling the processing is finished.

5.361.4.4 attr

`pthread_attr_t MHAPlugin_Split::posix_threads_t::attr [private]`

Thread attributes.

5.361.4.5 priority

`struct sched_param MHAPlugin_Split::posix_threads_t::priority [private]`

Thread scheduling priority.

5.361.4.6 scheduler

`int MHAPlugin_Split::posix_threads_t::scheduler [private]`

5.361.4.7 thread

`pthread_t MHAPlugin_Split::posix_threads_t::thread [private]`

The thread object.

5.361.4.8 kicked

`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.

5.361.4.9 processing_done

`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.

5.361.4.10 termination_request `bool MHAPlugin_Split::posix_threads_t::termination_request [private]`

Set to true by the destructor.

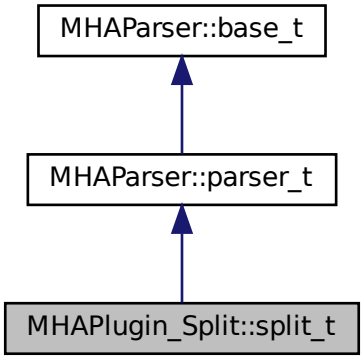
The documentation for this class was generated from the following file:

- **split.cpp**

5.362 MHAPlugin_Split::split_t Class Reference

Implements split plugin.

Inheritance diagram for MHAPlugin_Split::split_t:



Public Member Functions

- **split_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.362.1 Detailed Description

Implements split plugin.

An instance of class **split_t** (p. 1324) implements the split plugin functionality: The audio channels are splitted and groups of audio channels are processed by different plugins in parallel.

5.362.2 Constructor & Destructor Documentation

5.362.2.1 split_t() MHAPlugin_Split::split_t::split_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Plugin constructor.

5.362.2.2 ~split_t() MHAPlugin_Split::split_t::~split_t ( )

Plugin destructor. Unloads nested plugins.

5.362.3 Member Function Documentation

5.362.3.1 prepare_() void MHAPlugin_Split::split_t::prepare_ (

```
mhaconfig_t & signal_parameters )
```

Check signal parameters, prepare chains, and allocate output signal holders.

5.362.3.2 release_() `void MHAPlugin_Split::split_t::release_ ( )`

Delete output signal holder and release chains.

5.362.3.3 process() `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.

5.362.3.4 update() `void MHAPlugin_Split::split_t::update ( ) [private]`

Load plugins in response to a value change in the algos variable.

5.362.3.5 clear_chains() `void MHAPlugin_Split::split_t::clear_chains ( ) [private]`

Unload the plugins.

5.362.3.6 copy_output_wave() `mha_wave_t* MHAPlugin_Split::split_t::copy_output_wave ( ) [private]`

5.362.3.7 copy_output_spec() `mha_spec_t* MHAPlugin_Split::split_t::copy_output_spec ( ) [private]`

5.362.3.8 trigger_processing() 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.

5.362.3.9 collect_result() template<class SigType >

void MHAPlugin_Split::split_t::collect_result (
SigType * *s_out*) [private]

Combine the output signal from the plugins.

5.362.3.10 signal_out() [1/2] 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.

5.362.3.11 signal_out() [2/2] MHASignal::spectrum_t* MHAPlugin_Split::split_t::signal_out (

mha_spec_t **) [inline], [private]

Spectrum domain output signal structure. Parameter is ignored.

5.362.4 Member Data Documentation

5.362.4.1 patchbay MHAEvents::patchbay_t< split_t> MHAPlugin_Split::split_t::patchbay [private]

Reload plugins when the algos variable changes.

5.362.4.2 algos

MHAParser::vstring_t MHAPlugin_Split::split_t::algos [private]

Vector of plugins to load in parallel.

5.362.4.3 channels

MHAParser::vint_t MHAPlugin_Split::split_t::channels [private]

Number of channels to route through each plugin.

5.362.4.4 thread_platform

MHAParser::kw_t MHAPlugin_Split::split_t::thread_platform [private]

Thread platform chooser.

5.362.4.5 worker_thread_scheduler

MHAParser::kw_t MHAPlugin_Split::split_t::worker_thread_scheduler [private]

Scheduler used for worker threads.

5.362.4.6 worker_thread_priority

MHAParser::int_t MHAPlugin_Split::split_t::worker_thread_priority [private]

Priority of worker threads.

5.362.4.7 framework_thread_scheduler

MHAParser::string_mon_t MHAPlugin_Split::split_t::framework_thread_scheduler [private]

Scheduler of the signal processing thread.

5.362.4.8 framework_thread_priority

MHAParser::int_mon_t MHAPlugin_Split::split↩
_t::framework_thread_priority [private]

Priority of signal processing thread.

5.362.4.9 delay

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.

5.362.4.10 chains

std::vector< **splitted_part_t***> MHAPlugin_Split::split_t::chains
[private]

Interfaces to parallel plugins.

5.362.4.11 wave_out

MHASignal::waveform_t* MHAPlugin_Split::split_t::wave_out
[private]

Combined output waveforms structure.

5.362.4.12 spec_out

MHASignal::spectrum_t* MHAPlugin_Split::split_t::spec_out
[private]

Combined output spectra structure.

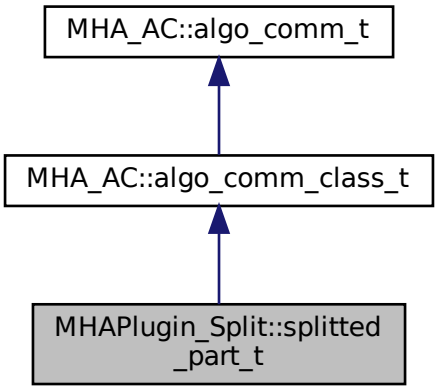
The documentation for this class was generated from the following file:

- **split.cpp**

5.363 MHAPlugin_Split::splitted_part_t Class Reference

The **splitted_part_t** (p. 1331) 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. 1336).*
- 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. 1310) instance.*
- void **release** ()
  *Delegates the release method to the plugin and deletes the **MHAPlugin_Split::domain_handler_t** (p. 1310) 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.

5.363.1 Detailed Description

The **splitted_part_t** (p. 1331) 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.

5.363.2 Constructor & Destructor Documentation

5.363.2.1 splitted_part_t() [1/3] MHAPlugin_Split::splitted_part_t::splitted_part_t (const **splitted_part_t** &) [private]

Disallow copy constructor.

5.363.2.2 splitted_part_t() [2/3] 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).

5.363.2.3 splitted_part_t() [3/3] 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. 1332) instance will take ownership of this instance and release it in the destructor.

5.363.2.4 ~splitted_part_t() MHAPlugin_Split::splitted_part_t::~splitted_part_t ( ) throw ( )

Destructor. Deletes the plugin **plug** (p. 1336).

5.363.3 Member Function Documentation

5.363.3.1 operator=()

```
splitted_part_t& MHAPlugin_Split::splitted_part_t::operator=
(
        const splitted_part_t & ) [private]
```

Disallow assignment operator.

5.363.3.2 prepare()

```
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. 1310) 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 n support different thread schedulers. This value is used for platforms other than "posix".
thread_priority	The new thread priority. Interpretation and permi range depend on the thread platform and possibl the scheduler.

5.363.3.3 release()

```
void MHAPlugin_Split::splitted_part_t::release (
        void )
```

Delegates the release method to the plugin and deletes the **MHAPlugin_Split::domain_handler_t** (p. 1310) instance.

Release the loaded plugin.

Plugin release.

5.363.3.4 parse() std::string MHAPlugin_Split::splitted_part_t::parse (const std::string & *str*) [inline]

Delegates parser incovation to plugin.

5.363.3.5 trigger_processing() 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

5.363.3.6 collect_result() 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

5.363.4 Member Data Documentation

5.363.4.1 plug `PluginLoader::fourway_processor_t* MHAPlugin_Split::splitted_part↩_t::plug [private]`

The plugin that performs the signal processing on the prepared channels.

5.363.4.2 domain `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.

5.363.4.3 thread `thread_platform_t* MHAPlugin_Split::splitted_part_t::thread [private]`

The platform-dependent thread synchronization implementation.

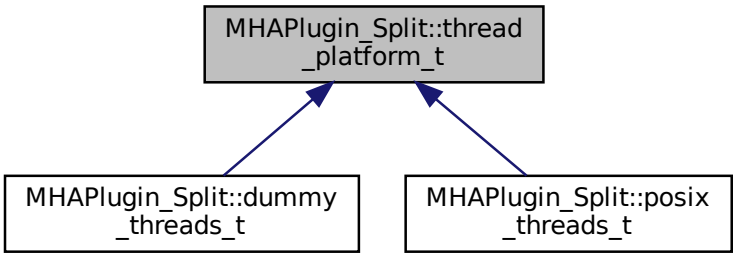
The documentation for this class was generated from the following file:

- **split.cpp**

5.364 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.

5.364.1 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.

5.364.2 Constructor & Destructor Documentation

5.364.2.1 thread_platform_t() [1/2] MHAPlugin_Split::thread_platform_t::thread_platform_t (const **thread_platform_t** &) [private]

Disallow copy constructor.

5.364.2.2 thread_platform_t() [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. 1331) instance.

5.364.2.3 ~thread_platform_t() 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.

5.364.3 Member Function Documentation

5.364.3.1 operator=() thread_platform_t& MHAPlugin_Split::thread_platform_t::operator= (const thread_platform_t &) [private]

Disallow assignment operator.

5.364.3.2 kick_thread() `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. 1321), and **MHAPlugin_Split↩ ::dummy_threads_t** (p. 1318).

5.364.3.3 catch_thread() `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. 1321), and **MHAPlugin_Split↩ ::dummy_threads_t** (p. 1318).

5.364.4 Member Data Documentation

5.364.4.1 processor **`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. 1340) interface instead of the mhaplugin-loader class directly for testability (no need to load real plugins for testing the thread platform).

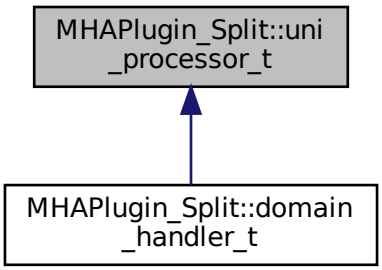
The documentation for this class was generated from the following file:

- **split.cpp**

5.365 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.

5.365.1 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.

5.365.2 Constructor & Destructor Documentation

5.365.2.1 ∼uni_processor_t() `virtual MHAPlugin_Split::uni_processor_t::∼uni_↩ processor_t ( ) [inline], [virtual]`

Classes containing virtual methods need virtual destructors.

5.365.3 Member Function Documentation

5.365.3.1 process() `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. 1315).

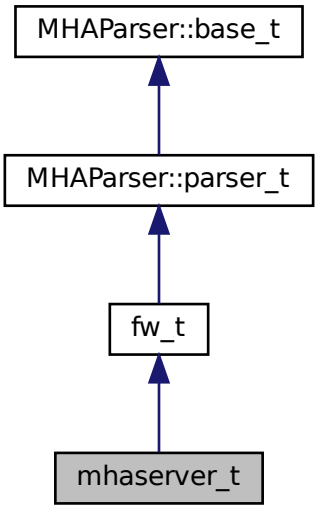
The documentation for this class was generated from the following file:

- **split.cpp**

5.366 mhaserver_t Class Reference

MHA Framework listening on TCP port for commands.

Inheritance diagram for mhaserver_t:



Classes

- class **tcp_server_t**

Public Member Functions

- **mhaserver_t** (const std::string &ao, const std::string &af, const std::string &lf, bool b_↩ interactive_)
- ∼**mhaserver_t** ()
- virtual std::string **on_received_line** (const std::string &line)

  A line of text was received from network client.
- virtual void **acceptor_started** ()

  Notification: "TCP port is open".
- virtual void **send_port_announcement** ()

  sends an announcement which port this MHA is listening on to the creator of the process.
- virtual void **start_stdin_thread** ()

  Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.
- 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

- std::shared_ptr< **tcp_server_t** > **tcpserver**
- std::string **ack_ok**
- std::string **ack_fail**
- std::string **logfile**
- unsigned short **announce_port**
- bool **b_interactive**
- **MHAParser::int_mon_t pid_mon**

Additional Inherited Members

5.366.1 Detailed Description

MHA Framework listening on TCP port for commands.

5.366.2 Constructor & Destructor Documentation

5.366.2.1 mhaserver_t() `mhaserver_t::mhaserver_t (`

```
const std::string & ao,
const std::string & af,
const std::string & lf,
bool b_interactive_ )
```

Parameters

ao	Acknowledgement string at end of successful command responses
af	Acknowledgement string at end of failed command responses
lf	File system path of file to use as log file. MHA appends.

5.366.2.2 ~mhaserver_t() `mhaserver_t::~mhaserver_t ( )`

5.366.3 Member Function Documentation

5.366.3.1 on_received_line() `std::string mhaserver_t::on_received_line (`

```
const std::string & line ) [virtual]
```

A line of text was received from network client.

5.366.3.2 acceptor_started() `void mhaserver_t::acceptor_started ( ) [virtual]`

Notification: "TCP port is open".

5.366.3.3 send_port_announcement() `void mhaserver_t::send_port_announcement ( ) [virtual]`

sends an announcement which port this MHA is listening on to the creator of the process.

See command line option –announce

5.366.3.4 start_stdin_thread() `void mhaserver_t::start_stdin_thread ( ) [virtual]`

Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.

Enables users to type mha configuration language commands directly into the terminal where MHA was started, without the need to use third-party tools like nc or putty.

5.366.3.5 set_announce_port() `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.

5.366.3.6 logstring() `void mhaserver_t::logstring ( const std::string & s ) [inline]`

Log a message to log file.

5.366.3.7 run() `int mhaserver_t::run ( unsigned short port, const std::string & _interface )`

Accept network connections and act on commands.

Calls **acceptor_started()** (p. 1343) when the TCP port is opened. Calls on_received_line for every line received.

Returns

exit code that can be used as process exit code

5.366.4 Member Data Documentation

5.366.4.1 tcpserver `std::shared_ptr< tcp_server_t> mhaserver_t::tcpserver [private]`

5.366.4.2 ack_ok `std::string mhaserver_t::ack_ok [private]`

5.366.4.3 ack_fail `std::string mhaserver_t::ack_fail [private]`

5.366.4.4 logfile `std::string mhaserver_t::logfile [private]`

5.366.4.5 announce_port `unsigned short mhaserver_t::announce_port [private]`

5.366.4.6 b_interactive `bool mhaserver_t::b_interactive [private]`

5.366.4.7 pid_mon `MHAParser::int_mon_t mhaserver_t::pid_mon [private]`

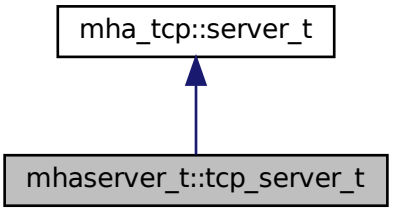
5.366.4.8 port `MHAParser::int_t mhaserver_t::port`

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

- **mhamain.cpp**

5.367 mhaserver_t::tcp_server_t Class Reference

Inheritance diagram for mhaserver_t::tcp_server_t:



Public Member Functions

- **tcp_server_t** (const std::string &interface, uint16_t **port**, **mhaserver_t** * **mha**)
- virtual bool **on_received_line** (std::shared_ptr< **mha_tcp::buffered_socket_t** > c, const std::string &l) override

  This method is invoked when a line of text is received on one of the accepted connections.

Private Attributes

- **mhaserver_t** * **mha**

5.367.1 Constructor & Destructor Documentation

5.367.1.1 tcp_server_t() `mhaserver_t::tcp_server_t::tcp_server_t (`

```
const std::string & interface,
uint16_t port,
 mhaserver_t * mha ) [inline]
```

5.367.2 Member Function Documentation

5.367.2.1 on_received_line()

```
virtual bool mhaserver_t::tcp_server_t::on_received_line (
            std::shared_ptr< mha_tcp::buffered_socket_t > c,
            const std::string & l ) [inline], [override], [virtual]
```

This method is invoked when a line of text is received on one of the accepted connections.

Override this method to process the communication with the client.

Parameters

c	the connection that has received this line
l	the line that has been received, without the line ending

Returns

client should return true when client wants to read another line of text, else false.

Reimplemented from **mha_tcp::server_t** (p. 966).

5.367.3 Member Data Documentation

5.367.3.1 mha

```
mhaserver_t* mhaserver_t::tcp_server_t::mha [private]
```

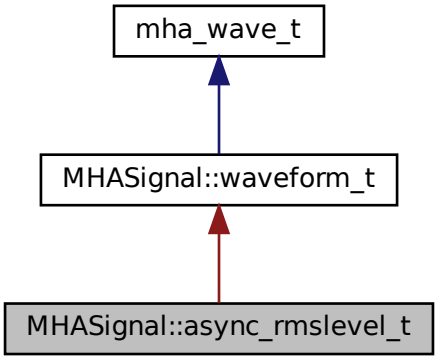
The documentation for this class was generated from the following file:

- **mhamain.cpp**

5.368 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

5.368.1 Detailed Description

Class for asynchronous level metering.

5.368.2 Constructor & Destructor Documentation

5.368.2.1 async_rmslevel_t() `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.

5.368.3 Member Function Documentation

5.368.3.1 rmslevel() `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).

5.368.3.2 peaklevel() `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).

5.368.3.3 process() `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).

5.368.4 Member Data Documentation

5.368.4.1 pos `unsigned int MHASignal::async_rmslevel_t::pos [private]`

5.368.4.2 filled `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**

5.369 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**

5.369.1 Constructor & Destructor Documentation

5.369.1.1 delay_spec_t() `MHASignal::delay_spec_t::delay_spec_t (`

```
unsigned int delay,
unsigned int frames,
unsigned int channels )
```

5.369.1.2 ∼delay_spec_t() `MHASignal::delay_spec_t::∼delay_spec_t ( )`

5.369.2 Member Function Documentation

5.369.2.1 process() **`mha_spec_t`** `∗ MHASignal::delay_spec_t::process (`

`mha_spec_t` `∗ s )`

5.369.3 Member Data Documentation

5.369.3.1 delay `unsigned int MHASignal::delay_spec_t::delay [private]`

5.369.3.2 buffer `MHASignal::spectrum_t** MHASignal::delay_spec_t::buffer [private]`

5.369.3.3 pos `unsigned int MHASignal::delay_spec_t::pos [private]`

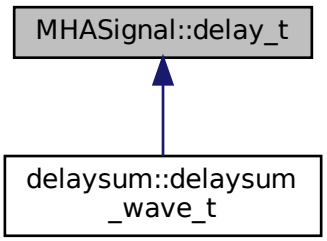
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.370 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**

5.370.1 Detailed Description

Class to realize a simple delay of waveform streams.

5.370.2 Constructor & Destructor Documentation

5.370.2.1 delay_t() `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.

5.370.2.2 ~delay_t() `MHASignal::delay_t::~delay_t ( )`

5.370.3 Member Function Documentation

5.370.3.1 process() `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.

5.370.3.2 inspect() `std::string MHASignal::delay_t::inspect ( ) const [inline]`

5.370.4 Member Data Documentation

5.370.4.1 channels `unsigned int MHASignal::delay_t::channels [private]`

5.370.4.2 delays `unsigned int* MHASignal::delay_t::delays [private]`

5.370.4.3 pos `unsigned int* MHASignal::delay_t::pos [private]`

5.370.4.4 buffer `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**

5.371 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**

5.371.1 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).

5.371.2 Constructor & Destructor Documentation

5.371.2.1 delay_wave_t() `MHASignal::delay_wave_t::delay_wave_t (`

```
unsigned int delay,
unsigned int frames,
unsigned int channels )
```

5.371.2.2 ∼delay_wave_t() `MHASignal::delay_wave_t::~delay_wave_t ( )`

5.371.3 Member Function Documentation

5.371.3.1 process() `mha_wave_t * MHASignal::delay_wave_t::process ( mha_wave_t * s )`

5.371.4 Member Data Documentation

5.371.4.1 delay `unsigned int MHASignal::delay_wave_t::delay [private]`

5.371.4.2 buffer `MHASignal::waveform_t** MHASignal::delay_wave_t::buffer [private]`

5.371.4.3 pos `unsigned int MHASignal::delay_wave_t::pos [private]`

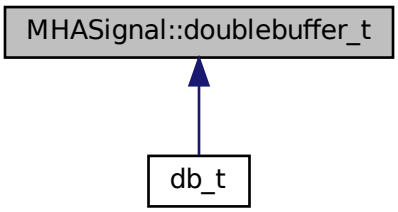
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.372 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**

5.372.1 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. 1358) is called whenever enough data is available.

5.372.2 Constructor & Destructor Documentation

5.372.2.1 doublebuffer_t()

```
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)

5.372.2.2 ~doublebuffer_t()

```
MHASignal::doublebuffer_t::~doublebuffer_t ( ) [virtual]
```

5.372.3 Member Function Documentation

5.372.3.1 outer_process()

```
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.

5.372.3.2 inner_process()

```
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. 444).

5.372.3.3 min()

```
unsigned int MHASignal::doublebuffer_t::min (
            unsigned int a,
            unsigned int b ) [inline], [private]
```

5.372.4 Member Data Documentation

5.372.4.1 outer_out

```
waveform_t MHASignal::doublebuffer_t::outer_out [private]
```

5.372.4.2 this_outer_out

```
mha_wave_t MHASignal::doublebuffer_t::this_outer_out [private]
```

5.372.4.3 inner_in

```
waveform_t MHASignal::doublebuffer_t::inner_in [private]
```

5.372.4.4 inner_out `waveform_t MHASignal::doublebuffer_t::inner_out [private]`

5.372.4.5 k_inner `unsigned int MHASignal::doublebuffer_t::k_inner [private]`

5.372.4.6 k_outer `unsigned int MHASignal::doublebuffer_t::k_outer [private]`

5.372.4.7 ch `unsigned int MHASignal::doublebuffer_t::ch [private]`

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.373 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**

5.373.1 Constructor & Destructor Documentation

5.373.1.1 fft_t() `MHASignal::fft_t::fft_t (`

```
const unsigned int & n )
```

5.373.1.2 ~fft_t() `MHASignal::fft_t::~fft_t ( )`

5.373.2 Member Function Documentation

5.373.2.1 wave2spec() `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.

5.373.2.2 spec2wave() [1/2]

```
void MHASignal::fft_t::spec2wave (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

5.373.2.3 spec2wave() [2/2]

```
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.

5.373.2.4 forward()

```
void MHASignal::fft_t::forward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

5.373.2.5 backward()

```
void MHASignal::fft_t::backward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

5.373.2.6 wave2spec_scale()

```
void MHASignal::fft_t::wave2spec_scale (
    const mha_wave_t * wave,
    mha_spec_t * spec,
    bool swap )
```

5.373.2.7 spec2wave_scale()

```
void MHASignal::fft_t::spec2wave_scale (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

5.373.2.8 forward_scale()

```
void MHASignal::fft_t::forward_scale (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

5.373.2.9 backward_scale()

```
void MHASignal::fft_t::backward_scale (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

5.373.2.10 sort_fftw2spec()

```
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 –].

5.373.2.11 sort_spec2fftw()

```
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.

5.373.3 Member Data Documentation

5.373.3.1 nfft

```
unsigned int MHASignal::fft_t::nfft [private]
```

5.373.3.2 n_re

```
unsigned int MHASignal::fft_t::n_re [private]
```

5.373.3.3 n_im unsigned int MHASignal::fft_t::n_im [private]

5.373.3.4 scale mha_real_t MHASignal::fft_t::scale [private]

5.373.3.5 buf_in mha_real_t* MHASignal::fft_t::buf_in [private]

5.373.3.6 buf_out mha_real_t* MHASignal::fft_t::buf_out [private]

5.373.3.7 fftw_plan_wave2spec rfftw_plan MHASignal::fft_t::fftw_plan_wave2spec [private]

5.373.3.8 fftw_plan_spec2wave rfftw_plan MHASignal::fft_t::fftw_plan_spec2wave [private]

5.373.3.9 fftw_plan_fft fftw_plan MHASignal::fft_t::fftw_plan_fft [private]

5.373.3.10 fftw_plan_ifft 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**

5.374 MHASignal::hilbert_fftw_t Class Reference

Public Member Functions

- **hilbert_fftw_t** (unsigned int len)
  *C'tor of **hilbert_fftw_t** (p. 1365).*
- ∼**hilbert_fftw_t** ()
  *D'tor of **hilbert_fftw_t** (p. 1365).*
- 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**

5.374.1 Constructor & Destructor Documentation

5.374.1.1 hilbert_fftw_t() `MHASignal::hilbert_fftw_t::hilbert_fftw_t ( unsigned int len )`

C'tor of **hilbert_fftw_t** (p. 1365).

Parameters

len	fft length

5.374.1.2 ∼hilbert_fftw_t() `MHASignal::hilbert_fftw_t::∼hilbert_fftw_t ( )`

D'tor of **hilbert_fftw_t** (p. 1365).

5.374.2 Member Function Documentation

5.374.2.1 hilbert() `void MHASignal::hilbert_fftw_t::hilbert (`

```
const  mha_wave_t * s_in,
 mha_wave_t * s_out )
```

5.374.3 Member Data Documentation

5.374.3.1 n `unsigned int MHASignal::hilbert_fftw_t::n [private]`

5.374.3.2 p1 `rfftw_plan MHASignal::hilbert_fftw_t::p1 [private]`

5.374.3.3 p2 `fftw_plan MHASignal::hilbert_fftw_t::p2 [private]`

5.374.3.4 buf_r_in `fftw_real* MHASignal::hilbert_fftw_t::buf_r_in [private]`

5.374.3.5 buf_r_out `fftw_real* MHASignal::hilbert_fftw_t::buf_r_out [private]`

5.374.3.6 buf_c_in `fftw_complex* MHASignal::hilbert_fftw_t::buf_c_in [private]`

5.374.3.7 buf_c_out `fftw_complex* MHASignal::hilbert_fftw_t::buf_c_out [private]`

5.374.3.8 sc `mha_real_t MHASignal::hilbert_fftw_t::sc [private]`

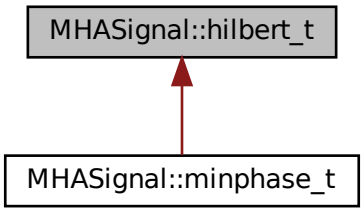
The documentation for this class was generated from the following file:

- **mha_signal.cpp**

5.375 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**

5.375.1 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.

5.375.2 Constructor & Destructor Documentation

5.375.2.1 hilbert_t() `MHASignal::hilbert_t::hilbert_t ( unsigned int len )`

Parameters

len	Length of waveform segment

5.375.2.2 ~hilbert_t() `MHASignal::hilbert_t::~hilbert_t ( )`

5.375.3 Member Function Documentation

5.375.3.1 operator()() `void MHASignal::hilbert_t::operator() ( const mha_wave_t * s_in, mha_wave_t * s_out )`

Apply Hilbert transformation on a waveform segment.

5.375.4 Member Data Documentation

5.375.4.1 h `void* MHASignal::hilbert_t::h [private]`

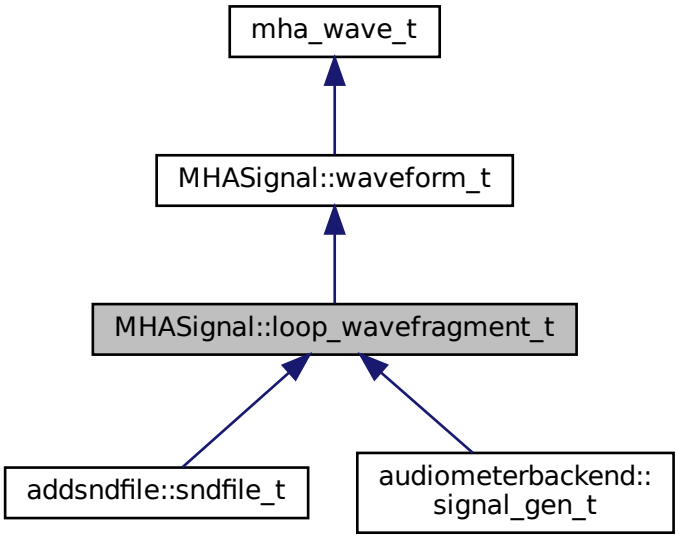
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.376 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

- enum **level_mode_t** { **relative** , **peak** , **rms** , **rms_limit40** }
  Switch for playback level mode.
- enum **playback_mode_t** { **add** , **replace** , **input** , **mute** }
  Switch for playback mode.

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. 1369) 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

5.376.1 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.

5.376.2 Member Enumeration Documentation

5.376.2.1 level_mode_t 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	

5.376.2.2 playback_mode_t 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).

5.376.3 Constructor & Destructor Documentation

5.376.3.1 loop_wavefragment_t() 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. 1369) 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. 1370) for details)
channels	Mapping of input to output channels.
startpos	Starting position

5.376.4 Member Function Documentation

5.376.4.1 get_mapping()

```
std::vector< int > MHASignal::loop_wavefragment_t::get_mapping (
            unsigned int channels )
```

5.376.4.2 playback() [1/3]

```
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

5.376.4.3 playback() [2/3] 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.

5.376.4.4 playback() [3/3] 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).

5.376.4.5 set_level_lin() void MHASignal::loop_wavefragment_t::set_level_lin (

```
mha_real_t l )
```

5.376.4.6 set_level_db() void MHASignal::loop_wavefragment_t::set_level_db (

```
mha_real_t l )
```

5.376.4.7 rewind() void MHASignal::loop_wavefragment_t::rewind ( ) [inline]

5.376.4.8 locate_end() void MHASignal::loop_wavefragment_t::locate_end ( ) [inline]

5.376.4.9 is_playback_active() bool MHASignal::loop_wavefragment_t::is_playback_active ( ) const [inline]

5.376.5 Member Data Documentation

5.376.5.1 playback_channels std::vector<int> MHASignal::loop_wavefragment_t::playback_channels [private]

5.376.5.2 b_loop bool MHASignal::loop_wavefragment_t::b_loop [private]

5.376.5.3 pos unsigned int MHASignal::loop_wavefragment_t::pos [private]

5.376.5.4 intern_level **MHASignal::waveform_t** MHASignal::loop_wavefragment_t::intern_level [private]

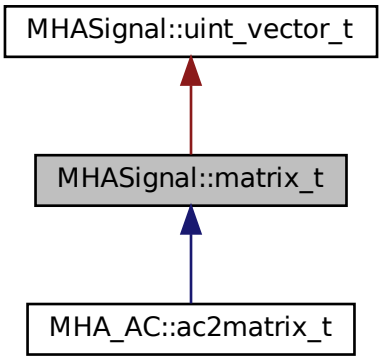
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.377 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 **MHA_AC::comm_var_t** &v)
  Fill matrix with data of an AC variable object.
- **MHA_AC::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

5.377.1 Detailed Description

n-dimensional matrix with real or complex floating point values.

Warning

The member functions **imag()** (p. 1381) and operator() should only be called if the matrix is defined to hold complex values.

5.377.2 Constructor & Destructor Documentation

5.377.2.1 matrix_t() [1/5]

```
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

5.377.2.2 matrix_t() [2/5] 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

5.377.2.3 matrix_t() [3/5] 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

5.377.2.4 matrix_t() [4/5] MHASignal::matrix_t::matrix_t (const **MHASignal::matrix_t** & *src*)

5.377.2.5 matrix_t() [5/5] MHASignal::matrix_t::matrix_t (const uint8_t * *buf*, unsigned int *len*)

Construct from memory area.

Warning

This constructor is not real time safe

5.377.2.6 ~matrix_t() MHASignal::matrix_t::~matrix_t ( )

5.377.3 Member Function Documentation

5.377.3.1 operator=() [1/2] matrix_t & MHASignal::matrix_t::operator= (const MHASignal::matrix_t & src)

5.377.3.2 operator=() [2/2] MHASignal::matrix_t & MHASignal::matrix_t::operator= (const MHA_AC::comm_var_t & v)

Fill matrix with data of an AC variable object.

Parameters

v	Source AC variable (comm_var_t)

Note

The type and dimension of the AC variable must match the type and dimension of the matrix.

5.377.3.3 get_comm_var() MHA_AC::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), valid for the life time of the matrix.

5.377.3.4 dimension() `unsigned int MHASignal::matrix_t::dimension ( ) const [inline]`

Return the dimension of the matrix.

Returns

Dimension of the matrix

5.377.3.5 size() `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

5.377.3.6 get_nelements() `unsigned int MHASignal::matrix_t::get_nelements ( ) const`

Return total number of elements.

5.377.3.7 is_same_size() `bool MHASignal::matrix_t::is_same_size ( const MHASignal::matrix_t & src )`

Test if matrix has same size as other.

5.377.3.8 iscomplex() `bool MHASignal::matrix_t::iscomplex ( ) const [inline]`

Return information about complexity.

5.377.3.9 real() [1/4] 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

5.377.3.10 imag() [1/4] 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

5.377.3.11 operator()() [1/4] 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

5.377.3.12 real() [2/4] 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

5.377.3.13 imag() [2/4]

```
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

5.377.3.14 operator()() [2/4]

```
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

5.377.3.15 real() [3/4]

```
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

Parameters

col	Column number of element

5.377.3.16 imag() [3/4]

```
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

5.377.3.17 operator()() [3/4]

```
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

5.377.3.18 real() [4/4] 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

5.377.3.19 imag() [4/4] 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

5.377.3.20 operator()() [4/4] 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

Parameters

col	Column number of element

5.377.3.21 get_nreals() `unsigned int MHASignal::matrix_t::get_nreals ( ) const [inline]`

5.377.3.22 get_index() [1/2] `unsigned int MHASignal::matrix_t::get_index (`
`unsigned int row,`
`unsigned int col ) const`

5.377.3.23 get_index() [2/2] `unsigned int MHASignal::matrix_t::get_index (`
`const` **MHASignal::uint_vector_t** `& index ) const`

5.377.3.24 numbytes() `unsigned int MHASignal::matrix_t::numbytes ( ) const`

Return number of bytes needed to store into memory.

5.377.3.25 write() `unsigned int MHASignal::matrix_t::write (`
`uint8_t * buf,`
`unsigned int len ) const`

Copy to memory area.

5.377.3.26 get_rdata() `const` **mha_real_t*** `MHASignal::matrix_t::get_rdata ( ) const [inline]`

Return pointer of real data.

5.377.3.27 get_cdata() const mha_complex_t* MHASignal::matrix_t::get_cdata ( ) const [inline]

Return pointer of complex data.

5.377.4 Member Data Documentation

5.377.4.1 complex_ofs uint32_t MHASignal::matrix_t::complex_ofs [private]

5.377.4.2 nelements uint32_t MHASignal::matrix_t::nelements [private]

5.377.4.3 rdata mha_real_t* MHASignal::matrix_t::rdata

5.377.4.4 cdata mha_complex_t* MHASignal::matrix_t::cdata

5.377.4.5 union { ... } [private]

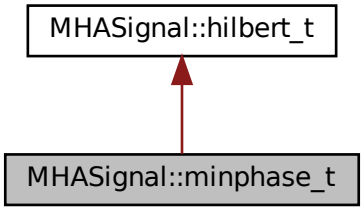
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.378 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

5.378.1 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 $\mathcal{H}\{\cdots\}$.

5.378.2 Constructor & Destructor Documentation

5.378.2.1 minphase_t() `MHASignal::minphase_t::minphase_t (`

```
unsigned int fftlen,
unsigned int ch )
```

Constructor.

Parameters

fftlen	FFT length
ch	Number of channels

5.378.3 Member Function Documentation

5.378.3.1 operator()() `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.

5.378.4 Member Data Documentation

5.378.4.1 phase `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**

5.379 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**

5.379.1 Detailed Description

Simple simulation of fixpoint quantization.

5.379.2 Constructor & Destructor Documentation

5.379.2.1 quantizer_t() `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.

5.379.3 Member Function Documentation

5.379.3.1 operator()() `void MHASignal::quantizer_t::operator() ( mha_wave_t & s )`

Quantization of a waveform fragment.

Parameters

s	Waveform fragment to be quantized.

5.379.4 Member Data Documentation

5.379.4.1 limit `bool MHASignal::quantizer_t::limit [private]`

5.379.4.2 upscale `mha_real_t MHASignal::quantizer_t::upscale [private]`

5.379.4.3 downscale `mha_real_t MHASignal::quantizer_t::downscale [private]`

5.379.4.4 up_limit `mha_real_t MHASignal::quantizer_t::up_limit [private]`

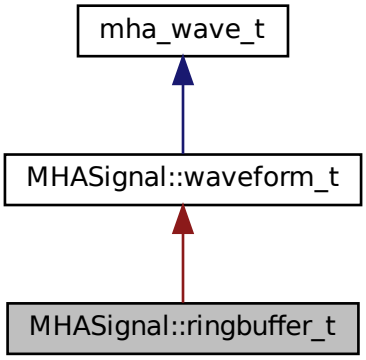
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.380 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

5.380.1 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. 1394) method. Individual audio samples can be accessed and altered using the **value** (p. 1393) method. Blocks of audio data can be deleted from the ringbuffer using the **discard** (p. 1393) method.

5.380.2 Constructor & Destructor Documentation

5.380.2.1 ringbuffer_t()

```
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. Maximu number of frames that can be stored in the ringbuf 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 (zero into the ringbuffer before the start of the actual sig inserted for the first time.

Exceptions

MHA_Error *(p. 906)*	if prefilled_frames > frames

5.380.3 Member Function Documentation

5.380.3.1 contained_frames() `unsigned MHASignal::ringbuffer_t::contained_frames ( ) const [inline]`

number of currently contained frames

5.380.3.2 value() `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. 1393) 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. 906)*	if channel or frame out of bounds.

5.380.3.3 discard() `void MHASignal::ringbuffer_t::discard ( unsigned frames ) [inline]`

Discards the oldest frames.

Makes room for new **write** (p. 1394), alters base frame index for **value** (p. 1393)

Parameters

frames	how many frames to discard.

Exceptions

MHA_Error *(p. 906)*	if frames > **contained_frames** (p. 1393)

5.380.3.4 write() `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. 906)*	if there is not enough space or if the channel co mismatches. Nothing is copied if the space is in

5.380.4 Member Data Documentation

5.380.4.1 next_read_frame_index `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. 1393) method, and this value is altered when **discard** (p. 1393) is called.

5.380.4.2 next_write_frame_index `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.

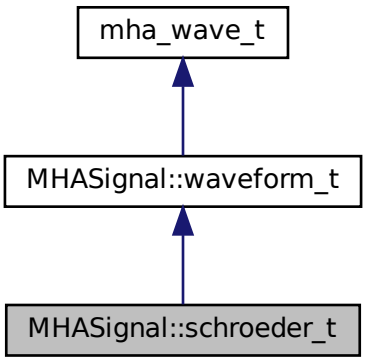
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.381 MHASignal::schroeder_t Class Reference

Schroeder tone complex class.

Inheritance diagram for MHASignal::schroeder_t:



Public Types

- enum **sign_t** { **up** , **down** }
  Enumerator for sign of Schroeder tone complex sweep direction.
- 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

5.381.1 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.

5.381.2 Member Typedef Documentation

5.381.2.1 groupdelay_t `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.

5.381.3 Member Enumeration Documentation

5.381.3.1 sign_t 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 ($\sigma = +1$)

5.381.4 Constructor & Destructor Documentation

5.381.4.1 schroeder_t() [1/2] 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)

5.381.4.2 schroeder_t() [2/2]

```
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.

5.381.5 Member Function Documentation

5.381.5.1 identity()

```
static float MHASignal::schroeder_t::identity (
    float x,
    float ,
    float  ) [inline], [static]
```

5.381.5.2 log_up()

```
static float MHASignal::schroeder_t::log_up (
    float x,
    float fmin,
    float fmax ) [inline], [static]
```

5.381.5.3 log_down()

```
static float MHASignal::schroeder_t::log_down (
    float x,
    float fmin,
    float fmax ) [inline], [static]
```

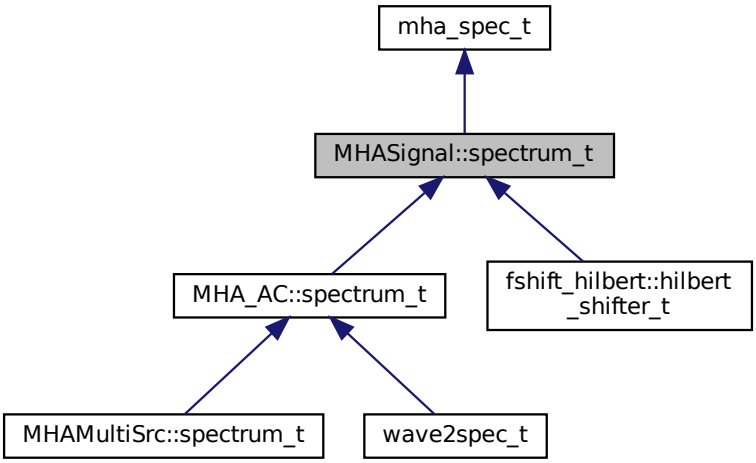
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.382 MHASignal::spectrum_t Class Reference

a signal processing class for spectral data (based on **mha_spec_t** (p. 937))

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

5.382.1 Detailed Description

a signal processing class for spectral data (based on **mha_spec_t** (p. 937))

5.382.2 Constructor & Destructor Documentation

5.382.2.1 spectrum_t() [1/4]

```
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

5.382.2.2 spectrum_t() [2/4]

```
spectrum_t::spectrum_t (
    const mha_spec_t & src ) [explicit]
```

Copy constructor.

5.382.2.3 spectrum_t() [3/4]

```
spectrum_t::spectrum_t (
    const MHASignal::spectrum_t & src )
```

Copy constructor.

5.382.2.4 spectrum_t() [4/4]

```
spectrum_t::spectrum_t (
    const std::vector< mha_complex_t > & src )
```

5.382.2.5 ~spectrum_t()

```
spectrum_t::~spectrum_t (
    void ) [virtual]
```

Reimplemented in **MHA_AC::spectrum_t** (p. 875).

5.382.3 Member Function Documentation

5.382.3.1 operator()() `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

5.382.3.2 operator[]() `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

5.382.3.3 value()

```
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

5.382.3.4 copy()

```
void spectrum_t::copy (
    const mha_spec_t & src )
```

copy all elements from a spectrum

Parameters

src	input spectrum

5.382.3.5 copy_channel()

```
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

Parameters

dch	Channel index in destination (this) signal

5.382.3.6 export_to() `void spectrum_t::export_to ( mha_spec_t & dest )`

copy elements to spectrum structure

Parameters

dest	destination spectrum structure

5.382.3.7 scale() `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

5.382.3.8 scale_channel() `void spectrum_t::scale_channel ( const unsigned int & ch, const mha_real_t & src )`

scale all elements in one channel

Parameters

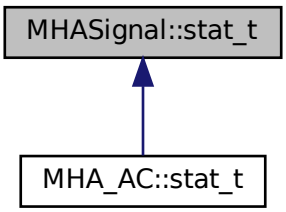
ch	channel number
src	scale factor

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.383 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**

5.383.1 Constructor & Destructor Documentation

5.383.1.1 stat_t()

```
MHASignal::stat_t::stat_t (
    const unsigned int & frames,
    const unsigned int & channels )
```

5.383.2 Member Function Documentation

5.383.2.1 mean()

```
void MHASignal::stat_t::mean (
    mha_wave_t & m )
```

5.383.2.2 mean_std()

```
void MHASignal::stat_t::mean_std (
    mha_wave_t & m,
    mha_wave_t & s )
```

5.383.2.3 push() [1/2]

```
void MHASignal::stat_t::push (
    const mha_wave_t & x )
```

5.383.2.4 push() [2/2]

```
void MHASignal::stat_t::push (
    const mha_real_t & x,
    const unsigned int & k,
    const unsigned int & ch )
```

5.383.3 Member Data Documentation

5.383.3.1 n `MHASignal::waveform_t MHASignal::stat_t::n [private]`

5.383.3.2 sum `MHASignal::waveform_t MHASignal::stat_t::sum [private]`

5.383.3.3 sum2 `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**

5.384 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.

5.384.1 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.

5.384.2 Constructor & Destructor Documentation

5.384.2.1 subsample_delay_t() `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. 906)*	if the parameters are out of range

5.384.3 Member Function Documentation

5.384.3.1 process() [1/2] void MHASignal::subsample_delay_t::process (mha_spec_t * s)

Apply the phase_gains to s to achieve the subsample delay.

5.384.3.2 process() [2/2] 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. 906)*	if idx >= s->num_channels

5.384.4 Member Data Documentation

5.384.4.1 phase_gains spectrum_t MHASignal::subsample_delay_t::phase_gains

The complex factors to apply to achieve the necessary phase shift.

5.384.4.2 last_complex_bin unsigned MHASignal::subsample_delay_t::last_complex_bin [private]

index of the last complex fft bin for the used fft length.

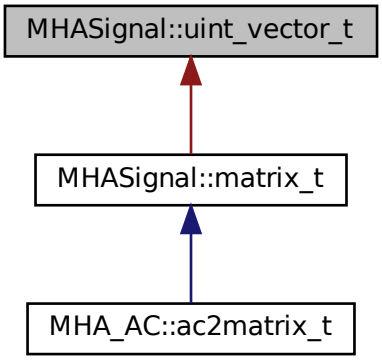
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.385 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. 1410).
- 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**

5.385.1 Detailed Description

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

5.385.2 Constructor & Destructor Documentation

5.385.2.1 uint_vector_t() [1/3] MHASignal::uint_vector_t::uint_vector_t (unsigned int *len*)

Constructor, initializes all elements to zero.

Parameters

len	Length of vector.

5.385.2.2 uint_vector_t() [2/3] MHASignal::uint_vector_t::uint_vector_t (const **uint_vector_t** & *src*)

5.385.2.3 uint_vector_t() [3/3] 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

5.385.2.4 ~uint_vector_t() MHASignal::uint_vector_t::~uint_vector_t ( )

5.385.3 Member Function Documentation

5.385.3.1 operator==() bool MHASignal::uint_vector_t::operator== (const uint_vector_t & src) const

Check for equality.

5.385.3.2 operator=() uint_vector_t & MHASignal::uint_vector_t::operator= (const uint_vector_t & src)

Assign from other **uint_vector_t** (p. 1410).

Warning

This assignment will fail if the lengths mismatch.

5.385.3.3 get_length() unsigned int MHASignal::uint_vector_t::get_length ( ) const [inline]

Return the length of the vector.

5.385.3.4 operator[]() [1/2] const uint32_t& MHASignal::uint_vector_t::operator[] (unsigned int k) const [inline]

Read-only access to elements.

5.385.3.5 operator[]() [2/2] uint32_t& MHASignal::uint_vector_t::operator[] (unsigned int *k*) [inline]

Access to elements.

5.385.3.6 numbytes() unsigned int MHASignal::uint_vector_t::numbytes ( ) const

Return number of bytes needed to store into memory.

5.385.3.7 write() unsigned int MHASignal::uint_vector_t::write (uint8_t * *buf*, unsigned int *len*) const

Copy to memory area.

5.385.3.8 getdata() const uint32_t* MHASignal::uint_vector_t::getdata ( ) const [inline]

Return pointer to the data field.

5.385.4 Member Data Documentation

5.385.4.1 length uint32_t MHASignal::uint_vector_t::length [protected]

5.385.4.2 data uint32_t* MHASignal::uint_vector_t::data [protected]

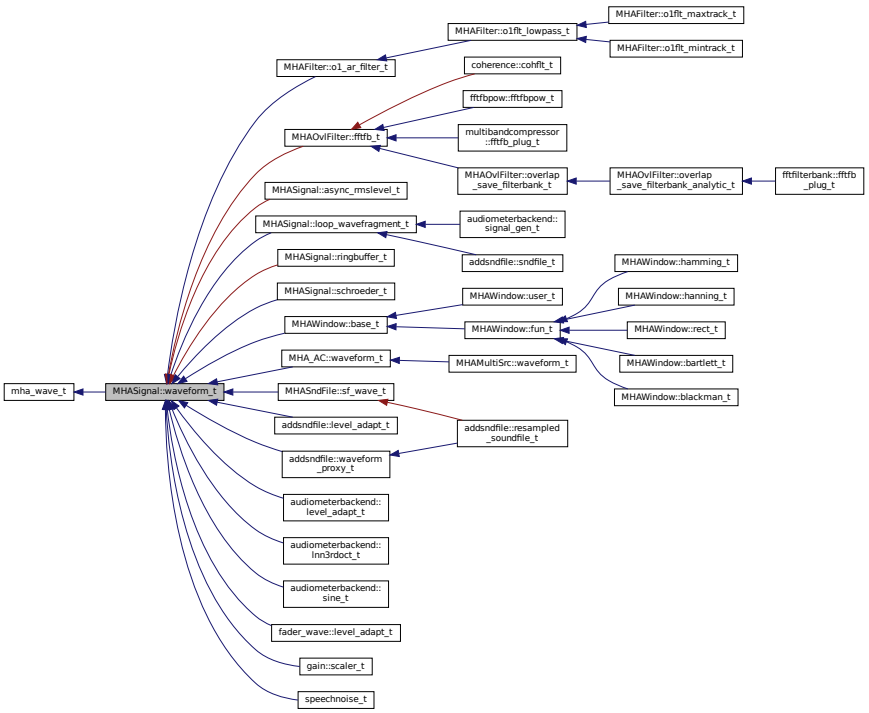
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.386 MHASignal::waveform_t Class Reference

signal processing class for waveform data (based on **mha_wave_t** (p. 985))

Inheritance diagram for MHASignal::waveform_t:



Public Member Functions

- **waveform_t** (const unsigned int &frames, const unsigned int & **channels**)
  *constructor of **waveform_t*** (p. 1414)
- **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. 985) *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)
- std::vector< **mha_real_t** > **flatten** () const
- **operator std::vector**< **mha_real_t** > () const
- 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. 985) *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

5.386.1 Detailed Description

signal processing class for waveform data (based on **mha_wave_t** (p. 985))

5.386.2 Constructor & Destructor Documentation

5.386.2.1 waveform_t() [1/5]

```
waveform_t::waveform_t (
    const unsigned int & frames,
    const unsigned int & channels )
```

constructor of **waveform_t** (p. 1414)

Allocates buffer memory and initializes values to zero.

Parameters

frames	number of frames in each channel
channels	number of channels

5.386.2.2 waveform_t() [2/5]

```
waveform_t::waveform_t (
    const mhaconfig_t & cf ) [explicit]
```

Constructor to create a waveform from plugin configuration.

Parameters

cf	Plugin configuration

5.386.2.3 waveform_t() [3/5] waveform_t::waveform_t (const **mha_wave_t** & *src*) [explicit]

Copy contructor for **mha_wave_t** (p. 985) source.

5.386.2.4 waveform_t() [4/5] waveform_t::waveform_t (const **MHASignal::waveform_t** & *src*)

Copy contructor.

5.386.2.5 waveform_t() [5/5] 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.

5.386.2.6 ~waveform_t() waveform_t::~waveform_t (void) [virtual]

Reimplemented in **MHA_AC::waveform_t** (p. 880).

5.386.3 Member Function Documentation

5.386.3.1 flatten() std::vector< mha_real_t > waveform_t::flatten ( ) const

5.386.3.2 operator std::vector< mha_real_t >() MHASignal::waveform_t::operator std::vector< mha_real_t > ( ) const [explicit]

5.386.3.3 operator=() void MHASignal::waveform_t::operator= (const mha_real_t & v) [inline]

5.386.3.4 operator[]() [1/2] mha_real_t& MHASignal::waveform_t::operator[] (unsigned int k) [inline]

5.386.3.5 operator[]() [2/2] const mha_real_t& MHASignal::waveform_t::operator[] (unsigned int k) const [inline]

5.386.3.6 value() [1/2] 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

5.386.3.7 operator()() [1/2]

```
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

5.386.3.8 value() [2/2]

```
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

5.386.3.9 operator()() [2/2] 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

5.386.3.10 sum() [1/3] 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

5.386.3.11 sum() [2/3] 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

5.386.3.12 sum() [3/3] mha_real_t waveform_t::sum ( )

sum of all elements

Returns

sum of all elements

5.386.3.13 sumsqr() mha_real_t waveform_t::sumsqr ( )

sum of square of all elements

Returns

sum of square of all elements

5.386.3.14 sum_channel() `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

5.386.3.15 assign() [1/2] `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

5.386.3.16 assign() [2/2] `void waveform_t::assign ( const mha_real_t & val )`

set all elements to value

Parameters

val	new value

5.386.3.17 assign_frame()

```
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

5.386.3.18 assign_channel()

```
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

5.386.3.19 copy() [1/3]

```
void waveform_t::copy (
const std::vector<  mha_real_t > & v )
```

5.386.3.20 copy() [2/3]

```
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)

5.386.3.21 copy() [3/3] void waveform_t::copy (const **mha_wave_t** * *src*)

5.386.3.22 copy_channel() 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

5.386.3.23 copy_from_at() 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

Parameters

len	Number of frames copied
src	Source
from_pos	Offset in source

5.386.3.24 export_to() `void waveform_t::export_to ( mha_wave_t & dest )`

copy data into allocated **mha_wave_t** (p. 985) structure

Parameters

dest	destination structure

5.386.3.25 limit() `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

5.386.3.26 power() `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)

5.386.3.27 powspec() `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)

5.386.3.28 scale() [1/2] `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

5.386.3.29 scale() [2/2]

```
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

5.386.3.30 scale_channel()

```
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

5.386.3.31 scale_frame()

```
void waveform_t::scale_frame (
const unsigned int & frame,
const  mha_real_t & val )
```

5.386.3.32 get_size()

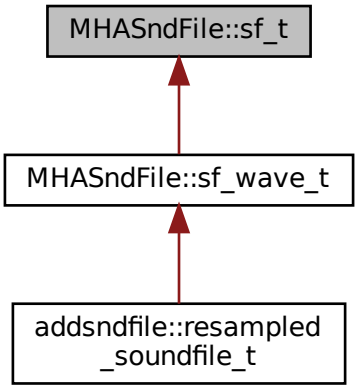
```
unsigned int MHASignal::waveform_t::get_size ( ) const  [inline]
```

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.387 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**

5.387.1 Constructor & Destructor Documentation

5.387.1.1 sf_t() `MHASndFile::sf_t::sf_t (`

`const std::string & fname )`

5.387.1.2 ~sf_t() `MHASndFile::sf_t::~sf_t ( )`

5.387.2 Member Data Documentation

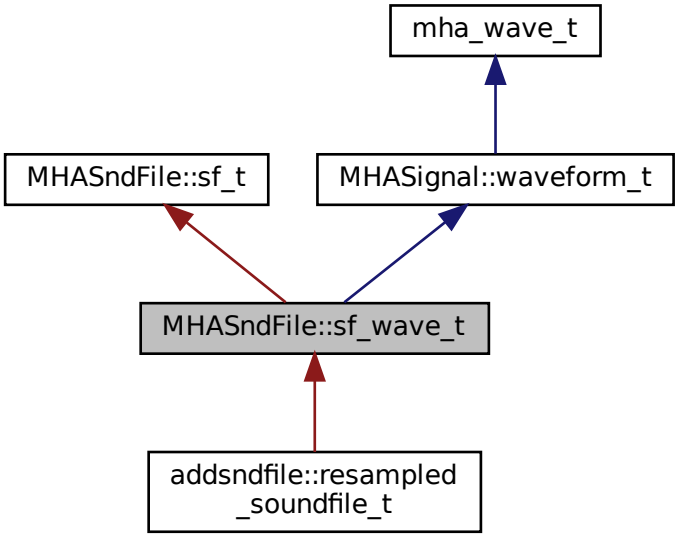
5.387.2.1 sf `SNDFILE* MHASndFile::sf_t::sf`

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

- **mhasndfile.h**
- **mhasndfile.cpp**

5.388 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

5.388.1 Constructor & Destructor Documentation

5.388.1.1 sf_wave_t()

```
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>() )
```

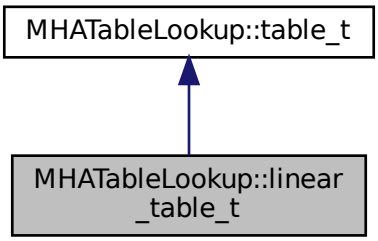
The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

5.389 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. 1430) 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

5.389.1 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.

5.389.2 Constructor & Destructor Documentation

5.389.2.1 linear_table_t() `linear_table_t::linear_table_t ( void )`

contructor creates an empty **linear_table_t** (p. 1430) 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.

5.389.2.2 ~linear_table_t() `linear_table_t::~linear_table_t ( void )`

destructor

5.389.3 Member Function Documentation

5.389.3.1 lookup() `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. 1436).

5.389.3.2 interp() `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. 1436).

5.389.3.3 set_xmin() `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.

5.389.3.4 add_entry() `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.

5.389.3.5 set_xmax() `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;

5.389.3.6 prepare() `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.

5.389.3.7 clear() `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. 1436).

5.389.4 Member Data Documentation

5.389.4.1 vy `mha_real_t* MHATableLookup::linear_table_t::vy [protected]`

5.389.4.2 len `unsigned int MHATableLookup::linear_table_t::len [protected]`

5.389.4.3 vec_y `vector< mha_real_t> MHATableLookup::linear_table_t::vec_y [private]`

5.389.4.4 xmin `mha_real_t MHATableLookup::linear_table_t::xmin [private]`

5.389.4.5 xmax `mha_real_t MHATableLookup::linear_table_t::xmax [private]`

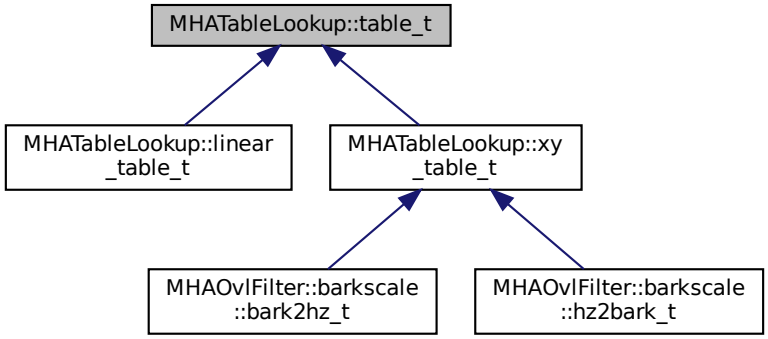
5.389.4.6 scalefac `mha_real_t MHATableLookup::linear_table_t::scalefac [private]`

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

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.390 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

5.390.1 Constructor & Destructor Documentation

5.390.1.1 table_t() `table_t::table_t (void )`

5.390.1.2 ~table_t() `table_t::~table_t (void ) [virtual]`

5.390.2 Member Function Documentation

5.390.2.1 lookup() `virtual mha_real_t MHATableLookup::table_t::lookup (mha_real_t ) const [pure virtual]`

Implemented in **MHATableLookup::xy_table_t** (p. 1439), and **MHATableLookup::linear_table_t** (p. 1432).

5.390.2.2 interp() `virtual mha_real_t MHATableLookup::table_t::interp (mha_real_t ) const [pure virtual]`

Implemented in **MHATableLookup::xy_table_t** (p. 1439), and **MHATableLookup::linear_table_t** (p. 1432).

5.390.2.3 clear() `virtual void MHATableLookup::table_t::clear ( void ) [protected], [pure virtual]`

Implemented in **MHATableLookup::linear_table_t** (p. 1434), and **MHATableLookup::xy_table_t** (p. 1440).

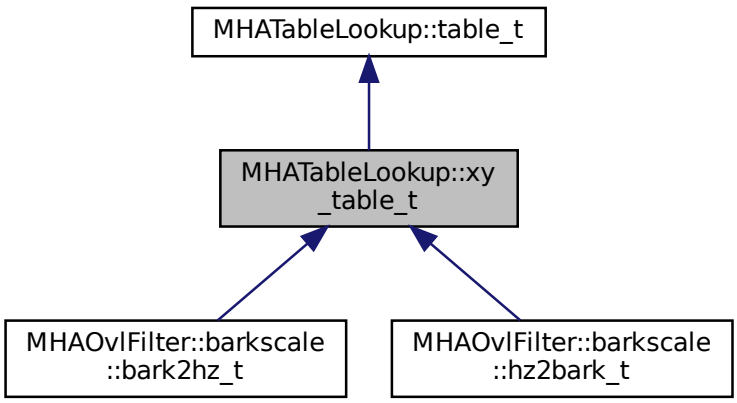
The documentation for this class was generated from the following files:

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.391 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

5.391.1 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. 1439) and **xy_table_t::lookup()** (p. 1439). 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.

5.391.2 Constructor & Destructor Documentation

5.391.2.1 xy_table_t() `xy_table_t::xy_table_t ( )`

5.391.3 Member Function Documentation

5.391.3.1 lookup() `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. 1436).

5.391.3.2 interp() `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. 1436).

5.391.3.3 add_entry() [1/2] 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

5.391.3.4 add_entry() [2/2] 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

5.391.3.5 clear() void xy_table_t::clear (

```
void ) [virtual]
```

Clear the table and transformation functions.

Implements **MHATableLookup::table_t** (p. 1436).

5.391.3.6 set_xfun() void xy_table_t::set_xfun (float(*)(float) *fun*)

Set transformation function for x values.

Parameters

fun	Transformation function.

5.391.3.7 set_yfun() void xy_table_t::set_yfun (float(*)(float) *fun*)

Set transformation function for y values during insertion.

Parameters

fun	Transformation function.

5.391.3.8 set_xyfun() 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.

5.391.3.9 get_xlimits() 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

5.391.4 Member Data Documentation

5.391.4.1 mXY `std::map< mha_real_t, mha_real_t> MHATableLookup::xy_table_t::mXY [private]`

5.391.4.2 xfun `float(* MHATableLookup::xy_table_t::xfun) (float) [private]`

5.391.4.3 yfun `float(* MHATableLookup::xy_table_t::yfun) (float) [private]`

5.391.4.4 xyfun `float(* MHATableLookup::xy_table_t::xyfun) (float, float) [private]`

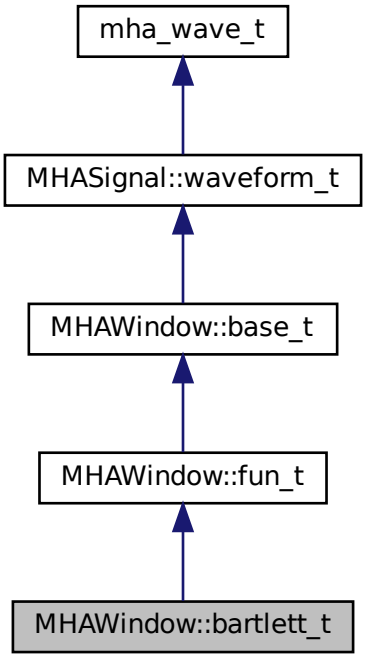
The documentation for this class was generated from the following files:

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

5.392 MHAWindow::bartlett_t Class Reference

Bartlett window.

Inheritance diagram for MHAWindow::bartlett_t:



Public Member Functions

- **bartlett_t** (unsigned int n)

Additional Inherited Members

5.392.1 Detailed Description

Bartlett window.

5.392.2 Constructor & Destructor Documentation

5.392.2.1 bartlett_t() `MHAWindow::bartlett_t::bartlett_t ( unsigned int n ) [inline]`

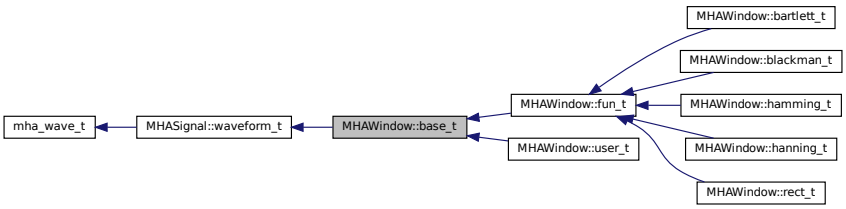
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.393 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

5.393.1 Detailed Description

Common base for window types.

5.393.2 Constructor & Destructor Documentation

5.393.2.1 base_t() [1/2] MHAWindow::base_t::base_t (unsigned int *len*)

Constructor.

Parameters

len	Window length in samples.

5.393.2.2 base_t() [2/2] MHAWindow::base_t::base_t (const **MHAWindow::base_t** & *src*)

Copy constructor.

Parameters

src	Source to be copied

5.393.3 Member Function Documentation

5.393.3.1 operator()() [1/2] void MHAWindow::base_t::operator() (**mha_wave_t** & *s*) const

Apply window to waveform segment (reference)

5.393.3.2 operator()() [2/2] void MHAWindow::base_t::operator() (mha_wave_t * s) const

Apply window to waveform segment (pointer)

5.393.3.3 ramp_begin() void MHAWindow::base_t::ramp_begin (mha_wave_t & s) const

Apply a ramp at the begining.

5.393.3.4 ramp_end() void MHAWindow::base_t::ramp_end (mha_wave_t & s) const

Apply a ramp at the end.

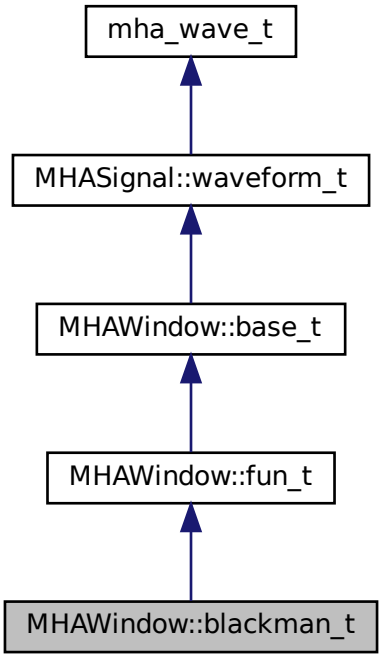
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.394 MHAWindow::blackman_t Class Reference

Blackman window.

Inheritance diagram for MHAWindow::blackman_t:



Public Member Functions

- **blackman_t** (unsigned int n)

Additional Inherited Members

5.394.1 Detailed Description

Blackman window.

5.394.2 Constructor & Destructor Documentation

5.394.2.1 blackman_t() `MHAWindow::blackman_t::blackman_t ( unsigned int n ) [inline]`

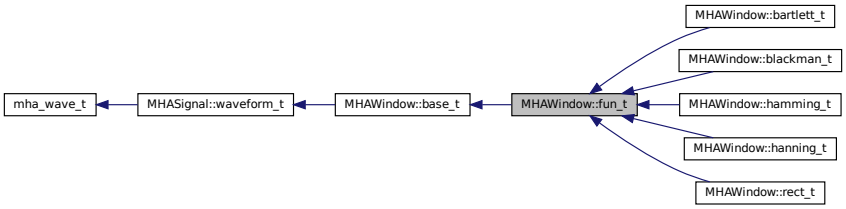
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.395 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

5.395.1 Detailed Description

Generic window based on a generator function.

The generator function should return a valid window function in the interval [-1,1[.

5.395.2 Constructor & Destructor Documentation

5.395.2.1 fun_t()

```
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. 158)
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

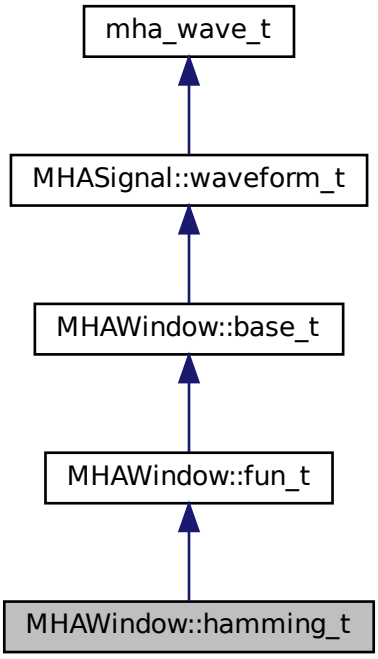
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.396 MHAWindow::hamming_t Class Reference

Hamming window.

Inheritance diagram for MHAWindow::hamming_t:



Public Member Functions

- **hamming_t** (unsigned int n)

Additional Inherited Members

5.396.1 Detailed Description

Hamming window.

5.396.2 Constructor & Destructor Documentation

5.396.2.1 hamming_t() `MHAWindow::hamming_t::hamming_t ( unsigned int n ) [inline]`

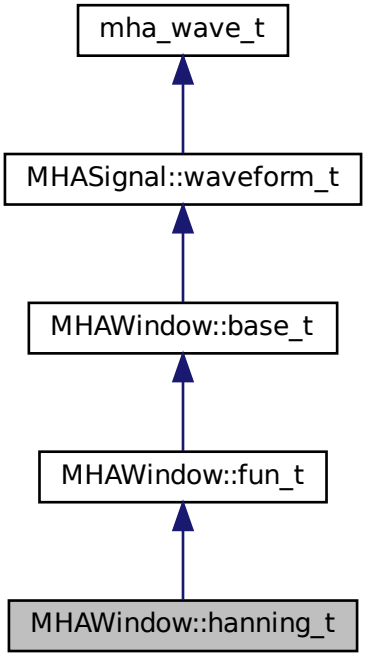
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.397 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

5.397.1 Detailed Description

von-Hann window

5.397.2 Constructor & Destructor Documentation

5.397.2.1 hanning_t() `MHAWindow::hanning_t::hanning_t ( unsigned int n ) [inline]`

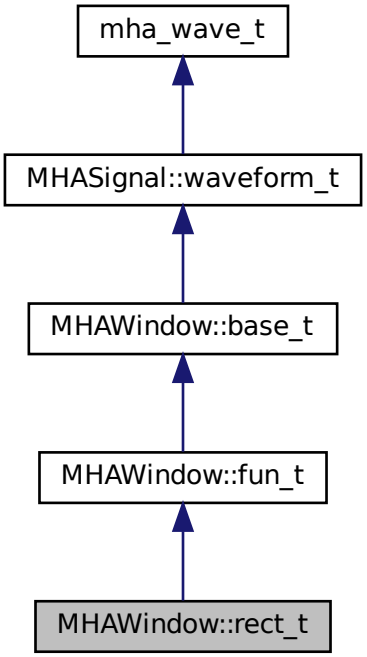
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.398 MHAWindow::rect_t Class Reference

Rectangular window.

Inheritance diagram for MHAWindow::rect_t:



Public Member Functions

- **rect_t** (unsigned int n)

Additional Inherited Members

5.398.1 Detailed Description

Rectangular window.

5.398.2 Constructor & Destructor Documentation

5.398.2.1 rect_t() `MHAWindow::rect_t::rect_t ( unsigned int n ) [inline]`

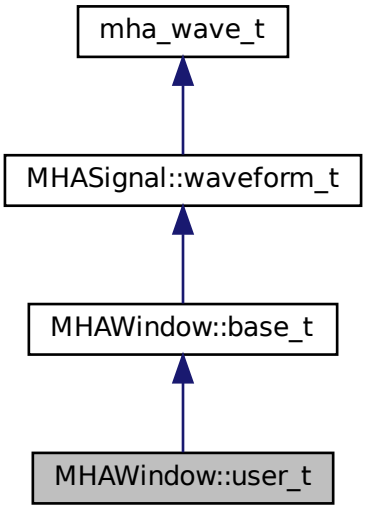
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.399 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

5.399.1 Detailed Description

User defined window.

5.399.2 Constructor & Destructor Documentation

5.399.2.1 user_t() `MHAWindow::user_t::user_t (const std::vector< mha_real_t > & wnd )`

Constructor.

Parameters

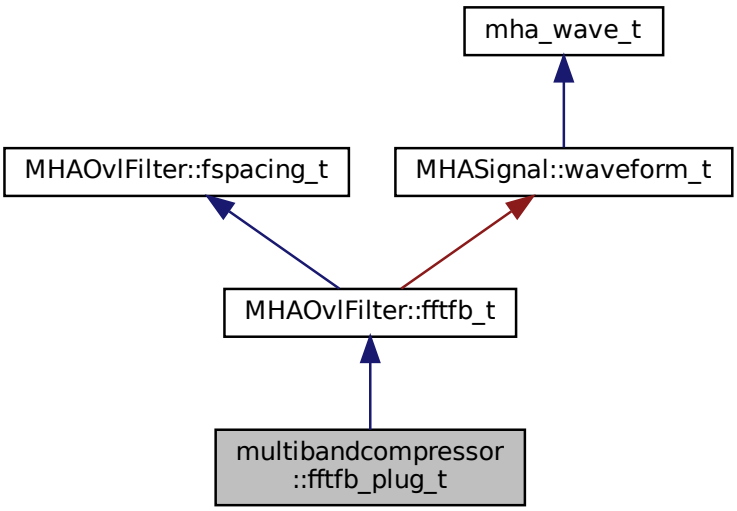
wnd	User defined window

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.400 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, **MHA_AC↩ ::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

5.400.1 Constructor & Destructor Documentation

5.400.1.1 fftfb_plug_t()

```
multibandcompressor::fftfb_plug_t::fftfb_plug_t (
    MHAOvlFilter::fftfb_vars_t & vars,
    const mhaconfig_t & cfg,
    MHA_AC::algo_comm_t & ac,
    std::string alg )
```

5.400.2 Member Function Documentation

5.400.2.1 insert()

```
void multibandcompressor::fftfb_plug_t::insert ( )
```

5.400.3 Member Data Documentation

5.400.3.1 cfv

```
MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::cfv [private]
```

vector of nominal center frequencies / Hz

5.400.3.2 efv

```
MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::efv [private]
```

vector of edge frequencies / Hz

5.400.3.3 bwv

```
MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::bwv [private]
```

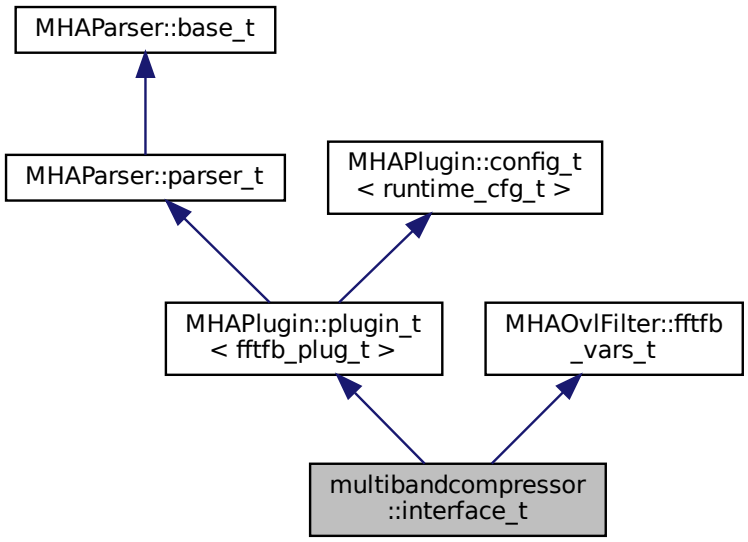
vector of band-weigths (sum of squared fft-bin-weigths)/num_frames

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

- **multibandcompressor.cpp**

5.401 multibandcompressor::interface_t Class Reference

Inheritance diagram for multibandcompressor::interface_t:



Public Member Functions

- **interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHA_AC::int_t 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

5.401.1 Constructor & Destructor Documentation

5.401.1.1 interface_t() `multibandcompressor::interface_t::interface_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Default values are set and MHA configuration variables registered into the parser.

Parameters

*ac_*	algorithm communication handle
th	chain name
al	algorithm name

5.401.2 Member Function Documentation

5.401.2.1 prepare() `void multibandcompressor::interface_t::prepare (`

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 1301).

5.401.2.2 release() `void multibandcompressor::interface_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< fftfb_plug_t >** (p. 1302).

5.401.2.3 process() `mha_spec_t * multibandcompressor::interface_t::process ( mha_spec_t * s )`

5.401.2.4 update_cfg() `void multibandcompressor::interface_t::update_cfg ( void ) [private]`

5.401.3 Member Data Documentation

5.401.3.1 num_channels `MHA_AC::int_t multibandcompressor::interface_t::num_channels [private]`

5.401.3.2 burn `DynComp::dc_afterburn_t multibandcompressor::interface_t::burn [private]`

5.401.3.3 patchbay `MHAEvents::patchbay_t< interface_t> multibandcompressor::interface_t::patchbay [private]`

5.401.3.4 algo `std::string multibandcompressor::interface_t::algo [private]`

5.401.3.5 plug `MHAParser::mhapluginloader_t multibandcompressor::interface_t::plug [private]`

5.401.3.6 plug_sigs `plugin_signals_t* multibandcompressor::interface_t::plug_sigs [private]`

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

- **multibandcompressor.cpp**

5.402 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**

5.402.1 Constructor & Destructor Documentation

5.402.1.1 plugin_signals_t() `multibandcompressor::plugin_signals_t::plugin_signals_t (`

```
unsigned int channels,
unsigned int bands )
```

5.402.2 Member Function Documentation

5.402.2.1 update_levels()

```
void multibandcompressor::plugin_signals_t::update_levels
(
        MHAOvlFilter::fftfb_t * pFb,
        mha_spec_t * s_in )
```

5.402.2.2 apply_gains()

```
void multibandcompressor::plugin_signals_t::apply_gains (
        MHAOvlFilter::fftfb_t * pFb,
        DynComp::dc_afterburn_t & burn,
        mha_spec_t * s_out )
```

5.402.3 Member Data Documentation

5.402.3.1 plug_level

```
MHASignal::waveform_t multibandcompressor::plugin_signals_t↩
::plug_level [private]
```

5.402.3.2 gain

```
MHASignal::waveform_t multibandcompressor::plugin_signals_t::gain
[private]
```

5.402.3.3 plug_output

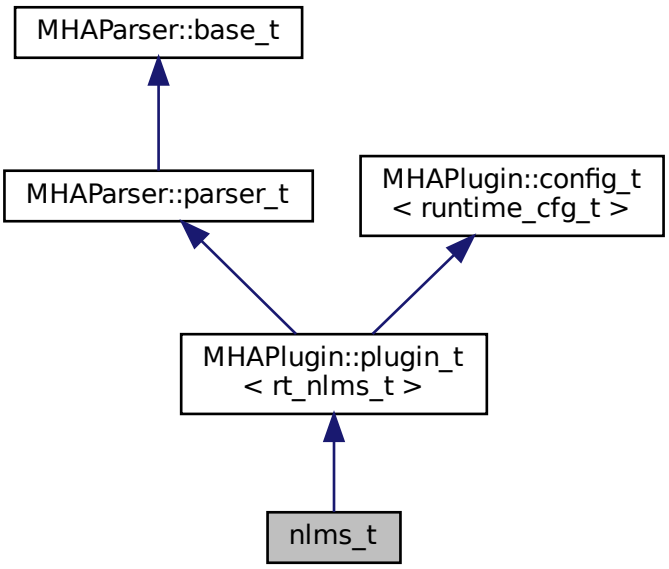
```
mha_wave_t* multibandcompressor::plugin_signals_t::plug_↩
output
```

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

- **multibandcompressor.cpp**

5.403 nlms_t Class Reference

Inheritance diagram for nlms_t:



Public Member Functions

- **nlms_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.403.1 Constructor & Destructor Documentation

5.403.1.1 nlms_t() `nlms_t::nlms_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.403.2 Member Function Documentation

5.403.2.1 prepare() `void nlms_t::prepare (`

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< rt_nlms_t >** (p. 1301).

5.403.2.2 release() `void nlms_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< rt_nlms_t >** (p. 1302).

5.403.2.3 process() `mha_wave_t * nlms_t::process ( mha_wave_t * s )`

5.403.2.4 update() `void nlms_t::update ( ) [private]`

5.403.3 Member Data Documentation

5.403.3.1 rho `MHAParser::float_t nlms_t::rho [private]`

5.403.3.2 c `MHAParser::float_t nlms_t::c [private]`

5.403.3.3 ntaps `MHAParser::int_t nlms_t::ntaps [private]`

5.403.3.4 name_u `MHAParser::string_t nlms_t::name_u [private]`

5.403.3.5 name_d `MHAParser::string_t nlms_t::name_d [private]`

5.403.3.6 normtype `MHAParser::kw_t nlms_t::normtype [private]`

5.403.3.7 estimtype

`MHAParser::kw_t nlms_t::estimtype [private]`

5.403.3.8 lambda_smoothing_power

`MHAParser::float_t nlms_t::lambda_smoothing_power [private]`

5.403.3.9 name_e

`MHAParser::string_t nlms_t::name_e [private]`

5.403.3.10 name_f

`MHAParser::string_t nlms_t::name_f [private]`

5.403.3.11 n_no_update

`MHAParser::int_t nlms_t::n_no_update [private]`

5.403.3.12 algo

`std::string nlms_t::algo [private]`

5.403.3.13 patchbay

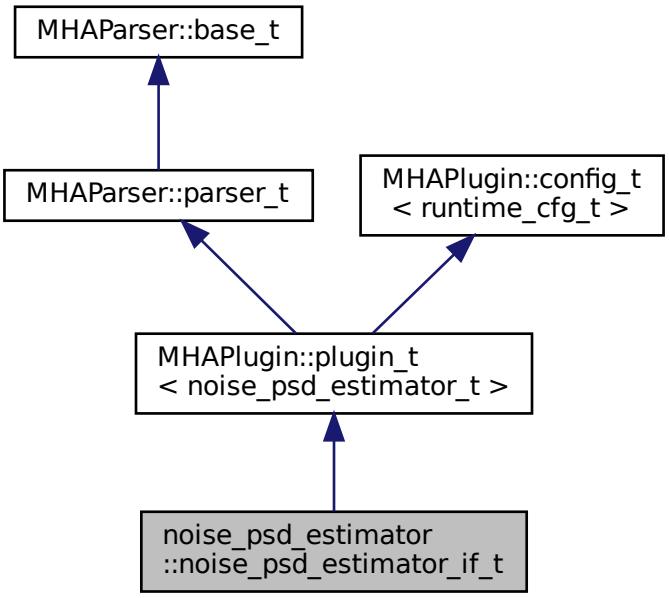
`MHAEvents::patchbay_t< nlms_t> nlms_t::patchbay [private]`

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

- **nlms_wave.cpp**

5.404 noise_psd_estimator::noise_psd_estimator_if_t Class Reference

Inheritance diagram for noise_psd_estimator::noise_psd_estimator_if_t:



Public Member Functions

- **noise_psd_estimator_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _name)
- **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**< **noise_psd_estimator_if_t** > **patchbay**

Additional Inherited Members

5.404.1 Constructor & Destructor Documentation

5.404.1.1 noise_psd_estimator_if_t() noise_psd_estimator::noise_psd_estimator_if_t::noise_psd_estimator_if_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.404.2 Member Function Documentation

5.404.2.1 process() mha_spec_t * noise_psd_estimator::noise_psd_estimator_if_t::process (

```
mha_spec_t * s )
```

5.404.2.2 prepare() void noise_psd_estimator::noise_psd_estimator_if_t::prepare (

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< noise_psd_estimator_t >** (p. 1301).

5.404.2.3 update_cfg() void noise_psd_estimator::noise_psd_estimator_if_t::update_cfg (

```
void ) [private]
```

5.404.3 Member Data Documentation

5.404.3.1 alphaPH1mean

MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::alphaPH1mean [private]

5.404.3.2 alphaPSD

MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::alphaPSD [private]

5.404.3.3 q

MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::q [private]

5.404.3.4 xiOptDb

MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::xiOptDb [private]

5.404.3.5 name

std::string noise_psd_estimator::noise_psd_estimator_if_t::name [private]

5.404.3.6 patchbay

MHAEvents::patchbay_t< **noise_psd_estimator_if_t**> noise_psd_estimator::noise_psd_estimator_if_t::patchbay [private]

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

- **noise_psd_estimator.cpp**

5.405 noise_psd_estimator::noise_psd_estimator_t Class Reference

Public Member Functions

- **noise_psd_estimator_t** (const **mhaconfig_t** &cf, **MHA_AC::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**

5.405.1 Constructor & Destructor Documentation

5.405.1.1 noise_psd_estimator_t() noise_psd_estimator::noise_psd_estimator_t↩ ::noise_psd_estimator_t (

```
const  mhaconfig_t & cf,
 MHA_AC::algo_comm_t & ac,
const std::string & name,
float alphaPH1mean,
float alphaPSD,
float q,
float xiOptDb )
```

5.405.2 Member Function Documentation

5.405.2.1 process() void noise_psd_estimator::noise_psd_estimator_t::process (

```
mha_spec_t * noisyDftFrame )
```

5.405.2.2 insert() void noise_psd_estimator::noise_psd_estimator_t::insert ( ) [inline]

5.405.3 Member Data Documentation

5.405.3.1 noisyPer **MHASignal::waveform_t** noise_psd_estimator::noise_psd_estimator_t::noisyPer [private]

5.405.3.2 PH1mean **MHASignal::waveform_t** noise_psd_estimator::noise_psd_estimator_t::PH1mean [private]

5.405.3.3 noisePow **MHA_AC::waveform_t** noise_psd_estimator::noise_psd_estimator_t::noisePow [private]

5.405.3.4 inputPow **MHA_AC::waveform_t** noise_psd_estimator::noise_psd_estimator_t::inputPow [private]

5.405.3.5 snrPost1Debug **MHA_AC::waveform_t** noise_psd_estimator::noise_psd_estimator_t::snrPost1Debug [private]

5.405.3.6 GLRDebug **MHA_AC::waveform_t** noise_psd_estimator::noise_psd_estimator_t::GLRDebug [private]

5.405.3.7 PH1Debug

MHA_AC::waveform_t noise_psd_estimator::noise_psd_estimator_t::PH1Debug [private]

5.405.3.8 estimateDebug

MHA_AC::waveform_t noise_psd_estimator::noise_psd_estimator_t::estimateDebug [private]

5.405.3.9 inputSpec

MHA_AC::spectrum_t noise_psd_estimator::noise_psd_estimator_t::inputSpec [private]

5.405.3.10 alphaPH1mean_

float noise_psd_estimator::noise_psd_estimator_t::alphaPH1mean_ [private]

5.405.3.11 alphaPSD_

float noise_psd_estimator::noise_psd_estimator_t::alphaPSD_ [private]

5.405.3.12 priorFact

float noise_psd_estimator::noise_psd_estimator_t::priorFact [private]

5.405.3.13 xiOpt

float noise_psd_estimator::noise_psd_estimator_t::xiOpt [private]

5.405.3.14 logGLRFact

float noise_psd_estimator::noise_psd_estimator_t::logGLRFact [private]

5.405.3.15 GLRexp `float noise_psd_estimator::noise_psd_estimator_t::GLRexp [private]`

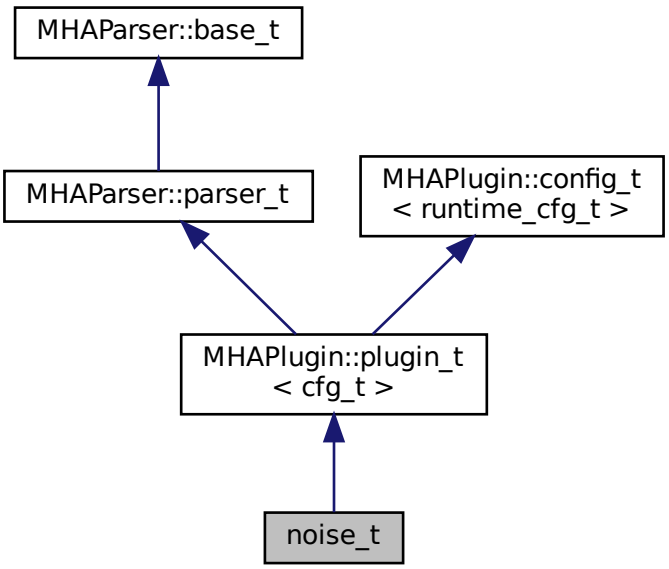
5.405.3.16 frameno `int noise_psd_estimator::noise_psd_estimator_t::frameno [private]`

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

- **noise_psd_estimator.cpp**

5.406 noise_t Class Reference

Inheritance diagram for noise_t:



Public Member Functions

- **noise_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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**
- **MHAParser::int_t seed**
- **MHAEvents::patchbay_t< noise_t > patchbay**

Additional Inherited Members

5.406.1 Constructor & Destructor Documentation

5.406.1.1 noise_t()

```
noise_t::noise_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.406.2 Member Function Documentation

5.406.2.1 process() [1/2]

```
mha_wave_t * noise_t::process (
    mha_wave_t * s )
```

5.406.2.2 process() [2/2]

```
mha_spec_t * noise_t::process (
    mha_spec_t * s )
```

5.406.2.3 prepare()

```
void noise_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.406.2.4 update_cfg() void noise_t::update_cfg (void)

5.406.3 Member Data Documentation

5.406.3.1 lev MHAParser::float_t noise_t::lev [private]

5.406.3.2 mode MHAParser::kw_t noise_t::mode [private]

5.406.3.3 frozennoise_length MHAParser::float_t noise_t::frozennoise_length [private]

5.406.3.4 seed MHAParser::int_t noise_t::seed [private]

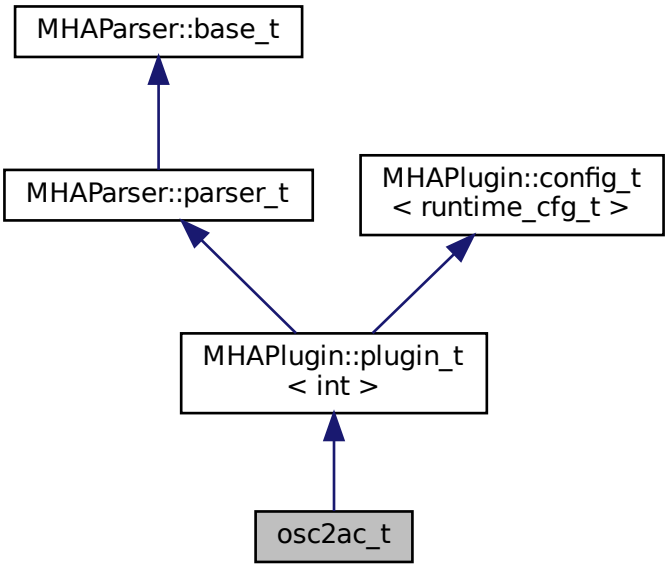
5.406.3.5 patchbay MHAEvents::patchbay_t< noise_t> noise_t::patchbay [private]

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

- **noise.cpp**

5.407 osc2ac_t Class Reference

Inheritance diagram for osc2ac_t:



Public Member Functions

- **osc2ac_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)
- void **process** ()

Private Member Functions

- void **setlock** (bool b)

Private Attributes

- **MHAParser::string_t host**
- **MHAParser::string_t port**
- **MHAParser::vstring_t vars**
- **MHAParser::vint_t size**
- **MHAEvents::patchbay_t**< **osc2ac_t** > **patchbay**
- std::unique_ptr< **osc_server_t** > **srv**

Additional Inherited Members

5.407.1 Constructor & Destructor Documentation

5.407.1.1 osc2ac_t() `osc2ac_t::osc2ac_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.407.2 Member Function Documentation

5.407.2.1 prepare() `void osc2ac_t::prepare (`

```
mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin_t< int >** (p. 1301).

5.407.2.2 release() `void osc2ac_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1302).

5.407.2.3 process() [1/3] `mha_wave_t* osc2ac_t::process (`

```
mha_wave_t * s ) [inline]
```

5.407.2.4 process() [2/3] `mha_spec_t* osc2ac_t::process (`

```
mha_spec_t * s ) [inline]
```

5.407.2.5 process() [3/3] void osc2ac_t::process (void)

5.407.2.6 setlock() void osc2ac_t::setlock (bool *b*) [private]

5.407.3 Member Data Documentation

5.407.3.1 host MHAParser::string_t osc2ac_t::host [private]

5.407.3.2 port MHAParser::string_t osc2ac_t::port [private]

5.407.3.3 vars MHAParser::vstring_t osc2ac_t::vars [private]

5.407.3.4 size MHAParser::vint_t osc2ac_t::size [private]

5.407.3.5 patchbay MHAEvents::patchbay_t< osc2ac_t> osc2ac_t::patchbay [private]

5.407.3.6 srv std::unique_ptr< osc_server_t> osc2ac_t::srv [private]

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

- **osc2ac.cpp**

5.408 osc_server_t Class Reference

OSC receiver implemented using liblo.

Public Member Functions

- **osc_server_t** (const std::string &multicast_addr, const std::string &port)
- ~**osc_server_t** ()
- void **server_stop** ()
- void **server_start** ()
- void **insert_variable** (const std::string &name, unsigned int **size**, **MHA_AC::algo_comm_t** &hAC)
- void **sync_osc2ac** ()
- void **ac_insert** ()

Static Public Member Functions

- static void **error_h** (int num, const char *msg, const char *path)

Private Attributes

- std::vector< std::unique_ptr< **osc_variable_t** > > **pVars**
- lo_server_thread **lost**
- bool **is_running**

5.408.1 Detailed Description

OSC receiver implemented using liblo.

5.408.2 Constructor & Destructor Documentation

5.408.2.1 osc_server_t() `osc_server_t::osc_server_t (`

```
const std::string & multicast_addr,
const std::string & port )
```

5.408.2.2 ~osc_server_t() osc_server_t::~osc_server_t ( )

5.408.3 Member Function Documentation

5.408.3.1 server_stop() void osc_server_t::server_stop ( )

5.408.3.2 server_start() void osc_server_t::server_start ( )

5.408.3.3 insert_variable() void osc_server_t::insert_variable (
const std::string & *name*,
unsigned int *size*,
MHA_AC::algo_comm_t & *hAC*)

5.408.3.4 sync_osc2ac() void osc_server_t::sync_osc2ac ( )

5.408.3.5 ac_insert() void osc_server_t::ac_insert ( )

5.408.3.6 error_h() void osc_server_t::error_h (
int *num*,
const char * *msg*,
const char * *path*) [static]

5.408.4 Member Data Documentation

5.408.4.1 pVars

`std::vector<std::unique_ptr< osc_variable_t> > osc_server_t::pVars [private]`

5.408.4.2 lost

`lo_server_thread osc_server_t::lost [private]`

5.408.4.3 is_running

`bool osc_server_t::is_running [private]`

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

- **osc2ac.cpp**

5.409 osc_variable_t Class Reference

Class for converting messages received at a single osc address to a single AC variable.

Public Member Functions

- **osc_variable_t** (const **osc_variable_t** &)=delete
  An instance of this class cannot safely be copied.
- **osc_variable_t** (const std::string &name, unsigned int **size**, **MHA_AC::algo_comm_t** &hAC, lo_server_thread lost)
  Constructor.
- void **sync_osc2ac** ()
  Copies the latest OSC data from the OSC storage to the AC storage.
- void **ac_insert** ()
  Insert/Re-insert the AC variable into AC space.
- int **handler** (const char *types, lo_arg **argv, int argc)
  Callback function called by network thread managed by liblo when a new OSC message has been received.

Static Public Member Functions

- static int **handler** (const char *path, const char *types, lo_arg **argv, int argc, lo_message msg, void *user_data)
  Callback function called by network thread managed by liblo when a new OSC message has been received.

Private Attributes

- std::string **acname**

  Name of the ac variable.
- std::string **oscaddr**

  OSC address.
- **MHA_AC::waveform_t ac_data**

  AC variable storage.
- **MHASignal::waveform_t osc_data**

  OSC variable storage.
- std::string **name_**

  Name of AC variable and OSC address without the initial slash.

5.409.1 Detailed Description

Class for converting messages received at a single osc address to a single AC variable.

OSC variables are received asynchronously in a network thread and must not modify their AC variables directly, because MHA plugins may only access their AC variables while executing their prepare, release, or process callbacks.

One osc2ac plugin uses multiple instances of **osc_variable_t** (p. 1480), one for each mapping of an OSC address to an AC variable.

5.409.2 Constructor & Destructor Documentation

5.409.2.1 osc_variable_t() [1/2] osc_variable_t::osc_variable_t (const osc_variable_t &) [delete]

An instance of this class cannot safely be copied.

5.409.2.2 osc_variable_t() [2/2] osc_variable_t::osc_variable_t (const std::string & *name*, unsigned int *size*, **MHA_AC::algo_comm_t** & *hAC*, lo_server_thread *lost*)

Constructor.

Allocates memory.

Parameters

name	The name of the AC variable that stores the latest value.
size	Number of elements to copy from OSC message to AC variable.
hAC	Handle of Algorithm Communication Variable space.
lost	libLO Server Thread.

5.409.3 Member Function Documentation

5.409.3.1 sync_osc2ac() void osc_variable_t::sync_osc2ac ( ) [inline]

Copies the latest OSC data from the OSC storage to the AC storage.

To be executed during process callback of osc2ac plugin.

5.409.3.2 ac_insert() void osc_variable_t::ac_insert ( ) [inline]

Insert/Re-insert the AC variable into AC space.

Should be done in each process callback.

5.409.3.3 handler() [1/2] int osc_variable_t::handler (

```
const char * path,
const char * types,
lo_arg ** argv,
int argc,
lo_message msg,
void * user_data ) [static]
```

Callback function called by network thread managed by liblo when a new OSC message has been received.

This static method forwards to the instance method by casting user_data to osc_variable_t*.

Parameters

path	Unused.
types	The OSC data type indicator of the received message.
argv	Array of received OSC data.
argc	Number of elements in array of received OSC data.
msg	Unused.
user_data	Pointer to **osc_variable_t** (p. 1480) instance.

Returns

1 if the message was accepted, 0 if not.

5.409.3.4 handler() [2/2] int osc_variable_t::handler (

```
const char * types,
lo_arg ** argv,
int argc )
```

Callback function called by network thread managed by liblo when a new OSC message has been received.

This instance method checks if the received data is of expected length and contains only floats, and if yes, copies the data into the buffer osc_data where the latest received data for this osc address is stored until it is either overwritten by the next data for the same osc address or copied to an AC variable.

Parameters

types	The OSC data type indicator of the received message.
argv	Array of received OSC data.
argc	Number of elements in array of received OSC data.

Returns

1 if the message had correct length and contained only floats, 0 if not.

5.409.4 Member Data Documentation

5.409.4.1 acname `std::string osc_variable_t::acname [private]`

Name of the ac variable.

5.409.4.2 oscaddr `std::string osc_variable_t::oscaddr [private]`

OSC address.

5.409.4.3 ac_data `MHA_AC::waveform_t osc_variable_t::ac_data [private]`

AC variable storage.

5.409.4.4 osc_data `MHASignal::waveform_t osc_variable_t::osc_data [private]`

OSC variable storage.

5.409.4.5 name_ `std::string osc_variable_t::name_ [private]`

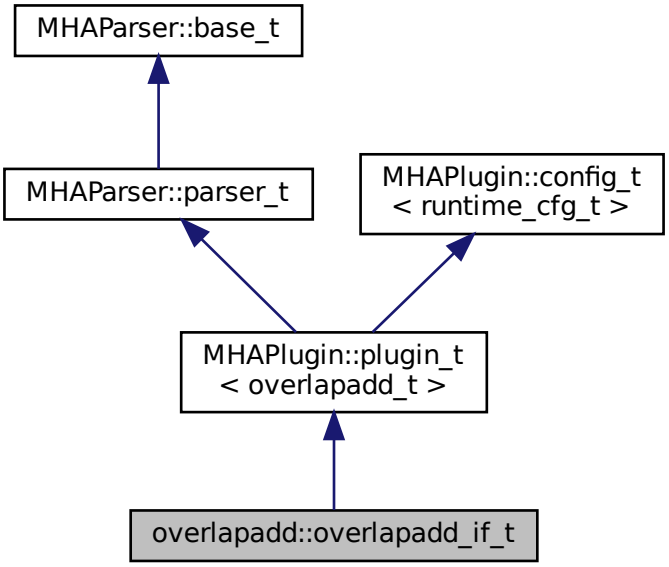
Name of AC variable and OSC address without the initial slash.

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

- **osc2ac.cpp**

5.410 overlapadd::overlapadd_if_t Class Reference

Inheritance diagram for overlapadd::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- ~**overlapadd_if_t** ()=default
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update** ()
- void **setlock** (bool b)

  Lock/Unlock all configuration variables.

Private Attributes

- **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::bool_t strict_window_ratio**

  Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.
- **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

5.410.1 Constructor & Destructor Documentation

5.410.1.1 overlapadd_if_t() `overlapadd::overlapadd_if_t::overlapadd_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.410.1.2 ~overlapadd_if_t() `overlapadd::overlapadd_if_t::~overlapadd_if_t ( ) [default]`

5.410.2 Member Function Documentation

5.410.2.1 prepare() `void overlapadd::overlapadd_if_t::prepare ( mhaconfig_t & t ) [virtual]`

Implements **MHAPlugin::plugin_t< overlapadd_t >** (p. 1301).

5.410.2.2 release() `void overlapadd::overlapadd_if_t::release ( void ) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< overlapadd_t >** (p. 1302).

5.410.2.3 process() `mha_wave_t * overlapadd::overlapadd_if_t::process ( mha_wave_t * wave_in )`

5.410.2.4 update() `void overlapadd::overlapadd_if_t::update ( ) [private]`

5.410.2.5 setlock() `void overlapadd::overlapadd_if_t::setlock ( bool b ) [inline], [private]`

Lock/Unlock all configuration variables.

Parameters

b	Desired lock state

5.410.3 Member Data Documentation

5.410.3.1 nfft `MHAParser::int_t overlapadd::overlapadd_if_t::nfft [private]`

FFT length to be used, zero-padding is FFT length-wndlength.

5.410.3.2 nwnd `MHAParser::int_t overlapadd::overlapadd_if_t::nwnd [private]`

Window length to be used (overlap is 1-fragsize/wndlength)

5.410.3.3 wndpos `MHAParser::float_t overlapadd::overlapadd_if_t::wndpos [private]`

Relative position of zero padding (0 end, 0.5 center, 1 start)

5.410.3.4 window `MHAParser::window_t overlapadd::overlapadd_if_t::window [private]`

5.410.3.5 wndexp `MHAParser::float_t overlapadd::overlapadd_if_t::wndexp [private]`

5.410.3.6 zerowindow `MHAParser::window_t overlapadd::overlapadd_if_t::zerowindow [private]`

5.410.3.7 strict_window_ratio `MHAParser::bool_t overlapadd::overlapadd_if_t::strict_window_ratio [private]`

Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.

5.410.3.8 plugloader

MHAParser::mhapluginloader_t overlapadd::overlapadd_if_t↩ ::plugloader [private]

5.410.3.9 prescale

MHAParser::float_mon_t overlapadd::overlapadd_if_t::prescale [private]

5.410.3.10 postscale

MHAParser::float_mon_t overlapadd::overlapadd_if_t::postscale [private]

5.410.3.11 algo

std::string overlapadd::overlapadd_if_t::algo [private]

5.410.3.12 cf_in

mhaconfig_t overlapadd::overlapadd_if_t::cf_in [private]

5.410.3.13 cf_out

mhaconfig_t overlapadd::overlapadd_if_t::cf_out [private]

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

- **overlapadd.hh**
- **overlapadd.cpp**

5.411 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** ∗ **wave2spec** (**mha_wave_t** ∗)
- **mha_wave_t** ∗ **spec2wave** (**mha_spec_t** ∗)

Private Member Functions

- void **wave2spec_hop_forward** (**mha_wave_t** ∗)
- void **wave2spec_apply_window** (void)
- **mha_spec_t** ∗ **wave2spec_compute_fft** (void)

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**

5.411.1 Constructor & Destructor Documentation

5.411.1.1 overlapadd_t() `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 )
```

5.411.1.2 ∼overlapadd_t() `overlapadd::overlapadd_t::∼overlapadd_t ( )`

5.411.2 Member Function Documentation

5.411.2.1 wave2spec()

```
mha_spec_t * overlapadd::overlapadd_t::wave2spec (
            mha_wave_t * s )
```

5.411.2.2 spec2wave()

```
mha_wave_t * overlapadd::overlapadd_t::spec2wave (
            mha_spec_t * s )
```

5.411.2.3 wave2spec_hop_forward()

```
void overlapadd::overlapadd_t::wave2spec_hop_↩
forward (
            mha_wave_t * s ) [private]
```

5.411.2.4 wave2spec_apply_window()

```
void overlapadd::overlapadd_t::wave2spec_↩
apply_window (
           void ) [private]
```

5.411.2.5 wave2spec_compute_fft()

```
mha_spec_t * overlapadd::overlapadd_t::wave2spec↩
_compute_fft (
           void ) [private]
```

5.411.3 Member Data Documentation

5.411.3.1 fft

```
mha_fft_t overlapadd::overlapadd_t::fft [private]
```

5.411.3.2 prewnd

```
MHAWindow::base_t overlapadd::overlapadd_t::prewnd [private]
```

5.411.3.3 postwnd

MHAWindow::base_t overlapadd::overlapadd_t::postwnd [private]

5.411.3.4 wave_in1

MHASignal::waveform_t overlapadd::overlapadd_t::wave_in1 [private]

5.411.3.5 wave_out1

MHASignal::waveform_t overlapadd::overlapadd_t::wave_out1 [private]

5.411.3.6 spec_in

MHASignal::spectrum_t overlapadd::overlapadd_t::spec_in [private]

5.411.3.7 calc_out

MHASignal::waveform_t overlapadd::overlapadd_t::calc_out [private]

5.411.3.8 out_buf

MHASignal::waveform_t overlapadd::overlapadd_t::out_buf [private]

5.411.3.9 write_buf

MHASignal::waveform_t overlapadd::overlapadd_t::write_buf [private]

5.411.3.10 n_zero

unsigned int overlapadd::overlapadd_t::n_zero [private]

5.411.3.11 n_pad1

unsigned int overlapadd::overlapadd_t::n_pad1 [private]

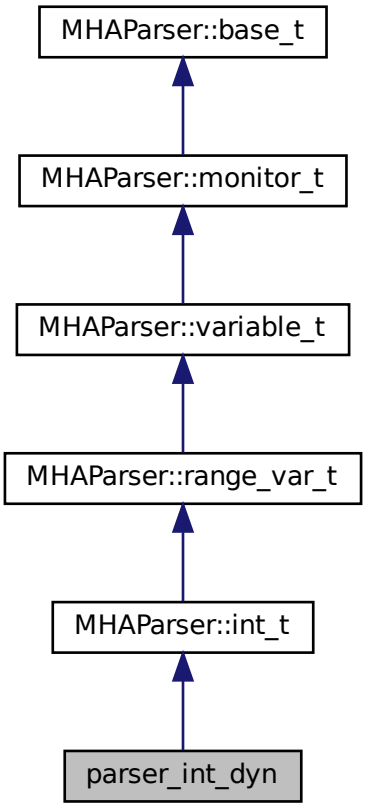
5.411.3.12 n_pad2 `unsigned int overlapadd::overlapadd_t::n_pad2 [private]`

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

- **overlapadd.hh**
- **overlapadd.cpp**

5.412 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

5.412.1 Constructor & Destructor Documentation

5.412.1.1 parser_int_dyn()

```
parser_int_dyn::parser_int_dyn (
    const std::string & help_text,
    const std::string & initial_value,
    const std::string & range ) [inline]
```

5.412.2 Member Function Documentation

5.412.2.1 set_max_angle_ind()

```
void parser_int_dyn::set_max_angle_ind (
    unsigned int max_ind ) [inline]
```

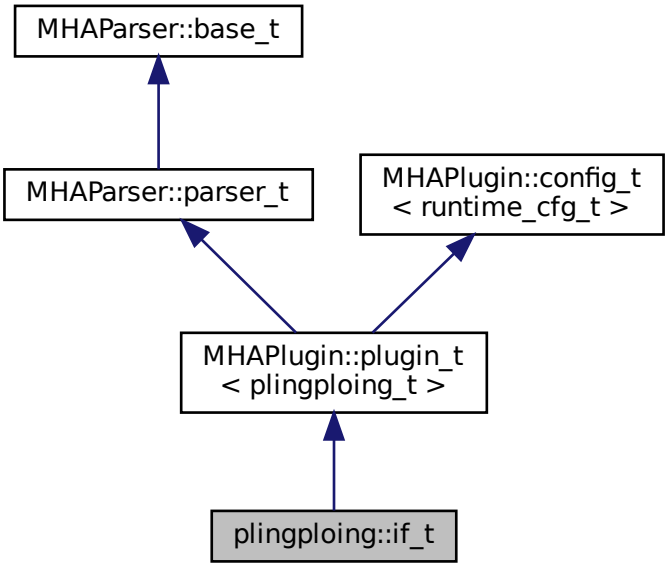
The documentation for this class was generated from the following file:

- **steerbf.h**

5.413 plingploing::if_t Class Reference

Plugin class of the plingploing music generator.

Inheritance diagram for plingploing::if_t:



Public Member Functions

- **if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.413.1 Detailed Description

Plugin class of the plingploing music generator.

5.413.2 Constructor & Destructor Documentation

5.413.2.1 if_t() plingploing::if_t::if_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.413.3 Member Function Documentation

5.413.3.1 process() mha_wave_t * plingploing::if_t::process (mha_wave_t * s)

5.413.3.2 prepare() void plingploing::if_t::prepare (mhaconfig_t & cf) [virtual]

Implements **MHAPlugin::plugin_t< plingploing_t >** (p. 1301).

5.413.3.3 update() void plingploing::if_t::update ( ) [private]

5.413.4 Member Data Documentation

5.413.4.1 patchbay MHAEvents::patchbay_t< if_t> plingploing::if_t::patchbay [private]

5.413.4.2 level `MHAParser::float_t plingploing::if_t::level [private]`

Output level in dB SPL.

5.413.4.3 pitch `MHAParser::float_t plingploing::if_t::pitch [private]`

Bass pitch in Hz.

5.413.4.4 fun1_key `MHAParser::float_t plingploing::if_t::fun1_key [private]`

Key1.

5.413.4.5 fun1_range `MHAParser::float_t plingploing::if_t::fun1_range [private]`

Range1.

5.413.4.6 fun2_key `MHAParser::float_t plingploing::if_t::fun2_key [private]`

Key 2.

5.413.4.7 fun2_range `MHAParser::float_t plingploing::if_t::fun2_range [private]`

Range 2.

5.413.4.8 bpm `MHAParser::float_t plingploing::if_t::bpm [private]`

Speed in beats per minute (bpm)

5.413.4.9 minlen `MHAParser::float_t plingploing::if_t::minlen [private]`

Minimum note length in beats.

5.413.4.10 maxlen `MHAParser::float_t plingploing::if_t::maxlen [private]`

Maximum note length in beats.

5.413.4.11 bassmod `MHAParser::float_t plingploing::if_t::bassmod [private]`

Bass key modulation depth.

5.413.4.12 bassperiod `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**

5.414 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**

5.414.1 Detailed Description

Run-time configuration of the plingploing music generator.

5.414.2 Constructor & Destructor Documentation

5.414.2.1 plingploing_t() `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 )
```

5.414.3 Member Function Documentation

5.414.3.1 process() void plingploing::plingploing_t::process (mha_wave_t * s)

5.414.4 Member Data Documentation

5.414.4.1 cf mhaconfig_t plingploing::plingploing_t::cf [private]

5.414.4.2 pitch_ mha_real_t plingploing::plingploing_t::pitch_ [private]

5.414.4.3 bt unsigned int plingploing::plingploing_t::bt [private]

5.414.4.4 t unsigned int plingploing::plingploing_t::t [private]

5.414.4.5 len unsigned int plingploing::plingploing_t::len [private]

5.414.4.6 dur_ mha_real_t plingploing::plingploing_t::dur_ [private]

5.414.4.7 minlen_ `mha_real_t plingploing::plingploing_t::minlen_ [private]`

5.414.4.8 maxlen_ `mha_real_t plingploing::plingploing_t::maxlen_ [private]`

5.414.4.9 bass `mha_real_t plingploing::plingploing_t::bass [private]`

5.414.4.10 freq `mha_real_t plingploing::plingploing_t::freq [private]`

5.414.4.11 fun1_key `mha_real_t plingploing::plingploing_t::fun1_key [private]`

5.414.4.12 fun1_range `mha_real_t plingploing::plingploing_t::fun1_range [private]`

5.414.4.13 fun1 `mha_real_t plingploing::plingploing_t::fun1 [private]`

5.414.4.14 fun2 `mha_real_t plingploing::plingploing_t::fun2 [private]`

5.414.4.15 fun2_key `mha_real_t plingploing::plingploing_t::fun2_key [private]`

5.414.4.16 fun2_range **mha_real_t** plingploing::plingploing_t::fun2_range [private]

5.414.4.17 dist **mha_real_t** plingploing::plingploing_t::dist [private]

5.414.4.18 dist1 **mha_real_t** plingploing::plingploing_t::dist1 [private]

5.414.4.19 alph **mha_real_t** plingploing::plingploing_t::alph [private]

5.414.4.20 rms **mha_real_t** plingploing::plingploing_t::rms [private]

5.414.4.21 bassmod_ **mha_real_t** plingploing::plingploing_t::bassmod_ [private]

5.414.4.22 bassperiod_ **mha_real_t** plingploing::plingploing_t::bassperiod_ [private]

5.414.4.23 hann1 **MHAWindow::hanning_t** plingploing::plingploing_t::hann1 [private]

5.414.4.24 hann2 **MHAWindow::hanning_t** plingploing::plingploing_t::hann2 [private]

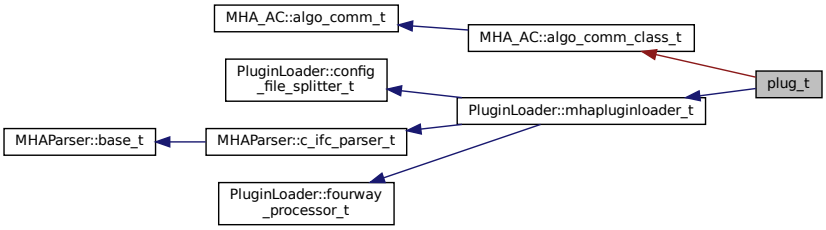
5.414.4.25 level `mha_real_t plingploing::plingploing_t::level [private]`

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

- **plingploing.cpp**

5.415 plug_t Class Reference

Inheritance diagram for plug_t:



Public Member Functions

- **plug_t** (const std::string & **libname**)
- ~**plug_t** () throw ()
- **MHAProc_wave2wave_t get_process_wave** ()
- **MHAProc_wave2spec_t get_process_spec** ()
- void * **get_handle** ()
- **MHA_AC::algo_comm_t** & **get_ac** ()
- void **prepare** (**mhaconfig_t** &) override
- void **release** () override

Additional Inherited Members

5.415.1 Constructor & Destructor Documentation

5.415.1.1 plug_t() `plug_t::plug_t (`
`const std::string & libname )`

5.415.1.2 ~plug_t() `plug_t::~plug_t ( ) throw ( )  [inline]`

5.415.2 Member Function Documentation

5.415.2.1 get_process_wave() `MHAProc_wave2wave_t plug_t::get_process_wave ( )`

5.415.2.2 get_process_spec() `MHAProc_wave2spec_t plug_t::get_process_spec ( )`

5.415.2.3 get_handle() `void * plug_t::get_handle ( )`

5.415.2.4 get_ac() `MHA_AC::algo_comm_t& plug_t::get_ac ( )  [inline]`

5.415.2.5 prepare()

```
void plug_t::prepare (
    mhaconfig_t & signal_dimensions )  [override], [virtual]
```

Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1527).

5.415.2.6 release()

```
void plug_t::release (
    void  )  [override], [virtual]
```

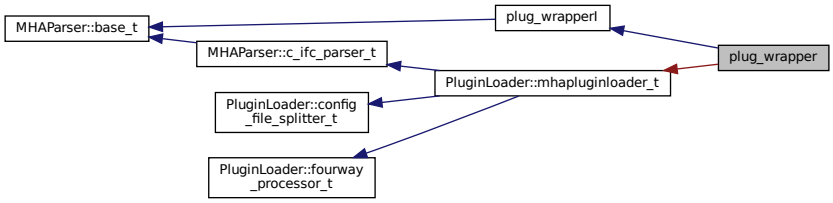
Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1527).

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

- **analysispath.cpp**

5.416 plug_wrapper Class Reference

Inheritance diagram for plug_wrapper:



Public Member Functions

- **plug_wrapper** (**MHA_AC::algo_comm_t** &iac, const std::string & **libname**)
- virtual ∼**plug_wrapper** ()=default
- virtual std::vector< std::string > **get_categories** ()
- virtual std::string **parse** (const std::string &str)
- virtual bool **has_parser** ()
- virtual std::string **get_documentation** ()
- virtual bool **has_process** (**mha_domain_t** in, **mha_domain_t** out)

Additional Inherited Members

5.416.1 Constructor & Destructor Documentation

5.416.1.1 plug_wrapper() `plug_wrapper::plug_wrapper ( MHA_AC::algo_comm_t & iac, const std::string & libname ) [inline]`

5.416.1.2 ∼plug_wrapper() `virtual plug_wrapper::∼plug_wrapper ( ) [virtual], [default]`

5.416.2 Member Function Documentation

5.416.2.1 get_categories() `virtual std::vector<std::string> plug_wrapper::get_categories ( ) [inline], [virtual]`

Implements **plug_wrapperI** (p. 1508).

5.416.2.2 parse() `virtual std::string plug_wrapper::parse ( const std::string & str ) [inline], [virtual]`

Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1526).

5.416.2.3 has_parser() `virtual bool plug_wrapper::has_parser ( ) [inline], [virtual]`

Implements **plug_wrapperI** (p. 1508).

5.416.2.4 get_documentation() `virtual std::string plug_wrapper::get_documentation ( ) [inline], [virtual]`

Implements **plug_wrapperI** (p. 1508).

5.416.2.5 has_process() `virtual bool plug_wrapper::has_process ( mha_domain_t in, mha_domain_t out ) [inline], [virtual]`

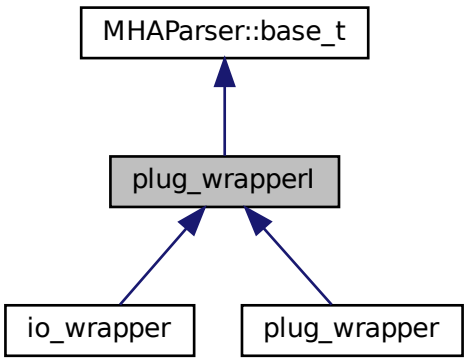
Implements **plug_wrapperI** (p. 1508).

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

- **generatemhaplugindoc.cpp**

5.417 plug_wrapperI Class Reference

Inheritance diagram for plug_wrapperI:



Public Member Functions

- **plug_wrapperI** ()
- virtual ~**plug_wrapperI** ()=default
- virtual std::vector< std::string > **get_categories** ()=0
- virtual std::string **parse** (const std::string &str)=0
- virtual bool **has_parser** ()=0
- virtual std::string **get_documentation** ()=0
- virtual bool **has_process** (**mha_domain_t**, **mha_domain_t**)=0

Additional Inherited Members

5.417.1 Constructor & Destructor Documentation

5.417.1.1 plug_wrapperI() `plug_wrapperI::plug_wrapperI ( ) [inline]`

5.417.1.2 ~plug_wrapperI() `virtual plug_wrapperI::~plug_wrapperI ( ) [virtual], [default]`

5.417.2 Member Function Documentation

5.417.2.1 get_categories() `virtual std::vector<std::string> plug_wrapperI::get_↩ categories ( ) [pure virtual]`

Implemented in **plug_wrapper** (p. 1506), and **io_wrapper** (p. 753).

5.417.2.2 parse() `virtual std::string plug_wrapperI::parse ( const std::string & str ) [pure virtual]`

Reimplemented from **MHAParser::base_t** (p. 1177).

Implemented in **plug_wrapper** (p. 1506), and **io_wrapper** (p. 753).

5.417.2.3 has_parser() `virtual bool plug_wrapperI::has_parser ( ) [pure virtual]`

Implemented in **plug_wrapper** (p. 1506), and **io_wrapper** (p. 753).

5.417.2.4 get_documentation() `virtual std::string plug_wrapperI::get_documentation ( ) [pure virtual]`

Implemented in **plug_wrapper** (p. 1506), and **io_wrapper** (p. 754).

5.417.2.5 has_process() `virtual bool plug_wrapperI::has_process ( mha_domain_t , mha_domain_t ) [pure virtual]`

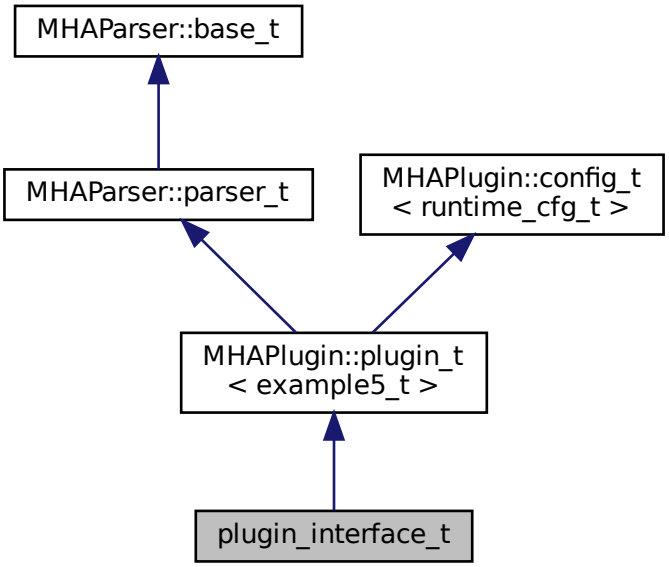
Implemented in **plug_wrapper** (p. 1506), and **io_wrapper** (p. 754).

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

- **generatemhaplugindoc.cpp**

5.418 plugin_interface_t Class Reference

Inheritance diagram for plugin_interface_t:



Public Member Functions

- **plugin_interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
- **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

5.418.1 Constructor & Destructor Documentation

5.418.1.1 plugin_interface_t() `plugin_interface_t::plugin_interface_t (`

```
  MHA_AC::algo_comm_t & iac,
 const std::string & configured_name )
```

5.418.2 Member Function Documentation

5.418.2.1 process() `mha_spec_t * plugin_interface_t::process (`

```
  mha_spec_t * spec )
```

5.418.2.2 prepare() `void plugin_interface_t::prepare (`

```
  mhaconfig_t & tfcfg ) [virtual]
```

Implements **MHAPlugin::plugin_t< example5_t >** (p. 1301).

5.418.2.3 update_cfg() `void plugin_interface_t::update_cfg ( ) [private]`

5.418.3 Member Data Documentation

5.418.3.1 scale_ch `MHAParser::int_t plugin_interface_t::scale_ch [private]`

5.418.3.2 factor `MHAParser::float_t plugin_interface_t::factor [private]`

5.418.3.3 patchbay `MHAEvents::patchbay_t< plugin_interface_t> plugin_interface_t::patchbay [private]`

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

- **example5.cpp**

5.419 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**

5.419.1 Constructor & Destructor Documentation

5.419.1.1 pluginbrowser_t() `pluginbrowser_t::pluginbrowser_t ( )`

5.419.2 Member Function Documentation

5.419.2.1 get_paths() void pluginbrowser_t::get_paths ( )

5.419.2.2 scan_plugin() **plugindescription_t** pluginbrowser_t::scan_plugin (const std::string & *name*)

5.419.2.3 add_plugins() void pluginbrowser_t::add_plugins ( )

5.419.2.4 clear_plugins() void pluginbrowser_t::clear_plugins ( )

5.419.2.5 scan_plugins() void pluginbrowser_t::scan_plugins ( )

5.419.2.6 add_plugin() void pluginbrowser_t::add_plugin (const std::string & *name*)

5.419.2.7 get_plugins() std::list< **plugindescription_t**> pluginbrowser_t::get_↩ plugins ( ) const [inline]

5.419.3 Member Data Documentation

5.419.3.1 plugin_extension `std::string pluginbrowser_t::plugin_extension [private]`

5.419.3.2 library_paths `std::list<std::string> pluginbrowser_t::library_paths [private]`

5.419.3.3 plugins `std::list< plugindescription_t> pluginbrowser_t::plugins [private]`

5.419.3.4 p `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**

5.420 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**

5.420.1 Member Data Documentation

5.420.1.1 name `std::string plugindescription_t::name`

5.420.1.2 fullname `std::string plugindescription_t::fullname`

5.420.1.3 documentation `std::string plugindescription_t::documentation`

5.420.1.4 categories `std::vector<std::string> plugindescription_t::categories`

5.420.1.5 wave2wave `bool plugindescription_t::wave2wave`

5.420.1.6 wave2spec `bool plugindescription_t::wave2spec`

5.420.1.7 spec2wave `bool plugindescription_t::spec2wave`

5.420.1.8 spec2spec `bool plugindescription_t::spec2spec`

5.420.1.9 query_cmds `std::vector<std::string> plugindescription_t::query_cmds`

5.420.1.10 queries `std::map<std::string,std::string> plugindescription_t::queries`

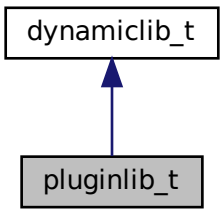
The documentation for this class was generated from the following file:

- **pluginbrowser.h**

5.421 pluginlib_t Class Reference

Specialisation of **dynamiclib_t** (p. 531) for mha plugin libraries.

Inheritance diagram for pluginlib_t:



Public Member Functions

- **pluginlib_t** (const std::string &name_)
  C'tor of the wrapper class.
- virtual void * **resolve** (const std::string &name_) override
  Resolves the plugin callback specified by name_, e.g.
- virtual ~**pluginlib_t** ()
  D'tor.

Additional Inherited Members

5.421.1 Detailed Description

Specialisation of **dynamiclib_t** (p. 531) for mha plugin libraries.

5.421.2 Constructor & Destructor Documentation

5.421.2.1 pluginlib_t() `pluginlib_t::pluginlib_t ( const std::string & name_ ) [explicit]`

C'tor of the wrapper class.

Takes a the the file name of a shared libary w/o the suffix as argument, searches for the library in the system-dependent standard paths for libraries and in MHA_LIBRARY_DIR. Calls load_lib for the actual work.

Parameters

*name_*	File name of the shared library, without suffix

5.421.2.2 ~pluginlib_t() `pluginlib_t::~pluginlib_t ( ) [virtual]`

D'tor.

5.421.3 Member Function Documentation

5.421.3.1 resolve() `void * pluginlib_t::resolve ( const std::string & name_ ) [override], [virtual]`

Resolves the plugin callback specified by name_, e.g.

'process', 'prepare', etc... and returns a pointer to it or a nullptr if the function was not found. Automatically adds the required prefixes and suffixes for dynamically/statically compiled mha plugins

Parameters

*name_*	Name of the function to be resolved

Returns

Pointer to the function

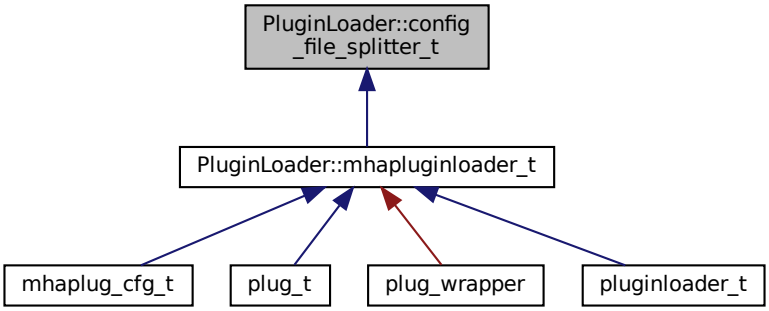
Reimplemented from **dynamiclib_t** (p. 533).

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

- **mha_os.h**
- **mha_os.cpp**

5.422 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**

5.422.1 Constructor & Destructor Documentation

5.422.1.1 config_file_splitter_t() `PluginLoader::config_file_splitter_t::config_file_splitter_t (`

```
            const std::string & name )
```

5.422.2 Member Function Documentation

5.422.2.1 get_configname() const std::string& PluginLoader::config_file_splitter_t::get_configname ( ) const [inline]

5.422.2.2 get_libname() const std::string& PluginLoader::config_file_splitter_t::get_libname ( ) const [inline]

5.422.2.3 get_origname() const std::string& PluginLoader::config_file_splitter_t::get_origname ( ) const [inline]

5.422.2.4 get_configfile() const std::string& PluginLoader::config_file_splitter_t::get_configfile ( ) const [inline]

5.422.3 Member Data Documentation

5.422.3.1 libname std::string PluginLoader::config_file_splitter_t::libname [private]

5.422.3.2 configname std::string PluginLoader::config_file_splitter_t::configname [private]

5.422.3.3 origname `std::string PluginLoader::config_file_splitter_t::origname [private]`

5.422.3.4 configfile `std::string PluginLoader::config_file_splitter_t::configfile [private]`

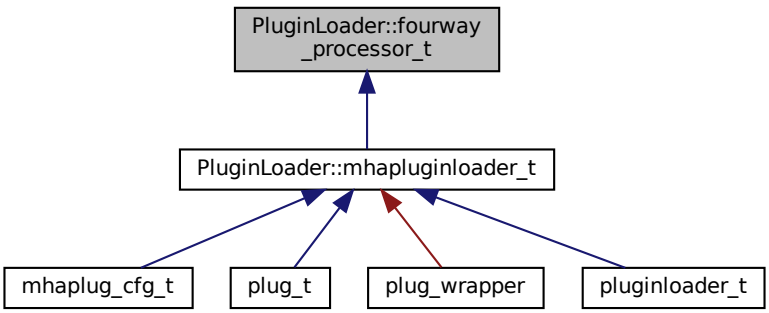
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.423 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. 1522) *are released here in **fourway_processor_t::release*** (p. 1523).

- virtual std::string **parse** (const std::string &query)=0

  Parser interface.

- virtual ~**fourway_processor_t** ()

  Classes with virtual methods need virtual destructor.

5.423.1 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.

5.423.2 Constructor & Destructor Documentation

5.423.2.1 ~**fourway_processor_t()** `virtual PluginLoader::fourway_processor_t::~fourway_processor_t ( ) [inline], [virtual]`

Classes with virtual methods need virtual destructor.

This destructor is empty.

5.423.3 Member Function Documentation

5.423.3.1 process() [1/4]

```
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. 1527).

5.423.3.2 process() [2/4]

```
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. 1528).

5.423.3.3 process() [3/4]

```
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. 1528).

5.423.3.4 process() [4/4]

```
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. 1528).

5.423.3.5 prepare()

```
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. 1522), settings reflects domain and dimensions of the input signal. When **fourway_processor_t::prepare** (p. 1522) returns, settings reflects domain and dimensions of the output signal.

Implemented in **plug_t** (p. 1504), **mhaplug_cfg_t** (p. 1289), and **PluginLoader↩ ::mhapluginloader_t** (p. 1527).

5.423.3.6 release() `virtual void PluginLoader::fourway_processor_t::release ( ) [pure virtual]`

Resources allocated for signal processing in **fourway_processor_t::prepare** (p. 1522) are released here in **fourway_processor_t::release** (p. 1523).

Implemented in **plug_t** (p. 1504), **mhaplug_cfg_t** (p. 1289), and **PluginLoader↩ ::mhapluginloader_t** (p. 1527).

5.423.3.7 parse() `virtual std::string PluginLoader::fourway_processor_t::parse ( const std::string & query ) [pure virtual]`

Parser interface.

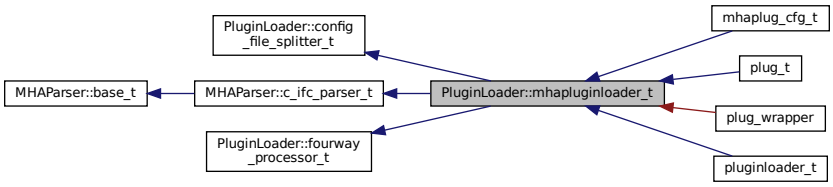
Implemented in **PluginLoader::mhapluginloader_t** (p. 1526), and **plug_wrapper** (p. 1506).

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

- **mhapluginloader.h**

5.424 PluginLoader::mhapluginloader_t Class Reference

Inheritance diagram for PluginLoader::mhapluginloader_t:



Public Member Functions

- std::string **parse** (const std::string &str) override
- **mhapluginloader_t** (**MHA_AC::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** &) override
- void **release** () override
- void **process** (**mha_wave_t** ∗, **mha_wave_t** ∗∗) override
- void **process** (**mha_spec_t** ∗, **mha_spec_t** ∗∗) override
- void **process** (**mha_wave_t** ∗, **mha_spec_t** ∗∗) override
- void **process** (**mha_spec_t** ∗, **mha_wave_t** ∗∗) override
- 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**
- **MHA_AC::algo_comm_t** & **ac**
- **pluginlib_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**
- **MHASetcpp_t MHASetcpp_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

5.424.1 Constructor & Destructor Documentation

5.424.1.1 mhapluginloader_t()

```
PluginLoader::mhapluginloader_t::mhapluginloader_t (
    MHA_AC::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)

Parameters

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.

5.424.1.2 ~mhapluginloader_t() `PluginLoader::mhapluginloader_t::~mhapluginloader_t ( ) throw ( )`

5.424.2 Member Function Documentation

5.424.2.1 parse() `std::string PluginLoader::mhapluginloader_t::parse ( const std::string & str ) [override], [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 1523).

Reimplemented in **plug_wrapper** (p. 1506).

5.424.2.2 has_process()

```
bool PluginLoader::mhapluginloader_t::has_process (
    mha_domain_t in,
    mha_domain_t out ) const
```

5.424.2.3 has_parser()

```
bool PluginLoader::mhapluginloader_t::has_parser ( ) const
```

5.424.2.4 input_domain()

```
mha_domain_t PluginLoader::mhapluginloader_t::input_domain ( ) const
```

5.424.2.5 output_domain()

```
mha_domain_t PluginLoader::mhapluginloader_t::output_domain ( ) const
```

5.424.2.6 prepare()

```
void PluginLoader::mhapluginloader_t::prepare (
    mhaconfig_t & tf ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1522).

Reimplemented in **plug_t** (p. 1504), and **mhaplug_cfg_t** (p. 1289).

5.424.2.7 release()

```
void PluginLoader::mhapluginloader_t::release ( ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1523).

Reimplemented in **plug_t** (p. 1504), and **mhaplug_cfg_t** (p. 1289).

5.424.2.8 process() [1/4]

```
void PluginLoader::mhapluginloader_t::process (
    mha_wave_t * s_in,
    mha_wave_t ** s_out ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1520).

5.424.2.9 process() [2/4]

```
void PluginLoader::mhapluginloader_t::process (
    mha_spec_t * s_in,
    mha_spec_t ** s_out ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1521).

5.424.2.10 process() [3/4]

```
void PluginLoader::mhapluginloader_t::process (
    mha_wave_t * s_in,
    mha_spec_t ** s_out ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1521).

5.424.2.11 process() [4/4]

```
void PluginLoader::mhapluginloader_t::process (
    mha_spec_t * s_in,
    mha_wave_t ** s_out ) [override], [virtual]
```

Implements **PluginLoader::fourway_processor_t** (p. 1522).

5.424.2.12 getfullname()

```
std::string PluginLoader::mhapluginloader_t::getfullname (
) const [inline]
```

5.424.2.13 get_documentation()

```
std::string PluginLoader::mhapluginloader_t::get_↩
documentation ( ) const [inline]
```

5.424.2.14 get_categories()

```
std::vector<std::string> PluginLoader::mhapluginloader_t::get_categories ( ) const [inline]
```

5.424.2.15 is_prepared()

```
bool PluginLoader::mhapluginloader_t::is_prepared ( ) const [inline]
```

5.424.2.16 test_error()

```
void PluginLoader::mhapluginloader_t::test_error ( ) [protected]
```

5.424.2.17 test_version()

```
void PluginLoader::mhapluginloader_t::test_version ( ) [protected]
```

5.424.2.18 mha_test_struct_size()

```
void PluginLoader::mhapluginloader_t::mha_test_struct_size (
            unsigned int s ) [protected]
```

5.424.2.19 resolve_and_init()

```
void PluginLoader::mhapluginloader_t::resolve_and_init ( ) [protected]
```

5.424.3 Member Data Documentation

5.424.3.1 lib_err

```
int PluginLoader::mhapluginloader_t::lib_err [protected]
```

5.424.3.2 ac

```
MHA_AC::algo_comm_t& PluginLoader::mhapluginloader_t::ac [protected]
```

5.424.3.3 lib_handle **pluginlib_t** PluginLoader::mhapluginloader_t::lib_handle [protected]

5.424.3.4 lib_data void* PluginLoader::mhapluginloader_t::lib_data [protected]

5.424.3.5 MHAGetVersion_cb **MHAGetVersion_t** PluginLoader::mhapluginloader_t::↩ MHAGetVersion_cb [protected]

5.424.3.6 MHAInit_cb **MHAInit_t** PluginLoader::mhapluginloader_t::MHAInit_cb [protected]

5.424.3.7 MHADestroy_cb **MHADestroy_t** PluginLoader::mhapluginloader_t::MHADestroy↩ _cb [protected]

5.424.3.8 MHAPrepare_cb **MHAPrepare_t** PluginLoader::mhapluginloader_t::MHAPrepare↩ _cb [protected]

5.424.3.9 MHARelease_cb **MHARelease_t** PluginLoader::mhapluginloader_t::MHARelease↩ _cb [protected]

5.424.3.10 MHAProc_wave2wave_cb **MHAProc_wave2wave_t** PluginLoader::mhapluginloader↩ _t::MHAProc_wave2wave_cb [protected]

5.424.3.11 MHAProc_spec2spec_cb

MHAProc_spec2spec_t PluginLoader::mhapluginloader_t::MHAProc_spec2spec_cb [protected]

5.424.3.12 MHAProc_wave2spec_cb

MHAProc_wave2spec_t PluginLoader::mhapluginloader_t::MHAProc_wave2spec_cb [protected]

5.424.3.13 MHAProc_spec2wave_cb

MHAProc_spec2wave_t PluginLoader::mhapluginloader_t::MHAProc_spec2wave_cb [protected]

5.424.3.14 MHASet_cb

MHASet_t PluginLoader::mhapluginloader_t::MHASet_cb [protected]

5.424.3.15 MHASetcpp_cb

MHASetcpp_t PluginLoader::mhapluginloader_t::MHASetcpp_cb [protected]

5.424.3.16 MHAStrError_cb

MHAStrError_t PluginLoader::mhapluginloader_t::MHAStrError_cb [protected]

5.424.3.17 cf_input

mhaconfig_t PluginLoader::mhapluginloader_t::cf_input [protected]

5.424.3.18 cf_output

mhaconfig_t PluginLoader::mhapluginloader_t::cf_output [protected]

5.424.3.19 plugin_documentation `std::string PluginLoader::mhapluginloader_t::plugin_documentation [protected]`

5.424.3.20 plugin_categories `std::vector<std::string> PluginLoader::mhapluginloader_t::plugin_categories [protected]`

5.424.3.21 b_check_version `bool PluginLoader::mhapluginloader_t::b_check_version [protected]`

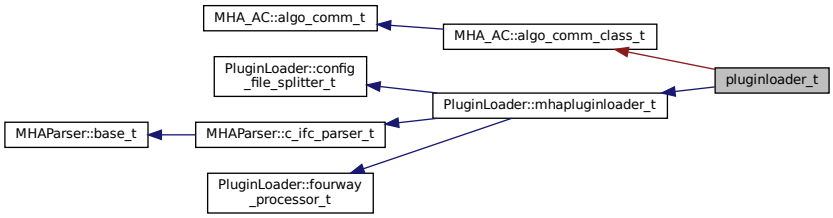
5.424.3.22 b_is_prepared `bool PluginLoader::mhapluginloader_t::b_is_prepared [protected]`

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

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.425 pluginloader_t Class Reference

Inheritance diagram for pluginloader_t:



Public Member Functions

- **pluginloader_t** (const std::string &name)
- ~**pluginloader_t** () throw ()

Additional Inherited Members

5.425.1 Constructor & Destructor Documentation

5.425.1.1 pluginloader_t() `pluginloader_t::pluginloader_t ( const std::string & name )`

5.425.1.2 ~pluginloader_t() `pluginloader_t::~pluginloader_t ( ) throw ( )`

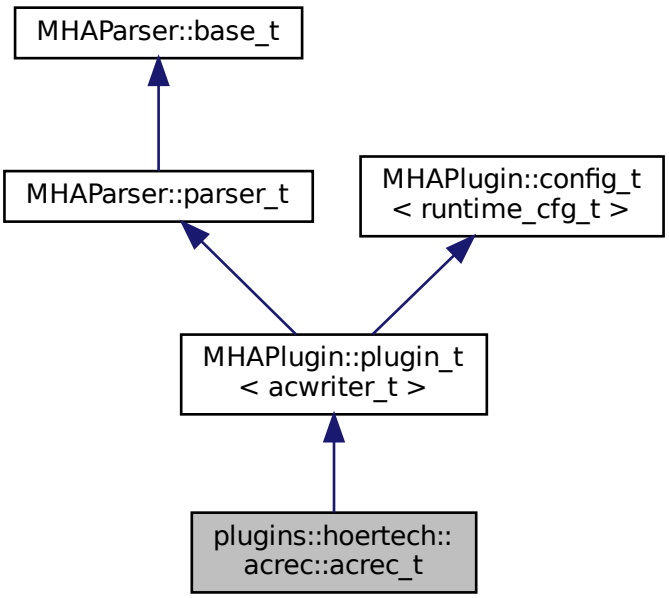
The documentation for this class was generated from the following files:

- **pluginbrowser.h**
- **pluginbrowser.cpp**

5.426 plugins::hoertech::acrec::acrec_t Class Reference

Plugin interface class of plugin acrec.

Inheritance diagram for plugins::hoertech::acrec::acrec_t:



Public Member Functions

- template<class mha_signal_t >
  mha_signal_t * **process** (mha_signal_t *s)
  Process callback.
- void **prepare** (**mhaconfig_t** &cf)
  Prepare callback.
- void **release** ()
  Ensure recorded data is flushed to disk.
- **acrec_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Plugin interface constructor.

Private Member Functions

- void **start_new_session** ()
  Configuration callback called whenever configuaration variable "record" is written to.

Private Attributes

- **MHAParser::bool_t record**
- **MHAParser::int_t fifolen**
- **MHAParser::int_t minwrite**
- **MHAParser::string_t prefix**
- **MHAParser::string_t varname**
- **MHAParser::bool_t use_date**
- **MHAEvents::patchbay_t< acrec_t > patchbay**
- **MHA_AC::comm_var_t cv**
- **MHA_AC::algo_comm_t** & **ac**

Additional Inherited Members

5.426.1 Detailed Description

Plugin interface class of plugin acrec.

5.426.2 Constructor & Destructor Documentation

5.426.2.1 acrec_t() acrec_t::acrec_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

Plugin interface constructor.

Parameters

iac	Algorithm communication variable space.

5.426.3 Member Function Documentation

5.426.3.1 process() template<class mha_signal_t > mha_signal_t * acrec_t::process (mha_signal_t * s)

Process callback.

Pushes the data from one AC variable into the fifo.

Returns

the unmodified input signal.

Parameters

s	input signal. The audio signal is not used or modified.

5.426.3.2 prepare() `void acrec_t::prepare ( mhaconfig_t & cf ) [virtual]`

Prepare callback.

acrec does not modify the signal parameters.

Parameters

cf	The signal parameters.

Implements **MHAPlugin::plugin_t< acwriter_t >** (p. 1301).

5.426.3.3 release() `void acrec_t::release ( void ) [virtual]`

Ensure recorded data is flushed to disk.

Reimplemented from **MHAPlugin::plugin_t< acwriter_t >** (p. 1302).

5.426.3.4 start_new_session() `void acrec_t::start_new_session ( ) [private]`

Configuration callback called whenever configuaration variable "record" is written to.

5.426.4 Member Data Documentation

5.426.4.1 record `MHAParser::bool_t plugins::hoertech::acrec::acrec_t::record [private]`

5.426.4.2 fifolen `MHAParser::int_t plugins::hoertech::acrec::acrec_t::fifolen [private]`

5.426.4.3 minwrite

MHAParser::int_t plugins::hoertech::acrec::acrec_t::minwrite [private]

5.426.4.4 prefix

MHAParser::string_t plugins::hoertech::acrec::acrec_t::prefix [private]

5.426.4.5 varname

MHAParser::string_t plugins::hoertech::acrec::acrec_t::varname [private]

5.426.4.6 use_date

MHAParser::bool_t plugins::hoertech::acrec::acrec_t::use_date [private]

5.426.4.7 patchbay

MHAEvents::patchbay_t< **acrec_t**> plugins::hoertech::acrec↩ ::acrec_t::patchbay [private]

5.426.4.8 cv

MHA_AC::comm_var_t plugins::hoertech::acrec::acrec_t::cv [private]

5.426.4.9 ac

MHA_AC::algo_comm_t& plugins::hoertech::acrec::acrec_t::ac [private]

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

- **acrec.hh**
- **acrec.cpp**

5.427 plugins::hoertech::acrec::acwriter_t Class Reference

acwriter_t (p. 1537) decouples signal processing from writing to disk.

Public Types

- typedef double **output_type**
  The numeric data type used for outputting the data to disk.

Public Member Functions

- **acwriter_t** (bool **active**, unsigned fifosize, unsigned minwrite, const std::string &prefix, bool use_date, const std::string & **varname**)
  Constructor allocates fifo and disk output buffer.
- ∼**acwriter_t** ()=default
  Deallocates memory but does not terminate the write_thread.
- void **process** (**MHA_AC::comm_var_t** ∗)
  Place the data present in the algorithm communication variable into the fifo for output to disk.
- void **exit_request** ()
  Terminate output thread.
- const std::string & **get_varname** () const
  getter for ac variable name

Private Member Functions

- void **write_thread** ()
  Main method of the disk writer thread.
- void **create_datafile** (const std::string &prefix, bool use_date)
  Open data file for output.

Private Attributes

- std::atomic< bool > **close_session**
  cross-thread-synchronization.
- const bool **active**
  The writer thread and the output file will only be created when active is true.
- std::unique_ptr< **mha_fifo_lf_t**< **output_type** > > **fifo**
  Fifo for decoupling signal processing thread from disk writer thread.
- const unsigned int **disk_write_threshold_min_num_samples**
  Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.
- std::thread **writethread**
  The thread that writes to disk.
- std::unique_ptr< **output_type**[ ]> **diskbuffer**
  Intermediate buffer to receive data from fifo and store on disk.
- std::fstream **outfile**
  Ouput file.

- unsigned **num_channels** = 0U

  Number of channels of AC variable using stride.
- bool **is_num_channels_known** = false

  The number of channels is determined during the first process callback.
- bool **is_complex** = false

  If the AC variable is of complex valued type or not.
- const std::string **varname**

  The name of the ac variable to publish.

5.427.1 Detailed Description

acwriter_t (p. 1537) decouples signal processing from writing to disk.

Data arriving in numeric AC variables is converted to data type double, placed into a fifo pipeline to transport the data from the signal processing thread to the disk writing thread, and finally written to disk in chunks of at least minwrite numbers. All numbers are written to disk as binary doubles (8 bytes) in host byte order.

5.427.2 Member Typedef Documentation

5.427.2.1 output_type `typedef double` **plugins::hoertech::acrec::acwriter_t::output_type**

The numeric data type used for outputting the data to disk.

5.427.3 Constructor & Destructor Documentation

5.427.3.1 acwriter_t() `acwriter_t::acwriter_t (`

```
bool active,
unsigned fifosize,
unsigned minwrite,
const std::string & prefix,
bool use_date,
const std::string & varname )
```

Constructor allocates fifo and disk output buffer.

It spawns a new thread for writing data to disk when active==true. In order to terminate the thread, method exit_request **must** be called before this object is destroyed.

Parameters

active	Only write data to disk when this is true.
fifosize	Capacity of both the fifo pipeline and of the disk buffer.
minwrite	Wait for a fifo fill count of at least minwrite doubles before flushing the contents of the fifo to disk. Fifo is also flushed before this object is destroyed.
prefix	Path and start of output file name. Will be extended with file name extension ".dat".
use_date	When true, the current date and time will be appended to the output file name before the file name extension.
varname	Name of AC variable to save into file. Can be accessed through getter method **get_varname()** (p. 1541). Stored here to avoid races between processing thread and configuration thread.

5.427.3.2 ~acwriter_t() `plugins::hoertech::acrec::acwriter_t::~acwriter_t ( ) [default]`

Deallocates memory but does not terminate the write_thread.

write_thread must be terminated before the destructor executes by calling exit_request.

5.427.4 Member Function Documentation

5.427.4.1 process() void acwriter_t::process (MHA_AC::comm_var_t * s)

Place the data present in the algorithm communication variable into the fifo for output to disk.

5.427.4.2 exit_request() void acwriter_t::exit_request ( )

Terminate output thread.

5.427.4.3 get_varname() const std::string& plugins::hoertech::acrec::acwriter_t::get_varname ( ) const [inline]

getter for ac variable name

Returns

name as char* as needed by get_var

5.427.4.4 write_thread() void acwriter_t::write_thread ( ) [private]

Main method of the disk writer thread.

Periodically wakes up and checks if data needs to be written to disk.

5.427.4.5 create_datafile() void acwriter_t::create_datafile (const std::string & prefix, bool use_date) [private]

Open data file for output.

Combine prefix, date, and file name extension

Parameters

prefix	Path and start of output file name. Will be extended with file name extension ".dat".
use_date	When true, the current date and time will be appended to the output file name before the file name extension.

5.427.5 Member Data Documentation

5.427.5.1 close_session `std::atomic<bool> plugins::hoertech::acrec::acwriter_t::close_session [private]`

cross-thread-synchronization.

write_thread() (p. 1541) terminates after this is set to true by **exit_request()** (p. 1541).

5.427.5.2 active `const bool plugins::hoertech::acrec::acwriter_t::active [private]`

The writer thread and the output file will only be created when active is true.

5.427.5.3 fifo `std::unique_ptr< mha_fifo_lf_t< output_type> > plugins::hoertech::acrec::acwriter_t::fifo [private]`

Fifo for decoupling signal processing thread from disk writer thread.

5.427.5.4 disk_write_threshold_min_num_samples `const unsigned int plugins::hoertech::acrec::acwriter_t::disk_write_threshold_min_num_samples [private]`

Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.

5.427.5.5 writethread

`std::thread plugins::hoertech::acrec::acwriter_t::writethread [private]`

The thread that writes to disk.

5.427.5.6 diskbuffer

`std::unique_ptr< output_type[] > plugins::hoertech::acrec::acwriter_t::diskbuffer [private]`

Intermediate buffer to receive data from fifo and store on disk.

5.427.5.7 outfile

`std::fstream plugins::hoertech::acrec::acwriter_t::outfile [private]`

Ouput file.

5.427.5.8 num_channels

`unsigned plugins::hoertech::acrec::acwriter_t::num_channels = 0U [private]`

Number of channels of AC variable using stride.

If the number of channels changes during processing, an exception is thrown.

5.427.5.9 is_num_channels_known

`bool plugins::hoertech::acrec::acwriter_t::is_num_channels_known = false [private]`

The number of channels is determined during the first process callback.

is_num_channels_known is set to true after the first process callback.

5.427.5.10 is_complex

`bool plugins::hoertech::acrec::acwriter_t::is_complex = false [private]`

If the AC variable is of complex valued type or not.

If this changes during processing, then an exception is thrown.

5.427.5.11 varname `const std::string plugins::hoertech::acrec::acwriter_t::varname [private]`

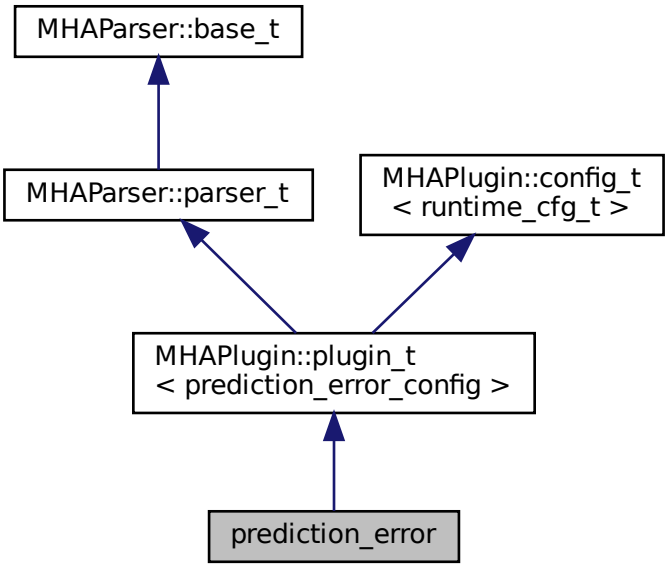
The name of the ac variable to publish.

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

- **acrec.hh**
- **acrec.cpp**

5.428 prediction_error Class Reference

Inheritance diagram for prediction_error:



Public Member Functions

- **prediction_error** (**MHA_AC::algo_comm_t** & **ac**, const std::string &configured_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 afc_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

5.428.1 Constructor & Destructor Documentation

5.428.1.1 prediction_error() `prediction_error::prediction_error (`

```
  MHA_AC::algo_comm_t & ac,
 const std::string & configured_name )
```

Constructs our plugin.

5.428.1.2 ~prediction_error() `prediction_error::~prediction_error ( )`

5.428.2 Member Function Documentation

5.428.2.1 process() `mha_wave_t * prediction_error::process ( mha_wave_t * signal )`

Checks for the most recent configuration and defers processing to it.

5.428.2.2 prepare() `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. 1301).

5.428.2.3 release() `void prediction_error::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< prediction_error_config >** (p. 1302).

5.428.2.4 update_cfg() `void prediction_error::update_cfg ( void ) [private]`

5.428.3 Member Data Documentation

5.428.3.1 rho **MHAParser::float_t** prediction_error::rho

5.428.3.2 c **MHAParser::float_t** prediction_error::c

5.428.3.3 ntaps **MHAParser::int_t** prediction_error::ntaps

5.428.3.4 gains **MHAParser::vfloat_t** prediction_error::gains

5.428.3.5 name_e **MHAParser::string_t** prediction_error::name_e

5.428.3.6 name_f **MHAParser::string_t** prediction_error::name_f

5.428.3.7 name_lpc **MHAParser::string_t** prediction_error::name_lpc

5.428.3.8 lpc_order **MHAParser::int_t** prediction_error::lpc_order

5.428.3.9 afc_delay **MHAParser::vint_t** prediction_error::afc_delay

5.428.3.10 delay_w **MHAParser::vint_t** prediction_error::delay_w

5.428.3.11 delay_d **MHAParser::vint_t** prediction_error::delay_d

5.428.3.12 n_no_update **MHAParser::int_t** prediction_error::n_no_update

5.428.3.13 patchbay **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**

5.429 prediction_error_config Class Reference

Public Member Functions

- **prediction_error_config** (**MHA_AC::algo_comm_t** & **ac**, const **mhaconfig_t** in_cfg, **prediction_error** ∗afc)
- ∼**prediction_error_config** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s_Y, **mha_real_t** rho, **mha_real_t** c)
- void **insert** ()

Private Attributes

- **MHA_AC::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**
- double **PSD_val**
- **MHASignal::waveform_t v_G**
- **MHASignal::waveform_t s_U**
- **MHASignal::delay_t s_E_afc_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**

5.429.1 Constructor & Destructor Documentation

5.429.1.1 prediction_error_config() `prediction_error_config::prediction_error_config (`

```
            MHA_AC::algo_comm_t & ac,
           const  mhaconfig_t in_cfg,
            prediction_error * afc )
```

5.429.1.2 ~prediction_error_config() prediction_error_config::~prediction_error_config ( )

5.429.2 Member Function Documentation

5.429.2.1 process()

```
mha_wave_t * prediction_error_config::process (
    mha_wave_t * s_Y,
    mha_real_t rho,
    mha_real_t c )
```

5.429.2.2 insert() void prediction_error_config::insert ( )

5.429.3 Member Data Documentation

5.429.3.1 ac MHA_AC::algo_comm_t& prediction_error_config::ac [private]

5.429.3.2 ntaps unsigned int prediction_error_config::ntaps [private]

5.429.3.3 frames unsigned int prediction_error_config::frames [private]

5.429.3.4 channels unsigned int prediction_error_config::channels [private]

5.429.3.5 s_E

`MHA_AC::waveform_t prediction_error_config::s_E [private]`

5.429.3.6 F

`MHA_AC::waveform_t prediction_error_config::F [private]`

5.429.3.7 Pu

`MHASignal::waveform_t prediction_error_config::Pu [private]`

Power of input signal delayline.

5.429.3.8 name_d_

`std::string prediction_error_config::name_d_ [private]`

5.429.3.9 name_lpc_

`std::string prediction_error_config::name_lpc_ [private]`

5.429.3.10 n_no_update_

`int prediction_error_config::n_no_update_ [private]`

5.429.3.11 no_iter

`int prediction_error_config::no_iter [private]`

5.429.3.12 iter

`int prediction_error_config::iter [private]`

5.429.3.13 PSD_val

`double prediction_error_config::PSD_val [private]`

5.429.3.14 v_G

MHASignal::waveform_t prediction_error_config::v_G [private]

5.429.3.15 s_U

MHASignal::waveform_t prediction_error_config::s_U [private]

5.429.3.16 s_E_afc_delay

MHASignal::delay_t prediction_error_config::s_E_afc_↩ delay [private]

5.429.3.17 s_W

MHASignal::delay_t prediction_error_config::s_W [private]

5.429.3.18 s_Wflt

MHASignal::ringbuffer_t prediction_error_config::s_Wflt [private]

5.429.3.19 s_U_delay

MHASignal::delay_t prediction_error_config::s_U_delay [private]

5.429.3.20 s_U_delayflt

MHASignal::ringbuffer_t prediction_error_config::s_U_↩ delayflt [private]

5.429.3.21 F_Uflt

MHASignal::waveform_t prediction_error_config::F_Uflt [private]

5.429.3.22 s_Y_delay

MHASignal::delay_t prediction_error_config::s_Y_delay [private]

5.429.3.23 s_Y_delayflt

MHASignal::ringbuffer_t prediction_error_config::s_Y_↩ delayflt [private]

5.429.3.24 UbufferPrew

MHASignal::ringbuffer_t prediction_error_config::Ubuffer↩ Prew [private]

5.429.3.25 s_LPC

mha_wave_t prediction_error_config::s_LPC [private]

5.429.3.26 UPrew

mha_wave_t prediction_error_config::UPrew [private]

5.429.3.27 YPrew

mha_wave_t prediction_error_config::YPrew [private]

5.429.3.28 EPrew

mha_wave_t prediction_error_config::EPrew [private]

5.429.3.29 UPrewW

mha_wave_t prediction_error_config::UPrewW [private]

5.429.3.30 smpl

mha_wave_t prediction_error_config::smpl [private]

5.429.3.31 s_Usmpl

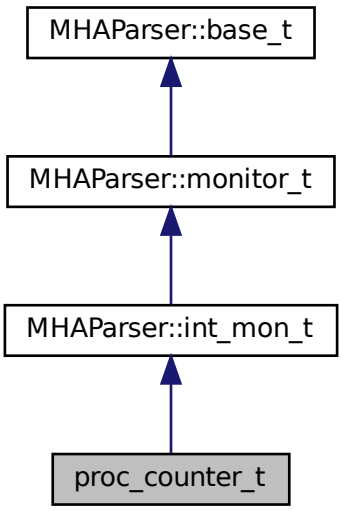
mha_wave_t* prediction_error_config::s_Usmpl [private]

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

- **prediction_error.h**
- **prediction_error.cpp**

5.430 proc_counter_t Class Reference

Inheritance diagram for proc_counter_t:



Public Member Functions

- **proc_counter_t** (**MHA_AC::algo_comm_t** &iac, const std::string & **configured_name**)
- ~**proc_counter_t** ()
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)
- void **prepare_** (**mhaconfig_t** &)
- void **release_** ()

Private Member Functions

- void **insert** ()

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
- const std::string **configured_name**

Additional Inherited Members

5.430.1 Constructor & Destructor Documentation

5.430.1.1 proc_counter_t() proc_counter_t::proc_counter_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.430.1.2 ~proc_counter_t() proc_counter_t::~proc_counter_t ( )

5.430.2 Member Function Documentation

5.430.2.1 process() [1/2] mha_wave_t * proc_counter_t::process (mha_wave_t * s)

5.430.2.2 process() [2/2] mha_spec_t * proc_counter_t::process (mha_spec_t * s)

5.430.2.3 prepare_() void proc_counter_t::prepare_ (mhaconfig_t &) [inline]

5.430.2.4 release_() void proc_counter_t::release_ ( ) [inline]

5.430.2.5 insert() `void proc_counter_t::insert ( ) [private]`

5.430.3 Member Data Documentation

5.430.3.1 ac `MHA_AC::algo_comm_t& proc_counter_t::ac [private]`

5.430.3.2 configured_name `const std::string proc_counter_t::configured_name [private]`

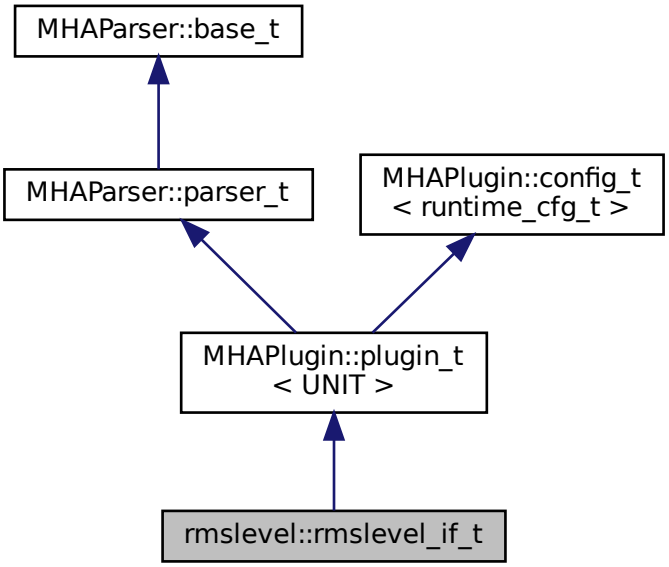
The documentation for this class was generated from the following file:

- **proc_counter.cpp**

5.431 rmslevel::rmslevel_if_t Class Reference

Rmslevel plugin.

Inheritance diagram for rmslevel::rmslevel_if_t:



Public Member Functions

- **rmslevel_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor of rmslevel plugin.
- **mha_spec_t** * **process** (**mha_spec_t** *s)
  Extract level from current STFT spectrum.
- **mha_wave_t** * **process** (**mha_wave_t** *s)
  Extract level from current time signal block.
- void **prepare** (**mhaconfig_t** &signal_dimensions) override
  Prepare rmslevel plugin for signal processing: Resize and reinitialize monitor variables according to number of audio channels specified in parameter, publish applicable monitor variables as AC variables (depends on signal domain).
- void **release** () override
  Release removes published AC variables from AC space.

Private Member Functions

- void **update** ()
  Called on write access to the configuration variable `unit`.
- void **insert_ac_variables_levels** ()
  (Re-)insert AC variables for spectral processing into AC space.
- void **insert_ac_variables_peaks_and_levels** ()
  (Re-)insert AC variables for waveform processing into AC space.
- void **insert_ac_variable_float_vector** (std::vector< float > &v, const std::string &acname)
  (Re-)insert a single AC variable.
- void **remove_ac_variables** ()
  Remove AC variables from AC space.

Private Attributes

- **MHAEvents::patchbay_t**< **rmslevel_if_t** > **patchbay**
  Configuration language event dispatcher.
- **MHAParser::vfloat_mon_t** **level** = {"RMS level in W/m^2"}
  Sound power.
- **MHAParser::vfloat_mon_t** **level_db** = {"RMS **level** in dB"}
  Sound pressure level.
- **MHAParser::vfloat_mon_t** **peak** = {"peak amplitude in Pa"}
  Peak amplitude.
- **MHAParser::vfloat_mon_t** **peak_db** = {"peak amplitude in dB"}
  dB value corresponding to peak amplitude.
- const std::string **level_acname**
  AC variable name for `level`.
- const std::string **level_db_acname**

AC variable name for `level_db`.

- const std::string **peak_acname**

  AC variable name for `peak`.

- const std::string **peak_db_acname**

  AC variable name for `peak_db`.

- **MHAParser::kw_t unit**

  Configuration variable for selecting result dB scale.

- std::vector< **mha_real_t** > **freq_offsets**

  freq_offsets provides the conversion of dB(SPL) to dB(HL) for every frequency bin in the stft used by coloured_intensity.

Additional Inherited Members

5.431.1 Detailed Description

Rmslevel plugin.

Measures sound for each block and publishes measured result in monitor variables and AC variables.

5.431.2 Constructor & Destructor Documentation

5.431.2.1 rmslevel_if_t() `rmslevel::rmslevel_if_t::rmslevel_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor of rmslevel plugin.

Parameters

iac	Algorithm communication variable space, used to extracted levels
configured_name	Configured name of this plugins: either "rmslevel" the name explicitly given after the colon. Used as for all published AC variables.

5.431.3 Member Function Documentation

5.431.3.1 process() [1/2] mha_spec_t * rmslevel::rmslevel_if_t::process (mha_spec_t * s)

Extract level from current STFT spectrum.

Parameters

s	input spectrum, not modified by this method.

5.431.3.2 process() [2/2] mha_wave_t * rmslevel::rmslevel_if_t::process (mha_wave_t * s)

Extract level from current time signal block.

Parameters

s	input audio block, not modified by this method.

5.431.3.3 prepare() void rmslevel::rmslevel_if_t::prepare (mhaconfig_t & signal_dimensions) [override], [virtual]

Prepare rmslevel plugin for signal processing: Resize and reinitialize monitor variables according to number of audio channels specified in parameter, publish applicable monitor variables as AC variables (depends on signal domain).

Parameters

signal_dimensions	Audio signal metadata, not modified by this method.

Implements **MHAPlugin::plugin_t< UNIT >** (p. 1301).

5.431.3.4 release() void rmslevel::rmslevel_if_t::release (void) [override], [virtual]

Release removes published AC variables from AC space.

Reimplemented from **MHAPlugin::plugin_t< UNIT >** (p. 1302).

5.431.3.5 update() void rmslevel::rmslevel_if_t::update ( ) [private]

Called on write access to the configuration variable `unit`.

5.431.3.6 insert_ac_variables_levels() void rmslevel::rmslevel_if_t::insert_ac_↩ variables_levels ( ) [private]

(Re-)insert AC variables for spectral processing into AC space.

Needs to be called during **prepare()** (p. 1559) and at the end of every invocation of **process()** (p. 1558) when signal domain is MHA_SPECTRUM.

5.431.3.7 insert_ac_variables_peaks_and_levels() void rmslevel::rmslevel_if_t↩ ::insert_ac_variables_peaks_and_levels ( ) [private]

(Re-)insert AC variables for waveform processing into AC space.

Needs to be called during **prepare()** (p. 1559) and at the end of every invocation of **process()** (p. 1558) when signal domain is MHA_WAVEFORM.

5.431.3.8 insert_ac_variable_float_vector() void rmslevel::rmslevel_if_t::insert_↩ ac_variable_float_vector (std::vector< float > & *v*, const std::string & *acname*) [private]

(Re-)insert a single AC variable.

Helper method used by insert_ac_variables_levels and insert_ac_variables_peaks_and_levels. The stride of the AC variable will be set to v.size().

Parameters

v	Vector of floats to insert into the AC space. Its memory at v.data() must be valid until the next call to **process()** (p. 1558) or **release()** (p. 1559) (whichever occurs earlier). Values may be accessed or altered by other plugins.
acname	Name of the AC variable in the AC space.

5.431.3.9 remove_ac_variables() void rmslevel::rmslevel_if_t::remove_ac_variables ( ) [private]

Remove AC variables from AC space.

Called from **release()** (p. 1559).

5.431.4 Member Data Documentation

5.431.4.1 patchbay MHAEvents::patchbay_t< rmslevel_if_t> rmslevel::rmslevel_if↩ _t::patchbay [private]

Configuration language event dispatcher.

5.431.4.2 level MHAParser::vfloat_mon_t rmslevel::rmslevel_if_t::level = {"RMS level in W/m^2"} [private]

Sound power.

5.431.4.3 level_db MHAParser::vfloat_mon_t rmslevel::rmslevel_if_t::level_db = {"RMS **level** in dB"} [private]

Sound pressure level.

5.431.4.4 peak `MHAParser::vfloat_mon_t rmslevel::rmslevel_if_t::peak = {"peak amplitude in Pa"} [private]`

Peak amplitude.

5.431.4.5 peak_db `MHAParser::vfloat_mon_t rmslevel::rmslevel_if_t::peak_db = {"peak amplitude in dB"} [private]`

dB value corresponding to peak amplitude.

5.431.4.6 level_acname `const std::string rmslevel::rmslevel_if_t::level_acname [private]`

AC variable name for `level`.

5.431.4.7 level_db_acname `const std::string rmslevel::rmslevel_if_t::level_db_acname [private]`

AC variable name for `level_db`.

5.431.4.8 peak_acname `const std::string rmslevel::rmslevel_if_t::peak_acname [private]`

AC variable name for `peak`.

5.431.4.9 peak_db_acname `const std::string rmslevel::rmslevel_if_t::peak_db_acname [private]`

AC variable name for `peak_db`.

5.431.4.10 unit `MHAParser::kw_t rmslevel::rmslevel_if_t::unit [private]`

Configuration variable for selecting result dB scale.

5.431.4.11 freq_offsets `std::vector< mha_real_t> rmslevel::rmslevel_if_t::freq_offsets [private]`

freq_offsets provides the conversion of dB(SPL) to dB(HL) for every frequency bin in the stft used by coloured_intensity.

Unused when not in spectral domain and unit=hl.

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

- **rmslevel.cpp**

5.432 RNNModel Struct Reference

Public Attributes

- int **input_scale_size**
- const **ScalerLayer** * **scaler**
- int **dense_projection_size**
- const **DenseLayer** * **dense_projection**
- int **dense_map_1_size**
- const **DenseLayer** * **dense_map_1**
- int **dconv_5_size**
- const **DConvLayer** * **dconv_5**
- int **dense_deconv_5_size**
- const **DenseLayer** * **dense_deconv_5**
- int **dconv_3_size**
- const **DConvLayer** * **dconv_3**
- int **dense_deconv_3_size**
- const **DenseLayer** * **dense_deconv_3**
- int **dconv_skip_1_size**
- const **DConvLayer1x1** * **dconv_skip_1**
- int **dense_tac1_1_size**
- const **DenseLayer** * **dense_tac1_1**
- int **dense_tac1_2_size**
- const **DenseLayer** * **dense_tac1_2**
- int **dense_tac1_3_size**
- const **DenseLayer** * **dense_tac1_3**

- int **gru_2_1_size**
- const **GRULayer** * **gru_2_1**
- int **gru_2_2_size**
- const **GRULayer** * **gru_2_2**
- int **dconv_skip_2_size**
- const **DConvLayer1x1** * **dconv_skip_2**
- int **dense_tac2_1_size**
- const **DenseLayer** * **dense_tac2_1**
- int **dense_tac2_2_size**
- const **DenseLayer** * **dense_tac2_2**
- int **dense_tac2_3_size**
- const **DenseLayer** * **dense_tac2_3**
- int **dense_map_2_size**
- const **DenseLayer** * **dense_map_2**
- int **dense_filt_b_size**
- const **DenseLayer** * **dense_filt_b**
- int **b_scale_size**
- const **ScalerLayer** * **scaler_b**
- int **dense_filt_t_size**
- const **DenseLayer** * **dense_filt_t**
- int **t_scale_size**
- const **ScalerLayer** * **scaler_t**

5.432.1 Member Data Documentation

5.432.1.1 input_scale_size `int RNNModel::input_scale_size`

5.432.1.2 scaler `const ScalerLayer * RNNModel::scaler`

5.432.1.3 dense_projection_size `int RNNModel::dense_projection_size`

5.432.1.4 dense_projection `const DenseLayer * RNNModel::dense_projection`

5.432.1.5 dense_map_1_size int RNNModel::dense_map_1_size

5.432.1.6 dense_map_1 const DenseLayer * RNNModel::dense_map_1

5.432.1.7 dconv_5_size int RNNModel::dconv_5_size

5.432.1.8 dconv_5 const DConvLayer * RNNModel::dconv_5

5.432.1.9 dense_deconv_5_size int RNNModel::dense_deconv_5_size

5.432.1.10 dense_deconv_5 const DenseLayer * RNNModel::dense_deconv_5

5.432.1.11 dconv_3_size int RNNModel::dconv_3_size

5.432.1.12 dconv_3 const DConvLayer * RNNModel::dconv_3

5.432.1.13 dense_deconv_3_size int RNNModel::dense_deconv_3_size

5.432.1.14 dense_deconv_3 const DenseLayer * RNNModel::dense_deconv_3

5.432.1.15 dconv_skip_1_size int RNNModel::dconv_skip_1_size

5.432.1.16 dconv_skip_1 const DConvLayer1x1 * RNNModel::dconv_skip_1

5.432.1.17 dense_tac1_1_size int RNNModel::dense_tac1_1_size

5.432.1.18 dense_tac1_1 const DenseLayer * RNNModel::dense_tac1_1

5.432.1.19 dense_tac1_2_size int RNNModel::dense_tac1_2_size

5.432.1.20 dense_tac1_2 const DenseLayer * RNNModel::dense_tac1_2

5.432.1.21 dense_tac1_3_size int RNNModel::dense_tac1_3_size

5.432.1.22 dense_tac1_3 const DenseLayer * RNNModel::dense_tac1_3

5.432.1.23 gru_2_1_size int RNNModel::gru_2_1_size

5.432.1.24 gru_2_1 const GRULayer * RNNModel::gru_2_1

5.432.1.25 gru_2_2_size int RNNModel::gru_2_2_size

5.432.1.26 gru_2_2 const GRULayer * RNNModel::gru_2_2

5.432.1.27 dconv_skip_2_size int RNNModel::dconv_skip_2_size

5.432.1.28 dconv_skip_2 const DConvLayer1x1 * RNNModel::dconv_skip_2

5.432.1.29 dense_tac2_1_size int RNNModel::dense_tac2_1_size

5.432.1.30 dense_tac2_1 const DenseLayer * RNNModel::dense_tac2_1

5.432.1.31 dense_tac2_2_size int RNNModel::dense_tac2_2_size

5.432.1.32 dense_tac2_2 const DenseLayer * RNNModel::dense_tac2_2

5.432.1.33 dense_tac2_3_size int RNNModel::dense_tac2_3_size

5.432.1.34 dense_tac2_3 const DenseLayer * RNNModel::dense_tac2_3

5.432.1.35 dense_map_2_size int RNNModel::dense_map_2_size

5.432.1.36 dense_map_2 const DenseLayer * RNNModel::dense_map_2

5.432.1.37 dense_filt_b_size int RNNModel::dense_filt_b_size

5.432.1.38 dense_filt_b const DenseLayer * RNNModel::dense_filt_b

5.432.1.39 b_scale_size int RNNModel::b_scale_size

5.432.1.40 scaler_b const ScalerLayer * RNNModel::scaler_b

5.432.1.41 dense_filt_t_size int RNNModel::dense_filt_t_size

5.432.1.42 dense_filt_t const DenseLayer * RNNModel::dense_filt_t

5.432.1.43 t_scale_size int RNNModel::t_scale_size

5.432.1.44 scaler_t const ScalerLayer * RNNModel::scaler_t

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

- **gcfsnet_bin/rnn_data.h**

5.433 RNNState Struct Reference

Public Attributes

- const **RNNModel** * **model**
- float * **gru_2_1_state**
- float * **gru_2_2_state**
- float * **dconv_5_buffer**
- float * **dconv_3_buffer**
- **counter dconv_5_idx_start**
- **counter dconv_3_idx_start**
- **counter dconv_5_idx_write**
- **counter dconv_3_idx_write**

5.433.1 Member Data Documentation

5.433.1.1 model const RNNModel * RNNState::model

5.433.1.2 gru_2_1_state `float * RNNState::gru_2_1_state`

5.433.1.3 gru_2_2_state `float * RNNState::gru_2_2_state`

5.433.1.4 dconv_5_buffer `float * RNNState::dconv_5_buffer`

5.433.1.5 dconv_3_buffer `float * RNNState::dconv_3_buffer`

5.433.1.6 dconv_5_idx_start `counter RNNState::dconv_5_idx_start`

5.433.1.7 dconv_3_idx_start `counter RNNState::dconv_3_idx_start`

5.433.1.8 dconv_5_idx_write `counter RNNState::dconv_5_idx_write`

5.433.1.9 dconv_3_idx_write `counter RNNState::dconv_3_idx_write`

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

- **gcfsnet_bin/rnn_data.h**

5.434 rohBeam::configOptions Struct Reference

Public Attributes

- bool **enable_adaptive_beam**
- int **binaural_type_index**
- float **alpha_postfilter**
- float **alpha_blocking_XkXi**
- float **alpha_blocking_XkY**

5.434.1 Member Data Documentation

5.434.1.1 enable_adaptive_beam `bool rohBeam::configOptions::enable_adaptive_beam`

5.434.1.2 binaural_type_index `int rohBeam::configOptions::binaural_type_index`

5.434.1.3 alpha_postfilter `float rohBeam::configOptions::alpha_postfilter`

5.434.1.4 alpha_blocking_XkXi `float rohBeam::configOptions::alpha_blocking_XkXi`

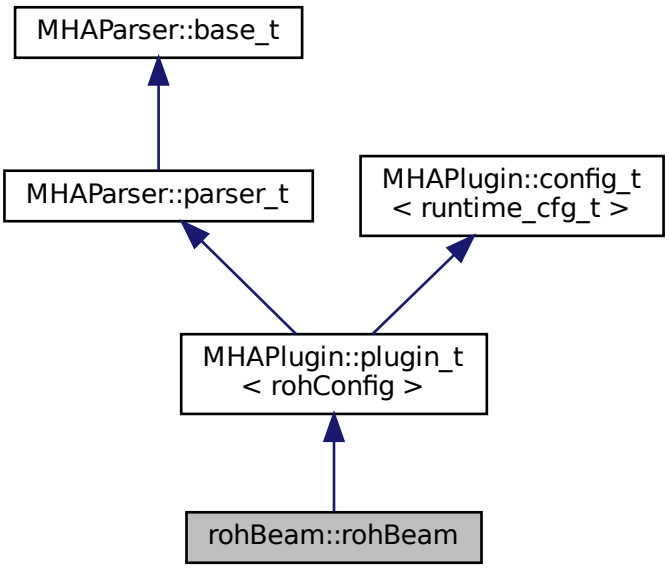
5.434.1.5 alpha_blocking_XkY `float rohBeam::configOptions::alpha_blocking_XkY`

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

- **rohBeam.hh**

5.435 rohBeam::rohBeam Class Reference

Inheritance diagram for rohBeam::rohBeam:



Public Member Functions

- **rohBeam** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- ∼**rohBeam** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)
- void **release** (void)

Private Types

- typedef const Eigen::MatrixXf(rohBeam::∗ **noiseFuncPtr**) (float)

Private Member Functions

- void **update_cfg** ()
- float **compute_head_model_T** (float)
- float **compute_head_model_alpha** (float)
- Eigen::MatrixXcf ∗ **compute_head_model_mat** (float src_az_degrees)

- **MHASignal::matrix_t** * **compute_delaycomp_vec** (Eigen::MatrixXcf *headModel)
- std::vector< Eigen::MatrixXcf > * **noise_integrate_hrtf** ()
- Eigen::VectorXcf **solve_MVDR** (Eigen::VectorXcf propVec, Eigen::MatrixXcf noiseM)
- const Eigen::MatrixXf **compute_uncorr** (float w)
- const Eigen::MatrixXf **compute_diff2D** (float)
- const Eigen::MatrixXf **compute_diff3D** (float)
- **MHASignal::matrix_t** * **compute_beamW** (Eigen::MatrixXcf *)
- float **compute_wng** (Eigen::VectorXcf freqRes, Eigen::VectorXcf propVec)
- void **export_beam_design** (const **MHASignal::matrix_t** &beamW, const Eigen::↩ MatrixXcf &headModel)
- **noiseFuncPtr** **get_noise_model_func** (void)
- void **on_model_param_valuechanged** ()

Private Attributes

- **MHAParser::kw_t** **prop_type**
- **MHAParser::string_t** **sampled_hrir_path**
- **MHAParser::float_t** **source_azimuth_degrees**
- **MHAParser::vfloat_t** **mic_azimuth_degrees_vec**
- **MHAParser::float_t** **head_model_sphere_radius_cm**
- **MHAParser::mfloat_t** **intermic_distance_cm**
- **MHAParser::kw_t** **noise_field_model**
- **MHAParser::bool_t** **enable_adaptive_beam**
- **MHAParser::kw_t** **binaural_type**
- **MHAParser::float_t** **diag_loading_mu**
- **MHAParser::bool_t** **enable_export**
- **MHAParser::bool_t** **enable_wng_optimization**
- **MHAParser::float_t** **tau_postfilter_ms**
- **MHAParser::float_t** **tau_blocking_XkXi_ms**
- **MHAParser::float_t** **tau_blocking_XkY_ms**
- **MHAEvents::patchbay_t**< **rohBeam** > **patchbay**
- bool **prepared**
- **MHA_AC::spectrum_t** * **beamExport**
- **MHA_AC::waveform_t** * **noiseModelExport**
- **MHA_AC::spectrum_t** * **propExport**

Additional Inherited Members

5.435.1 Member Typedef Documentation

5.435.1.1 noiseFuncPtr

```
typedef const Eigen::MatrixXf(rohBeam::* rohBeam::rohBeam↩
::noiseFuncPtr) (float) [private]
```

5.435.2 Constructor & Destructor Documentation

5.435.2.1 rohBeam()

```
rohBeam::rohBeam::rohBeam (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

5.435.2.2 ~rohBeam()

```
rohBeam::rohBeam::~rohBeam ( )
```

5.435.3 Member Function Documentation

5.435.3.1 process()

```
mha_spec_t* rohBeam::rohBeam::process (
    mha_spec_t * )
```

5.435.3.2 prepare()

```
void rohBeam::rohBeam::prepare (
    mhaconfig_t & ) [virtual]
```

Implements **MHAPlugin::plugin_t< rohConfig >** (p. 1301).

5.435.3.3 release()

```
void rohBeam::rohBeam::release (
    void ) [inline], [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< rohConfig >** (p. 1302).

5.435.3.4 update_cfg() void rohBeam::rohBeam::update_cfg ( ) [private]

5.435.3.5 compute_head_model_T() float rohBeam::rohBeam::compute_head_model_T (float) [private]

5.435.3.6 compute_head_model_alpha() float rohBeam::rohBeam::compute_head_↩ model_alpha (float) [private]

5.435.3.7 compute_head_model_mat() Eigen::MatrixXcf* rohBeam::rohBeam::compute↩ _head_model_mat (float *src_az_degrees*) [private]

5.435.3.8 compute_delaycomp_vec() **MHASignal::matrix_t*** rohBeam::rohBeam::compute↩ _delaycomp_vec (Eigen::MatrixXcf * *headModel*) [private]

5.435.3.9 noise_integrate_hrtf() std::vector<Eigen::MatrixXcf>* rohBeam::rohBeam↩ ::noise_integrate_hrtf ( ) [private]

5.435.3.10 solve_MVDR() Eigen::VectorXcf rohBeam::rohBeam::solve_MVDR (Eigen::VectorXcf *propVec*, Eigen::MatrixXcf *noiseM*) [private]

5.435.3.11 compute_uncorr()

```
const Eigen::MatrixXf rohBeam::rohBeam::compute_uncorr (
        float w ) [private]
```

5.435.3.12 compute_diff2D()

```
const Eigen::MatrixXf rohBeam::rohBeam::compute_diff2D (
        float ) [private]
```

5.435.3.13 compute_diff3D()

```
const Eigen::MatrixXf rohBeam::rohBeam::compute_diff3D (
        float ) [private]
```

5.435.3.14 compute_beamW()

```
MHASignal::matrix_t* rohBeam::rohBeam::compute_beamW (
        Eigen::MatrixXcf * ) [private]
```

5.435.3.15 compute_wng()

```
float rohBeam::rohBeam::compute_wng (
        Eigen::VectorXcf freqRes,
        Eigen::VectorXcf propVec ) [private]
```

5.435.3.16 export_beam_design()

```
void rohBeam::rohBeam::export_beam_design (
        const MHASignal::matrix_t & beamW,
        const Eigen::MatrixXcf & headModel ) [private]
```

5.435.3.17 get_noise_model_func()

```
noiseFuncPtr rohBeam::rohBeam::get_noise_model_func (
        void ) [private]
```

5.435.3.18 on_model_param_valuechanged() void rohBeam::rohBeam::on_model_↩ param_valuechanged ( ) [private]

5.435.4 Member Data Documentation

5.435.4.1 prop_type MHAParser::kw_t rohBeam::rohBeam::prop_type [private]

5.435.4.2 sampled_hrir_path MHAParser::string_t rohBeam::rohBeam::sampled_hrir_↩ path [private]

5.435.4.3 source_azimuth_degrees MHAParser::float_t rohBeam::rohBeam::source_↩ azimuth_degrees [private]

5.435.4.4 mic_azimuth_degrees_vec MHAParser::vfloat_t rohBeam::rohBeam::mic_↩ azimuth_degrees_vec [private]

5.435.4.5 head_model_sphere_radius_cm MHAParser::float_t rohBeam::rohBeam↩ ::head_model_sphere_radius_cm [private]

5.435.4.6 intermic_distance_cm MHAParser::mfloat_t rohBeam::rohBeam::intermic_↩ distance_cm [private]

5.435.4.7 noise_field_model

MHAParser::kw_t rohBeam::rohBeam::noise_field_model [private]

5.435.4.8 enable_adaptive_beam

MHAParser::bool_t rohBeam::rohBeam::enable_↩ adaptive_beam [private]

5.435.4.9 binaural_type

MHAParser::kw_t rohBeam::rohBeam::binaural_type [private]

5.435.4.10 diag_loading_mu

MHAParser::float_t rohBeam::rohBeam::diag_loading_mu [private]

5.435.4.11 enable_export

MHAParser::bool_t rohBeam::rohBeam::enable_export [private]

5.435.4.12 enable_wng_optimization

MHAParser::bool_t rohBeam::rohBeam::enable_↩ wng_optimization [private]

5.435.4.13 tau_postfilter_ms

MHAParser::float_t rohBeam::rohBeam::tau_postfilter↩ _ms [private]

5.435.4.14 tau_blocking_XkXi_ms

MHAParser::float_t rohBeam::rohBeam::tau_↩ blocking_XkXi_ms [private]

5.435.4.15 tau_blocking_XkY_ms

MHAParser::float_t rohBeam::rohBeam::tau_blocking_XkY_ms [private]

5.435.4.16 patchbay

MHAEvents::patchbay_t< **rohBeam**> rohBeam::rohBeam::patchbay [private]

5.435.4.17 prepared

bool rohBeam::rohBeam::prepared [private]

5.435.4.18 beamExport

MHA_AC::spectrum_t* rohBeam::rohBeam::beamExport [private]

5.435.4.19 noiseModelExport

MHA_AC::waveform_t* rohBeam::rohBeam::noiseModelExport [private]

5.435.4.20 propExport

MHA_AC::spectrum_t* rohBeam::rohBeam::propExport [private]

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

- **rohBeam.hh**

5.436 rohBeam::rohConfig Class Reference

Public Member Functions

- **rohConfig** (const **mhaconfig_t** **in_cfg**, const **mhaconfig_t** **out_cfg**, std::unique_ptr< Eigen::MatrixXcf > headModel_, std::unique_ptr< **MHASignal::matrix_t** > beamW_, std::unique_ptr< **MHASignal::matrix_t** > delayComp_, const **configOptions** &options)
- **rohConfig** (**rohConfig** *lastConfig, const **mhaconfig_t**, const **mhaconfig_t** **out_cfg**, std::unique_ptr< Eigen::MatrixXcf > headModel_, std::unique_ptr< **MHASignal::matrix_t** > beamW_, std::unique_ptr< **MHASignal::matrix_t** > delayComp_, const **configOptions** &options)
- ~**rohConfig** ()
- **rohConfig** (const **rohConfig** &)=delete
- **rohConfig** & **operator=** (const **rohConfig** &)=delete
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **init_dynamic** ()

Private Member Functions

- void **phasereconstruction** (**MHASignal::spectrum_t** $*$)
- void **postfilter** (**mha_spec_t** $*$, **MHASignal::spectrum_t** $*$)
- void **copyfixedbfoutput** (**MHASignal::spectrum_t** $*$)

Private Attributes

- int **nfreq**
- int **nchan_block**
- **mhaconfig_t** **in_cfg**
- **mhaconfig_t** **out_cfg**
- bool **enable_adaptive_beam**
- int **binaural_type_index**
- std::unique_ptr$<$ Eigen::MatrixXcf $>$ **headModel**
- std::unique_ptr$<$ **MHASignal::matrix_t** $>$ **beamW**
- std::unique_ptr$<$ **MHASignal::matrix_t** $>$ **delayComp**
- **MHASignal::spectrum_t** $*$ **beam1**
- **MHASignal::spectrum_t** $*$ **beamA**
- **MHASignal::spectrum_t** $*$ **blockSpec**
- **MHASignal::spectrum_t** $*$ **outSpec**
- float **alpha_postfilter**
- float **alpha_blocking_XkXi**
- float **alpha_blocking_XkY**
- std::vector$<$ Eigen::MatrixXcf $>$ **corrXpXp**
- std::vector$<$ Eigen::VectorXcf $>$ **corrXpYf**
- Eigen::VectorXf **corrZZ**
- Eigen::VectorXf **corrLL**
- Eigen::VectorXf **corrRR**
- Eigen::HouseholderQR$<$ Eigen::MatrixXcf $>$ **hhCorrXpXp**
- Eigen::VectorXcf **nextXpYf**
- Eigen::VectorXcf **blockXp**
- Eigen::VectorXcf **freqResp**
- Eigen::ArrayXf **magResp**
- float **minLim**
- float **maxLim**

5.436.1 Constructor & Destructor Documentation

5.436.1.1 rohConfig() [1/3]

```
rohBeam::rohConfig::rohConfig (
    const mhaconfig_t in_cfg,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

5.436.1.2 rohConfig() [2/3]

```
rohBeam::rohConfig::rohConfig (
    rohConfig * lastConfig,
    const mhaconfig_t,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

5.436.1.3 ~rohConfig()

```
rohBeam::rohConfig::~rohConfig ( )
```

5.436.1.4 rohConfig() [3/3]

```
rohBeam::rohConfig::rohConfig (
    const rohConfig & ) [delete]
```

5.436.2 Member Function Documentation

5.436.2.1 operator=()

```
rohConfig& rohBeam::rohConfig::operator= (
    const rohConfig & ) [delete]
```

5.436.2.2 process()

```
mha_spec_t * rohBeam::rohConfig::process (
    mha_spec_t * inSpec )
```

5.436.2.3 init_dynamic() void rohBeam::rohConfig::init_dynamic ( )

5.436.2.4 phasereconstruction() void rohBeam::rohConfig::phasereconstruction (MHASignal::spectrum_t * prevSpecPost) [private]

5.436.2.5 postfilter() void rohBeam::rohConfig::postfilter (mha_spec_t * inSpec, MHASignal::spectrum_t * prevSpecPost) [private]

5.436.2.6 copyfixedbfoutput() void rohBeam::rohConfig::copyfixedbfoutput (MHASignal::spectrum_t * prevSpecPost) [private]

5.436.3 Member Data Documentation

5.436.3.1 nfreq int rohBeam::rohConfig::nfreq [private]

5.436.3.2 nchan_block int rohBeam::rohConfig::nchan_block [private]

5.436.3.3 in_cfg mhaconfig_t rohBeam::rohConfig::in_cfg [private]

5.436.3.4 out_cfg mhaconfig_t rohBeam::rohConfig::out_cfg [private]

5.436.3.5 enable_adaptive_beam

bool rohBeam::rohConfig::enable_adaptive_beam [private]

5.436.3.6 binaural_type_index

int rohBeam::rohConfig::binaural_type_index [private]

5.436.3.7 headModel

std::unique_ptr<Eigen::MatrixXcf> rohBeam::rohConfig::headModel [private]

5.436.3.8 beamW

std::unique_ptr< **MHASignal::matrix_t**> rohBeam::rohConfig::beamW [private]

5.436.3.9 delayComp

std::unique_ptr< **MHASignal::matrix_t**> rohBeam::rohConfig::delayComp [private]

5.436.3.10 beam1

MHASignal::spectrum_t* rohBeam::rohConfig::beam1 [private]

5.436.3.11 beamA

MHASignal::spectrum_t* rohBeam::rohConfig::beamA [private]

5.436.3.12 blockSpec

MHASignal::spectrum_t* rohBeam::rohConfig::blockSpec [private]

5.436.3.13 outSpec

MHASignal::spectrum_t* rohBeam::rohConfig::outSpec [private]

5.436.3.14 alpha_postfilter

`float rohBeam::rohConfig::alpha_postfilter [private]`

5.436.3.15 alpha_blocking_XkXi

`float rohBeam::rohConfig::alpha_blocking_XkXi [private]`

5.436.3.16 alpha_blocking_XkY

`float rohBeam::rohConfig::alpha_blocking_XkY [private]`

5.436.3.17 corrXpXp

`std::vector<Eigen::MatrixXcf> rohBeam::rohConfig::corrXpXp [private]`

5.436.3.18 corrXpYf

`std::vector<Eigen::VectorXcf> rohBeam::rohConfig::corrXpYf [private]`

5.436.3.19 corrZZ

`Eigen::VectorXf rohBeam::rohConfig::corrZZ [private]`

5.436.3.20 corrLL

`Eigen::VectorXf rohBeam::rohConfig::corrLL [private]`

5.436.3.21 corrRR

`Eigen::VectorXf rohBeam::rohConfig::corrRR [private]`

5.436.3.22 hhCorrXpXp

`Eigen::HouseholderQR<Eigen::MatrixXcf> rohBeam::rohConfig::hhCorrXpXp [private]`

5.436.3.23 nextXpYf `Eigen::VectorXcf rohBeam::rohConfig::nextXpYf [private]`

5.436.3.24 blockXp `Eigen::VectorXcf rohBeam::rohConfig::blockXp [private]`

5.436.3.25 freqResp `Eigen::VectorXcf rohBeam::rohConfig::freqResp [private]`

5.436.3.26 magResp `Eigen::ArrayXf rohBeam::rohConfig::magResp [private]`

5.436.3.27 minLim `float rohBeam::rohConfig::minLim [private]`

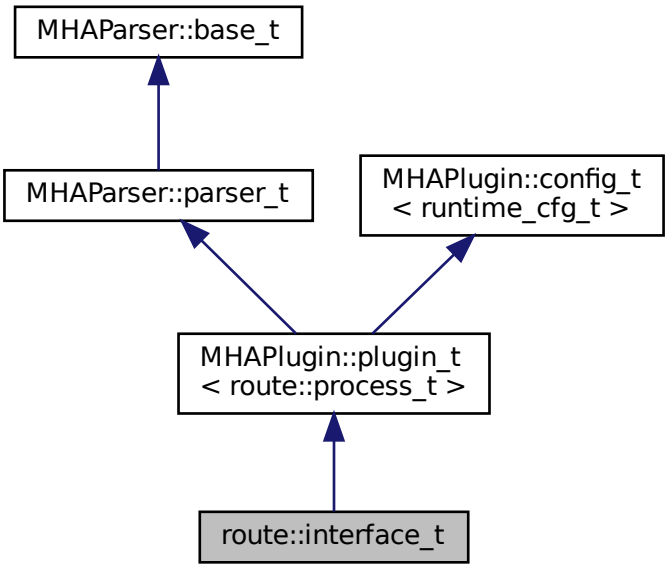
5.436.3.28 maxLim `float rohBeam::rohConfig::maxLim [private]`

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

- **rohBeam.hh**
- **rohBeam.cpp**

5.437 route::interface_t Class Reference

Inheritance diagram for route::interface_t:



Public Member Functions

- **interface_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- 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

5.437.1 Constructor & Destructor Documentation

5.437.1.1 interface_t() route::interface_t::interface_t (

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.437.2 Member Function Documentation

5.437.2.1 prepare() void route::interface_t::prepare (

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< route::process_t >** (p. 1301).

5.437.2.2 release() void route::interface_t::release (

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< route::process_t >** (p. 1302).

5.437.2.3 process() [1/2] mha_wave_t * route::interface_t::process (

```
mha_wave_t * s )
```

5.437.2.4 process() [2/2] mha_spec_t * route::interface_t::process (

```
mha_spec_t * s )
```

5.437.2.5 update() void route::interface_t::update ( ) [private]

5.437.3 Member Data Documentation

5.437.3.1 patchbay MHAEvents::patchbay_t< route::interface_t> route::interface_t::patchbay [private]

5.437.3.2 route_out MHAParser::vstring_t route::interface_t::route_out [private]

5.437.3.3 route_ac MHAParser::vstring_t route::interface_t::route_ac [private]

5.437.3.4 cfin mhaconfig_t route::interface_t::cfin [private]

5.437.3.5 cfout mhaconfig_t route::interface_t::cfout [private]

5.437.3.6 cfac mhaconfig_t route::interface_t::cfac [private]

5.437.3.7 prepared bool route::interface_t::prepared [private]

5.437.3.8 stopped `bool route::interface_t::stopped [private]`

5.437.3.9 algo `std::string route::interface_t::algo [private]`

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

- **route.cpp**

5.438 route::process_t Class Reference

Public Member Functions

- **process_t** (**MHA_AC::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**

5.438.1 Constructor & Destructor Documentation

5.438.1.1 process_t() `route::process_t::process_t (`

```
 MHA_AC::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 )
```

5.438.2 Member Function Documentation

5.438.2.1 process() [1/2] mha_wave_t * route::process_t::process (mha_wave_t * s)

5.438.2.2 process() [2/2] mha_spec_t * route::process_t::process (mha_spec_t * s)

5.438.3 Member Data Documentation

5.438.3.1 wout MHAMultiSrc::waveform_t route::process_t::wout [private]

5.438.3.2 sout MHAMultiSrc::spectrum_t route::process_t::sout [private]

5.438.3.3 wout_ac MHAMultiSrc::waveform_t route::process_t::wout_ac [private]

5.438.3.4 sout_ac MHAMultiSrc::spectrum_t route::process_t::sout_ac [private]

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

- **route.cpp**

5.439 rt_nlms_t Class Reference

Public Member Functions

- **rt_nlms_t** (**MHA_AC::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

- **MHA_AC::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**

5.439.1 Constructor & Destructor Documentation

5.439.1.1 rt_nlms_t() rt_nlms_t::rt_nlms_t (

```
 MHA_AC::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 )
```

5.439.1.2 ~rt_nlms_t() rt_nlms_t::~rt_nlms_t ( ) [inline]

5.439.2 Member Function Documentation

5.439.2.1 process() 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 )
```

5.439.2.2 insert() void rt_nlms_t::insert ( )

5.439.3 Member Data Documentation

5.439.3.1 ac MHA_AC::algo_comm_t& rt_nlms_t::ac [private]

5.439.3.2 ntaps `unsigned int rt_nlms_t::ntaps [private]`

5.439.3.3 frames `unsigned int rt_nlms_t::frames [private]`

5.439.3.4 channels `unsigned int rt_nlms_t::channels [private]`

5.439.3.5 F `MHA_AC::waveform_t rt_nlms_t::F [private]`

5.439.3.6 U `MHASignal::waveform_t rt_nlms_t::U [private]`

Input signal cache.

5.439.3.7 Uflt `MHASignal::waveform_t rt_nlms_t::Uflt [private]`

Input signal cache (second filter)

5.439.3.8 Pu `MHASignal::waveform_t rt_nlms_t::Pu [private]`

Power of input signal delayline.

5.439.3.9 fu `MHASignal::waveform_t rt_nlms_t::fu [private]`

Filtered input signal.

5.439.3.10 fuflt `MHASignal::waveform_t rt_nlms_t::fuflt [private]`

Filtered input signal.

5.439.3.11 fu_previous `MHASignal::waveform_t rt_nlms_t::fu_previous [private]`

5.439.3.12 y_previous `MHASignal::waveform_t rt_nlms_t::y_previous [private]`

5.439.3.13 P_Sum `MHASignal::waveform_t rt_nlms_t::P_Sum [private]`

5.439.3.14 name_u_ `std::string rt_nlms_t::name_u_ [private]`

5.439.3.15 name_d_ `std::string rt_nlms_t::name_d_ [private]`

5.439.3.16 name_e_ `std::string rt_nlms_t::name_e_ [private]`

5.439.3.17 n_no_update_ `int rt_nlms_t::n_no_update_ [private]`

5.439.3.18 no_iter `int rt_nlms_t::no_iter [private]`

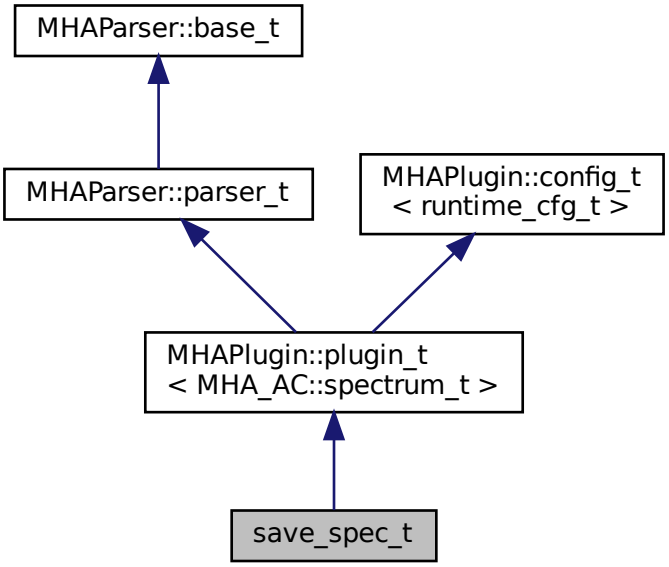
5.439.3.19 s_E `mha_wave_t rt_nlms_t::s_E [private]`

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

- **nlms_wave.cpp**

5.440 save_spec_t Class Reference

Inheritance diagram for save_spec_t:



Public Member Functions

- **save_spec_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗s)
- void **prepare** (**mhaconfig_t** &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.440.1 Constructor & Destructor Documentation

5.440.1.1 save_spec_t()

```
save_spec_t::save_spec_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name ) [inline]
```

5.440.2 Member Function Documentation

5.440.2.1 process()

```
mha_spec_t* save_spec_t::process (
    mha_spec_t * s ) [inline]
```

5.440.2.2 prepare()

```
void save_spec_t::prepare (
    mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin_t< MHA_AC::spectrum_t >** (p. 1301).

5.440.3 Member Data Documentation

5.440.3.1 basename

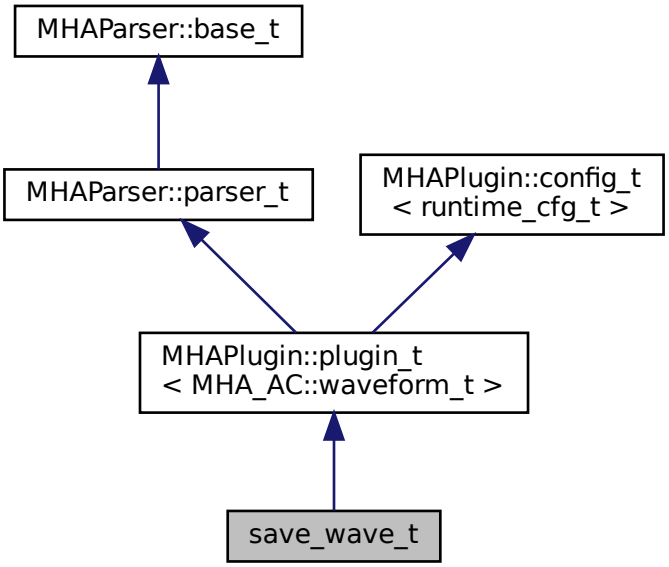
```
std::string save_spec_t::basename [private]
```

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

- **save_spec.cpp**

5.441 save_wave_t Class Reference

Inheritance diagram for save_wave_t:



Public Member Functions

- **save_wave_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
- void **prepare** (**mhaconfig_t** &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.441.1 Constructor & Destructor Documentation

5.441.1.1 save_wave_t() save_wave_t::save_wave_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name) [inline]

5.441.2 Member Function Documentation

5.441.2.1 process() mha_wave_t* save_wave_t::process (mha_wave_t * s) [inline]

5.441.2.2 prepare() void save_wave_t::prepare (mhaconfig_t & tf) [inline], [virtual]

Implements **MHAPlugin::plugin_t**< **MHA_AC::waveform_t** > (p. 1301).

5.441.3 Member Data Documentation

5.441.3.1 basename std::string save_wave_t::basename [private]

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

- **save_wave.cpp**

5.442 ScalerLayer Struct Reference

Public Attributes

- const float * **bias**
- const float * **input_weights**
- **counter nb_inputs**
- **counter nb_neurons**
- **counter activation**

5.442.1 Member Data Documentation

5.442.1.1 bias `const float * ScalerLayer::bias`

5.442.1.2 input_weights `const float * ScalerLayer::input_weights`

5.442.1.3 nb_inputs **`counter`** `ScalerLayer::nb_inputs`

5.442.1.4 nb_neurons **`counter`** `ScalerLayer::nb_neurons`

5.442.1.5 activation **`counter`** `ScalerLayer::activation`

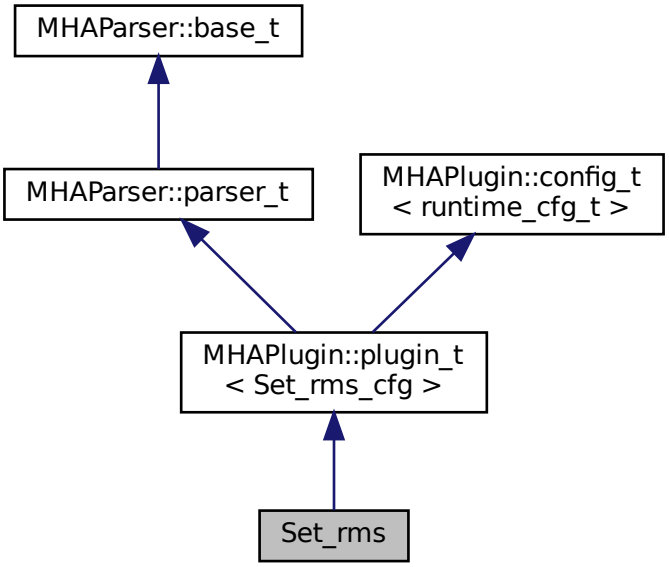
The documentation for this struct was generated from the following file:

- **gcfsnet_bin/rnn.h**

5.443 Set_rms Class Reference

Plugin interface class for seting the channels of an output signal(e.g.

Inheritance diagram for Set_rms:



Public Member Functions

- **Set_rms** (**algo_comm_t** & **ac**, const std::string & **algo_name**)
  Constructor of the plugin interface class.
- void **prepare** (**mhaconfig_t** &signal_info)
  Prepare function of the plugin interface class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal)
  Process function of the plugin interface class.
- void **release** ()
  Release function of the plugin interface class.

Private Member Functions

- void **update_cfg** ()
  Runtime configuration update function of the plugin interface class.

Private Attributes

- std::string **algo_name**

  Name of the algorithm within the plugin chain.
- **MHAParser::string_t ac_name_in**

  Name of the AC variable containing the (exponentially averaged) RMS of the original input signal to be applied to the output signal.
- **MHAParser::string_t ac_name_out**

  Name of the AC variable containing the (exponentially averaged) RMS of the output signal used for normalization.
- **MHAEvents::patchbay_t< Set_rms > patchbay**

  Data member connecting an event emitter (i.e.

Additional Inherited Members

5.443.1 Detailed Description

Plugin interface class for seting the channels of an output signal(e.g.

of some signal processing operation) to the RMS values of the original input signal

5.443.2 Constructor & Destructor Documentation

5.443.2.1 Set_rms()

```
Set_rms::Set_rms (
    algo_comm_t & ac,
  const std::string & algo_name )
```

Constructor of the plugin interface class.

Parameters

ac	Reference to the processing chain structure
algo_name	Reference to the algorithm name

5.443.3 Member Function Documentation

5.443.3.1 prepare() `void Set_rms::prepare (` **`mhaconfig_t`** `& signal_info ) [virtual]`

Prepare function of the plugin interface class.

Parameters

signal_info	Reference to the prepare configuration structure

Implements **MHAPlugin::plugin_t< Set_rms_cfg >** (p. 1301).

5.443.3.2 process() **`mha_wave_t`** `* Set_rms::process (` **`mha_wave_t`** `* signal )`

Process function of the plugin interface class.

Parameters

signal	Pointer to the current input signal fragment

Returns

Pointer to the output signal fragment (with adjusted RMS)

5.443.3.3 release() `void Set_rms::release ( void ) [virtual]`

Release function of the plugin interface class.

Reimplemented from **MHAPlugin::plugin_t< Set_rms_cfg >** (p. 1302).

5.443.3.4 update_cfg() `void Set_rms::update_cfg ( void ) [private]`

Runtime configuration update function of the plugin interface class.

5.443.4 Member Data Documentation

5.443.4.1 algo_name `std::string Set_rms::algo_name [private]`

Name of the algorithm within the plugin chain.

5.443.4.2 ac_name_in `MHAParser::string_t Set_rms::ac_name_in [private]`

Name of the AC variable containing the (exponentially averaged) RMS of the original input signal to be applied to the output signal.

5.443.4.3 ac_name_out `MHAParser::string_t Set_rms::ac_name_out [private]`

Name of the AC variable containing the (exponentially averaged) RMS of the output signal used for normalization.

5.443.4.4 patchbay `MHAEvents::patchbay_t< Set_rms> Set_rms::patchbay [private]`

Data member connecting an event emitter (i.e.

configuration variable) with a callback function of the plugin interface class

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

- **set_rms.hh**
- **set_rms.cpp**

5.444 Set_rms_cfg Class Reference

Runtime configuration class for setting the channels of an output signal (e.g.

Public Member Functions

- **Set_rms_cfg** (std::string ac_name_in, std::string ac_name_out)

  Constructor of the runtime configuration class.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗signal, **mha_wave_t** channel_rms_in, **mha_wave_t** channel_rms_out)

  Process function of the runtime configuration class (main signal processing function).

Private Attributes

- std::string **ac_name_in_cfg**

  Name of the AC variable containing the (exponentially averaged) RMS of the original input signal to be applied to the output signal.
- std::string **ac_name_out_cfg**

  Name of the AC variable containing the (exponentially averaged) RMS of the output signal used for normalization.

5.444.1 Detailed Description

Runtime configuration class for setting the channels of an output signal (e.g.

of some signal processing operation) to the RMS values of the original input signal

5.444.2 Constructor & Destructor Documentation

5.444.2.1 Set_rms_cfg() `Set_rms_cfg::Set_rms_cfg (`

```
std::string ac_name_in,
std::string ac_name_out )
```

Constructor of the runtime configuration class.

Parameters

ac_name_in	Name of the AC variable containing the (exponentially averaged) RMS of the original input signal to be applied to the output signal (cannot be changed at runtime)

Parameters

ac_name_out	Name of the AC variable containing the (exponentially averaged) RMS of the output signal used for normalization (cannot be changed at runtime)

5.444.3 Member Function Documentation

5.444.3.1 process()

```
mha_wave_t * Set_rms_cfg::process (
    mha_wave_t * signal,
    mha_wave_t channel_rms_in,
    mha_wave_t channel_rms_out )
```

Process function of the runtime configuration class (main signal processing function).

It sets the RMS of the output signal fragment to the value specified by the parameter channel↩ _rms_in

Parameters

signal	Pointer to the current input signal fragment
channel_rms_in	Contains a vector of RMS values for each channel the original input signal to be applied to the output signal
channel_rms_out	Contains a vector of RMS values for each channel the output signal used for normalization

Returns

Pointer to the output signal fragment (with adjusted RMS)

5.444.4 Member Data Documentation

5.444.4.1 ac_name_in_cfg

```
std::string Set_rms_cfg::ac_name_in_cfg [private]
```

Name of the AC variable containing the (exponentially averaged) RMS of the original input signal to be applied to the output signal.

5.444.4.2 ac_name_out_cfg `std::string Set_rms_cfg::ac_name_out_cfg [private]`

Name of the AC variable containing the (exponentially averaged) RMS of the output signal used for normalization.

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

- **set_rms.hh**
- **set_rms.cpp**

5.445 shadowfilter_begin::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft, int inch, int outch, **MHA_AC::algo_comm_t** &ac, std::string name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **insert_ac_variables** ()
  Inserts or reinserts AC variables in_spec_copy, nch, ntracks into AC variable space.

Private Attributes

- **MHA_AC::spectrum_t in_spec_copy**
- **MHASignal::spectrum_t out_spec**
- **MHA_AC::int_t nch**
- **MHA_AC::int_t ntracks**

5.445.1 Constructor & Destructor Documentation

5.445.1.1 cfg_t() `cfg_t::cfg_t (`

```
int nfft,
int inch,
int outch,
MHA_AC::algo_comm_t & ac,
std::string name )
```

5.445.2 Member Function Documentation

5.445.2.1 process() `mha_spec_t * cfg_t::process ( mha_spec_t * s )`

5.445.2.2 insert_ac_variables() `void cfg_t::insert_ac_variables ( )`

Inserts or reinserts AC variables in_spec_copy, nch, ntracks into AC variable space.

5.445.3 Member Data Documentation

5.445.3.1 in_spec_copy `MHA_AC::spectrum_t shadowfilter_begin::cfg_t::in_spec_copy [private]`

5.445.3.2 out_spec `MHASignal::spectrum_t shadowfilter_begin::cfg_t::out_spec [private]`

5.445.3.3 nch `MHA_AC::int_t shadowfilter_begin::cfg_t::nch [private]`

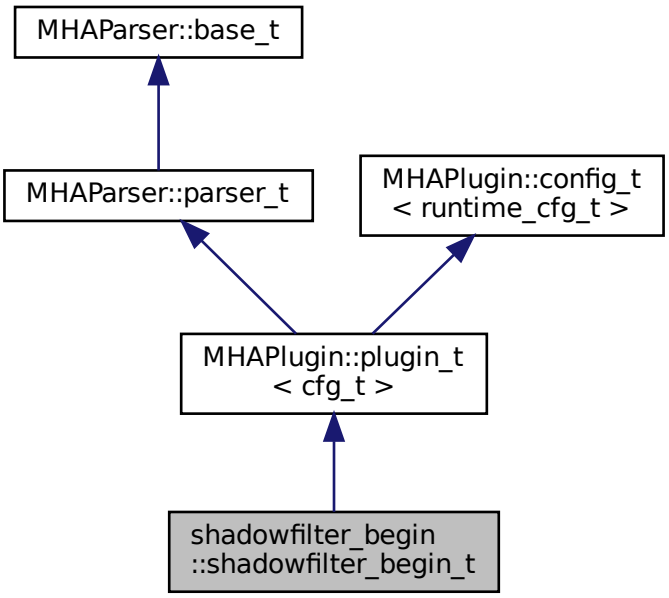
5.445.3.4 ntracks `MHA_AC::int_t shadowfilter_begin::cfg_t::ntracks [private]`

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

- **shadowfilter_begin.cpp**

5.446 shadowfilter_begin::shadowfilter_begin_t Class Reference

Inheritance diagram for shadowfilter_begin::shadowfilter_begin_t:



Public Member Functions

- **shadowfilter_begin_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _name)
- **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

5.446.1 Constructor & Destructor Documentation

5.446.1.1 shadowfilter_begin_t()

```
shadowfilter_begin::shadowfilter_begin_t::shadowfilter_begin_t (
            MHA_AC::algo_comm_t & iac,
          const std::string & configured_name )
```

5.446.2 Member Function Documentation

5.446.2.1 process()

```
mha_spec_t * shadowfilter_begin::shadowfilter_begin_t::process (
            mha_spec_t * s )
```

5.446.2.2 prepare()

```
void shadowfilter_begin::shadowfilter_begin_t::prepare (
            mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.446.3 Member Data Documentation

5.446.3.1 basename

```
std::string shadowfilter_begin::shadowfilter_begin_t::basename [private]
```

5.446.3.2 nch

```
MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::nch [private]
```

5.446.3.3 ntracks

```
MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::ntracks [private]
```

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

- **shadowfilter_begin.cpp**

5.447 shadowfilter_end::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft_, **MHA_AC::algo_comm_t** &ac_, std::string name_)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- **MHA_AC::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**

5.447.1 Constructor & Destructor Documentation

5.447.1.1 cfg_t()

```
cfg_t::cfg_t (
    int nfft_,
    MHA_AC::algo_comm_t & ac_,
    std::string name_ )
```

5.447.2 Member Function Documentation

5.447.2.1 process()

```
mha_spec_t * cfg_t::process (
    mha_spec_t * s )
```

5.447.3 Member Data Documentation

5.447.3.1 ac

`MHA_AC::algo_comm_t& shadowfilter_end::cfg_t::ac [private]`

5.447.3.2 name

`std::string shadowfilter_end::cfg_t::name [private]`

5.447.3.3 nfft

`int shadowfilter_end::cfg_t::nfft [private]`

5.447.3.4 ntracks

`int shadowfilter_end::cfg_t::ntracks [private]`

5.447.3.5 nch_out

`int shadowfilter_end::cfg_t::nch_out [private]`

5.447.3.6 in_spec

`mha_spec_t shadowfilter_end::cfg_t::in_spec [private]`

5.447.3.7 out_spec

`MHASignal::spectrum_t shadowfilter_end::cfg_t::out_spec [private]`

5.447.3.8 gains

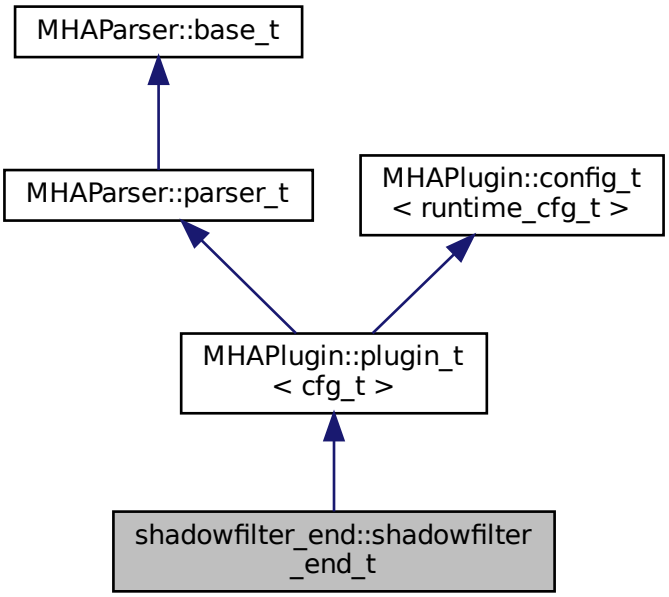
`MHA_AC::spectrum_t shadowfilter_end::cfg_t::gains [private]`

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

- **shadowfilter_end.cpp**

5.448 shadowfilter_end::shadowfilter_end_t Class Reference

Inheritance diagram for shadowfilter_end::shadowfilter_end_t:



Public Member Functions

- **shadowfilter_end_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_↩ name)
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- **MHAParser::string_t basename**

Additional Inherited Members

5.448.1 Constructor & Destructor Documentation

5.448.1.1 shadowfilter_end_t()

```
shadowfilter_end::shadowfilter_end_t::shadowfilter_end_t (
            MHA_AC::algo_comm_t & iac,
          const std::string & configured_name )
```

5.448.2 Member Function Documentation

5.448.2.1 process()

```
mha_spec_t * shadowfilter_end::shadowfilter_end_t::process (
            mha_spec_t * s )
```

5.448.2.2 prepare()

```
void shadowfilter_end::shadowfilter_end_t::prepare (
            mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.448.3 Member Data Documentation

5.448.3.1 basename

```
MHAParser::string_t shadowfilter_end::shadowfilter_end_t::basename [private]
```

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

- **shadowfilter_end.cpp**

5.449 sine_cfg_t Struct Reference

Runtime configuration of the sine plugin.

Public Member Functions

- **sine_cfg_t** (double sampling_rate, **mha_real_t** frequency, **mha_real_t** newlev, int _mix, const std::vector< int > &_channels)

  Constructor computes data members from input parameters.

Public Attributes

- double **phase_increment_div_2pi**
  Phase increment per sample, divided by 2 pi for easier phase wrapping.
- double **amplitude**
  Amplitude of the sinusoid in Pascal.
- int **mix**
  0 for mode replace, 1 for mode mix. Used as factor on input signal.
- const std::vector< int > **channels**
  Indices of affected audio channels.

5.449.1 Detailed Description

Runtime configuration of the sine plugin.

5.449.2 Constructor & Destructor Documentation

5.449.2.1 sine_cfg_t()

```
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]
```

Constructor computes data members from input parameters.

5.449.3 Member Data Documentation

5.449.3.1 phase_increment_div_2pi

```
double sine_cfg_t::phase_increment_div_2pi
```

Phase increment per sample, divided by 2 pi for easier phase wrapping.

5.449.3.2 amplitude `double sine_cfg_t::amplitude`

Amplitude of the sinusoid in Pascal.

5.449.3.3 mix `int sine_cfg_t::mix`

0 for mode replace, 1 for mode mix. Used as factor on input signal.

5.449.3.4 channels `const std::vector<int> sine_cfg_t::channels`

Indices of affected audio channels.

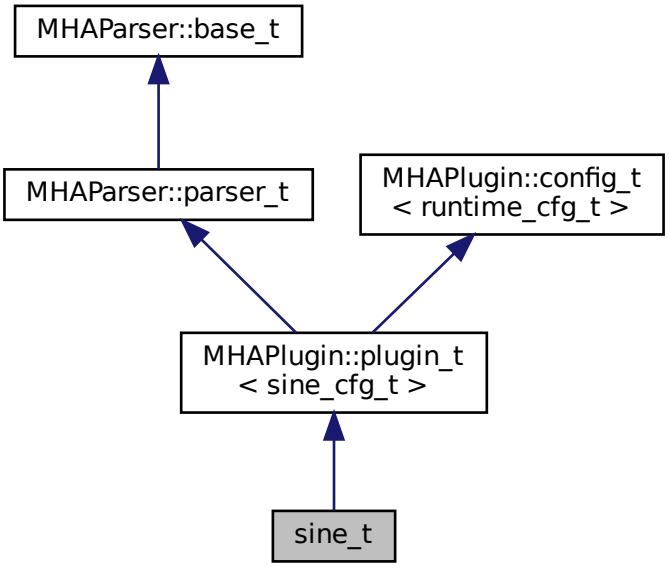
The documentation for this struct was generated from the following file:

- **sine.cpp**

5.450 sine_t Class Reference

Interface class of plugin `sine`, a sinusoid generator plugin.

Inheritance diagram for sine_t:



Public Member Functions

- **sine_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor initializes and connects configuration variables.
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗)
  Computes sinusoid and mixes/replaces input signal.
- void **prepare** (**mhaconfig_t** &)
  Adapts range of channel variable and prepares.
- void **release** ()
  Reset channel range to default.

Private Member Functions

- void **update_cfg** ()
  Computes new runtime configuration.

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

5.450.1 Detailed Description

Interface class of plugin `sine`, a sinusoid generator plugin.

5.450.2 Constructor & Destructor Documentation

5.450.2.1 sine_t() `sine_t::sine_t (`

```
  MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor initializes and connects configuration variables.

5.450.3 Member Function Documentation

5.450.3.1 process() `mha_wave_t * sine_t::process ( mha_wave_t * s )`

Computes sinusoid and mixes/replaces input signal.

If the amplitude has changed since the last process callback, spread out the amplitude change linearly across all samples of the buffer to avoid clicks.

5.450.3.2 prepare() `void sine_t::prepare ( mhaconfig_t & tf ) [virtual]`

Adapts range of channel variable and prepares.

Implements **MHAPlugin::plugin_t< sine_cfg_t >** (p. 1301).

5.450.3.3 release() `void sine_t::release ( void ) [virtual]`

Reset channel range to default.

Reimplemented from **MHAPlugin::plugin_t< sine_cfg_t >** (p. 1302).

5.450.3.4 update_cfg() `void sine_t::update_cfg ( void ) [private]`

Computes new runtime configuration.

5.450.4 Member Data Documentation

5.450.4.1 lev `MHAParser::float_t sine_t::lev [private]`

5.450.4.2 frequency `MHAParser::float_t sine_t::frequency [private]`

5.450.4.3 mode `MHAParser::kw_t sine_t::mode [private]`

5.450.4.4 channels `MHAParser::vint_t sine_t::channels [private]`

5.450.4.5 phase_div_2pi `double sine_t::phase_div_2pi [private]`

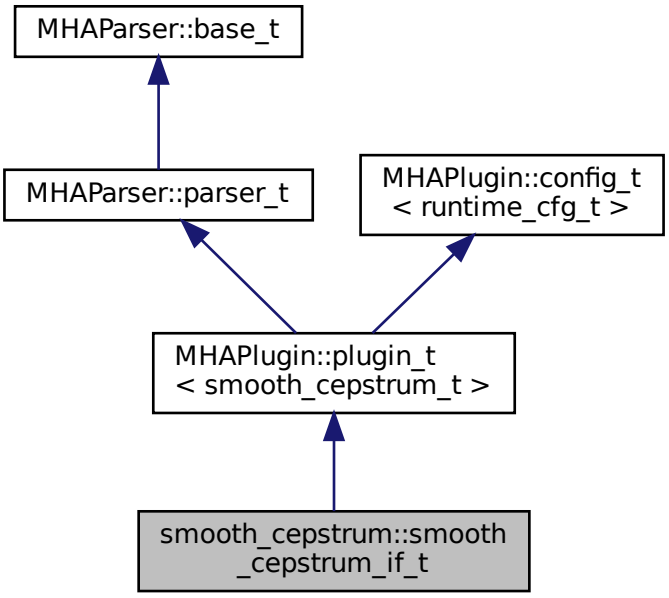
5.450.4.6 patchbay `MHAEvents::patchbay_t< sine_t> sine_t::patchbay [private]`

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

- **sine.cpp**

5.451 smooth_cepstrum::smooth_cepstrum_if_t Class Reference

Inheritance diagram for smooth_cepstrum::smooth_cepstrum_if_t:



Public Member Functions

- **smooth_cepstrum_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured↩ _name)

  Constructs the beamforming plugin.
- **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**< **smooth_cepstrum_if_t** > **patchbay**
- bool **prepared**

Additional Inherited Members

5.451.1 Constructor & Destructor Documentation

5.451.1.1 smooth_cepstrum_if_t() `smooth_cepstrum::smooth_cepstrum_if_t::smooth_cepstrum_if_t (`

```
            MHA_AC::algo_comm_t & iac,
           const std::string & configured_name )
```

Constructs the beamforming plugin.

5.451.2 Member Function Documentation

5.451.2.1 process() `mha_spec_t * smooth_cepstrum::smooth_cepstrum_if_t::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.

5.451.2.2 prepare() `void smooth_cepstrum::smooth_cepstrum_if_t::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< smooth_cepstrum_t >** (p. 1301).

5.451.2.3 release() `void smooth_cepstrum::smooth_cepstrum_if_t::release ( void ) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< smooth_cepstrum_t >** (p. 1302).

5.451.2.4 update_cfg() void smooth_cepstrum::smooth_cepstrum_if_t::update_cfg (void) [private]

5.451.2.5 on_model_param_valuechanged() void smooth_cepstrum::smooth_cepstrum_if_t::on_model_param_valuechanged ( ) [private]

5.451.3 Member Data Documentation

5.451.3.1 xi_min_db MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::xi_min_db [private]

5.451.3.2 f0_low MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::f0_low [private]

5.451.3.3 f0_high MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::f0_high [private]

5.451.3.4 delta_pitch MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::delta_pitch [private]

5.451.3.5 lambda_thresh MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::lambda_thresh [private]

5.451.3.6 alpha_pitch

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t↩
::alpha_pitch [private]

5.451.3.7 beta_const

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t↩
::beta_const [private]

5.451.3.8 kappa_const

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t↩
::kappa_const [private]

5.451.3.9 gain_min_db

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t↩
::gain_min_db [private]

5.451.3.10 win_f0

MHAParser::vfloat_t smooth_cepstrum::smooth_cepstrum_if_t↩
::win_f0 [private]

5.451.3.11 alpha_const_vals

MHAParser::vfloat_t smooth_cepstrum::smooth_cepstrum↩
_if_t::alpha_const_vals [private]

5.451.3.12 alpha_const_limits_hz

MHAParser::vfloat_t smooth_cepstrum::smooth_↩
cepstrum_if_t::alpha_const_limits_hz [private]

5.451.3.13 noisePow_name

MHAParser::string_t smooth_cepstrum::smooth_cepstrum↩
_if_t::noisePow_name [private]

5.451.3.14 spp

MHAParser::parser_t smooth_cepstrum::smooth_cepstrum_if_t::spp [private]

5.451.3.15 prior_q

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::prior_q [private]

5.451.3.16 xi_opt_db

MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::xi_opt_db [private]

5.451.3.17 patchbay

MHAEvents::patchbay_t< **smooth_cepstrum_if_t**> smooth_cepstrum::smooth_cepstrum_if_t::patchbay [private]

5.451.3.18 prepared

bool smooth_cepstrum::smooth_cepstrum_if_t::prepared [private]

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

- **smooth_cepstrum.hh**
- **smooth_cepstrum.cpp**

5.452 smooth_cepstrum::smooth_cepstrum_t Class Reference

Public Member Functions

- **smooth_cepstrum_t** (**MHA_AC::algo_comm_t** & **ac**, **smooth_params** & **params**)
- **smooth_cepstrum_t** (const **smooth_cepstrum_t** &)=delete
- **smooth_cepstrum_t** & **operator=** (const **smooth_cepstrum_t** &)=delete
- ∼**smooth_cepstrum_t** ()
- **mha_spec_t** ∗ **process** (**mha_spec_t** ∗)

Private Attributes

- **MHA_AC::algo_comm_t** & **ac**
- **smooth_params params**
- unsigned int **fftlen**
- **mha_fft_t mha_fft**
- unsigned int **nfreq**
- unsigned int **nchan**
- 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**

5.452.1 Constructor & Destructor Documentation

5.452.1.1 smooth_cepstrum_t() [1/2]

smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (
MHA_AC::algo_comm_t & *ac*,
smooth_params & *params*)

5.452.1.2 smooth_cepstrum_t() [2/2]

smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (
const **smooth_cepstrum_t** &) [delete]

5.452.1.3 ~smooth_cepstrum_t()

smooth_cepstrum::smooth_cepstrum_t::~smooth_cepstrum_t ( )

5.452.2 Member Function Documentation

5.452.2.1 operator=()

smooth_cepstrum_t& smooth_cepstrum::smooth_cepstrum_t::operator=
(
const **smooth_cepstrum_t** &) [delete]

5.452.2.2 process()

mha_spec_t * smooth_cepstrum::smooth_cepstrum_t::process (
mha_spec_t * *noisyFrame*)

5.452.3 Member Data Documentation

5.452.3.1 ac

MHA_AC::algo_comm_t& smooth_cepstrum::smooth_cepstrum_t::ac [private]

5.452.3.2 params

`smooth_params` smooth_cepstrum::smooth_cepstrum_t::params [private]

5.452.3.3 fftlen

unsigned int smooth_cepstrum::smooth_cepstrum_t::fftlen [private]

5.452.3.4 mha_fft

`mha_fft_t` smooth_cepstrum::smooth_cepstrum_t::mha_fft [private]

5.452.3.5 nfreq

unsigned int smooth_cepstrum::smooth_cepstrum_t::nfreq [private]

5.452.3.6 nchan

unsigned int smooth_cepstrum::smooth_cepstrum_t::nchan [private]

5.452.3.7 ola_powspec_scale

float smooth_cepstrum::smooth_cepstrum_t::ola_powspec↩
_scale [private]

5.452.3.8 q_low

float smooth_cepstrum::smooth_cepstrum_t::q_low [private]

5.452.3.9 q_high

float smooth_cepstrum::smooth_cepstrum_t::q_high [private]

5.452.3.10 winF0

`MHASignal::waveform_t` smooth_cepstrum::smooth_cepstrum_t::winF0 [private]

5.452.3.11 xi_min float smooth_cepstrum::smooth_cepstrum_t::xi_min [private]

5.452.3.12 gain_min float smooth_cepstrum::smooth_cepstrum_t::gain_min [private]

5.452.3.13 alpha_const MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t↩ ::alpha_const [private]

5.452.3.14 alpha_prev MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t↩ ::alpha_prev [private]

5.452.3.15 noisePow MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t↩ ::noisePow [private]

5.452.3.16 powSpec MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t↩ ::powSpec [private]

5.452.3.17 gamma_post MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_↩ t::gamma_post [private]

5.452.3.18 xi_ml MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::xi_ml [private]

5.452.3.19 lambda_ml_full

MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::lambda_ml_full [private]

5.452.3.20 lambda_ml_ceps

MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::lambda_ml_ceps [private]

5.452.3.21 lambda_ml_smooth

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::lambda_ml_smooth [private]

5.452.3.22 alpha_hat

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::alpha_hat [private]

5.452.3.23 alpha_frame

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::alpha_frame [private]

5.452.3.24 lambda_ceps

MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::lambda_ceps [private]

5.452.3.25 lambda_ceps_prev

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::lambda_ceps_prev [private]

5.452.3.26 log_lambda_spec

MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::log_lambda_spec [private]

5.452.3.27 lambda_spec

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::lambda_spec [private]

5.452.3.28 xi_est

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::xi_est [private]

5.452.3.29 gain_wiener

MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::gain_wiener [private]

5.452.3.30 spec_out

MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::spec_out [private]

5.452.3.31 max_val

double* smooth_cepstrum::smooth_cepstrum_t::max_val [private]

5.452.3.32 max_q

int* smooth_cepstrum::smooth_cepstrum_t::max_q [private]

5.452.3.33 pitch_set_first

int* smooth_cepstrum::smooth_cepstrum_t::pitch_set_first [private]

5.452.3.34 pitch_set_last

int* smooth_cepstrum::smooth_cepstrum_t::pitch_set_last [private]

5.452.3.35 priorFact `float smooth_cepstrum::smooth_cepstrum_t::priorFact [private]`

5.452.3.36 xiOpt `float smooth_cepstrum::smooth_cepstrum_t::xiOpt [private]`

5.452.3.37 logGLRFact `float smooth_cepstrum::smooth_cepstrum_t::logGLRFact [private]`

5.452.3.38 GLRexp `float smooth_cepstrum::smooth_cepstrum_t::GLRexp [private]`

5.452.3.39 GLR **`MHASignal::waveform_t`** `smooth_cepstrum::smooth_cepstrum_t::GLR [private]`

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

- **smooth_cepstrum.hh**
- **smooth_cepstrum.cpp**

5.453 smooth_cepstrum::smooth_params Class Reference

Public Member Functions

- **smooth_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**

5.453.1 Constructor & Destructor Documentation

5.453.1.1 smooth_params()

```
smooth_cepstrum::smooth_params::smooth_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]
```

5.453.2 Member Data Documentation

5.453.2.1 in_cfg const **mhaconfig_t** smooth_cepstrum::smooth_params::in_cfg

5.453.2.2 xi_min_db float smooth_cepstrum::smooth_params::xi_min_db

5.453.2.3 f0_low float smooth_cepstrum::smooth_params::f0_low

5.453.2.4 f0_high float smooth_cepstrum::smooth_params::f0_high

5.453.2.5 delta_pitch float smooth_cepstrum::smooth_params::delta_pitch

5.453.2.6 lambda_thresh float smooth_cepstrum::smooth_params::lambda_thresh

5.453.2.7 alpha_pitch float smooth_cepstrum::smooth_params::alpha_pitch

5.453.2.8 beta_const float smooth_cepstrum::smooth_params::beta_const

5.453.2.9 kappa_const float smooth_cepstrum::smooth_params::kappa_const

5.453.2.10 prior_q `float smooth_cepstrum::smooth_params::prior_q`

5.453.2.11 xi_opt_db `float smooth_cepstrum::smooth_params::xi_opt_db`

5.453.2.12 gain_min_db `float smooth_cepstrum::smooth_params::gain_min_db`

5.453.2.13 winF0 `std::vector<float> smooth_cepstrum::smooth_params::winF0`

5.453.2.14 alpha_const_vals `std::vector<float> smooth_cepstrum::smooth_params::alpha_const_vals`

5.453.2.15 alpha_const_limits_hz `std::vector<float> smooth_cepstrum::smooth_params::alpha_const_limits_hz`

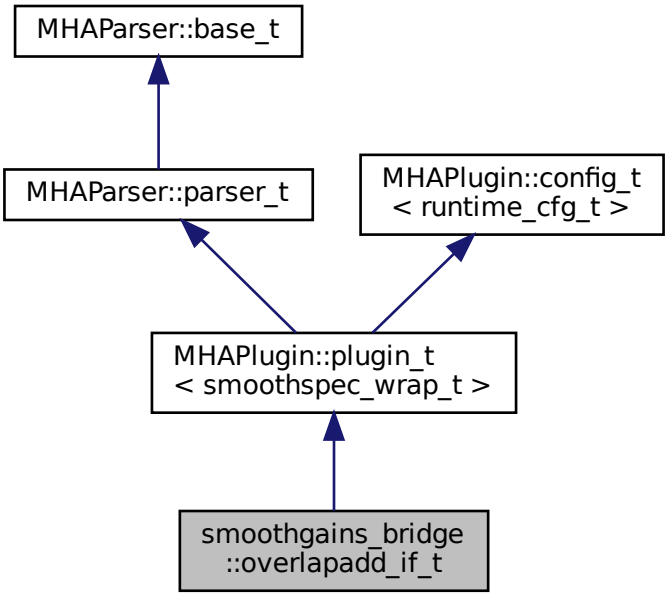
5.453.2.16 noisePow_name `std::string smooth_cepstrum::smooth_params::noisePow_name`

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

- **smooth_cepstrum.hh**

5.454 smoothgains_bridge::overlapadd_if_t Class Reference

Inheritance diagram for smoothgains_bridge::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- ~**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

5.454.1 Constructor & Destructor Documentation

5.454.1.1 overlapadd_if_t() `smoothgains_bridge::overlapadd_if_t::overlapadd_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.454.1.2 ~overlapadd_if_t() `smoothgains_bridge::overlapadd_if_t::~overlapadd_if_t ( )`

5.454.2 Member Function Documentation

5.454.2.1 prepare() `void smoothgains_bridge::overlapadd_if_t::prepare (`

```
mhaconfig_t & t ) [virtual]
```

Implements **MHAPlugin::plugin_t< smoothspec_wrap_t >** (p. 1301).

5.454.2.2 release() `void smoothgains_bridge::overlapadd_if_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< smoothspec_wrap_t >** (p. 1302).

5.454.2.3 process() `mha_spec_t * smoothgains_bridge::overlapadd_if_t::process (`

```
mha_spec_t * spec )
```

5.454.2.4 update() void smoothgains_bridge::overlapadd_if_t::update ( ) [private]

5.454.3 Member Data Documentation

5.454.3.1 patchbay MHAEvents::patchbay_t< overlapadd_if_t> smoothgains_bridge::overlapadd_if_t::patchbay [private]

5.454.3.2 mode MHAParser::kw_t smoothgains_bridge::overlapadd_if_t::mode [private]

5.454.3.3 irswnd MHAParser::window_t smoothgains_bridge::overlapadd_if_t::irswnd [private]

5.454.3.4 epsilon MHAParser::float_t smoothgains_bridge::overlapadd_if_t::epsilon [private]

5.454.3.5 plugloader MHAParser::mhapluginloader_t smoothgains_bridge::overlapadd_if_t::plugloader [private]

5.454.3.6 algo std::string smoothgains_bridge::overlapadd_if_t::algo [private]

5.454.3.7 cf_in mhaconfig_t smoothgains_bridge::overlapadd_if_t::cf_in [private]

5.454.3.8 cf_out `mhaconfig_t smoothgains_bridge::overlapadd_if_t::cf_out [private]`

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

- **smoothgains_bridge.cpp**

5.455 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**

5.455.1 Constructor & Destructor Documentation

5.455.1.1 smoothspec_wrap_t() `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 )
```

5.455.2 Member Function Documentation

5.455.2.1 proc_1()

`mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_1 ( mha_spec_t * s )`

5.455.2.2 proc_2()

`mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_2 ( mha_spec_t * s )`

5.455.3 Member Data Documentation

5.455.3.1 spec_in_copy

`MHASignal::spectrum_t smoothgains_bridge::smoothspec_wrap_t::spec_in_copy [private]`

Copy of input spectrum for smoothspec.

5.455.3.2 smoothspec

`MHAFilter::smoothspec_t smoothgains_bridge::smoothspec_wrap_t::smoothspec [private]`

Smoothspec calculator.

5.455.3.3 use_smoothspec

`bool smoothgains_bridge::smoothspec_wrap_t::use_smoothspec [private]`

5.455.3.4 smoothspec_epsilon

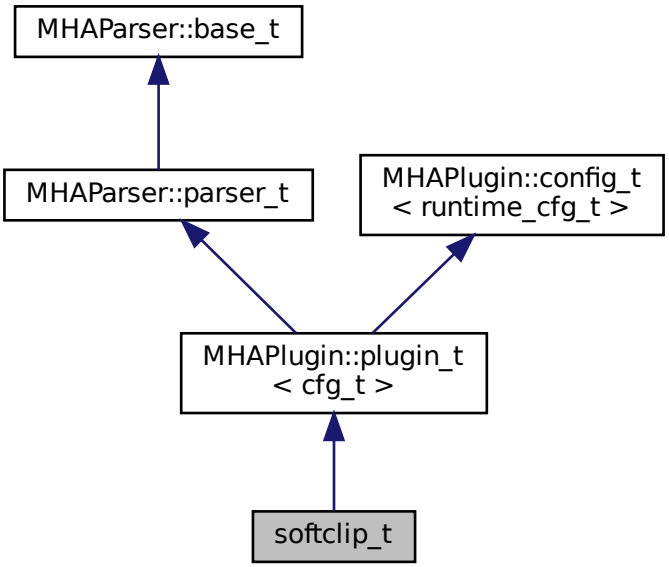
`float smoothgains_bridge::smoothspec_wrap_t::smoothspec_epsilon [private]`

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

- **smoothgains_bridge.cpp**

5.456 softclip_t Class Reference

Inheritance diagram for softclip_t:



Public Member Functions

- **softclip_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.456.1 Constructor & Destructor Documentation

5.456.1.1 softclip_t() `softclip_t::softclip_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.456.2 Member Function Documentation

5.456.2.1 process() `mha_wave_t * softclip_t::process (`

```
mha_wave_t * s )
```

5.456.2.2 prepare() `void softclip_t::prepare (`

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.456.2.3 update() `void softclip_t::update ( )`

5.456.3 Member Data Documentation

5.456.3.1 tftype `mhaconfig_t softclip_t::tftype [private]`

5.456.3.2 attack `MHAParser::float_t softclip_t::attack [private]`

5.456.3.3 decay `MHAParser::float_t softclip_t::decay [private]`

5.456.3.4 start_limit `MHAParser::float_t softclip_t::start_limit [private]`

5.456.3.5 slope_db `MHAParser::float_t softclip_t::slope_db [private]`

5.456.3.6 patchbay `MHAEvents::patchbay_t< softclip_t> softclip_t::patchbay [private]`

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

- **softclip.cpp**

5.457 softclipper_t Class Reference

Soft clipper signal processing implementation.

Public Member Functions

- **softclipper_t** (const **softclipper_variables_t** &v, const **mhaconfig_t** &tf)
  Constructor, copies information from parameters and initializes state.
- **mha_real_t process** (**mha_wave_t** ∗s)
  Process one block of audio signal.

Private Attributes

- **MHAFilter::o1flt_lowpass_t attack**
  Attack filter.
- **MHAFilter::o1flt_maxtrack_t decay**
  Decay filter.
- **MHAFilter::o1flt_lowpass_t clipmeter**
  Clipping ratio filter.
- **mha_real_t threshold**
  Compression onset value.
- **mha_real_t hardlimit**
  Maximum output amplitude of softclipper.
- **mha_real_t slope**
  Compression slope.
- bool **linear**
  Is compression done on linear or log scale.

5.457.1 Detailed Description

Soft clipper signal processing implementation.

5.457.2 Constructor & Destructor Documentation

5.457.2.1 softclipper_t()

```
softclipper_t::softclipper_t (
    const softclipper_variables_t & v,
    const mhaconfig_t & tf )
```

Constructor, copies information from parameters and initializes state.

Parameters

v	Configuration variables of the softclipper
tf	Signal dimensions

5.457.3 Member Function Documentation

5.457.3.1 process() `mha_real_t softclipper_t::process ( mha_wave_t * s )`

Process one block of audio signal.

Parameters

in,out	*s*	Input signal which is modified in-place

5.457.4 Member Data Documentation

5.457.4.1 attack `MHAFilter::o1flt_lowpass_t softclipper_t::attack [private]`

Attack filter.

5.457.4.2 decay `MHAFilter::o1flt_maxtrack_t softclipper_t::decay [private]`

Decay filter.

5.457.4.3 clipmeter `MHAFilter::o1flt_lowpass_t softclipper_t::clipmeter [private]`

Clipping ratio filter.

5.457.4.4 threshold `mha_real_t softclipper_t::threshold [private]`

Compression onset value.

5.457.4.5 hardlimit `mha_real_t softclipper_t::hardlimit [private]`

Maximum output amplitude of softclipper.

5.457.4.6 slope `mha_real_t softclipper_t::slope [private]`

Compression slope.

5.457.4.7 linear `bool softclipper_t::linear [private]`

Is compression done on linear or log scale.

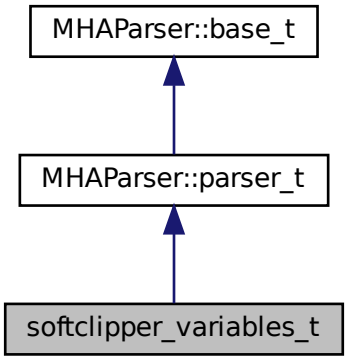
The documentation for this class was generated from the following file:

- **transducers.cpp**

5.458 softclipper_variables_t Class Reference

Parser aggregate of all configuration variables for the output soft clipper.

Inheritance diagram for softclipper_variables_t:



Public Member Functions

- **softclipper_variables_t** ()
  Constructor, initializes all variables and inserts them into ∗this.

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

5.458.1 Detailed Description

Parser aggregate of all configuration variables for the output soft clipper.

5.458.2 Constructor & Destructor Documentation

5.458.2.1 softclipper_variables_t() `softclipper_variables_t::softclipper_variables_t ( )`

Constructor, initializes all variables and inserts them into ∗this.

5.458.3 Member Data Documentation

5.458.3.1 tau_attack `MHAParser::float_t softclipper_variables_t::tau_attack`

5.458.3.2 tau_decay `MHAParser::float_t softclipper_variables_t::tau_decay`

5.458.3.3 tau_clip `MHAParser::float_t softclipper_variables_t::tau_clip`

5.458.3.4 threshold `MHAParser::float_t softclipper_variables_t::threshold`

5.458.3.5 hardlimit `MHAParser::float_t softclipper_variables_t::hardlimit`

5.458.3.6 slope `MHAParser::float_t softclipper_variables_t::slope`

5.458.3.7 linear `MHAParser::bool_t softclipper_variables_t::linear`

5.458.3.8 clipped `MHAParser::float_mon_t softclipper_variables_t::clipped`

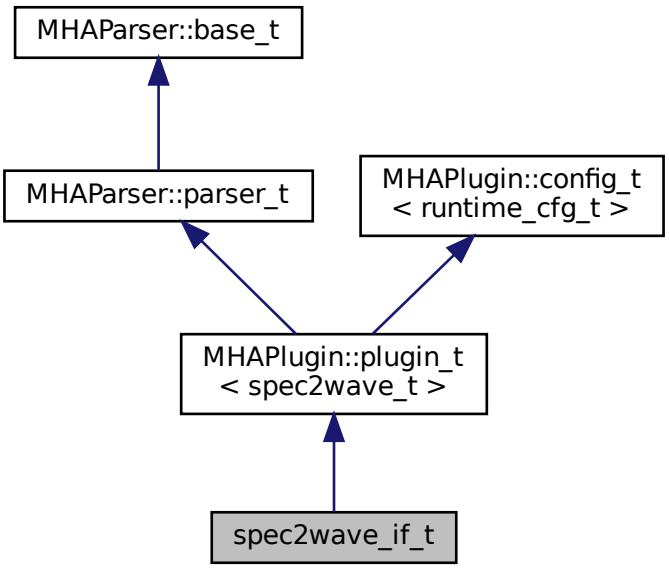
5.458.3.9 max_clipped `MHAParser::float_t softclipper_variables_t::max_clipped`

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

- **transducers.cpp**

5.459 spec2wave_if_t Class Reference

Inheritance diagram for spec2wave_if_t:



Public Member Functions

- **spec2wave_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** ∗ **process** (**mha_spec_t** ∗)

Private Member Functions

- void **update** ()
- void **setlock** (bool b)

Private Attributes

- **MHAParser::float_t ramplen**
- **windowselector_t window_config**

Additional Inherited Members

5.459.1 Constructor & Destructor Documentation

5.459.1.1 spec2wave_if_t() spec2wave_if_t::spec2wave_if_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.459.2 Member Function Documentation

5.459.2.1 prepare() void spec2wave_if_t::prepare (mhaconfig_t & t) [virtual]

Implements **MHAPlugin::plugin_t< spec2wave_t >** (p. 1301).

5.459.2.2 release() void spec2wave_if_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< spec2wave_t >** (p. 1302).

5.459.2.3 process() mha_wave_t * spec2wave_if_t::process (mha_spec_t * spec_in)

5.459.2.4 update() void spec2wave_if_t::update ( ) [private]

5.459.2.5 setlock() `void spec2wave_if_t::setlock ( bool b ) [private]`

5.459.3 Member Data Documentation

5.459.3.1 ramplen `MHAParser::float_t spec2wave_if_t::ramplen [private]`

5.459.3.2 window_config `windowselector_t spec2wave_if_t::window_config [private]`

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

- **spec2wave.cpp**

5.460 spec2wave_t Class Reference

Runtime config class for plugin spec2wave, the counterpart of wave2spec.

Public Member Functions

- **spec2wave_t** (unsigned nfft_, unsigned nwnd_, unsigned nwndshift_, unsigned nch, **mha_real_t** ramplen, const **MHAWindow::base_t** &postwin)
  Constructor.
- ∼**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**

5.460.1 Detailed Description

Runtime config class for plugin spec2wave, the counterpart of wave2spec.

5.460.2 Constructor & Destructor Documentation

5.460.2.1 spec2wave_t() `spec2wave_t::spec2wave_t (`

```
unsigned nfft_,
unsigned nwnd_,
unsigned nwndshift_,
unsigned nch,
 mha_real_t ramplen,
const  MHAWindow::base_t & postwin )
```

Constructor.

Parameters

*nfft_*	FFT length.
*nwnd_*	Overlap-add analysis window length.
*nwndshift_*	Overlap-add hop size. Needed to compute scaling.
nch	Number of audio channels.
ramplen	A user-configurable factor which is applied to the length of the zero paddings in samples before and after the analysis window. Rising and falling hanning ramps are computed for the resulting ramp lengths in member ramps.
postwin	Post windowing function. This window is applied in addition to the hanning ramps. It is applied to the whole output waveform computed by the inverse FFT before the signal is overlap-added to the previous output.

5.460.2.2 ~spec2wave_t() `spec2wave_t::~spec2wave_t ( )`

5.460.3 Member Function Documentation

5.460.3.1 process() `mha_wave_t * spec2wave_t::process ( mha_spec_t * spec_in )`

5.460.4 Member Data Documentation

5.460.4.1 ft `mha_fft_t spec2wave_t::ft [private]`

FFT class.

5.460.4.2 npad1 `unsigned int spec2wave_t::npad1 [private]`

length of zero padding before window

5.460.4.3 npad2 `unsigned int spec2wave_t::npad2 [private]`

length of zero padding after window

5.460.4.4 ramps `hanning_ramps_t spec2wave_t::ramps [private]`

5.460.4.5 calc_out `MHASignal::waveform_t spec2wave_t::calc_out [private]`

5.460.4.6 out_buf `MHASignal::waveform_t spec2wave_t::out_buf [private]`

5.460.4.7 write_buf `MHASignal::waveform_t spec2wave_t::write_buf [private]`

5.460.4.8 sc `mha_real_t spec2wave_t::sc [private]`

5.460.4.9 nfft `unsigned int spec2wave_t::nfft [private]`

5.460.4.10 nwndshift `unsigned int spec2wave_t::nwndshift [private]`

5.460.4.11 postwindow `MHAWindow::base_t spec2wave_t::postwindow [private]`

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

- **spec2wave.cpp**

5.461 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**

5.461.1 Constructor & Destructor Documentation

5.461.1.1 spec_fader_t()

```
spec_fader_t::spec_fader_t (
 unsigned int ch,
  mha_real_t fr,
  MHAParser::vfloat_t & ng,
  MHAParser::float_t & t )
```

5.461.1.2 ~spec_fader_t()

```
spec_fader_t::~spec_fader_t ( ) [inline]
```

5.461.2 Member Data Documentation

5.461.2.1 nch

```
unsigned int spec_fader_t::nch
```

5.461.2.2 gains

```
mha_real_t* spec_fader_t::gains
```

5.461.2.3 fr

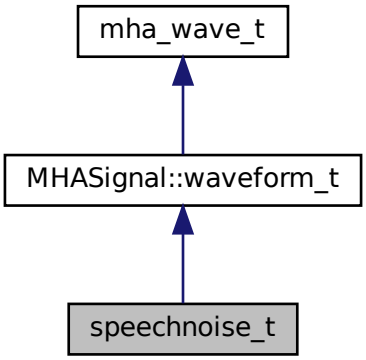
```
unsigned int spec_fader_t::fr
```

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

- **fader_spec.cpp**

5.462 speechnoise_t Class Reference

Inheritance diagram for speechnoise_t:



Public Types

- enum **noise_type_t** {
  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** }

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

5.462.1 Member Enumeration Documentation

5.462.1.1 noise_type_t `enum` `speechnoise_t::noise_type_t`

Enumerator

mha	
olnoise	
LTASS↩ _↩ combined	
LTASS↩ _female	
LTASS↩ _male	
white	
pink	
brown	

Enumerator

TEN_↩ SPL	
TEN_↩ SPL_↩ 250_8k	
TEN_↩ SPL_↩ 50_16k	
sin125	
sin250	
sin500	
sin1k	
sin2k	
sin4k	
sin8k	

5.462.2 Constructor & Destructor Documentation

5.462.2.1 speechnoise_t() [1/2]

```
speechnoise_t::speechnoise_t (
    float duration,
    float srate,
    unsigned int channels,
    speechnoise_t::noise_type_t noise_type = speechnoise_t::mha )
```

5.462.2.2 speechnoise_t() [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 )
```

5.462.3 Member Function Documentation

5.462.3.1 creator()

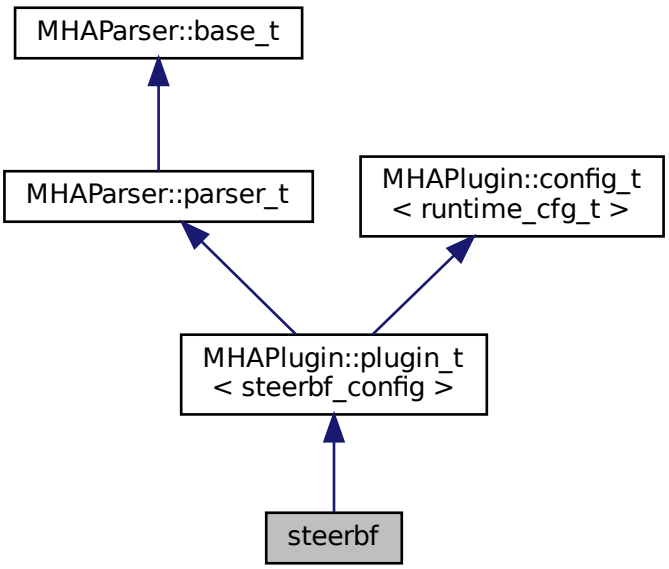
```
void speechnoise_t::creator (
    speechnoise_t::noise_type_t noise_type,
    float srate ) [private]
```

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

- **speechnoise.h**
- **speechnoise.cpp**

5.463 steerbf Class Reference

Inheritance diagram for steerbf:



Public Member Functions

- **steerbf** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_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

5.463.1 Constructor & Destructor Documentation

5.463.1.1 steerbf() steerbf::steerbf (MHA_AC::algo_comm_t & *iac*, const std::string & *configured_name*)

Constructs our plugin.

5.463.1.2 ~steerbf() steerbf::~steerbf ( )

5.463.2 Member Function Documentation

5.463.2.1 process() mha_spec_t * steerbf::process (mha_spec_t * *signal*)

Defers to configuration class.

5.463.2.2 prepare() 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. 1301).

5.463.2.3 release() void steerbf::release (void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t< steerbf_config >** (p. 1302).

5.463.2.4 update_cfg() void steerbf::update_cfg (void) [private]

5.463.3 Member Data Documentation

5.463.3.1 bf_src MHAParser::string_t steerbf::bf_src

5.463.3.2 angle_ind parser_int_dyn steerbf::angle_ind

5.463.3.3 angle_src `MHAParser::string_t steerbf::angle_src`

5.463.3.4 patchbay `MHAEvents::patchbay_t< steerbf> steerbf::patchbay [private]`

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

- **steerbf.h**
- **steerbf.cpp**

5.464 steerbf_config Class Reference

Public Member Functions

- **steerbf_config** (**MHA_AC::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**
- **MHA_AC::algo_comm_t** & **ac**
- std::string **bf_src_copy**

5.464.1 Constructor & Destructor Documentation

5.464.1.1 steerbf_config()

```
steerbf_config::steerbf_config (
    MHA_AC::algo_comm_t & ac,
  const  mhaconfig_t in_cfg,
    steerbf * steerbf )
```

5.464.1.2 ~steerbf_config() steerbf_config::~steerbf_config ( )

5.464.2 Member Function Documentation

5.464.2.1 process() **mha_spec_t** * steerbf_config::process (**mha_spec_t** * *inSpec*)

5.464.3 Member Data Documentation

5.464.3.1 nchan unsigned int steerbf_config::nchan [private]

5.464.3.2 nfreq unsigned int steerbf_config::nfreq [private]

5.464.3.3 outSpec **MHASignal::spectrum_t** steerbf_config::outSpec [private]

5.464.3.4 bf_vec **mha_spec_t** steerbf_config::bf_vec [private]

5.464.3.5 nangle unsigned int steerbf_config::nangle [private]

5.464.3.6 _steerbf **steerbf*** steerbf_config::_steerbf [private]

5.464.3.7 ac `MHA_AC::algo_comm_t& steerbf_config::ac [private]`

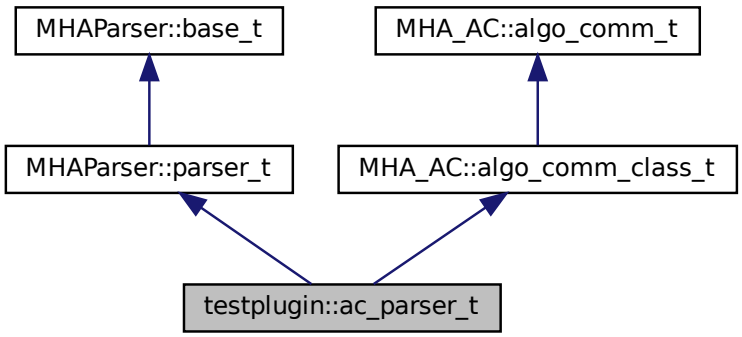
5.464.3.8 bf_src_copy `std::string steerbf_config::bf_src_copy [private]`

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

- **steerbf.h**
- **steerbf.cpp**

5.465 testplugin::ac_parser_t Class Reference

Inheritance diagram for testplugin::ac_parser_t:



Public Types

- enum **data_type_t** {
  _MHA_AC_CHAR , **_MHA_AC_INT** , **_MHA_AC_MHAREAL** , **_MHA_AC_FLOAT** ,
  _MHA_AC_DOUBLE , **_MHA_AC_MHACOMPLEX** , **_unknown** }

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

5.465.1 Member Enumeration Documentation

5.465.1.1 data_type_t 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	

5.465.2 Constructor & Destructor Documentation

5.465.2.1 ac_parser_t() testplugin::ac_parser_t::ac_parser_t ( ) [inline]

5.465.3 Member Function Documentation

5.465.3.1 do_insert_var() 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

5.465.3.2 do_get_var() void testplugin::ac_parser_t::do_get_var ( ) [inline]

5.465.4 Member Data Documentation

5.465.4.1 insert_var MHAParser::string_t testplugin::ac_parser_t::insert_var

5.465.4.2 get_var MHAParser::string_t testplugin::ac_parser_t::get_var

5.465.4.3 data_type MHAParser::kw_t testplugin::ac_parser_t::data_type

5.465.4.4 num_entries MHAParser::int_t testplugin::ac_parser_t::num_entries

5.465.4.5 stride

MHAParser::int_t testplugin::ac_parser_t::stride

5.465.4.6 char_data

MHAParser::string_t testplugin::ac_parser_t::char_data

5.465.4.7 int_data

MHAParser::vint_t testplugin::ac_parser_t::int_data

5.465.4.8 float_data

MHAParser::vfloat_t testplugin::ac_parser_t::float_data

5.465.4.9 complex_data

MHAParser::vcomplex_t testplugin::ac_parser_t::complex_↩data

5.465.4.10 patchbay

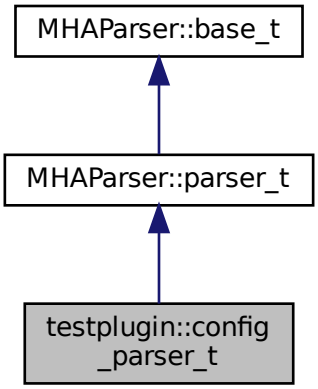
MHAEvents::patchbay_t< **ac_parser_t**> testplugin::ac_parser_↩t::patchbay

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

- **testplugin.cpp**

5.466 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

5.466.1 Constructor & Destructor Documentation

5.466.1.1 config_parser_t() `testplugin::config_parser_t::config_parser_t ( ) [inline]`

5.466.2 Member Function Documentation

5.466.2.1 setlock() `void testplugin::config_parser_t::setlock ( const bool & b ) [inline]`

5.466.2.2 get() **`mhaconfig_t`** `testplugin::config_parser_t::get ( ) const [inline]`

5.466.2.3 set() void testplugin::config_parser_t::set (mhaconfig_t c) [inline]

5.466.3 Member Data Documentation

5.466.3.1 channels MHAParser::int_t testplugin::config_parser_t::channels

5.466.3.2 domain MHAParser::kw_t testplugin::config_parser_t::domain

5.466.3.3 fragsize MHAParser::int_t testplugin::config_parser_t::fragsize

5.466.3.4 wndlen MHAParser::int_t testplugin::config_parser_t::wndlen

5.466.3.5 fftlen MHAParser::int_t testplugin::config_parser_t::fftlen

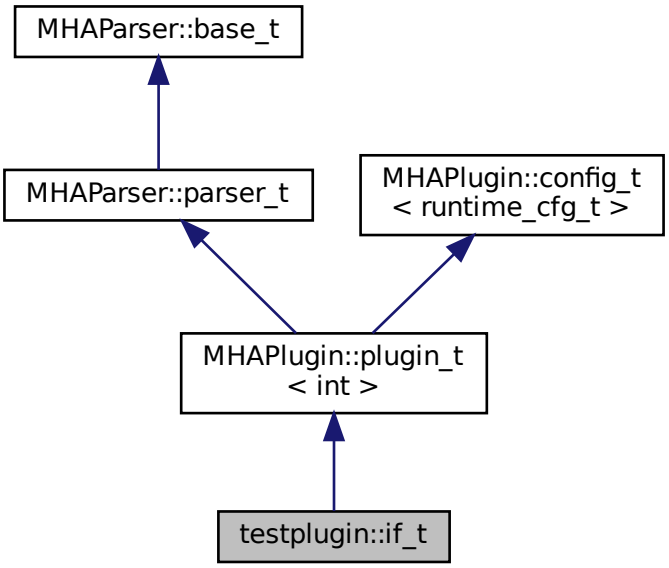
5.466.3.6 srate MHAParser::float_t testplugin::config_parser_t::srate

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

- **testplugin.cpp**

5.467 testplugin::if_t Class Reference

Inheritance diagram for testplugin::if_t:



Public Member Functions

- **if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.467.1 Constructor & Destructor Documentation

5.467.1.1 if_t() `testplugin::if_t::if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.467.2 Member Function Documentation

5.467.2.1 process() [1/2] `mha_spec_t* testplugin::if_t::process (`

```
mha_spec_t * s_in ) [inline]
```

5.467.2.2 process() [2/2] `mha_wave_t* testplugin::if_t::process (`

```
mha_wave_t * s_in ) [inline]
```

5.467.2.3 prepare() `void testplugin::if_t::prepare (`

```
mhaconfig_t & ) [inline], [virtual]
```

Implements **MHAPlugin::plugin_t**< **int** > (p. 1301).

5.467.2.4 test_prepare() `void testplugin::if_t::test_prepare ( ) [private]`

5.467.2.5 test_process() `void testplugin::if_t::test_process ( ) [private]`

5.467.3 Member Data Documentation

5.467.3.1 config_in `config_parser_t testplugin::if_t::config_in [private]`

5.467.3.2 config_out `config_parser_t testplugin::if_t::config_out [private]`

5.467.3.3 ac `ac_parser_t testplugin::if_t::ac [private]`

5.467.3.4 signal `signal_parser_t testplugin::if_t::signal [private]`

5.467.3.5 _prepare `MHAParser::bool_t testplugin::if_t::_prepare [private]`

5.467.3.6 patchbay `MHAEvents::patchbay_t< if_t> testplugin::if_t::patchbay [private]`

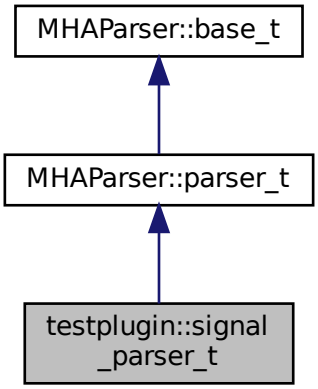
5.467.3.7 plug `MHAParser::mhapluginloader_t testplugin::if_t::plug [private]`

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

- **testplugin.cpp**

5.468 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

5.468.1 Constructor & Destructor Documentation

5.468.1.1 signal_parser_t() `testplugin::signal_parser_t::signal_parser_t ( ) [inline]`

5.468.2 Member Data Documentation

5.468.2.1 input_wave `MHAParser::mfloat_t testplugin::signal_parser_t::input_wave`

5.468.2.2 input_spec `MHAParser::mcomplex_t testplugin::signal_parser_t::input_spec`

5.468.2.3 output_wave `MHAParser::mfloat_mon_t testplugin::signal_parser_t::output_wave`

5.468.2.4 output_spec `MHAParser::mcomplex_mon_t testplugin::signal_parser_t::output_spec`

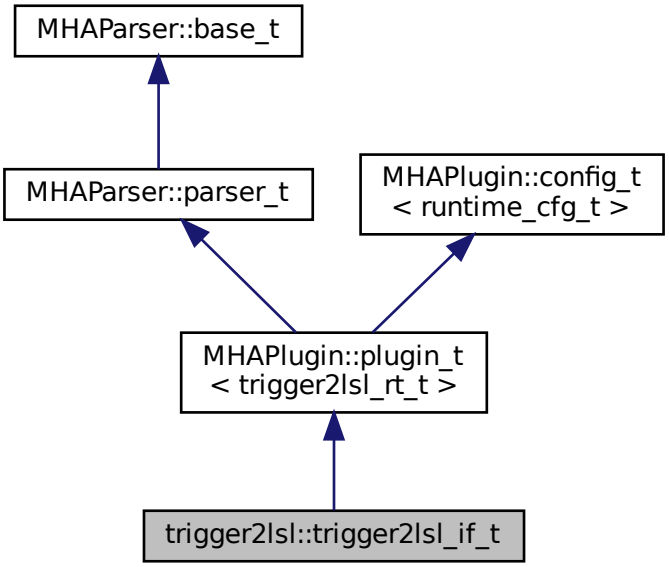
The documentation for this class was generated from the following file:

- **testplugin.cpp**

5.469 trigger2lsl::trigger2lsl_if_t Class Reference

Plugin interface class of plugin **trigger2lsl** (p. 166).

Inheritance diagram for trigger2lsl::trigger2lsl_if_t:



Public Member Functions

- template<class mha_signal_t >
  mha_signal_t * **process** (mha_signal_t *s)
- void **prepare** (**mhaconfig_t** &cf)
  Prepare callback.
- void **release** ()
  Ensure recorded data is flushed to disk.
- **trigger2lsl_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Plugin interface constructor.

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **trigger2lsl_if_t** > **patchbay**
- **MHAParser::string_t** **rising_edge** {"Marker string to be sent when a rising edge is detected","START"}

- **MHAParser::string_t falling_edge** {"Marker string to be sent when a falling edge is detected","STOP"}
- **MHAParser::float_t threshold** {"Threshold","0.5","[0,]"}
- **MHAParser::int_t channel** {"Channel index where edge detection should be run","0","[0,]"}
- **MHAParser::string_t stream_name** {"Name of the output stream",""}
- **MHAParser::bool_t use_edge_position** {"Offset timestamp by position of edge within block","no"}
- **MHAParser::int_t min_debounce** {"Number of consecutive samples the **threshold** must have been crossed before a trigger is issued","3","[0,]"}

Additional Inherited Members

5.469.1 Detailed Description

Plugin interface class of plugin **trigger2lsl** (p. 166).

5.469.2 Constructor & Destructor Documentation

5.469.2.1 trigger2lsl_if_t() `trigger2lsl_if_t::trigger2lsl_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Plugin interface constructor.

Parameters

iac	Algorithm communication variable space.

5.469.3 Member Function Documentation

5.469.3.1 process() template<class mha_signal_t > mha_signal_t * trigger2lsl_if_t::process (mha_signal_t * s)

5.469.3.2 prepare() void trigger2lsl_if_t::prepare (mhaconfig_t & cf) [virtual]

Prepare callback.

trigger2lsl (p. 166) does not modify the signal parameters.

Parameters

cf	The signal parameters.

Implements **MHAPlugin::plugin_t**< **trigger2lsl_rt_t** > (p. 1301).

5.469.3.3 release() void trigger2lsl_if_t::release (void) [virtual]

Ensure recorded data is flushed to disk.

Reimplemented from **MHAPlugin::plugin_t**< **trigger2lsl_rt_t** > (p. 1302).

5.469.3.4 update() void trigger2lsl_if_t::update ( ) [private]

5.469.4 Member Data Documentation

5.469.4.1 patchbay **MHAEvents::patchbay_t**< **trigger2lsl_if_t**> trigger2lsl::trigger2lsl_if_t::patchbay [private]

5.469.4.2 rising_edge

MHAParser::string_t trigger2lsl::trigger2lsl_if_t::rising_edge {"Marker string to be sent when a rising edge is detected","START"} [private]

5.469.4.3 falling_edge

MHAParser::string_t trigger2lsl::trigger2lsl_if_t::falling_edge {"Marker string to be sent when a falling edge is detected","STOP"} [private]

5.469.4.4 threshold

MHAParser::float_t trigger2lsl::trigger2lsl_if_t::threshold {"Threshold","0.5","[0,]"} [private]

5.469.4.5 channel

MHAParser::int_t trigger2lsl::trigger2lsl_if_t::channel {"Channel index where edge detection should be run","0","[0,]"} [private]

5.469.4.6 stream_name

MHAParser::string_t trigger2lsl::trigger2lsl_if_t::stream_name {"Name of the output stream",""} [private]

5.469.4.7 use_edge_position

MHAParser::bool_t trigger2lsl::trigger2lsl_if_t::use_edge_position {"Offset timestamp by position of edge within block","no"} [private]

5.469.4.8 min_debounce

MHAParser::int_t trigger2lsl::trigger2lsl_if_t::min_debounce {"Number of consecutive samples the **threshold** must have been crossed before a trigger is issued","3","[0,]"} [private]

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

- **trigger2lsl.hh**
- **trigger2lsl.cpp**

5.470 trigger2lsl::trigger2lsl_rt_t Class Reference

real-time configuration class for **trigger2lsl** (p. 166) plugin

Public Member Functions

- **trigger2lsl_rt_t** (const std::string &stream_name_, const std::string &rising_edge_↩ , const std::string &falling_edge_, **mha_real_t** threshold_, int channel_, **mha_real_t** sampling_rate_, bool use_edge_position_, int min_debounce_)

  C'tor of rt configuration.
- void **process** (**mha_wave_t** ∗wave)

Private Attributes

- lsl::stream_outlet **stream**

  Outlet stream.
- const std::string **rising_edge**

  String to be sent when a rising edge is detected.
- const std::string **falling_edge**

  String to be sent when a falling edge is detected.
- const **mha_real_t** **threshold**

  Threshold for state transition.
- const int **channel**

  Channel number where to look for threshold crossings.
- bool **state** =false

  Current state.
- const **mha_real_t** **sampling_rate**

  Sampling rate of the input signal.
- const bool **use_edge_position** =true

  Flag wether to use the position of the edge within the signal block to correct the timestamp of the output marker.
- const int **min_debounce**

  Minimum number of consecutive samples that need to cross the threshold to initiate a state transition.
- int **debounce_counter** =0

  Debounce counter.

5.470.1 Detailed Description

real-time configuration class for **trigger2lsl** (p. 166) plugin

5.470.2 Constructor & Destructor Documentation

5.470.2.1 trigger2lsl_rt_t() `trigger2lsl_rt_t::trigger2lsl_rt_t (`

```
const std::string & stream_name_,
const std::string & rising_edge_,
const std::string & falling_edge_,
 mha_real_t threshold_,
int channel_,
 mha_real_t sampling_rate_,
bool use_edge_position_,
int min_debounce_ )
```

C'tor of rt configuration.

Parameters

*stream_name_*	Name of the output stream
*rising_edge_*	String to be sent on detection of a rising edge
*falling_edge_*	String to be sent on detection of a falling edge
*threshold_*	Threshold for state transition
*channel_*	Channel index where to look for threshold crossi
*sampling_rate_*	Sampling rate of the input signal. Needed for tim offeset correction
*use_edge_position_*	Flag wether to use the position of the edge withir signal block to correct the timestamp of the outpu marker
*min_debounce_*	Minimum number of consecutive samples that ne cross the threshold to initiate a state transition

5.470.3 Member Function Documentation

5.470.3.1 process() `void trigger2lsl_rt_t::process (`

```
mha_wave_t * wave )
```

5.470.4 Member Data Documentation

5.470.4.1 stream `lsl::stream_outlet trigger2lsl::trigger2lsl_rt_t::stream [private]`

Outlet stream.

5.470.4.2 rising_edge `const std::string trigger2lsl::trigger2lsl_rt_t::rising_edge [private]`

String to be sent when a rising edge is detected.

5.470.4.3 falling_edge `const std::string trigger2lsl::trigger2lsl_rt_t::falling_edge [private]`

String to be sent when a falling edge is detected.

5.470.4.4 threshold `const mha_real_t trigger2lsl::trigger2lsl_rt_t::threshold [private]`

Threshold for state transition.

5.470.4.5 channel `const int trigger2lsl::trigger2lsl_rt_t::channel [private]`

Channel number where to look for threshold crossings.

5.470.4.6 state `bool trigger2lsl::trigger2lsl_rt_t::state =false [private]`

Current state.

false means HIGH and true means LOW. LOW state means we are below the threshold, looking for rising edges, HIGH state means we are above, looking for falling edges.

5.470.4.7 sampling_rate `const mha_real_t trigger2lsl::trigger2lsl_rt_t::sampling_rate [private]`

Sampling rate of the input signal.

Needed for timestamp offeset correction

5.470.4.8 use_edge_position `const bool trigger2lsl::trigger2lsl_rt_t::use_edge_position =true [private]`

Flag wether to use the position of the edge within the signal block to correct the timestamp of the output marker.

5.470.4.9 min_debounce `const int trigger2lsl::trigger2lsl_rt_t::min_debounce [private]`

Minimum number of consecutive samples that need to cross the threshold to initiate a state transition.

5.470.4.10 debounce_counter `int trigger2lsl::trigger2lsl_rt_t::debounce_counter =0 [private]`

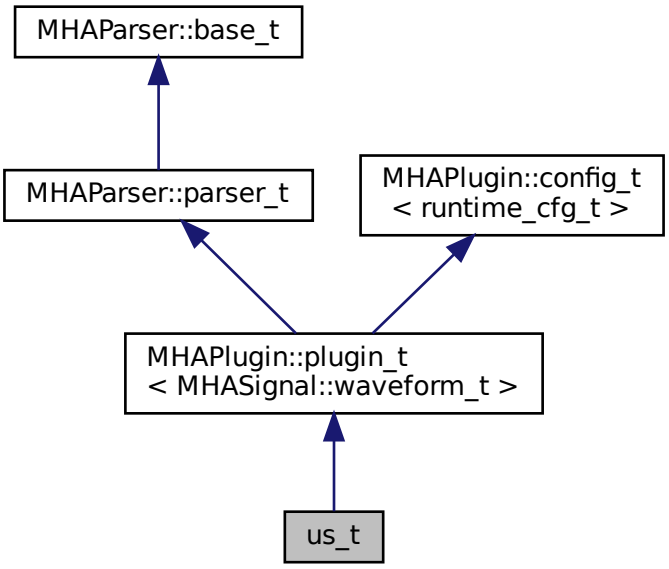
Debounce counter.

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

- **trigger2lsl.hh**
- **trigger2lsl.cpp**

5.471 us_t Class Reference

Inheritance diagram for us_t:



Public Member Functions

- **us_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- **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

5.471.1 Constructor & Destructor Documentation

5.471.1.1 us_t() us_t::us_t (MHA_AC::algo_comm_t & iac, const std::string & configured_name)

5.471.2 Member Function Documentation

5.471.2.1 process() mha_wave_t * us_t::process (mha_wave_t * s)

5.471.2.2 prepare() void us_t::prepare (mhaconfig_t & cf) [virtual]

Implements **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 1301).

5.471.2.3 release() void us_t::release (void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. 1302).

5.471.3 Member Data Documentation

5.471.3.1 ratio MHAParser::int_t us_t::ratio [private]

5.471.3.2 antialias MHAFilter::iir_filter_t us_t::antialias [private]

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

- **upsample.cpp**

5.472 wave2lsl::cfg_t Class Reference

Runtime configuration class of the **wave2lsl** (p. 166) plugin.

Public Member Functions

- **cfg_t** (unsigned skip_, unsigned num_channels_, unsigned num_samples_, const std::string &source_id, const std::string varname_, double rate)

  *C'tor of **wave2lsl** (p. 166) run time configuration.*
- void **process** (**mha_wave_t** *s)

Private Attributes

- unsigned **skipcnt**

  Counter of frames to skip.
- const unsigned **skip**

  Number of frames to skip after each send.
- lsl::stream_info **info**

  LSL stream info.
- lsl::stream_outlet **stream**

  LSL stream outlet.

5.472.1 Detailed Description

Runtime configuration class of the **wave2lsl** (p. 166) plugin.

5.472.2 Constructor & Destructor Documentation

5.472.2.1 cfg_t() cfg_t::cfg_t (

```
unsigned skip_,
unsigned num_channels_,
unsigned num_samples_,
const std::string & source_id,
const std::string varname_,
double rate )
```

C'tor of **wave2lsl** (p. 166) run time configuration.

Parameters

*skip_*	Number of frames to skip after each send
*num_channels_*	Number of channels in the LSL stream
*num_samples_*	Number of samples within one frame
*source_id_*	LSL identifier for this data stream
*varname_*	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_

5.472.3 Member Function Documentation

5.472.3.1 process() void cfg_t::process (

```
mha_wave_t * s )
```

5.472.4 Member Data Documentation

5.472.4.1 skipcnt unsigned wave2lsl::cfg_t::skipcnt [private]

Counter of frames to skip.

5.472.4.2 skip `const unsigned wave2lsl::cfg_t::skip [private]`

Number of frames to skip after each send.

5.472.4.3 info `lsl::stream_info wave2lsl::cfg_t::info [private]`

LSL stream info.

5.472.4.4 stream `lsl::stream_outlet wave2lsl::cfg_t::stream [private]`

LSL stream outlet.

Interface to lsl

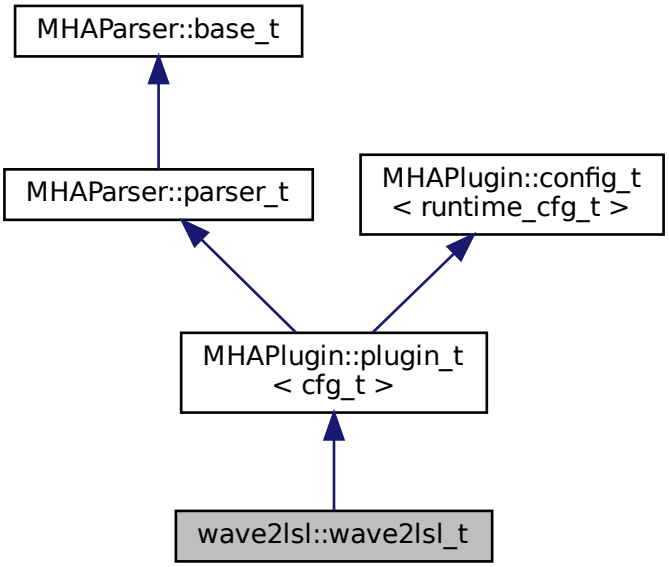
The documentation for this class was generated from the following file:

- **wave2lsl.cpp**

5.473 wave2lsl::wave2lsl_t Class Reference

Plugin class of **wave2lsl** (p. 166).

Inheritance diagram for wave2lsl::wave2lsl_t:



Public Member Functions

- **wave2lsl_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
- void **prepare** (**mhaconfig_t** &)
  *Prepare locks the configuration, then calls **update()** (p. 1687).*
- **mha_wave_t** ∗ **process** (**mha_wave_t** ∗s)
  Processing fct for waveforms.
- void **release** ()
  Release fct.

Private Member Functions

- void **update** ()
  Construct new runtime configuration.

Private Attributes

- **MHAParser::string_t name**
- **MHAParser::string_t source_id**
- **MHAParser::bool_t rt_strict**
- **MHAParser::bool_t activate**
- **MHAParser::int_t skip**
- **MHAEvents::patchbay_t**< **wave2lsl_t** > **patchbay**
- bool **is_first_run**

Additional Inherited Members

5.473.1 Detailed Description

Plugin class of **wave2lsl** (p. 166).

5.473.2 Constructor & Destructor Documentation

5.473.2.1 wave2lsl_t() `wave2lsl::wave2lsl_t::wave2lsl_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.473.3 Member Function Documentation

5.473.3.1 prepare() `void wave2lsl::wave2lsl_t::prepare ( mhaconfig_t & ) [virtual]`

Prepare locks the configuration, then calls **update()** (p. 1687).

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.473.3.2 process() `mha_wave_t * wave2lsl::wave2lsl_t::process ( mha_wave_t * s )`

Processing fct for waveforms.

Calls process of the cfg class.

5.473.3.3 release() `void wave2lsl::wave2lsl_t::release ( void ) [virtual]`

Release fct.

Unlocks the configuration

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1302).

5.473.3.4 update() `void wave2lsl::wave2lsl_t::update ( ) [private]`

Construct new runtime configuration.

5.473.4 Member Data Documentation

5.473.4.1 name

`MHAParser::string_t wave2lsl::wave2lsl_t::name [private]`

5.473.4.2 source_id

`MHAParser::string_t wave2lsl::wave2lsl_t::source_id [private]`

5.473.4.3 rt_strict

`MHAParser::bool_t wave2lsl::wave2lsl_t::rt_strict [private]`

5.473.4.4 activate

`MHAParser::bool_t wave2lsl::wave2lsl_t::activate [private]`

5.473.4.5 skip

`MHAParser::int_t wave2lsl::wave2lsl_t::skip [private]`

5.473.4.6 patchbay

`MHAEvents::patchbay_t< wave2lsl_t> wave2lsl::wave2lsl_t::patchbay [private]`

5.473.4.7 is_first_run

`bool wave2lsl::wave2lsl_t::is_first_run [private]`

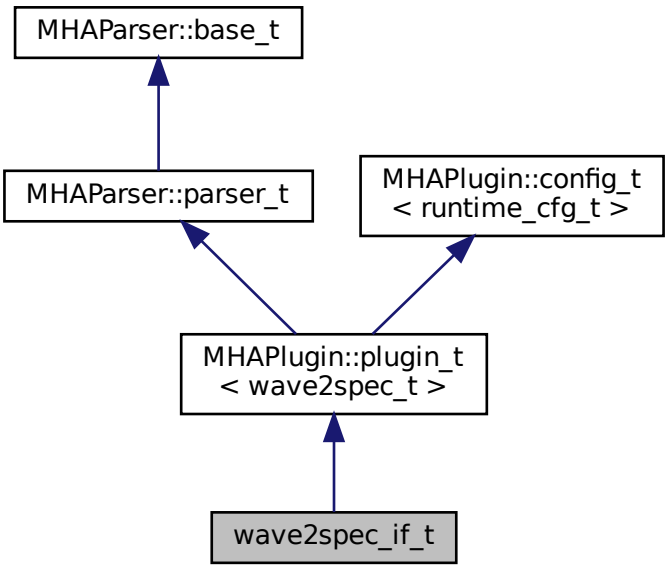
The documentation for this class was generated from the following file:

- **wave2lsl.cpp**

5.474 wave2spec_if_t Class Reference

Plugin wave2spec interface class, uses **wave2spec_t** (p. 1694) as runtime configuration.

Inheritance diagram for wave2spec_if_t:



Public Member Functions

- **wave2spec_if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor of wave2spec plugin, sets up configuration variables and callbacks.
- void **prepare** (**mhaconfig_t** &t)
  prepare for signal processing
- void **release** ()
  Unprepare signal processing.
- void **process** (**mha_wave_t** ∗wave_in, **mha_spec_t** ∗∗sout)
  processing callback used for domain transformation
- void **process** (**mha_wave_t** ∗wave_in, **mha_wave_t** ∗∗sout)
  processing callback used if output of original waveform is requested.

Private Member Functions

- void **update** ()
  Create a new runtime configuration from configuration parameters when the plugin is prepared, or when the window position or other window parameters change.
- void **setlock** (bool b)
  Lock/Unlock all configuration variables.

Private Attributes

- **MHAParser::int_t nfft**
  FFT length selector.
- **MHAParser::int_t nwnd**
  Window length selector.
- **MHAParser::float_t wndpos**
  Window position selector.
- **windowselector_t window_config**
- **MHAParser::bool_t strict_window_ratio**
  Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.
- **MHAParser::bool_t return_wave**
  Switch to select return domain.
- std::string **algo**
  configured name this plugin, used to name the AC variables
- **MHAParser::vfloat_mon_t zeropadding**

Additional Inherited Members

5.474.1 Detailed Description

Plugin wave2spec interface class, uses **wave2spec_t** (p. 1694) as runtime configuration.

5.474.2 Constructor & Destructor Documentation

5.474.2.1 wave2spec_if_t() `wave2spec_if_t::wave2spec_if_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

Constructor of wave2spec plugin, sets up configuration variables and callbacks.

Parameters

iac	algorithm communication storage accessor
ialg	configured name of this plugin, used to name the AC variables published by wave2spec

5.474.3 Member Function Documentation

5.474.3.1 prepare() void wave2spec_if_t::prepare (mhaconfig_t & t) [virtual]

prepare for signal processing

Parameters

in,out	*t*	signal dimenstions, modified by prepare as determined by the STFT configuration

Implements **MHAPlugin::plugin_t< wave2spec_t >** (p. 1301).

5.474.3.2 release() void wave2spec_if_t::release (void) [virtual]

Unprepare signal processing.

Reimplemented from **MHAPlugin::plugin_t< wave2spec_t >** (p. 1302).

5.474.3.3 process() [1/2] void wave2spec_if_t::process (mha_wave_t * wave_in, mha_spec_t ** sout)

processing callback used for domain transformation

Parameters

wave_in	latest block of audio signal (hop size samples per channel)
sout	output spectrum pointer

5.474.3.4 process() [2/2] void wave2spec_if_t::process (mha_wave_t * wave_in, mha_wave_t ** sout)

processing callback used if output of original waveform is requested.

The STFT spectrum is computed and can only be accessed by downstream plugins through the AC variable published by this plugin.

Parameters

wave_in	latest block of audio signal (hop size samples per channel)
sout	output waveform pointer (FFT length samples per channel)

5.474.3.5 update() void wave2spec_if_t::update ( ) [private]

Create a new runtime configuration from configuration parameters when the plugin is prepared, or when the window position or other window parameters change.

Exceptions

MHA_Error *(p. 906)*	if the configuration change is not compatible with current input and FFT length constraints.

5.474.3.6 setlock() void wave2spec_if_t::setlock (bool b) [private]

Lock/Unlock all configuration variables.

Parameters

b	Desired lock state

5.474.4 Member Data Documentation

5.474.4.1 nfft `MHAParser::int_t wave2spec_if_t::nfft [private]`

FFT length selector.

5.474.4.2 nwnd `MHAParser::int_t wave2spec_if_t::nwnd [private]`

Window length selector.

5.474.4.3 wndpos `MHAParser::float_t wave2spec_if_t::wndpos [private]`

Window position selector.

5.474.4.4 window_config `windowselector_t wave2spec_if_t::window_config [private]`

5.474.4.5 strict_window_ratio `MHAParser::bool_t wave2spec_if_t::strict_window_ratio [private]`

Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.

5.474.4.6 return_wave `MHAParser::bool_t wave2spec_if_t::return_wave [private]`

Switch to select return domain.

5.474.4.7 algo `std::string wave2spec_if_t::algo [private]`

configured name this plugin, used to name the AC variables

5.474.4.8 zeropadding `MHAParser::vfloat_mon_t wave2spec_if_t::zeropadding [private]`

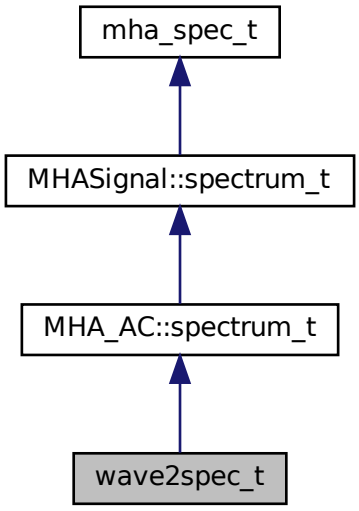
The documentation for this class was generated from the following files:

- **wave2spec.hh**
- **wave2spec.cpp**

5.475 wave2spec_t Class Reference

Runtime configuration class for plugin wave2spec.

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**, **MHA_AC::algo_comm_t** & **ac**, std::string algo)
  Constructor computes window and zeropadding, allocates storage and FFT plan.
- void **publish_ac_variables** ()
  Insert two AC variables into AC space:
- **mha_spec_t** * **process** (**mha_wave_t** *wave_in)
  Perform signal shift, windowing, zero-padding and FFT.
- ~**wave2spec_t** ()
  Destructor removes AC variables from AC space and deallocates memory.
- unsigned **get_zeropadding** (bool after) const
  Getter method to read zeropadding computed for the STFT configuration parameters.

Private Member Functions

- void **calc_pre_wnd** (**MHASignal::waveform_t** &dest, const **MHASignal::waveform_t** &src)

  Applies analysis window weights to current input signal and writes windowed signal to correct place in FFT buffer.

Private Attributes

- unsigned int **nwnd**

  window length
- unsigned int **nwndshift**

  window shift or hop size
- **mha_fft_t ft**

  FFT instance used for transformation.
- unsigned int **npad1**

  length of zero padding before window
- unsigned int **npad2**

  length of zero padding after window
- **MHAWindow::base_t window**

  Analysis window.
- **MHASignal::waveform_t calc_in**

  waveform buffer with FFT length samples per channel
- **MHASignal::waveform_t in_buf**

  waveform buffer with window length samples per channel
- **MHASignal::spectrum_t spec_in**

  spectrum buffer containing only the positive frequency bins
- std::string **ac_wndshape_name**

  name of window shape AC variable

Additional Inherited Members

5.475.1 Detailed Description

Runtime configuration class for plugin wave2spec.

Manages window shift, windowing, zero-padding, and FFT. Inserts current window shape and current STFT spectrum into AC space.

5.475.2 Constructor & Destructor Documentation

5.475.2.1 wave2spec_t() 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,
 MHA_AC::algo_comm_t & ac,
std::string algo )
```

Constructor computes window and zeropadding, allocates storage and FFT plan.

Parameters

nfft	FFT length
*nwnd_*	window length in samples
*nwndshift_*	window shift (hop size) in samples
nch	number of audio channels
wndpos	for cases nfft $>$ nwnd_, where to place the window inside the FFT buffer: 0 = at start, 1 = at end, 0.5 = centered. Position is rounded to full samples and determines zero-padding
window	Analysis window shape
ac	algorithm communication storage accessor
algo	configured name of this plugin, used to name the AC variables published by wave2spec

5.475.2.2 ~wave2spec_t() wave2spec_t::~wave2spec_t ( )

Destructor removes AC variables from AC space and deallocates memory.

5.475.3 Member Function Documentation

5.475.3.1 publish_ac_variables() void wave2spec_t::publish_ac_variables ( )

Insert two AC variables into AC space:

- $<$configured_name$>$: Contains the current STFT spectrum.
- $<$configured_name$>$_wnd: Contains the window shape as individual weights.

5.475.3.2 process() `mha_spec_t * wave2spec_t::process ( mha_wave_t * wave_in )`

Perform signal shift, windowing, zero-padding and FFT.

Parameters

wave_in	latest block of audio signal (hop size samples per channel)

Returns

pointer to current STFT spectrum. Storage is managed by this object. Downstream plugins may modify the signal in place.

5.475.3.3 get_zeropadding() `unsigned wave2spec_t::get_zeropadding ( bool after ) const [inline]`

Getter method to read zeropadding computed for the STFT configuration parameters.

Result is only valid after prepare() has been called.

Returns

Computed zeropadding before or after the analysis window in number of samples.

Parameters

after	When false, return length of zeropadding before the analysis window. When true, return length of zeropadding in samples after the analysis window.

5.475.3.4 calc_pre_wnd() `void wave2spec_t::calc_pre_wnd ( MHASignal::waveform_t & dest, const MHASignal::waveform_t & src ) [private]`

Applies analysis window weights to current input signal and writes windowed signal to correct place in FFT buffer.

Ensures zero-padding regions contain only zeros. To be invoked before applying FFT.

Parameters

out	*dest*	waveform buffer with FFT length audio samples, completely overwritten by this method
in	*src*	waveform buffer with window length audio samples, these samples are written to dest after window shape has been applied to the individual samples.

5.475.4 Member Data Documentation

5.475.4.1 nwnd `unsigned int wave2spec_t::nwnd [private]`

window length

5.475.4.2 nwndshift `unsigned int wave2spec_t::nwndshift [private]`

window shift or hop size

5.475.4.3 ft `mha_fft_t wave2spec_t::ft [private]`

FFT instance used for transformation.

5.475.4.4 npad1 unsigned int wave2spec_t::npad1 [private]

length of zero padding before window

5.475.4.5 npad2 unsigned int wave2spec_t::npad2 [private]

length of zero padding after window

5.475.4.6 window MHAWindow::base_t wave2spec_t::window [private]

Analysis window.

5.475.4.7 calc_in MHASignal::waveform_t wave2spec_t::calc_in [private]

waveform buffer with FFT length samples per channel

5.475.4.8 in_buf MHASignal::waveform_t wave2spec_t::in_buf [private]

waveform buffer with window length samples per channel

5.475.4.9 spec_in MHASignal::spectrum_t wave2spec_t::spec_in [private]

spectrum buffer containing only the positive frequency bins

5.475.4.10 ac_wndshape_name std::string wave2spec_t::ac_wndshape_name [private]

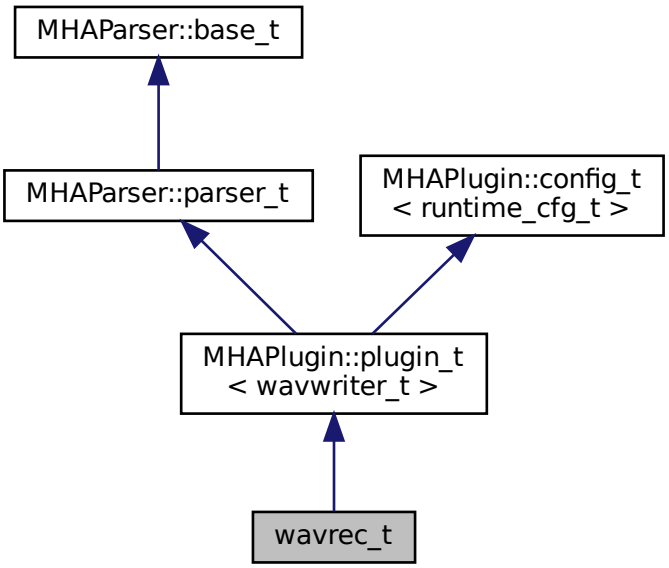
name of window shape AC variable

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

- **wave2spec.hh**
- **wave2spec.cpp**

5.476 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** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)

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**
- **MHAParser::kw_t output_sample_format**
- **MHAEvents::patchbay_t**< **wavrec_t** > **patchbay**

Additional Inherited Members

5.476.1 Constructor & Destructor Documentation

5.476.1.1 wavrec_t() `wavrec_t::wavrec_t (`

```
MHA_AC::algo_comm_t & iac,
const std::string & configured_name )
```

5.476.2 Member Function Documentation

5.476.2.1 process() `mha_wave_t * wavrec_t::process (`

```
mha_wave_t * s )
```

5.476.2.2 prepare() `void wavrec_t::prepare (`

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< wavwriter_t >** (p. 1301).

5.476.2.3 release() `void wavrec_t::release (`

```
void ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< wavwriter_t >** (p. 1302).

5.476.2.4 start_new_session() `void wavrec_t::start_new_session ( ) [private]`

5.476.3 Member Data Documentation

5.476.3.1 record `MHAParser::bool_t wavrec_t::record [private]`

5.476.3.2 fifolen `MHAParser::int_t wavrec_t::fifolen [private]`

5.476.3.3 minwrite `MHAParser::int_t wavrec_t::minwrite [private]`

5.476.3.4 prefix `MHAParser::string_t wavrec_t::prefix [private]`

5.476.3.5 use_date `MHAParser::bool_t wavrec_t::use_date [private]`

5.476.3.6 output_sample_format `MHAParser::kw_t wavrec_t::output_sample_format [private]`

5.476.3.7 patchbay `MHAEvents::patchbay_t< wavrec_t> wavrec_t::patchbay [private]`

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

- **wavrec.cpp**

5.477 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, const std::string &format_name_)
- ∼**wavwriter_t** ()
- void **process** (**mha_wave_t** ∗)
- void **exit_request** ()

Private Member Functions

- void **write_thread** ()
- void **create_soundfile** (const std::string &prefix, bool use_date)
- void **set_format** (SF_INFO &sf_info)

  Converts the format_name string to the corresponding int according to libsndfile and writes it into the format field of sf_info throws if no format of this name is available.

Static Private Member Functions

- static void * **write_thread** (void *this_)

Private Attributes

- std::atomic< bool > **close_session**
- bool **act_**
- **mhaconfig_t** **cf_**
- SNDFILE * **sf**
- **mha_fifo_lf_t**< **mha_real_t** > **fifo**
- unsigned int **minw_**
- pthread_t **writethread**
- float * **data**
- std::string **format_name**

5.477.1 Constructor & Destructor Documentation

5.477.1.1 wavwriter_t() `wavwriter_t::wavwriter_t (`

```
bool active,
const  mhaconfig_t & cf,
unsigned int fifosize,
unsigned int minwrite,
const std::string & prefix,
bool use_date,
const std::string & format_name_ )
```

5.477.1.2 ~wavwriter_t() `wavwriter_t::~wavwriter_t ( )`

5.477.2 Member Function Documentation

5.477.2.1 process() void wavwriter_t::process (mha_wave_t * s)

5.477.2.2 exit_request() void wavwriter_t::exit_request ( )

5.477.2.3 write_thread() [1/2] static void* wavwriter_t::write_thread (void * this_) [inline], [static], [private]

5.477.2.4 write_thread() [2/2] void wavwriter_t::write_thread ( ) [private]

5.477.2.5 create_soundfile() void wavwriter_t::create_soundfile (const std::string & prefix, bool use_date) [private]

5.477.2.6 set_format() void wavwriter_t::set_format (SF_INFO & *sf_info*) [private]

Converts the format_name string to the corresponding int according to libsndfile and writes it into the format field of sf_info throws if no format of this name is available.

Parameters

sf_info	Destination sf_info struct for the format

Exceptions

MHA_Error *(p. 906)*	If no sample format of name format_name is offered by libsndfile

5.477.3 Member Data Documentation

5.477.3.1 close_session std::atomic<bool> wavwriter_t::close_session [private]

5.477.3.2 act_ bool wavwriter_t::act_ [private]

5.477.3.3 cf_ **mhaconfig_t** wavwriter_t::cf_ [private]

5.477.3.4 sf SNDFILE* wavwriter_t::sf [private]

5.477.3.5 fifo **mha_fifo_lf_t**< **mha_real_t**> wavwriter_t::fifo [private]

5.477.3.6 minw_ unsigned int wavwriter_t::minw_ [private]

5.477.3.7 writethread pthread_t wavwriter_t::writethread [private]

5.477.3.8 data float* wavwriter_t::data [private]

5.477.3.9 format_name std::string wavwriter_t::format_name [private]

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

- **wavrec.cpp**

5.478 windnoise::cfg_t Class Reference

Runtime config class for windnoise plugin.

Public Member Functions

- **cfg_t** (const **mhaconfig_t** &signal_info, bool **UseChannel_LF_attenuation**, float tau↩ _Lowpass, float LowPassCutOffFrequency, float LowPassFraction_dB, float LowPass↩ WindGain_dB)

  constructor translates configuration variables to runtime config
- **mha_spec_t** * **process** (**mha_spec_t** *signal, std::vector< int > &detected, std↩ ::vector< float > &lowpass_quotient)

  Detect windnoise.
- void **update_PSD_Lowpass** (const **mha_spec_t** *signal)

  Low-pass filters the power spectrum.
- void **threshold_compare** (std::vector< int > &detected, std::vector< float > &lowpass↩ _quotient)

  Wind noise detection by comparing low-frequency intensity with broadband intensity.
- int **remapping** (const std::vector< float > &lowpass_quotient)
- void **compensation** (**mha_spec_t** *signal, int best_signal_channel_index)

Public Attributes

- bool **UseChannel_LF_attenuation** = false

  FIXME: documentation for UseChannel_LF_attenuation.
- float **alpha_Lowpass** = 0

  Filter coefficient for low-pass filtering each bin in the power spectrum with a first-order recursive low-pass filter.
- unsigned **FrequencyBinLowPass** = 0

  Only smoothed power spectrum bins $<$ FrequencyBinLowPass are added to the low-pass intensity.
- float **LowPassFraction** = 1

  The wind noise detection threshold: We have wind noise if lowFreqIntensity / broadBand↩ Intensity $>$ LowPassFraction.
- float **LowPassWindGain** = 1

  FIXME: documentation for LowPassWindGain.
- **MHASignal::waveform_t PSD_Lowpass**

  The smoothed-over-time power spectrum.
- **MHASignal::waveform_t powspec**

  Temporary storage for the power spectrum of the current input spectrum.

5.478.1 Detailed Description

Runtime config class for windnoise plugin.

Computes power spectra of incoming STFT spectra, smoothes the power spectrum over time by low-pass filtering the intensities of each bin over time, then detects wind noise presence by comparing intensity at low frequency bins to broadband intensity.

5.478.2 Constructor & Destructor Documentation

5.478.2.1 cfg_t()

```
cfg_t::cfg_t (
    const  mhaconfig_t & signal_info,
    bool UseChannel_LF_attenuation,
    float tau_Lowpass,
    float LowPassCutOffFrequency,
    float LowPassFraction_dB,
    float LowPassWindGain_dB )
```

constructor translates configuration variables to runtime config

5.478.3 Member Function Documentation

5.478.3.1 process()

```
mha_spec_t * cfg_t::process (
    mha_spec_t * signal,
    std::vector< int > & detected,
    std::vector< float > & lowpass_quotient )
```

Detect windnoise.

FIXME: cancel it. The process method calls update_PSD_Lowpass and threshold_compare to do its work.

Parameters

in,out	*signal*	The current STFT spe
out	*detected*	This Method changes the vector but not its s is set to 1 or 0, depen being detected in the audio channel.
out	*lowpass_quotient*	This Method changes vector but not its size. set to the ratio betwee signal, at low frequenc intensity, in the corres channel.

Exceptions

MHA_Error *(p. 906)*	if windnoise_indicators.size() != signal.num_channels.

5.478.3.2 update_PSD_Lowpass()

```
void cfg_t::update_PSD_Lowpass (
    const mha_spec_t * signal )
```

Low-pass filters the power spectrum.

5.478.3.3 threshold_compare() void cfg_t::threshold_compare (std::vector< int > & *detected*, std::vector< float > & *lowpass_quotient*)

Wind noise detection by comparing low-frequency intensity with broadband intensity.

Parameters

out	*detected*	This Method changes the the vector but not its size element is set to 1 or 0, windnoise being detecte corresponding audio cha
out	*lowpass_quotient*	This Method changes ele vector but not its size. Ea set to the ratio between i signal, at low frequencies intensity, in the correspo channel.

5.478.3.4 remapping() int cfg_t::remapping (const std::vector< float > & *lowpass_quotient*)

5.478.3.5 compensation() void cfg_t::compensation (**mha_spec_t** * *signal*, int *best_signal_channel_index*)

5.478.4 Member Data Documentation

5.478.4.1 UseChannel_LF_attenuation bool windnoise::cfg_t::UseChannel_LF_attenuation = false

FIXME: documentation for UseChannel_LF_attenuation.

5.478.4.2 alpha_Lowpass `float windnoise::cfg_t::alpha_Lowpass = 0`

Filter coefficient for low-pass filtering each bin in the power spectrum with a first-order recursive low-pass filter.

5.478.4.3 FrequencyBinLowPass `unsigned windnoise::cfg_t::FrequencyBinLowPass = 0`

Only smoothed power spectrum bins $<$ FrequencyBinLowPass are added to the low-pass intensity.

5.478.4.4 LowPassFraction `float windnoise::cfg_t::LowPassFraction = 1`

The wind noise detection threshold: We have wind noise if lowFreqIntensity / broadBand↩ Intensity $>$ LowPassFraction.

5.478.4.5 LowPassWindGain `float windnoise::cfg_t::LowPassWindGain = 1`

FIXME: documentation for LowPassWindGain.

5.478.4.6 PSD_Lowpass **`MHASignal::waveform_t`** `windnoise::cfg_t::PSD_Lowpass`

The smoothed-over-time power spectrum.

5.478.4.7 powspec **`MHASignal::waveform_t`** `windnoise::cfg_t::powspec`

Temporary storage for the power spectrum of the current input spectrum.

Only needed to hold the newest squared magnitudes until they are filtered into PSD_Lowpass.

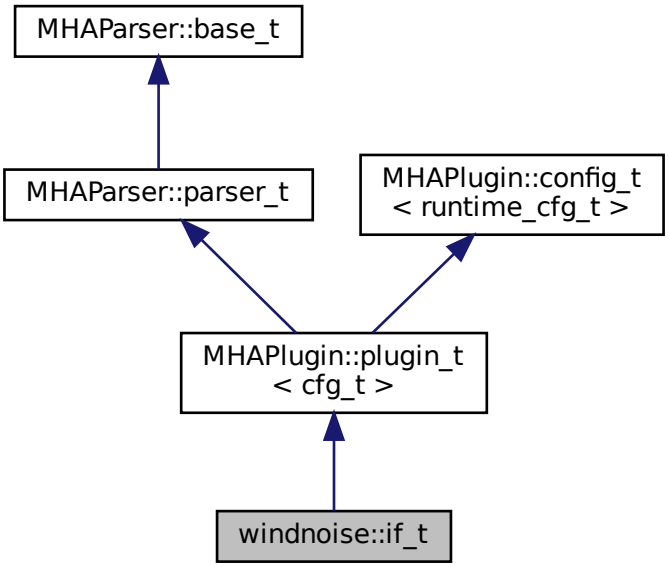
The documentation for this class was generated from the following files:

- **windnoise.hh**
- **windnoise.cpp**

5.479 windnoise::if_t Class Reference

interface class for windnoise plugin

Inheritance diagram for windnoise::if_t:



Public Member Functions

- **if_t** (**MHA_AC::algo_comm_t** &iac, const std::string &configured_name)
  Constructor instantiates one windnoise plugin.
- void **prepare** (**mhaconfig_t** &signal_info) override
  Prepare windnoise plugin for signal processing.
- void **release** (void) override
  Nothing needs to be deallocated on release.
- **mha_spec_t** * **process** (**mha_spec_t** *signal)
  *signal processing, delegates to **cfg_t::process*** (p. 1708)
- void **update** (void)
  update runtime config when configuration parameters have changed
- void **insert** ()
  inserts the windnoise detection vector into AC space

Public Attributes

- **MHAParser::bool_t UseChannel_LF_attenuation**
- **MHAParser::float_t tau_Lowpass**
- **MHAParser::float_t LowPassCutOffFrequency**
- **MHAParser::float_t LowPassFraction**
- **MHAParser::float_t LowPassWindGain**
- **MHAParser::kw_t WindNoiseDetector**
- **MHAParser::vint_mon_t detected**
- **MHAParser::vfloat_mon_t lowpass_quotient**
- const std::string **detected_acname**
  Name of AC variable mirroring the configuration monitor variable "detector".
- const std::string **lowpass_quotient_acname**
  Name of AC variable mirroring the configuration monitor variable "lowpass_quotient".

Private Attributes

- **MHAEvents::patchbay_t< if_t > patchbay**
  The Event connector.

Additional Inherited Members

5.479.1 Detailed Description

interface class for windnoise plugin

5.479.2 Constructor & Destructor Documentation

5.479.2.1 if_t()

```
windnoise::if_t::if_t (
    MHA_AC::algo_comm_t & iac,
    const std::string & configured_name )
```

Constructor instantiates one windnoise plugin.

5.479.3 Member Function Documentation

5.479.3.1 prepare() void windnoise::if_t::prepare (mhaconfig_t & signal_info) [override], [virtual]

Prepare windnoise plugin for signal processing.

Parameters

signal_info	signal dimensions, not changed by this plugin

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1301).

5.479.3.2 release() void windnoise::if_t::release (void) [inline], [override], [virtual]

Nothing needs to be deallocated on release.

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1302).

5.479.3.3 process() mha_spec_t * windnoise::if_t::process (mha_spec_t * signal)

signal processing, delegates to **cfg_t::process** (p. 1708)

5.479.3.4 update() void windnoise::if_t::update (void)

update runtime config when configuration parameters have changed

5.479.3.5 insert() void windnoise::if_t::insert ( ) [inline]

inserts the windnoise detection vector into AC space

5.479.4 Member Data Documentation

5.479.4.1 patchbay `MHAEvents::patchbay_t< if_t> windnoise::if_t::patchbay [private]`

The Event connector.

5.479.4.2 UseChannel_LF_attenuation `MHAParser::bool_t windnoise::if_t::UseChannel_LF_attenuation`

5.479.4.3 tau_Lowpass `MHAParser::float_t windnoise::if_t::tau_Lowpass`

5.479.4.4 LowPassCutOffFrequency `MHAParser::float_t windnoise::if_t::LowPassCutOffFrequency`

5.479.4.5 LowPassFraction `MHAParser::float_t windnoise::if_t::LowPassFraction`

5.479.4.6 LowPassWindGain `MHAParser::float_t windnoise::if_t::LowPassWindGain`

5.479.4.7 WindNoiseDetector `MHAParser::kw_t windnoise::if_t::WindNoiseDetector`

5.479.4.8 detected `MHAParser::vint_mon_t windnoise::if_t::detected`

5.479.4.9 lowpass_quotient `MHAParser::vfloat_mon_t windnoise::if_t::lowpass_↩ quotient`

5.479.4.10 detected_acname `const std::string windnoise::if_t::detected_acname`

Name of AC variable mirroring the configuration monitor variable "detector".

Usually "windnoise_detected".

5.479.4.11 lowpass_quotient_acname `const std::string windnoise::if_t::lowpass_↩ quotient_acname`

Name of AC variable mirroring the configuration monitor variable "lowpass_quotient".

Usually "windnoise_lowpass_quotient".

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

- **windnoise.hh**
- **windnoise.cpp**

5.480 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.
- void **setlock** (bool b_)
  Lock/Unlock variables.

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. 1717)*
- **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

5.480.1 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. 1444) when needed.

5.480.2 Constructor & Destructor Documentation

5.480.2.1 windowselector_t() `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"

5.480.2.2 ~windowselector_t() `windowselector_t::~windowselector_t ( )`

destructor frees window data that were allocated

5.480.3 Member Function Documentation

5.480.3.1 get_window_data() `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. 1444) instance. The referenced instance lives until the window parameters are changed, or this **windowselector_t** (p. 1715) instance is destroyed.

5.480.3.2 insert_items() `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.

5.480.3.3 setlock() `void windowselector_t::setlock ( bool b_ )`

Lock/Unlock variables.

Parameters

*b_*	Desired lock state

5.480.3.4 invalidate_window_data() `void windowselector_t::invalidate_window_data ( ) [private]`

invalidates any allocated window samples.

5.480.3.5 update_parser() `void windowselector_t::update_parser ( ) [private]`

invoked when a parser parameter changes.

Calls **invalidate_window_data()** (p. 1718) and emits the updated event.

5.480.4 Member Data Documentation

5.480.4.1 updated `MHAEvents::emitter_t windowselector_t::updated`

A collector event that fires when any of the window parameters managed here is written to.

5.480.4.2 wnd `MHAWindow::base_t* windowselector_t::wnd [private]`

Storage for the window data returned by **get_window_data()** (p. 1717)

5.480.4.3 wndtype `MHAParser::kw_t windowselector_t::wndtype [private]`

parser variable for window type

5.480.4.4 wndexp `MHAParser::float_t windowselector_t::wndexp [private]`

parser variable for window exponent

5.480.4.5 userwnd `MHAParser::vfloat_t windowselector_t::userwnd [private]`

parser variable for user window samples to use

5.480.4.6 patchbay `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::ac2wave_t**
  ***ac2wave** (p. 79) real-time configuration class*
- class **ac2wave::ac2wave_if_t**
  ***ac2wave** (p. 79) plugin interface class*

Namespaces

- **ac2wave**
  *Namespace containing all code for the **ac2wave** (p. 79) plugin.*

6.4 ac2xdf.cpp File Reference

Classes

- class **ac2xdf::ac2xdf_rt_t**
- class **ac2xdf::ac2xdf_if_t**
  *Plugin interface class of plugin **ac2xdf** (p. 79).*

Namespaces

- **ac2xdf**

Variables

- const std::unordered_map< std::type_index, std::string > **ac2xdf::types**

6.5 ac2xdf.hh File Reference

Classes

- class **ac2xdf::output_file_t**
  ***output_file_t** (p. 211) represents one XDF output file.*
- class **ac2xdf::acwriter_base_t**
  *Base class for all **acwriter_t** (p. 205)'s.*
- class **ac2xdf::acwriter_t< T >**

Namespaces

- **ac2xdf**

Macros

- #define **_CRT_SECURE_NO_WARNINGS**

Functions

- std::string **ac2xdf::to_iso8601** (time_t tm)

6.5.1 Macro Definition Documentation

6.5.1.1 _CRT_SECURE_NO_WARNINGS `#define _CRT_SECURE_NO_WARNINGS`

6.6 ac_monitor_type.cpp File Reference

6.7 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.

6.8 ac_mul.cpp File Reference

6.9 ac_mul.hh File Reference

Classes

- class **ac_mul_t**
  *The class which implements the **ac_mul_t** (p. 215) plugin.*

Enumerations

- enum **arg_type_t** { **ARG_RR** , **ARG_RC** , **ARG_CR** , **ARG_CC** }
  Indicates whether the factors of the product are real or complex valued.
- enum **val_type_t** { **VAL_REAL** , **VAL_COMPLEX** }
  Indicates whether an AC variable contains real or complex values.

6.9.1 Enumeration Type Documentation

6.9.1.1 arg_type_t enum arg_type_t

Indicates whether the factors of the product are real or complex valued.

Enumerator

ARG_RR	Both factors are real.
ARG_RC	First factor is real, second is complex.
ARG_CR	First factor is complex, second is real.
ARG_CC	Both factors are complex.

6.9.1.2 val_type_t enum val_type_t

Indicates whether an AC variable contains real or complex values.

Enumerator

VAL_REAL	AC variable contains real values.
VAL_COMPLEX	AC variable contains complex values.

6.10 ac_proc.cpp File Reference

Classes

- class **ac_proc::interface_t**

Namespaces

- **ac_proc**

6.11 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)

6.11.1 Macro Definition Documentation

6.11.1.1 PATCH_VAR

```
#define PATCH_VAR(
        var ) patchbay.connect(&var.valuechanged, this, & acConcat_wave::update_cfg)
```

6.11.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
        var ) insert_member(var); PATCH_VAR(var)
```

6.12 acConcat_wave.h File Reference

Classes

- class **acConcat_wave_config**
- class **acConcat_wave**

6.13 acmon.cpp File Reference

Classes

- class **acmon::acmon_t**
  acmon plugin interface class

Namespaces

- **acmon**
  Namespace for displaying ac variables as parser monitors.

6.14 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)

6.14.1 Macro Definition Documentation

6.14.1.1 PATCH_VAR

```
#define PATCH_VAR(
        var ) patchbay.connect(&var.valuechanged, this, & acPooling_wave::update_cfg)
```

6.14.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
        var ) insert_member(var); PATCH_VAR(var)
```

6.15 acPooling_wave.h File Reference

Classes

- class **acPooling_wave_config**
- class **acPooling_wave**

6.16 acrec.cpp File Reference

6.17 acrec.hh File Reference

Classes

- class **plugins::hoertech::acrec::acwriter_t**
  acwriter_t *(p. 1537) decouples signal processing from writing to disk.*
- class **plugins::hoertech::acrec::acrec_t**
  Plugin interface class of plugin acrec.

Namespaces

- **plugins**
- **plugins::hoertech**
- **plugins::hoertech::acrec**

Functions

- std::string **plugins::hoertech::acrec::to_iso8601** (time_t tm)

6.18 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"

6.18.1 Macro Definition Documentation

6.18.1.1 ACSAVE_FMT_TXT `#define ACSAVE_FMT_TXT 0`

6.18.1.2 ACSAVE_SFMT_TXT #define ACSAVE_SFMT_TXT "txt"

6.18.1.3 ACSAVE_FMT_MAT4 #define ACSAVE_FMT_MAT4 1

6.18.1.4 ACSAVE_SFMT_MAT4 #define ACSAVE_SFMT_MAT4 "mat4"

6.18.1.5 ACSAVE_FMT_M #define ACSAVE_FMT_M 2

6.18.1.6 ACSAVE_SFMT_M #define ACSAVE_SFMT_M "m"

6.19 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)

6.19.1 Macro Definition Documentation

6.19.1.1 PATCH_VAR #define PATCH_VAR(*var*) patchbay.connect(&var.valuechanged, this, & **acSteer::update_cfg**)

6.19.1.2 INSERT_PATCH #define INSERT_PATCH(*var*) **insert_member**(var); **PATCH_VAR**(var)

6.20 acSteer.h File Reference

Classes

- class **acSteer_config**
- class **acSteer**

6.21 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)

6.21.1 Macro Definition Documentation

6.21.1.1 PATCH_VAR

```
#define PATCH_VAR(
        var ) patchbay.connect(&var.valuechanged, this, & acTransform_wave
::update_cfg)
```

6.21.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
        var )  insert_member(var);  PATCH_VAR(var)
```

6.22 acTransform_wave.h File Reference

Classes

- class **acTransform_wave_config**
- class **acTransform_wave**

6.23 adaptive_feedback_canceller.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, & **adaptive_↩ feedback_canceller::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

Functions

- std::vector< int > **calcDelayValues** (const std::vector< int > &raw_latency, const unsigned int correction)
- void **make_friendly_number_by_limiting** (double &x)

6.23.1 Macro Definition Documentation

6.23.1.1 PATCH_VAR

```
#define PATCH_VAR(
        var ) patchbay.connect(&var.valuechanged, this, & adaptive_feedback_↩
canceller::update_cfg)
```

6.23.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
        var ) insert_member(var); PATCH_VAR(var)
```

6.23.2 Function Documentation

6.23.2.1 calcDelayValues()

```
std::vector<int> calcDelayValues (
        const std::vector< int > & raw_latency,
        const unsigned int correction )
```

6.23.2.2 make_friendly_number_by_limiting() `void make_friendly_number_by_limiting (`

`double & x ) [inline]`

6.24 adaptive_feedback_canceller.h File Reference

Classes

- class **adaptive_feedback_canceller_config**
  This is the runtime configuration, the main processing will be done in this class.
- class **adaptive_feedback_canceller**

6.25 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. 286).*
- 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

- enum **addsndfile::addsndfile_resampling_mode_t** { **addsndfile::DONT_↩ RESAMPLE_PERMISSIVE** , **addsndfile::DONT_RESAMPLE_STRICT** , **addsndfile↩ ::DO_RESAMPLE** }

  Specifies the resampling mode in resampled_soundfile_t.

Functions

- static unsigned **addsndfile::resampled_num_frames** (unsigned num_source_frames, float source_rate, float target_rate, addsndfile_resampling_mode_t resampling_mode)

6.25.1 Macro Definition Documentation

6.25.1.1 DEBUG `#define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl`

6.26 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)

6.26.1 Function Documentation

6.26.1.1 adm_fir_lp()

```
MHASignal::waveform_t* adm_fir_lp (
    unsigned int fs,
    unsigned f_pass,
    unsigned int f_stop,
    unsigned int order )
```

6.26.1.2 adm_fir_decomb()

```
MHASignal::waveform_t* adm_fir_decomb (
    unsigned int fs,
    float dist_m,
    unsigned int order )
```

6.27 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.
- 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

6.28 altconfig.cpp File Reference

6.29 altconfig.hh File Reference

Classes

- class **altconfig_t**
  Single class implementing plugin altconfig.

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.29.1 Macro Definition Documentation

6.29.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN `#define MHAPLUGIN_OVERLOAD_OUTDOMAIN`

6.30 altplugs.cpp File Reference

Classes

- class **mhaplug_cfg_t**
- class **altplugs_t**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.30.1 Macro Definition Documentation

6.30.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN `#define MHAPLUGIN_OVERLOAD_OUTDOMAIN`

6.31 analysemhaplugin.cpp File Reference

Functions

- std::string **strdom** (**mha_domain_t** d)
- void **print_ac** (**MHA_AC::algo_comm_t** &ac, std::string txt)
- int **document_plugin** (**MHA_AC::algo_comm_class_t** &ac, **PluginLoader**↩ **::mhapluginloader_t** &load, int argc, char ∗∗argv)
- void **document_io_plugin** (char ∗lib_name)
- int **main** (int argc, char ∗∗argv)

6.31.1 Function Documentation

6.31.1.1 strdom() std::string strdom (mha_domain_t d)

6.31.1.2 print_ac() void print_ac (MHA_AC::algo_comm_t & ac, std::string txt)

6.31.1.3 document_plugin() int document_plugin (MHA_AC::algo_comm_class_t & ac, PluginLoader::mhapluginloader_t & load, int argc, char ** argv)

6.31.1.4 document_io_plugin() void document_io_plugin (char * lib_name)

6.31.1.5 main()

```
int main (
    int argc,
    char ** argv )
```

6.32 analysispath.cpp File Reference

Classes

- class **analysepath_t**
- class **plug_t**
- class **analysispath_if_t**

Functions

- static void * **thread_start** (void *instance)

6.32.1 Function Documentation

6.32.1.1 thread_start()

```
static void* thread_start (
    void * instance )  [static]
```

6.33 attenuate20.cpp File Reference

Classes

- class **attenuate20_t**

6.34 audiometerbackend.cpp File Reference

Classes

- class **audiometerbackend::lnn3rdoct_t**
- class **audiometerbackend::sine_t**
- class **audiometerbackend::signal_gen_t**
- class **audiometerbackend::level_adapt_t**
- class **audiometerbackend::audiometer_if_t**

Namespaces

- **audiometerbackend**

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

Typedefs

- typedef **MHAPlugin::config_t**< level_adapt_t > **audiometerbackend::level_adaptor**
- typedef **MHAPlugin::plugin_t**< signal_gen_t > **audiometerbackend::generator**

Functions

- static unsigned int **audiometerbackend::gcd** (unsigned int a, unsigned int b)
- **MHASignal::waveform_t audiometerbackend::return_sig** (unsigned int sigtype, unsigned int fs, unsigned int f)

6.34.1 Macro Definition Documentation

6.34.1.1 DEBUG `#define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl`

6.35 auditory_profile.cpp File Reference

6.36 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)

6.37 bmfwf.cpp File Reference

Classes

- class **bmfwf_t**

6.38 browsemhaplugins.cpp File Reference

Macros

- #define **DEBUG**(x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x<<"="<<x << std::endl

Functions

- int **main** (int argc, char **argv)

6.38.1 Macro Definition Documentation

6.38.1.1 DEBUG

```
#define DEBUG(
            x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x<<"="<<x
<< std::endl
```

6.38.2 Function Documentation

6.38.2.1 main()

```
int main (
          int argc,
          char ** argv )
```

6.39 ci_auralization_ace.cpp File Reference

6.40 ci_auralization_ace.hh File Reference

Classes

- class **Ci_auralization_ace_cfg**

  Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.
- class **Ci_auralization_ace**

  Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

6.41 ci_auralization_cis.cpp File Reference

6.42 ci_auralization_cis.hh File Reference

Classes

- class **Ci_auralization_cis_cfg**

  Runtime configuration class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.
- class **Ci_auralization_cis**

  Plugin interface class for generating an auralized audio signal from the specified AC variable, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

6.43 ci_simulation_ace.cpp File Reference

Variables

- const unsigned int **CHANNELS** = 1

  Constant describing the required number of input audio channels per side.
- const unsigned int **FFTLEN** = 128

  Constant describing the required FFT length / bins.
- const **mha_real_t** **SRATE** = 16000

  Constant describing the required sampling rate / Hz.
- const unsigned int **M_ELECTRODES** = 22

  Constant describing the required total number of electrodes per side.
- const std::vector< unsigned int > **BIN_INDICES**

  Constant vector describing the required FFT bin indices for composing the filterbank bands.
- const std::vector< **mha_real_t** > **WEIGHTS**

  Constant vector describing the required weights for the filterbank bands.

6.43.1 Variable Documentation

6.43.1.1 CHANNELS `const unsigned int CHANNELS = 1`

Constant describing the required number of input audio channels per side.

6.43.1.2 FFTLEN `const unsigned int FFTLEN = 128`

Constant describing the required FFT length / bins.

6.43.1.3 SRATE `const mha_real_t SRATE = 16000`

Constant describing the required sampling rate / Hz.

6.43.1.4 M_ELECTRODES `const unsigned int M_ELECTRODES = 22`

Constant describing the required total number of electrodes per side.

6.43.1.5 BIN_INDICES `const std::vector<unsigned int> BIN_INDICES`

Constant vector describing the required FFT bin indices for composing the filterbank bands.

6.43.1.6 WEIGHTS `const std::vector< mha_real_t> WEIGHTS`

Constant vector describing the required weights for the filterbank bands.

6.44 ci_simulation_ace.hh File Reference

Classes

- class **Ci_simulation_ace_cfg**

  Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.
- class **Ci_simulation_ace**

  Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical ACE (advanced combination encoder, n-of-m) coding strategy with 22 channels.

Variables

- const unsigned int **CHANNELS**

  Constant describing the required number of input audio channels per side.
- const unsigned int **FFTLEN**

  Constant describing the required FFT length / bins.
- const **mha_real_t** **SRATE**

  Constant describing the required sampling rate / Hz.
- const unsigned int **M_ELECTRODES**

  Constant describing the required total number of electrodes per side.
- const std::vector< unsigned int > **BIN_INDICES**

  Constant vector describing the required FFT bin indices for composing the filterbank bands.
- const std::vector< **mha_real_t** > **WEIGHTS**

  Constant vector describing the required weights for the filterbank bands.

6.44.1 Variable Documentation

6.44.1.1 CHANNELS `const unsigned int CHANNELS [extern]`

Constant describing the required number of input audio channels per side.

6.44.1.2 FFTLEN `const unsigned int FFTLEN [extern]`

Constant describing the required FFT length / bins.

6.44.1.3 SRATE const mha_real_t SRATE [extern]

Constant describing the required sampling rate / Hz.

6.44.1.4 M_ELECTRODES const unsigned int M_ELECTRODES [extern]

Constant describing the required total number of electrodes per side.

6.44.1.5 BIN_INDICES const std::vector<unsigned int> BIN_INDICES [extern]

Constant vector describing the required FFT bin indices for composing the filterbank bands.

6.44.1.6 WEIGHTS const std::vector< mha_real_t> WEIGHTS [extern]

Constant vector describing the required weights for the filterbank bands.

6.45 ci_simulation_cis.cpp File Reference

Variables

- const unsigned int **CHANNELS** = 1
  Constant describing the required number of input audio channels per side.
- const **mha_real_t** **SRATE** = 48000
  Constant describing the required sampling rate / Hz.
- const unsigned int **M_ELECTRODES** = 12
  Constant describing the required total number of electrodes per side.

6.45.1 Variable Documentation

6.45.1.1 CHANNELS `const unsigned int CHANNELS = 1`

Constant describing the required number of input audio channels per side.

6.45.1.2 SRATE `const` **`mha_real_t`** `SRATE = 48000`

Constant describing the required sampling rate / Hz.

6.45.1.3 M_ELECTRODES `const unsigned int M_ELECTRODES = 12`

Constant describing the required total number of electrodes per side.

6.46 ci_simulation_cis.hh File Reference

Classes

- class **Ci_simulation_cis_cfg**

  Runtime configuration class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.
- class **Ci_simulation_cis**

  Plugin interface class for generating an electrodogram from an audio signal, using a stimulation strategy similar to a typical CIS (continuous interleaved sampling) coding strategy with 12 channels.

Variables

- const unsigned int **CHANNELS**

  Constant describing the required number of input audio channels per side.
- const **mha_real_t** **SRATE**

  Constant describing the required sampling rate / Hz.
- const unsigned int **M_ELECTRODES**

  Constant describing the required total number of electrodes per side.

6.46.1 Variable Documentation

6.46.1.1 CHANNELS `const unsigned int CHANNELS [extern]`

Constant describing the required number of input audio channels per side.

6.46.1.2 SRATE `const` **mha_real_t** `SRATE [extern]`

Constant describing the required sampling rate / Hz.

6.46.1.3 M_ELECTRODES `const unsigned int M_ELECTRODES [extern]`

Constant describing the required total number of electrodes per side.

6.47 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)

6.48 combinechannels.cpp File Reference

Classes

- class **combc_t**
- class **combc_if_t**

6.49 compiler_id.cpp File Reference

6.50 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_↩ HELPER2**(x,y,z)
- #define **COMPILER_ID_VERSION**
- #define **COMPILER_ID** **COMPILER_ID_VENDOR** "-" **COMPILER_ID_VERSION** "-" COMPILER_ID_STANDARD

6.50.1 Macro Definition Documentation

6.50.1.1 COMPILER_ID_VENDOR `#define COMPILER_ID_VENDOR "gcc"`

6.50.1.2 COMPILER_ID_MAJOR `#define COMPILER_ID_MAJOR __GNUC__`

6.50.1.3 COMPILER_ID_MINOR `#define COMPILER_ID_MINOR __GNUC_MINOR__`

6.50.1.4 COMPILER_ID_PATCH `#define COMPILER_ID_PATCH __GNUC_PATCHLEVEL__`

6.50.1.5 COMPILER_ID_VERSION_HELPER2 #define COMPILER_ID_VERSION_HELPER2(

x,
y,
z) #x "." #y "." #z

6.50.1.6 COMPILER_ID_VERSION_HELPER1 #define COMPILER_ID_VERSION_HELPER1(

x,
y,
z) **COMPILER_ID_VERSION_HELPER2**(x,y,z)

6.50.1.7 COMPILER_ID_VERSION #define COMPILER_ID_VERSION

6.50.1.8 COMPILER_ID #define COMPILER_ID **COMPILER_ID_VENDOR** "-" **COMPILER_ID**↩ **_VERSION** "-" COMPILER_ID_STANDARD

6.51 complex_filter.cpp File Reference

6.52 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.

6.53 complex_scale_channel.cpp File Reference

Classes

- class **cfg_t**
- class **complex_scale_channel_t**

6.54 cpuload.cpp File Reference

Classes

- class **cpuload::cpuload_cfg_t**
- class **cpuload::cpuload_if_t**

Namespaces

- **cpuload**

6.55 db.cpp File Reference

Classes

- class **db_t**
- class **db_if_t**

6.56 dbasync.cpp File Reference

Classes

- class **dbasync_native::delay_check_t**
- class **dbasync_native::dbasync_t**
- class **dbasync_native::db_if_t**

Namespaces

- **dbasync_native**

Enumerations

- enum { **dbasync_native::INVALID_THREAD_PRIORITY** = 999999999 }

Functions

- static void * **dbasync_native::thread_start** (void *instance)
- static unsigned **dbasync_native::gcd** (unsigned a, unsigned b)

6.57 dc.cpp File Reference

Macros

- #define **DUPVEC**(x) v.x.data = **MHASignal::dupvec_chk**(v.x.data,s)

Functions

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

  The dynamic compressor implemented by plugin `dc` was created to perform multi-band dynamic compression as found in hearing aids.

6.57.1 Macro Definition Documentation

6.57.1.1 DUPVEC `#define DUPVEC( x ) v.x.data = MHASignal::dupvec_chk(v.x.data,s)`

6.57.2 Function Documentation

6.57.2.1 get_audiochannels()

```
static unsigned int get_audiochannels (
    unsigned int totalchannels,
    std::string acname,
    MHA_AC::algo_comm_t & ac ) [static]
```

The dynamic compressor implemented by plugin `dc` was created to perform multi-band dynamic compression as found in hearing aids.

In hearing aid simulation tasks, openMHA will normally simulate either one or two hearing aids (i.e., aid one or two ears).

The filter bank which splits the broadband input signal from left and/or right microphone is not part of plugin dc. openMHA researchers can use a filterbank plugin of their choice, e.g. `fftfilterbank`. Filter banks split broadband signal channels into multiple narrow-band signal channels: A filter bank with 10 frequency bands would split one broadband signal channel into 10 narrow-band signal channels, and the dc plugin will then only see these 10 narrow-band audio signal channels. The same filter bank would split 2 broadband channels into 20 narrow-band channels, i.e. 10 channels per ear.

For hearing aid fitting, the fitting rule should be able to detect if the dc plugin fits one hearing aid or two hearing aids. The dc plugin can normally not derive this information from the incoming audio signal dimensions alone, but the filterbank that was used to split the broadband signals into narrow-band signal knows the original number of broadband channels. MHA filterbank plugins therefore publish an AC variable containing the original number of broadband audio channels, which were present before the filterbank split the signal into frequency bands.

This function accesses the AC space and reads the number of broadband audio channels from the AC variable `acname`, which must be given by the configuration and should identify the AC variable published by the filter bank for this purpose.

If `acname` is empty, then the function parameter `totalchannels` is returned instead, assuming that there was no filterbank and all input channels that the `dc` plugin receives are broadband audio channels.

Parameters

totalchannels	The number of audio channels that dc receives from the MHA (MHA informs plugins about the number o input signal channels with the prepare() callback). If the `dc` plugin processes the output signal of a filter bank, then this will be the equal to the number of broadband input channels multiplied by the number filter bank bands. If not, this will already be the number of broadband input channels.
acname	If non-empty, name of an AC variable that contains the number of broadband input processed by an up-stream filter bank plugin. An empty acname denotes that no filter bank is present and that the `d` plugin directly processes broadband audio signals.

Parameters

ac	Algorithm communication variable space, needed to access AC variable `acname`.

Returns

The number of broadband audio channels from which the input signal of `dc` was generated. This will usually be 1 or 2 for normal hearing aid simulation task, but the `dc` plugin is not restricted to process only 1 or 2 broadband signals, therefore other return values are possible, but then hearing aid fitting rules querying the dc plugin may become confused.

Warning

Since this function accesses the AC variable space, it may only be called in situation where a plugin is allowed to interact with the AC variable space. This function should be called from prepare()

6.58 dc.hh File Reference

Classes

- class **dc::dc_vars_t**
  Collection of configuration variables of the `dc` plugin.
- class **dc::dc_vars_validator_t**
  Consistency checker.
- class **dc::dc_t**
  Runtime configuration class of dynamic compression plugin `dc`.
- class **dc::dc_if_t**
  Plugin interface class of the dynamic compression plugin `dc`.

Namespaces

- **dc**
  Namespace containing all classes of the `dc` plugin which performs dynamic compression.

6.59 dc_afterburn.cpp File Reference

Namespaces

- **DynComp**
  dynamic compression related classes and functions

Functions

- float **mylogf** (float x)

6.59.1 Function Documentation

6.59.1.1 mylogf() `float mylogf ( float x )`

6.60 dc_afterburn.h File Reference

Classes

- class **DynComp::dc_afterburn_vars_t**
  *Variables for **dc_afterburn_t*** (p. 538) *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

6.61 dc_simple.cpp File Reference

Namespaces

- **dc_simple**

Functions

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

  Checks size of vector.
- std::vector< float > **dc_simple::force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)

  Creates a copy of vector v with s elements, provided that \v has either s elements or 1 elements.
- **mha_real_t dc_simple::not_zero** (**mha_real_t** x, const std::string &comment)

  Helper function to throw an error if x is 0.

6.62 dc_simple.hh File Reference

Classes

- class **dc_simple::dc_vars_t**

  *class for **dc_simple** (p. 89) plugin which registers variables to **MHAParser** (p. 125).*
- class **dc_simple::dc_vars_validator_t**

  Helper class to check sizes of configuration variable vectors.
- class **dc_simple::level_smoother_t**

  Class which computes smoothed input levels on individual bands, using an attack and release filter, which are a first order low pass filter and a maximum tracker filter, respectively.
- class **dc_simple::dc_t**

  *Runtime config class for **dc_simple** (p. 89) plugin.*
- class **dc_simple::dc_t::line_t**

  Helper class for usage in computing compression, expansion and limiting.
- class **dc_simple::dc_if_t**

  *interface class for **dc_simple** (p. 89)*

Namespaces

- **dc_simple**

Typedefs

- typedef **MHAPlugin::plugin_t**< dc_t > **dc_simple::DC**

  Define alternate name for runtime_cfg_t.
- typedef **MHAPlugin::config_t**< level_smoother_t > **dc_simple::LEVEL**

  Define alternate name for config_t.

Functions

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

  Checks size of vector.
- std::vector< float > **dc_simple::force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)

  Creates a copy of vector `v` with `s` elements, provided that \v has either s elements or 1 elements.
- **mha_real_t dc_simple::not_zero** (**mha_real_t** x, const std::string &comment)

  Helper function to throw an error if `x` is 0.

6.63 delay.cpp File Reference

Namespaces

- **delay**

6.64 delay.hh File Reference

Classes

- class **delay::interface_t**

Namespaces

- **delay**

6.65 delaysum_spec.cpp File Reference

Classes

- class **delaysum_spec::delaysum_t**
- class **delaysum_spec::delaysum_spec_if_t**

Namespaces

- **delaysum_spec**

6.66 delaysum_wave.cpp File Reference

Classes

- class **delaysum::delaysum_wave_t**
  Runtime configuration of the delaysum_wave plugin.
- class **delaysum::delaysum_wave_if_t**
  Interface class for the delaysum plugin.

Namespaces

- **delaysum**
  This namespace contains the delaysum plugin.

6.67 denoise.c File Reference

Classes

- struct **DenoiseState**

Macros

- #define **FEAT_LEN** 520
- #define **FFT_SIZE** 128
- #define **FFT_HALF** 65
- #define **NUM_CHAN** 4
- #define **LEN_FILT_T** 1
- #define **SQUARE**(x) ((x)∗(x))
- #define **MAX**(x, y) ((x) > (y) ? (x) : (y))
- #define **NB_FEATURES** **FEAT_LEN**
- #define **TRAINING** 0

Functions

- int **rnnoise_get_size** ()
- int **rnnoise_init** (**DenoiseState** ∗st)
- **DenoiseState** ∗ **rnnoise_create** ()
- void **rnnoise_destroy** (**DenoiseState** ∗st)
- static void **apply_filter_b** (**DenoiseState** ∗st, const float ∗real_part, const float ∗imag↩ _part)
- static void **apply_filter_t** (**DenoiseState** ∗st, float ∗real_out, float ∗imag_out)
- static void **compute_frame_features** (**DenoiseState** ∗st, const float ∗real_part, const float ∗imag_part)
- void **rnnoise_process_frame** (**DenoiseState** ∗st, float ∗real_output, float ∗imag_output, const float ∗real_input, const float ∗imag_input)

Variables

- const struct **RNNModel rnnoise_model_orig**

6.67.1 Macro Definition Documentation

6.67.1.1 FEAT_LEN `#define FEAT_LEN 520`

6.67.1.2 FFT_SIZE `#define FFT_SIZE 128`

6.67.1.3 FFT_HALF `#define FFT_HALF 65`

6.67.1.4 NUM_CHAN `#define NUM_CHAN 4`

6.67.1.5 LEN_FILT_T `#define LEN_FILT_T 1`

6.67.1.6 SQUARE `#define SQUARE(`

```
x ) ((x)*(x))
```

6.67.1.7 MAX `#define MAX(`

```
x,
y ) ((x) > (y) ?  (x) :  (y))
```

6.67.1.8 NB_FEATURES #define NB_FEATURES FEAT_LEN

6.67.1.9 TRAINING #define TRAINING 0

6.67.2 Function Documentation

6.67.2.1 rnnoise_get_size() int rnnoise_get_size ( )

6.67.2.2 rnnoise_init() int rnnoise_init (DenoiseState * st)

6.67.2.3 rnnoise_create() DenoiseState* rnnoise_create ( )

6.67.2.4 rnnoise_destroy() void rnnoise_destroy (DenoiseState * st)

6.67.2.5 apply_filter_b() static void apply_filter_b (DenoiseState * st, const float * real_part, const float * imag_part) [static]

6.67.2.6 apply_filter_t()

```
static void apply_filter_t (
    DenoiseState * st,
    float * real_out,
    float * imag_out ) [static]
```

6.67.2.7 compute_frame_features()

```
static void compute_frame_features (
    DenoiseState * st,
    const float * real_part,
    const float * imag_part ) [static]
```

6.67.2.8 rnnoise_process_frame()

```
void rnnoise_process_frame (
    DenoiseState * st,
    float * real_output,
    float * imag_output,
    const float * real_input,
    const float * imag_input )
```

6.67.3 Variable Documentation

6.67.3.1 rnnoise_model_orig

```
const struct RNNModel rnnoise_model_orig [extern]
```

6.68 denoise.c File Reference

Classes

- struct **DenoiseState**

Macros

- #define **FEAT_LEN** 260
- #define **FFT_SIZE** 128
- #define **FFT_HALF** 65
- #define **NUM_CHAN** 2
- #define **LEN_FILT_T** 1
- #define **SQUARE**(x) ((x)∗(x))
- #define **MAX**(x, y) ((x) > (y) ? (x) : (y))
- #define **NB_FEATURES FEAT_LEN**
- #define **TRAINING** 0

Functions

- int **rnnoise_get_size** ()
- int **rnnoise_init** (**DenoiseState** ∗st)
- **DenoiseState** ∗ **rnnoise_create** ()
- void **rnnoise_destroy** (**DenoiseState** ∗st)
- static void **apply_filter_b** (**DenoiseState** ∗st, const float ∗real_part, const float ∗imag↩ _part)
- static void **apply_filter_t** (**DenoiseState** ∗st, float ∗real_out, float ∗imag_out)
- static void **compute_frame_features** (**DenoiseState** ∗st, const float ∗real_part, const float ∗imag_part)
- void **rnnoise_process_frame** (**DenoiseState** ∗st, float ∗real_output, float ∗imag_output, const float ∗real_input, const float ∗imag_input)

Variables

- const struct **RNNModel** **rnnoise_model_orig**

6.68.1 Macro Definition Documentation

6.68.1.1 FEAT_LEN `#define FEAT_LEN 260`

6.68.1.2 FFT_SIZE `#define FFT_SIZE 128`

6.68.1.3 FFT_HALF `#define FFT_HALF 65`

6.68.1.4 NUM_CHAN `#define NUM_CHAN 2`

6.68.1.5 LEN_FILT_T #define LEN_FILT_T 1

6.68.1.6 SQUARE #define SQUARE(x) ((x)*(x))

6.68.1.7 MAX #define MAX(x, y) ((x) > (y) ? (x) : (y))

6.68.1.8 NB_FEATURES #define NB_FEATURES **FEAT_LEN**

6.68.1.9 TRAINING #define TRAINING 0

6.68.2 Function Documentation

6.68.2.1 rnnoise_get_size() int rnnoise_get_size ( )

6.68.2.2 rnnoise_init() int rnnoise_init (**DenoiseState** * *st*)

6.68.2.3 rnnoise_create() **DenoiseState*** rnnoise_create ( )

6.68.2.4 rnnoise_destroy()

```
void rnnoise_destroy (
    DenoiseState * st )
```

6.68.2.5 apply_filter_b()

```
static void apply_filter_b (
    DenoiseState * st,
    const float * real_part,
    const float * imag_part ) [static]
```

6.68.2.6 apply_filter_t()

```
static void apply_filter_t (
    DenoiseState * st,
    float * real_out,
    float * imag_out ) [static]
```

6.68.2.7 compute_frame_features()

```
static void compute_frame_features (
    DenoiseState * st,
    const float * real_part,
    const float * imag_part ) [static]
```

6.68.2.8 rnnoise_process_frame()

```
void rnnoise_process_frame (
    DenoiseState * st,
    float * real_output,
    float * imag_output,
    const float * real_input,
    const float * imag_input )
```

6.68.3 Variable Documentation

6.68.3.1 rnnoise_model_orig

```
const struct RNNModel rnnoise_model_orig [extern]
```

6.69 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)

6.69.1 Macro Definition Documentation

6.69.1.1 PATCH_VAR `#define PATCH_VAR( var ) patchbay.connect(&var.valuechanged, this, & doasvm_classification↩ ::update_cfg)`

6.69.1.2 INSERT_PATCH `#define INSERT_PATCH( var ) insert_member(var); PATCH_VAR(var)`

6.70 doasvm_classification.h File Reference

Classes

- class **doasvm_classification_config**
- class **doasvm_classification**

6.71 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)

6.71.1 Macro Definition Documentation

6.71.1.1 PATCH_VAR

```
#define PATCH_VAR(
        var ) patchbay.connect(&var.valuechanged, this, & doasvm_feature_↩
extraction::update_cfg)
```

6.71.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
        var )  insert_member(var);  PATCH_VAR(var)
```

6.72 doasvm_feature_extraction.h File Reference

Classes

- class **doasvm_feature_extraction_config**
- class **doasvm_feature_extraction**

6.73 doc_appendix.h File Reference

6.74 doc_examples.h File Reference

6.75 doc_frameworks.h File Reference

6.76 doc_general.h File Reference

6.77 doc_kernel.h File Reference

6.78 doc_matlab.h File Reference

6.79 doc_mhamain.h File Reference

6.80 doc_parser.h File Reference

6.81 doc_plugins.h File Reference

6.82 doc_system.h File Reference

6.83 doc_toolbox.h File Reference

6.84 double2acvar.cpp File Reference

Classes

- class **double2acvar::double2acvar_t**
  *Plugin interface class for **double2acvar** (p. 92).*

Namespaces

- **double2acvar**

6.85 downsample.cpp File Reference

Classes

- class **ds_t**

6.86 dropgen.cpp File Reference

Classes

- class **dropgen_t**

6.87 droptect.cpp File Reference

Classes

- class **droptect_t**
  Detect dropouts in a signal with a constant spectrum.

6.88 equalize.cpp File Reference

Classes

- class **equalize::cfg_t**
- class **equalize::freqgains_t**

Namespaces

- **equalize**

6.89 example1.cpp File Reference

Classes

- class **example1_t**
  This C++ class implements the simplest example plugin for the step-by-step tutorial.

6.90 example2.cpp File Reference

Classes

- class **example2_t**
  This C++ class implements the second example plugin for the step-by-step tutorial.

6.91 example3.cpp File Reference

Classes

- class **example3_t**
  A Plugin class using the openMHA Event mechanism.

6.92 example4.cpp File Reference

Classes

- class **example4_t**
  A Plugin class using the spectral signal.

6.93 example5.cpp File Reference

Classes

- class **example5_t**
- class **plugin_interface_t**

6.94 example6.cpp File Reference

Classes

- class **cfg_t**
- class **example6_t**

6.95 example7.cpp File Reference

6.96 example7.hh File Reference

Classes

- class **example7_t**

6.97 fader_spec.cpp File Reference

Classes

- class **spec_fader_t**
- class **fader_if_t**

6.98 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**

6.98.1 Macro Definition Documentation

6.98.1.1 DEBUG `#define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl`

6.99 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.

6.100 fftfilter.cpp File Reference

Classes

- class **fftfilter::fftfilter_t**
- class **fftfilter::interface_t**

Namespaces

- **fftfilter**

Functions

- unsigned int **fftfilter::irs_length** (const **MHAParser::mfloat_t** &irs)
- unsigned int **fftfilter::irs_validator** (const **MHAParser::mfloat_t** &irs, const unsigned int & **channels**, const unsigned int &fragsize, const unsigned int &fftlen)

6.101 fftfilterbank.cpp File Reference

Classes

- class **fftfilterbank::fftfb_plug_t**
- class **fftfilterbank::fftfb_interface_t**

Namespaces

- **fftfilterbank**

6.102 fshift.cpp File Reference

6.103 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.

6.104 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.

6.105 gain.cpp File Reference

Classes

- class **gain::scaler_t**
- class **gain::gain_if_t**

Namespaces

- **gain**

6.106 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)

6.106.1 Function Documentation

6.106.1.1 convert_f2logf() `std::vector< mha_real_t> convert_f2logf ( const std::vector< mha_real_t > & vF )`

6.106.1.2 isempty() `bool isempty ( const std::vector< std::vector< mha_real_t > > & arg )`

6.107 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.

6.108 gcfsnet_bin.cpp File Reference

Classes

- class **gcfsnet_bin_t**

6.109 gcfsnet_mono.cpp File Reference

Classes

- class **gcfsnet_mono_t**

6.110 generatemhaplugindoc.cpp File Reference

Classes

- class **plug_wrapperl**
- class **io_wrapper**
- class **plug_wrapper**
- class **latex_doc_t**
  Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUMENTATION 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)

6.110.1 Function Documentation

6.110.1.1 conv2latex()

```
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 "}"

6.110.1.2 print_plugin_references()

```
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

6.110.1.3 create_latex_doc()

```
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 appende 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", ...)

6.110.1.4 main()

```
int main (
    int argc,
    char ** argv )
```

6.111 get_rms.cpp File Reference

6.112 get_rms.hh File Reference

Classes

- class **Get_rms_cfg**

  Runtime configuration class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.
- class **Get_rms**

  Plugin interface class for computing the exponentially averaged RMS of the channels of an input signal and storing it in an AC variable.

6.113 gsc_adaptive_stage.cpp File Reference

6.114 gsc_adaptive_stage.hh File Reference

Classes

- class **gsc_adaptive_stage::gsc_adaptive_stage**

Namespaces

- **gsc_adaptive_stage**

Variables

- constexpr **mha_real_t gsc_adaptive_stage::DELT** =1e-12

  Small constant to ensure no division by zero occurs.

6.115 gsc_adaptive_stage_if.cpp File Reference

6.116 gsc_adaptive_stage_if.hh File Reference

Classes

- class **gsc_adaptive_stage::gsc_adaptive_stage_if**

  Plugin interface class.

Namespaces

- **gsc_adaptive_stage**

6.117 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.

6.117.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

6.117.2 Function Documentation

6.117.2.1 filter_complex()

```
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

6.117.2.2 filter_real()

```
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

Parameters

tmp_complex	A reference to a **mha_complex_t** (p. 886) 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. 886)
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

6.118 gtfb_simd.cpp File Reference

Gammatone Filterbank Analyzer Plugin using SIMD.

Classes

- class **gtfb_simd_cfg_t**
- class **gtfb_simd_t**

Macros

- #define **add4f**(a, b) __builtin_ia32_addps(a,b)
- #define **sub4f**(a, b) __builtin_ia32_subps(a,b)
- #define **mul4f**(a, b) __builtin_ia32_mulps(a,b)
- #define **MXCSR_DAZ** (1 << 6) /* Enable denormals are zero mode */
- #define **MXCSR_FTZ** (1 << 15) /* Enable flush to zero mode */
- #define **check_alignment**(ptr, alignment)

  Checks alignment of pointer address.

Functions

- void **filter_sisd_complex** (const unsigned bands, const unsigned order, const **mha_complex_t** ∗inputs, **mha_complex_t** ∗outputs, const **mha_complex_t** ∗coefficients, **mha_complex_t** ∗states)

  Filters one sample per band, using SISD operations and the mha_complex operations.
- void **filter_sisd_real** (const unsigned bands, const unsigned order, const **mha_complex_t** ∗inputs, **mha_complex_t** ∗outputs, const **mha_complex_t** ∗coefficients, **mha_complex_t** ∗states)

  Filters one sample per band, using SISD operations and real operations (operating on real and imaginary part as necessary).
- void **filter_simd** (const unsigned bands, const unsigned order, const **mha_real_t** ∗rinputs, const **mha_real_t** ∗iinputs, **mha_real_t** ∗routputs, **mha_real_t** ∗ioutputs, const **mha_real_t** ∗rcoefficients, const **mha_real_t** ∗icoefficients, **mha_real_t** ∗rstates, **mha_real_t** ∗istates)

  Filters one sample per band, using simd operations on float32 To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameters rstates and istates).

6.118.1 Detailed Description

Gammatone Filterbank Analyzer Plugin using SIMD.

A single-instruction-multiple-data (SIMD) implementation of a gammatone filterbank (GTFB)

Not all functions in this file are actually used. Read the functions in this file as a path to convert an algorithm, here complex-valued gammatone filtering as introduced in Hohmann 2002, from a single-instruction-single-data (SISD) implementation that uses complex arithmetic operations to a SIMD implementation of the same, splitting the complex arithmetic operations into their defining real operations, i.e (a+b).real==a.real+b.real, (a+b).imag==a.imag+b.imag, (a∗b).real==a.real∗b.real-a.imag∗b.imag, (a∗b).imag==a.real∗b.imag+a.imag∗b.real.

6.118.2 Macro Definition Documentation

6.118.2.1 add4f

```
#define add4f(
    a,
    b ) __builtin_ia32_addps(a,b)
```

6.118.2.2 sub4f

```
#define sub4f(
    a,
    b ) __builtin_ia32_subps(a,b)
```

6.118.2.3 mul4f

```
#define mul4f(
    a,
    b ) __builtin_ia32_mulps(a,b)
```

6.118.2.4 MXCSR_DAZ

```
#define MXCSR_DAZ (1 << 6) /* Enable denormals are zero mode */
```

6.118.2.5 MXCSR_FTZ

```
#define MXCSR_FTZ (1 << 15) /* Enable flush to zero mode */
```

6.118.2.6 check_alignment

```
#define check_alignment(
    ptr,
    alignment )
```

Checks alignment of pointer address.

Parameters

ptr	pointer to check
alignment	required alignment

Exceptions

MHA_Error *(p. 906)*	if ptr is not aligned as required.

6.118.3 Function Documentation

6.118.3.1 filter_sisd_complex()

```
void filter_sisd_complex (
    const unsigned bands,
    const unsigned order,
    const  mha_complex_t * inputs,
     mha_complex_t * outputs,
    const  mha_complex_t * coefficients,
     mha_complex_t * states )
```

Filters one sample per band, using SISD operations and the mha_complex operations.

To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameter states).

This function is not actually used in this plugin, but can be used for testing. It implements the Hohmann 2002 filtering in the most readable form, and is translated towards a SIMD implementation in the following functions.

Parameters

bands	Number of total bands to compute (i.e. input_channels ∗ num_frequencies)
order	Gammatone filter order
inputs	Pointer to array of complex input samples, only 1 sample per band
outputs	Pointer to array with space for output samples, 1 complex sample per band
coefficients	Pointer to array of recursive filter coefficient, 1 coefficient per band. The same coefficient is reused for all filter orders.
states	Pointer to array of complex filter states. Array size is bands∗order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state array from filtering the previous sample again, unmodified. The filter state of band b, order o can be found at index [b+o∗bands]

6.118.3.2 filter_sisd_real()

```
void filter_sisd_real (
    const unsigned bands,
    const unsigned order,
    const  mha_complex_t * inputs,
     mha_complex_t * outputs,
    const  mha_complex_t * coefficients,
     mha_complex_t * states )  [inline]
```

Filters one sample per band, using SISD operations and real operations (operating on real and imaginary part as necessary).

To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameter states).

This function is not actually used in this plugin, but can be used for testing. It reimplements filter_sisd_complex, but expands the complex operations into real arithmetics.

Parameters

bands	Number of total bands to compute (i.e. input_channels $*$ num_frequencies)
order	Gammatone filter order
inputs	Pointer to array of complex input samples, only 1 sample per band
outputs	Pointer to array with space for output samples, 1 complex sample per band
coefficients	Pointer to array of recursive filter coefficient, 1 coefficient per band. The same coefficient is reused for all filter orders.
states	Pointer to array of complex filter states. Array size is bands$*$order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state array from filtering the previous sample again, unmodified. The filter state of band b, order o can be found at index [b+o$*$bands]

6.118.3.3 filter_simd()

```
void filter_simd (
 const unsigned bands,
 const unsigned order,
 const  mha_real_t * rinputs,
 const  mha_real_t * iinputs,
  mha_real_t * routputs,
  mha_real_t * ioutputs,
 const  mha_real_t * rcoefficients,
 const  mha_real_t * icoefficients,
  mha_real_t * rstates,
  mha_real_t * istates )  [inline]
```

Filters one sample per band, using simd operations on float32 To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameters rstates and istates).

This reimplements filter_sisd_real, but uses the CPU's vector registers for the arithmetics, and is actually used by this plugin.

Parameters

bands	Number of total bands to compute (i.e. input_channels ∗ num_frequencies) bands is also the size of the arrays pointed to by rinputs, iinputs, routputs, ioutputs, rcoefficients, icoefficients.
order	Gammatone filter order
rinputs	Pointer to array of the real part of input samples
iinputs	Pointer to array of the imaginary part of input samples
routputs	Pointer to array with space for the real parts of the output samples
routputs	Pointer to array with space for the real parts of the output samples
rcoefficients	Pointer to array of the real parts of the recursive filte coefficients
icoefficients	Pointer to array of the imaginary parts of the filter coefficients
rstates	Pointer to array of real parts of filter states. Array size is bands∗order. Initialize all elements with zeros before filtering the first sample. Filter state values wi be modified by the function. For filtering the next sample, this function needs the filter state arrays from filtering the previous sample again, unmodified The real part of the filter state of band b, order o car be found at index [b+o∗bands]
istates	Pointer to array of imaginary parts of filter states. Array size is bands∗order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state arrays from filtering the previous sample again, unmodified. The imaginary part of the filter state of band b, order o can be found at index [b+o∗bands]

6.119 gtfb_simple_bridge.cpp File Reference

Classes

- class **gtfb_simple_rt_t**

  Runtime configuration class of gtfb_simple_bridge plugin.
- class **gtfb_simple_t**

  Interface class of gtfb_simple_bridge plugin.

6.120 hann.cpp File Reference

Macros

- #define **PI** 3.14159265358979323846

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

6.120.1 Macro Definition Documentation

6.120.1.1 PI `#define PI 3.14159265358979323846`

6.120.2 Function Documentation

6.120.2.1 hannf() `float* hannf ( const unsigned int N )`

6.120.2.2 hann() `double* hann ( const unsigned int N )`

6.121 hann.h File Reference

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

6.121.1 Function Documentation

6.121.1.1 hannf() `float* hannf ( const unsigned int N )`

6.121.1.2 hann() `double* hann ( const unsigned int N )`

6.122 identity.cpp File Reference

Classes

- class **identity_t**

6.123 ifftshift.cpp File Reference

Functions

- void **ifftshift** (**mha_wave_t** ∗spec)

6.123.1 Function Documentation

6.123.1.1 ifftshift() `void ifftshift ( mha_wave_t * spec )`

6.124 ifftshift.h File Reference

Functions

- void **ifftshift** (**mha_wave_t** ∗spec)

6.124.1 Function Documentation

6.124.1.1 ifftshift() `void ifftshift (`

mha_wave_t * *spec*)

6.125 iirfilter.cpp File Reference

Classes

- class **iirfilter_t**

6.126 level_matching.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, & **level_matching::level_matching_t::update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

6.126.1 Macro Definition Documentation

6.126.1.1 PATCH_VAR `#define PATCH_VAR(`

var) patchbay.connect(&var.valuechanged, this, & **level_matching::level_matching_t::update_cfg**)

6.126.1.2 INSERT_PATCH `#define INSERT_PATCH(`

var) **insert_member**(var); **PATCH_VAR**(var)

6.127 level_matching.hh File Reference

Classes

- class **level_matching::channel_pair**
- class **level_matching::level_matching_config_t**
- class **level_matching::level_matching_t**

Namespaces

- **level_matching**

6.128 levelmeter.cpp File Reference

Classes

- class **levelmeter_t**

Macros

- #define **PASCALE** 93.979400086720374929

6.128.1 Macro Definition Documentation

6.128.1.1 PASCALE `#define PASCALE 93.979400086720374929`

6.129 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)

6.129.1 Macro Definition Documentation

6.129.1.1 PATCH_VAR

```
#define PATCH_VAR(
    var ) patchbay.connect(&var.valuechanged, this, & lpc::update_cfg)
```

6.129.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
    var )  insert_member(var);  PATCH_VAR(var)
```

6.129.2 Function Documentation

6.129.2.1 Levinson2()

```
void Levinson2 (
    unsigned int P,
    const std::vector< mha_real_t > & R,
    std::vector< mha_real_t > & A )
```

6.130 lpc.h File Reference

Classes

- class **lpc_config**
- class **lpc**

6.131 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)

6.131.1 Macro Definition Documentation

6.131.1.1 PATCH_VAR

```
#define PATCH_VAR(
            var ) patchbay.connect(&var.valuechanged, this, & lpc_bl_predictor↩
::update_cfg)
```

6.131.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
            var )  insert_member(var);  PATCH_VAR(var)
```

6.132 lpc_bl_predictor.h File Reference

Classes

- class **lpc_bl_predictor_config**
- class **lpc_bl_predictor**

Macros

- #define **EPSILON** 1e-10

6.132.1 Macro Definition Documentation

6.132.1.1 EPSILON

```
#define EPSILON 1e-10
```

6.133 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)

6.133.1 Macro Definition Documentation

6.133.1.1 PATCH_VAR `#define PATCH_VAR( var ) patchbay.connect(&var.valuechanged, this, & lpc_burglattice::update_cfg)`

6.133.1.2 INSERT_PATCH `#define INSERT_PATCH( var ) insert_member(var); PATCH_VAR(var)`

6.134 lpc_burg-lattice.h File Reference

Classes

- class **lpc_burglattice_config**
- class **lpc_burglattice**

Macros

- #define **EPSILON** 1e-10

6.134.1 Macro Definition Documentation

6.134.1.1 EPSILON `#define EPSILON 1e-10`

6.135 lsl2ac.cpp File Reference

6.136 lsl2ac.hh File Reference

Classes

- class **lsl2ac::save_var_base_t**
- class **lsl2ac::save_var_t**< **T** >
  LSL to AC bridge variable.
- class **lsl2ac::save_var_t**< **std::string** >
  Specialication for marker streams.
- class **lsl2ac::cfg_t**
  *Runtime configuration class of the **lsl2ac*** (p. 100) *plugin.*
- class **lsl2ac::lsl2ac_t**
  *Plugin class of **lsl2ac*** (p. 100).

Namespaces

- **lsl2ac**

Enumerations

- enum class **lsl2ac::overrun_behavior** { **lsl2ac::Discard** =0 , **lsl2ac::Ignore** }

6.137 matlab_wrapper.cpp File Reference

6.138 matlab_wrapper.hh File Reference

Classes

- struct **matlab_wrapper::types**< **MHA_WAVEFORM** >
- struct **matlab_wrapper::types**< **MHA_SPECTRUM** >
- class **matlab_wrapper::matlab_wrapper_rt_cfg_t**
  Thin wrapper around the emxArray containing the user defined configuration variables.
- class **matlab_wrapper::callback**
  Utility class connecting a user_config_t instance to its corresponding configuration parser.
- class **matlab_wrapper::matlab_wrapper_t**
  Matlab wraper plugin interface class.
- class **matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t**
  Wrapper class around the matlab-generated library.

Namespaces

- **matlab_wrapper**
  Namespace where all classes of the matlab wrapper plugin live.

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.138.1 Macro Definition Documentation

6.138.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN `#define MHAPLUGIN_OVERLOAD_↩ OUTDOMAIN`

6.139 matrixmixer.cpp File Reference

Classes

- class **matrixmixer::cfg_t**
- class **matrixmixer::matmix_t**

Namespaces

- **matrixmixer**

6.140 mconv.cpp File Reference

Classes

- class **mconv::MConv**

Namespaces

- **mconv**

6.141 mha.cpp File Reference

Functions

- int **mhamain** (int argc, char ∗argv[ ])
- int **main** (int argc, char ∗argv[ ])

6.141.1 Function Documentation

6.141.1.1 mhamain() `int mhamain (`

```
int argc,
char * argv[] )
```

6.141.1.2 main() `int main (`

```
int argc,
char * argv[] )
```

6.142 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. 886) 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**

- struct **mha_audio_descriptor_t**
  *Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 996)).*
- struct **mha_audio_t**
  *An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 985) and **mha_spec_t** (p. 937)).*
- struct **mhaconfig_t**
  MHA prepare configuration structure.
- struct **MHA_AC::comm_var_t**
  Algorithm communication variable structure.

Namespaces

- **MHA_AC**

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** 18
  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**

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 unsigned int(* **MHAGetVersion_t**) (void)
- typedef int(* **MHAInit_t**) (**MHA_AC::algo_comm_t** &algo_comm, const char *algo_↩ name, 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 std::string(* **MHASetcpp_t**) (void *h, const std::string &command)
- typedef const char *(* **MHAStrError_t**) (void *h, int err)
- typedef int(* **MHAProc_wave2wave_t**) (void *h, **mha_wave_t** *sIn, **mha_wave_↩ t** **sOut)
- 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** **s↩ Out)
- typedef const char *(* **MHAPluginDocumentation_t**) (void)
- typedef const char *(* **MHAPluginCategory_t**) (void)

Enumerations

- enum **MHA_AC_TYPE_CONSTANTS** : unsigned int {
  MHA_AC_UNKNOWN = 0 , **MHA_AC_CHAR** = 1 , **MHA_AC_INT** = 2 , **MHA_AC_↩ MHAREAL** = 3 ,
  MHA_AC_FLOAT = 4 , **MHA_AC_DOUBLE** = 5 , **MHA_AC_MHACOMPLEX** = 6 , **MHA_AC_USER** = 1000 }

  *Values for the **MHA_AC::comm_var_t::data_type*** (p. 869) `data_type` *field of AC variables in **MHA_AC::comm_var_t*** (p. 868)*.*

6.142.1 Detailed Description

common types for MHA kernel, MHA framework applications and external plugins

6.142.2 Macro Definition Documentation

6.142.2.1 MHA_CALLBACK_TEST #define MHA_CALLBACK_TEST(

x)

Test macro to compare function type definition and declaration.

6.142.2.2 MHA_CALLBACK_TEST_PREFIX #define MHA_CALLBACK_TEST_PREFIX(

prefix,
x)

6.142.2.3 MHA_XSTRF #define MHA_XSTRF(

x) **MHA_STRF**(x)

6.142.2.4 MHA_STRF #define MHA_STRF(

x) #x

6.142.2.5 MHA_VERSION_MAJOR #define MHA_VERSION_MAJOR 4

Major version number of MHA.

6.142.2.6 MHA_VERSION_MINOR #define MHA_VERSION_MINOR 18

Minor version number of MHA.

6.142.2.7 MHA_VERSION_RELEASE #define MHA_VERSION_RELEASE 0

Release number of MHA.

6.142.2.8 MHA_VERSION_BUILD `#define MHA_VERSION_BUILD 0`

Build number of MHA (currently unused)

6.142.2.9 MHA_STRUCT_SIZEMATCH `#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.

6.142.2.10 MHA_VERSION `#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.

6.142.2.11 MHA_VERSION_STRING `#define MHA_VERSION_STRING MHA_XSTRF( MHA_↩ VERSION_MAJOR) "." MHA_XSTRF( MHA_VERSION_MINOR)`

Version string of MHA kernel (major.minor)

6.142.2.12 MHA_RELEASE_VERSION_STRING `#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)

6.142.2.13 MHA_WAVEFORM `#define MHA_WAVEFORM 0`

6.142.2.14 MHA_SPECTRUM `#define MHA_SPECTRUM 1`

6.142.2.15 MHA_DOMAIN_MAX `#define MHA_DOMAIN_MAX 2`

6.142.2.16 MHA_DOMAIN_UNKNOWN `#define MHA_DOMAIN_UNKNOWN` **MHA_DOMAIN_MAX**

6.142.3 Typedef Documentation

6.142.3.1 mha_domain_t `typedef unsigned int` **mha_domain_t**

6.142.3.2 MHAGetVersion_t `typedef unsigned int(* MHAGetVersion_t) (void)`

6.142.3.3 MHAInit_t `typedef int(* MHAInit_t) (` **MHA_AC::algo_comm_t** `&algo_comm, const char *algo_name, void **h)`

6.142.3.4 MHAPrepare_t `typedef int(* MHAPrepare_t) (void *h,` **mhaconfig_t** `*cfg)`

6.142.3.5 MHARelease_t `typedef int(* MHARelease_t) (void *h)`

6.142.3.6 MHADestroy_t

```
typedef void(* MHADestroy_t) (void *h)
```

6.142.3.7 MHASet_t

```
typedef int(* MHASet_t) (void *h, const char *cmd, char *retval,
unsigned int len)
```

6.142.3.8 MHASetcpp_t

```
typedef std::string(* MHASetcpp_t) (void *h, const std↩
::string &command)
```

6.142.3.9 MHAStrError_t

```
typedef const char*(* MHAStrError_t) (void *h, int err)
```

6.142.3.10 MHAProc_wave2wave_t

```
typedef int(* MHAProc_wave2wave_t) (void *h,
mha_wave_t *sIn,  mha_wave_t **sOut)
```

6.142.3.11 MHAProc_wave2spec_t

```
typedef int(* MHAProc_wave2spec_t) (void *h,
mha_wave_t *sIn,  mha_spec_t **sOut)
```

6.142.3.12 MHAProc_spec2wave_t

```
typedef int(* MHAProc_spec2wave_t) (void *h,
mha_spec_t *sIn,  mha_wave_t **sOut)
```

6.142.3.13 MHAProc_spec2spec_t

```
typedef int(* MHAProc_spec2spec_t) (void *h,
mha_spec_t *sIn,  mha_spec_t **sOut)
```

6.142.3.14 MHAPluginDocumentation_t `typedef const char*(* MHAPluginDocumentation↩ _t) (void)`

6.142.3.15 MHAPluginCategory_t `typedef const char*(* MHAPluginCategory_t) (void)`

6.142.4 Enumeration Type Documentation

6.142.4.1 MHA_AC_TYPE_CONSTANTS `enum MHA_AC_TYPE_CONSTANTS : unsigned int`

Values for the **MHA_AC::comm_var_t::data_type** (p. 869) `data_type` field of AC variables in **MHA_AC::comm_var_t** (p. 868).

Enumerator

MHA_AC_UNKNOWN	This value should not be used for AC variables space. It may be used to indicate that the **MHA_AC::comm_var_t** (p. 868) struct has no initialized.
MHA_AC_CHAR	The AC variable points to value(s) of type `cha`
MHA_AC_INT	The AC variable points to value(s) of type `int`
MHA_AC_MHAREAL	The AC variable points to value(s) of type `mha`
MHA_AC_FLOAT	The AC variable points to value(s) of type `flo`
MHA_AC_DOUBLE	The AC variable points to value(s) of type `dou`
MHA_AC_MHACOMPLEX	The AC variable points to value(s) of type `mha_comples_t`.
MHA_AC_USER	This value or any higher value for the **MHA_AC::comm_var_t::data_type** (p. 869) fi that the AC variable is of a user-defined data t

6.143 mha_algo_comm.cpp File Reference

6.144 mha_algo_comm.hh File Reference

Header file for Algorithm Communication.

Classes

- class **MHA_AC::spectrum_t**

  Convenience class for inserting a spectrum into the AC space.
- class **MHA_AC::waveform_t**

  Convenience class for inserting a waveform (a block of time-domain audio signal) 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.
- class **MHA_AC::comm_var_map_t**

  Storage class for the AC variable space.
- class **MHA_AC::algo_comm_t**

  Algorithm communication variable space interface.
- class **MHA_AC::algo_comm_class_t**

  AC variable space implementation.
- class **MHA_AC::scalar_t**< **numeric_t, MHA_AC_TYPECODE** >

  Template for convenience classes for inserting a numeric scalar into the AC space.

Namespaces

- **MHA_AC**

Typedefs

- typedef scalar_t< int, **MHA_AC_INT** > **MHA_AC::int_t**

  Convenience class for inserting an integer variable into the AC space.
- typedef scalar_t< float, **MHA_AC_FLOAT** > **MHA_AC::float_t**

  Convenience class for inserting a single-precision floating-point variable into the AC space.
- typedef scalar_t< double, **MHA_AC_DOUBLE** > **MHA_AC::double_t**

  Convenience class for inserting a double-precision floating-point variable into the AC space.

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.

6.144.1 Detailed Description

Header file for Algorithm Communication.

Functions and classes for Algorithm Communication (AC) support.

6.145 mha_defs.h File Reference

Preprocessor definitions common to all MHA components.

Macros

- #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 **M_PI** 3.14159265358979323846

  Define pi if it is not defined yet.

6.145.1 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.

6.145.2 Macro Definition Documentation

6.145.2.1 CHECK_EXPR #define CHECK_EXPR(

```
            x ) {if(!(x)){throw  MHA_Error(__FILE__,__LINE__,"The expression \"" #x
"\" is invalid.");}}
```

6.145.2.2 CHECK_VAR #define CHECK_VAR(

```
            x ) {if(!(x)){throw  MHA_Error(__FILE__,__LINE__,"The variable \"" #x
"\" is not defined.");}}
```

6.145.2.3 M_PI #define M_PI 3.14159265358979323846

Define pi if it is not defined yet.

6.146 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**]

6.146.1 Macro Definition Documentation

6.146.1.1 STRLEN `#define STRLEN 0x1000`

6.146.2 Function Documentation

6.146.2.1 mha_strerror() `const char* mha_strerror ( int mhaerrno )`

6.146.2.2 mha_set_user_error() `void mha_set_user_error ( const char * str )`

6.146.3 Variable Documentation

6.146.3.1 next_except_str `char next_except_str[ STRLEN] = ""`

6.146.3.2 cstr_strerror `const char* cstr_strerror[ MHA_ERR_USER]`

6.147 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)

6.147.1 Macro Definition Documentation

6.147.1.1 MHA_ERR_SUCCESS `#define MHA_ERR_SUCCESS 0`

6.147.1.2 MHA_ERR_UNKNOWN `#define MHA_ERR_UNKNOWN 1`

6.147.1.3 MHA_ERR_INVALID_HANDLE `#define MHA_ERR_INVALID_HANDLE 2`

6.147.1.4 MHA_ERR_NULL `#define MHA_ERR_NULL 3`

6.147.1.5 MHA_ERR_VARRANGE `#define MHA_ERR_VARRANGE 4`

6.147.1.6 MHA_ERR_VARFMT `#define MHA_ERR_VARFMT 5`

6.147.1.7 MHA_ERR_USER `#define MHA_ERR_USER 10000`

6.147.2 Function Documentation

6.147.2.1 mha_strerror() `const char* mha_strerror ( int mhaerrno )`

6.147.2.2 mha_set_user_error() `void mha_set_user_error ( const char * str )`

6.148 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,...)

6.148.1 Detailed Description

Implementation of openMHA error handling.

This file forms a seperate library.

6.148.2 Function Documentation

6.148.2.1 mha_debug() `void mha_debug (`

```
const char * fmt,
...  )
```

6.149 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. 906).*
- #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. 906) with the values.*

Functions

- void **mha_debug** (const char ∗fmt,...) __attribute__((__format__(printf
  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.

6.149.1 Macro Definition Documentation

6.149.1.1 Getmsg `#define Getmsg( e ) ((e).get_msg())`

6.150 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.

6.151 mha_events.cpp File Reference

6.152 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.

6.153 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)

  Transform the test frequency into the relative position on the filter flank of the given frequency band.

Variables

- **mha_real_t MHAOvlFilter::barkscale::vfreq** [**BARKSCALE_ENTRIES**]
- **mha_real_t MHAOvlFilter::barkscale::vbark** [**BARKSCALE_ENTRIES**]

6.153.1 Macro Definition Documentation

6.153.1.1 BARKSCALE_ENTRIES `#define BARKSCALE_ENTRIES 50`

6.153.2 Function Documentation

6.153.2.1 filtershapefun()

```
mha_real_t filtershapefun (
    mha_real_t f,
    MHAOvlFilter::band_descriptor_t b,
    mha_real_t plateau )
```

Transform the test frequency into the relative position on the filter flank of the given frequency band.

Parameters

f	Test frequency in units corresponding to the chosen frequency scale
b	Descriptor of a single filter bank band: E.g. contains center frequencies of this and the two adjacent bands, and the crossover ("edge") frequencies of this band.
plateau	For non-rectangular filter shapes, specifies what frequency portion of the band around its center frequency should have no attenuation applied.

Precondition

0 <= plateau <= 1

Returns

The position of frequency f on the filter flank as follows: A returned position of 0 means that f is equal to the band's center frequency, or should be treated the same as the center frequency (i.e. is within the band's plateau). A returned position of -1 means that f is <= the lowest frequency of the filter flank (or is an even lower frequency). A returned value of -0.5 means that f is equal to the lower edge frequency. Positive returned values have equivalent meanings for the high half of the filter flank.

6.154 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. 1148).*
- 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**)

6.155 mha_fifo.cpp File Reference

6.156 mha_fifo.h File Reference

Classes

- class **mha_fifo_t**< **T** >
  A FIFO class.
- class **mha_fifo_lf_t**< **T** >
  A lock-free FIFO class for transferring data from a producer thread to a consumer thread.
- 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. 933).
- 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**

6.156.1 Macro Definition Documentation

6.156.1.1 mha_fifo_thread_platform_implementation_t `#define mha_fifo_thread_platform_implementation_t` **`mha_fifo_posix_threads_t`**

6.157 mha_filter.cpp File Reference

Functions

- std::vector< **mha_real_t** > **diff_coeffs** ()

6.157.1 Function Documentation

6.157.1.1 diff_coeffs() `std::vector< mha_real_t> diff_coeffs ( )`

6.158 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

- template<typename T , typename std::enable_if< std::is_floating_point< T >::value, T >::type * = nullptr> void **MHAFilter::make_friendly_number** (T &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.
- std::vector< float > **MHAFilter::fir_lp** (float f_pass_, float f_stop_, float fs_, unsigned order_)
  Setup a nth order fir low pass 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.

6.158.1 Detailed Description

Header file for IIR filter classes.

6.159 mha_generic_chain.cpp File Reference

Functions

- void **mhaconfig_compare** (**mhaconfig_t** req, **mhaconfig_t** avail, const char *cpref)

6.159.1 Function Documentation

6.159.1.1 mhaconfig_compare()

```
void mhaconfig_compare (
    mhaconfig_t req,
    mhaconfig_t avail,
    const char * cpref )
```

6.160 mha_generic_chain.h File Reference

Classes

- class **mhachain::plugs_t**
- class **mhachain::chain_base_t**

Namespaces

- **mhachain**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.160.1 Macro Definition Documentation

6.160.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN

```
#define MHAPLUGIN_OVERLOAD_OUTDOMAIN
```

6.161 mha_git_commit_hash.cpp File Reference

Macros

- #define **GITCOMMITHASH** "independent-plugin-build"

Variables

- const char * **mha_git_commit_hash**
  store git commit hash in every binary plgin to support reproducible research

6.161.1 Macro Definition Documentation

6.161.1.1 GITCOMMITHASH `#define GITCOMMITHASH "independent-plugin-build"`

6.161.2 Variable Documentation

6.161.2.1 mha_git_commit_hash `const char* mha_git_commit_hash`

store git commit hash in every binary plgin to support reproducible research

6.162 mha_git_commit_hash.hh File Reference

Variables

- const char * **mha_git_commit_hash**
  store git commit hash in every binary plgin to support reproducible research

6.162.1 Variable Documentation

6.162.1.1 mha_git_commit_hash `const char* mha_git_commit_hash [extern]`

store git commit hash in every binary plgin to support reproducible research

6.163 mha_io_ifc.h File Reference

Macros

- #define **MHAIO_DOCUMENTATION_PREFIX**(prefix, cat, doc)
- #define **MHAIO_DOCUMENTATION**(plugname, cat, doc) **MHAIO_DOCUMENTATION_PREFIX**(MHA_STATIC_ ## plugname ## _,cat,doc)

Typedefs

- typedef int(∗ **IOProcessEvent_t**) (void ∗handle, **mha_wave_t** ∗sIn, **mha_wave_t** ∗∗s↩ Out)

  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, **IOStopped**↩ **Event_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)

6.163.1 Macro Definition Documentation

6.163.1.1 MHAIO_DOCUMENTATION_PREFIX

```
#define MHAIO_DOCUMENTATION_PREFIX(
    prefix,
    cat,
    doc )
```

6.163.1.2 MHAIO_DOCUMENTATION

```
#define MHAIO_DOCUMENTATION(
    plugname,
    cat,
    doc )   MHAIO_DOCUMENTATION_PREFIX(MHA_STATIC_ ## plugname ## _,cat,doc)
```

6.163.2 Typedef Documentation

6.163.2.1 IOProcessEvent_t

```
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.

6.163.2.2 IOStoppedEvent_t

```
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.

6.163.2.3 IOStartedEvent_t

```
typedef void(* IOStartedEvent_t) (void *handle)
```

Event handler for start event.

This event handler needs to be realtime compatible. The function must return immediatly.

6.163.2.4 IOInit_t

```
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)
```

6.163.2.5 IOPrepare_t

```
typedef int(* IOPrepare_t) (void *handle, int num_inchannels, int num_outchannels)
```

6.163.2.6 IOStart_t

```
typedef int(* IOStart_t) (void *handle)
```

6.163.2.7 IOStop_t

```
typedef int(* IOStop_t) (void *handle)
```

6.163.2.8 IORelease_t `typedef int(* IORelease_t) (void *handle)`

6.163.2.9 IOSetVar_t `typedef int(* IOSetVar_t) (void *handle, const char *cmd, char *retval, unsigned int len)`

6.163.2.10 IOStrError_t `typedef const char*(* IOStrError_t) (void *handle, int err)`

6.163.2.11 IODestroy_t `typedef void(* IODestroy_t) (void *handle)`

6.164 mha_io_utils.cpp File Reference

6.165 mha_io_utils.hh File Reference

Namespaces

- **mhaioutils**

Functions

- template<typename T >
  T **mhaioutils::to_int_clamped** (float val)

6.166 mha_multisrc.cpp File Reference

Namespaces

- **MHAMultiSrc**
  Collection of classes for selecting audio chunks from multiple sources.

6.167 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.

6.168 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)

6.168.1 Function Documentation

6.168.1.1 mha_hasenv() `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

6.168.1.2 mha_getenv() `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

6.168.1.3 mha_delenv() `void mha_delenv ( const std::string & envvar )`

Deletes environment variable from process environment if it exists.

Parameters

envvar	Name of environment variable to delete

6.168.1.4 mha_setenv() `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

6.168.1.5 mha_library_paths() `std::list<std::string> mha_library_paths ( )`

6.168.1.6 list_dir() `std::list<std::string> list_dir (`

```
const std::string & path,
const std::string & pattern )
```

6.169 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**

  Wrapper class around a shared library.
- class **pluginlib_t**

  *Specialisation of **dynamiclib_t** (p. 531) for mha plugin libraries.*

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)

6.169.1 Macro Definition Documentation

6.169.1.1 mha_loadlib

```
#define mha_loadlib(
  x ) dlopen(x,RTLD_NOW)
```

6.169.1.2 mha_freelib

```
#define mha_freelib(
  x ) dlclose(x)
```

6.169.1.3 mha_freelib_success

```
#define mha_freelib_success(
  x ) (x == 0)
```

6.169.1.4 mha_getlibfun

```
#define mha_getlibfun(
  h,
  x ) x ## _cb = (x ## _t)dlsym(h,#x)
```

6.169.1.5 mha_getlibfun_checked

```
#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.")
```

6.169.1.6 mha_loadlib_error

```
#define mha_loadlib_error(
  x ) dlerror()
```

6.169.1.7 mha_lib_extension

```
#define mha_lib_extension ".so"
```

6.169.1.8 mha_msleep

```
#define mha_msleep(
  milliseconds ) usleep((milliseconds)*1000)
```

6.169.1.9 FMTsz `#define FMTsz "%zu"`

printf modifier to print integers of type size_t

6.169.1.10 MHA_RESOLVE `#define MHA_RESOLVE(`

```
h,
t ) t ## _cb = (t ## _t)(h->resolve(#t))
```

6.169.1.11 MHA_RESOLVE_CHECKED `#define MHA_RESOLVE_CHECKED(`

```
h,
t ) t ## _cb = (t ## _t)(h->resolve_checked(#t))
```

6.169.2 Typedef Documentation

6.169.2.1 mha_libhandle_t `typedef void* mha_libhandle_t`

6.169.3 Function Documentation

6.169.3.1 mha_getenv() `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

6.169.3.2 mha_hasenv() `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

6.169.3.3 mha_setenv() `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

6.169.3.4 mha_delenv() `void mha_delenv ( const std::string & envvar )`

Deletes environment variable from process environment if it exists.

Parameters

envvar	Name of environment variable to delete

6.169.3.5 mha_library_paths() `std::list<std::string> mha_library_paths ( )`

6.169.3.6 list_dir() `std::list<std::string> list_dir (`

```
const std::string & path,
const std::string & pattern )
```

6.169.3.7 mha_hton() [1/3] `void mha_hton (`

```
float * data,
unsigned int len ) [inline]
```

6.169.3.8 mha_ntoh() [1/3] `void mha_ntoh (`

```
float * data,
unsigned int len ) [inline]
```

6.169.3.9 mha_hton() [2/3] `void mha_hton (`

```
uint32_t * data,
unsigned int len ) [inline]
```

6.169.3.10 mha_ntoh() [2/3] `void mha_ntoh (`

```
uint32_t * data,
unsigned int len ) [inline]
```

6.169.3.11 mha_hton() [3/3]

```
void mha_hton (
int32_t * data,
unsigned int len ) [inline]
```

6.169.3.12 mha_ntoh() [3/3]

```
void mha_ntoh (
int32_t * data,
unsigned int len ) [inline]
```

6.170 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)
  count number of brackets
- 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 void **check_parenthesis_complex** (const std::string &str)
  This function checks for unbalanced parenthesis in the string containing complex number.
- static int **check_sign_complex** (const std::string &str)
  This function checks for valid sign (b/w real & img.
- 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.

6.170.1 Macro Definition Documentation

6.170.1.1 MHAPLATFORM `#define MHAPLATFORM "undefined-linux"`

6.170.2 Function Documentation

6.170.2.1 write_float() `static std::ostream& write_float ( std::ostream & o, const float & f ) [inline], [static]`

6.170.2.2 parse_1_float() `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

Precondition

s.size() > 0

Parameters

v	The float variable to fill with a value

Returns

The rest of the string.

6.170.2.3 check_parenthesis_complex() `static void check_parenthesis_complex ( const std::string & str ) [static]`

This function checks for unbalanced parenthesis in the string containing complex number.

Parameters

str	The string to check. This function returns normally only when the string starts with an opening bracket and ends with a closing bracket.

Precondition

str.size() $>$ 0

Exceptions

MHA_ErrorMsg	This function raises an error with an appropriate er message if the string does not start with an openin bracket '(' or if it does not end with a closing bracke

6.170.2.4 check_sign_complex() `static int check_sign_complex ( const std::string & str ) [static]`

This function checks for valid sign (b/w real & img.

parts of complex number). It also checks if real part is missing in complex number.

Parameters

str	String containing sign and onward imaginary part.

Precondition

str.size() $>$ 0

Exceptions

This	function raises an error if wrong sign (other than '+' or '-') is found. It also raises error, if it finds 'i' instead of sign. This can happen when real part is missing in complex number and 'i' (of imaginary part) is found where sign was expected.

Returns

+1 for '+' sign and -1 for '-' sign

6.170.2.5 parse_1_complex() `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

Exceptions

MHA_ErrorMsg	This function raises an error if s does not have characters except spaces, tabs etc

Returns

The rest of the string.

6.171 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::mint_t**
  Matrix variable with int 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.
- int **MHAParser::StrCnv::num_brackets** (const std::string &s)
  count number of brackets

6.171.1 Detailed Description

Header file for the MHA-Parser script language.

6.171.2 Macro Definition Documentation

6.171.2.1 DEFAULT_RETSIZE `#define DEFAULT_RETSIZE 0x100000`

6.171.2.2 insert_member `#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. 1249).

6.172 mha_plugin.cpp File Reference

6.173 mha_plugin.hh File Reference

Header file for MHA C++ plugin class templates.

Classes

- class **MHAPlugin::cfg_node_t**< **runtime_cfg_t** >

  A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.
- 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 **MHAPLUGIN_PROC_CALLBACK_PREFIX**(prefix, classname, indom, outdom)
- #define **MHAPLUGIN_SETCPP_CALLBACK_PREFIX**(prefix, classname)
- #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) **MHAPLUGIN_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) **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.

6.173.1 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).

6.173.2 Macro Definition Documentation

6.173.2.1 __declspec `#define __declspec(`

p `)`

6.173.2.2 WINAPI #define WINAPI

6.173.2.3 HINSTANCE #define HINSTANCE int

6.173.2.4 MHAPLUGIN_PROC_CALLBACK_PREFIX #define MHAPLUGIN_PROC_CALLBACK↩ _PREFIX(

prefix,
classname,
indom,
outdom)

6.173.2.5 MHAPLUGIN_SETCPP_CALLBACK_PREFIX #define MHAPLUGIN_SETCPP_↩ CALLBACK_PREFIX(

prefix,
classname)

6.173.2.6 MHAPLUGIN_INIT_CALLBACKS_PREFIX #define MHAPLUGIN_INIT_CALLBACKS↩ _PREFIX(

prefix,
classname)

6.173.2.7 MHAPLUGIN_CALLBACKS_PREFIX #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. 1298) template class. Exceptions of type **MHA_Error** (p. 906) are caught and transformed into apropriate error codes with their corresponding error messages.

6.173.2.8 MHAPLUGIN_DOCUMENTATION_PREFIX

```
#define MHAPLUGIN_DOCUMENTATION_PREFIX(
            prefix,
            cat,
            doc )
```

6.173.2.9 MHAPLUGIN_PROC_CALLBACK

```
#define MHAPLUGIN_PROC_CALLBACK(
            plugname,
            classname,
            indom,
            outdom )   MHAPLUGIN_PROC_CALLBACK_PREFIX(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)
```

6.173.2.10 MHAPLUGIN_INIT_CALLBACKS

```
#define MHAPLUGIN_INIT_CALLBACKS(
            plugname,
            classname )   MHAPLUGIN_INIT_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _,classname)
```

6.173.2.11 MHAPLUGIN_CALLBACKS `#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 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. 1298) template class. Exceptions of type **MHA_Error** (p. 906) are caught and transformed into apropriate error codes with their corresponding error messages.

6.173.2.12 MHAPLUGIN_DOCUMENTATION `#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.

6.174 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)

6.174.1 Function Documentation

6.174.1.1 mha_tic() `void mha_tic (`

mha_tictoc_t * *t*)

6.174.1.2 mha_platform_tic() `void mha_platform_tic (`

mha_platform_tictoc_t * *t*)

6.174.1.3 mha_toc() `float mha_toc (`

mha_tictoc_t * *t*)

6.174.1.4 mha_platform_toc() `float mha_platform_toc (`

mha_platform_tictoc_t * *t*)

6.175 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)

6.175.1 Typedef Documentation

6.175.1.1 mha_platform_tictoc_t `typedef mha_tictoc_t mha_platform_tictoc_t`

6.175.2 Function Documentation

6.175.2.1 mha_platform_tic() `void mha_platform_tic ( mha_platform_tictoc_t * t )`

6.175.2.2 mha_platform_toc() `float mha_platform_toc ( mha_platform_tictoc_t * t )`

6.176 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** ()

6.176.1 Typedef Documentation

6.176.1.1 rb_f_t `typedef VALUE(* rb_f_t) (...)`

6.176.2 Function Documentation

6.176.2.1 mha_free() `static void mha_free (`
`void * mha ) [static]`

6.176.2.2 mha_alloc() `static VALUE mha_alloc (`
`VALUE klass ) [static]`

6.176.2.3 mha_exit_request() `static VALUE mha_exit_request (`
`VALUE self ) [static]`

6.176.2.4 mha_parse() `static VALUE mha_parse (`
`VALUE self,`
`VALUE request ) [static]`

6.176.2.5 Init_mha_ruby() `void Init_mha_ruby ( )`

6.177 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. 985) 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. 985) 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. 937) 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.

6.177.1 Macro Definition Documentation

6.177.1.1 MHA_ID_UINT_VECTOR `#define MHA_ID_UINT_VECTOR "MHASignal::uint_↩ vector_t"`

6.177.1.2 MHA_ID_MATRIX `#define MHA_ID_MATRIX "MHASignal::matrix_t"`

6.177.1.3 ASSERT_EQUAL_DIM `#define ASSERT_EQUAL_DIM(`

```
a,
b )
```

6.177.1.4 ASSERT_EQUAL_DIM_PTR #define ASSERT_EQUAL_DIM_PTR(*a*, *b*)

6.177.2 Function Documentation

6.177.2.1 set_minabs() void set_minabs (**mha_spec_t** & *self*, const **mha_real_t** & *m*)

6.177.2.2 safe_div() **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.

6.177.2.3 intensity() static **mha_real_t** intensity (const **mha_spec_t** & *s*, unsigned int *channel*, unsigned int *fftlen*, **mha_real_t** * *sqfreq_response* = *0*) [static]

6.178 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. 937))*
- class **MHASignal::waveform_t**
  *signal processing class for waveform data (based on **mha_wave_t** (p. 985))*
- 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. 985).
- **mha_real_t MHASignal::lin2db** (**mha_real_t** x, **mha_real_t** eps)
  Conversion from linear scale to dB (no SPL reference)
- **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::sq2db** (**mha_real_t** x, **mha_real_t** eps=0.0f)
  conversion from squared values to dB (no SPL reference)
- **mha_real_t MHASignal::db2sq** (**mha_real_t** x)
  conversion from dB to squared values (no SPL reference)
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x, **mha_real_t** eps)
  Conversion from linear Pascal scale to dB SPL.
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x)
  Conversion from linear 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::pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=0.0f)
  Conversion from squared Pascal scale to dB SPL.
- **mha_real_t MHASignal::dbspl2pa2** (**mha_real_t** x)
  conversion from dB SPL to squared 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. 985) *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. 985) *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. 937) *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=nullptr)
  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. 886) variable.*
- **mha_complex_t** **mha_complex** (**mha_real_t** real, **mha_real_t** imag=0)

  *Create a new **mha_complex_t** (p. 886) 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. 886) 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. 886).*
- **mha_complex_t** & **expi** (**mha_complex_t** &self, **mha_real_t** **angle**)

  *replaces the value of the given **mha_complex_t** (p. 886) 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. 886) 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. 886) with its conjugate.*
- void **conjugate** (**mha_spec_t** &self)

  *Replace (!) the value of this **mha_spec_t** (p. 937) 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. 886)*
- std::istream & **operator**>> (std::istream &i, **mha_complex_t** &c)
  *preliminary istream operator for **mha_complex_t** (p. 886) 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.

6.178.1 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.

6.178.2 Macro Definition Documentation

6.178.2.1 M_PI `#define M_PI 3.14159265358979323846`

6.178.2.2 mha_round `#define mha_round( x ) (int)((float)x+0.5)`

6.178.3 Function Documentation

6.178.3.1 mha_min_1()

```
unsigned int mha_min_1 (
    unsigned int a ) [inline]
```

6.178.3.2 value() [1/2]

```
mha_real_t& value (
    mha_wave_t * s,
    unsigned int k ) [inline]
```

6.178.3.3 value() [2/2]

```
mha_complex_t& value (
    mha_spec_t * s,
    unsigned int k ) [inline]
```

6.178.3.4 set_minabs()

```
void set_minabs (
    mha_spec_t & self,
    const mha_real_t & m )
```

6.178.3.5 safe_div()

```
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.

6.178.3.6 operator<<()

```
std::ostream& operator<< (
    std::ostream & o,
    const mha_complex_t & c ) [inline]
```

ostream operator for **mha_complex_t** (p. 886)

6.178.3.7 operator>>() std::istream& operator>> (std::istream & *i*, **mha_complex_t** & *c*) [inline]

preliminary istream operator for **mha_complex_t** (p. 886) without error checking

6.179 mha_signal_fft.h File Reference

Classes

- class **MHASignal::fft_t**

Namespaces

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

6.180 mha_tablelookup.cpp File Reference

6.181 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.

6.181.1 Detailed Description

Header file for table lookup classes.

6.182 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**

6.182.1 Macro Definition Documentation

6.182.1.1 INVALID_SOCKET `#define INVALID_SOCKET (-1)`

6.182.1.2 SOCKET_ERROR `#define SOCKET_ERROR (-1)`

6.182.1.3 closesocket `#define closesocket(`
fd `) (close((fd)))`

6.182.1.4 ASYNC_CONNECT_STARTED `#define ASYNC_CONNECT_STARTED EINPROGRESS`

6.182.2 Typedef Documentation

6.182.2.1 SOCKET `typedef int` **SOCKET**

6.182.3 Function Documentation

6.182.3.1 host_port_to_sock_addr()

```
static sockaddr_in host_port_to_sock_addr (
    const std::string & host,
    unsigned short port ) [static]
```

6.182.3.2 tcp_connect_to()

```
static SOCKET tcp_connect_to (
    const std::string & host,
    unsigned short port ) [static]
```

6.182.3.3 tcp_connect_to_with_timeout()

```
static SOCKET tcp_connect_to_with_timeout
(
    const std::string & host,
    unsigned short port,
    Timeout_Watcher & timeout_watcher ) [static]
```

6.182.3.4 thread_start_func()

```
static void* thread_start_func (
    void * thread ) [static]
```

6.183 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. 946) 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.

6.183.1 Macro Definition Documentation

6.183.1.1 Sleep `#define Sleep( x ) usleep((x)*1000);`

6.184 mha_tcp_server.cpp File Reference

Namespaces

- **mha_tcp**

  namespace for network communication classes of MHA

6.185 mha_tcp_server.hh File Reference

Classes

- class **mha_tcp::buffered_socket_t**

  An asio TCP socket with an associated streambuf buffer for receiving lines of text, as well as string buffers for sending responses.
- class **mha_tcp::server_t**

  Class for accepting TCP connections from clients.

Namespaces

- **mha_tcp**

  namespace for network communication classes of MHA

6.186 mha_toolbox.h File Reference

6.187 mha_utils.cpp File Reference

6.188 mha_utils.hh File Reference

Namespaces

- **MHAUtils**

Functions

- bool **MHAUtils::is_multiple_of** (const unsigned big, const unsigned small)
- bool **MHAUtils::is_power_of_two** (const unsigned n)
- bool **MHAUtils::is_multiple_of_by_power_of_two** (const unsigned big, const unsigned small)
- std::string **MHAUtils::strip** (const std::string &line)
- std::string **MHAUtils::remove** (const std::string &str_, char c)
- bool **MHAUtils::is_denormal** (**mha_real_t** x)

  Get the normal-ness of a mha_real_t.
- bool **MHAUtils::is_denormal** (const **mha_complex_t** &x)

  Get the normal-ness of a complex number.
- bool **MHAUtils::is_denormal** (const std::complex< **mha_real_t** > &x)

  Get the normal-ness of a complex number.
- **mha_real_t MHAUtils::spl2hl** (**mha_real_t** f)

  Get the offset of between dB(SPL) and dB(HL) for a given frequency according to ISO 389-7:2005 (freefield); e.g.

6.189 mha_windowparser.cpp File Reference

Variables

- float(* **wnd_funs** [])(float)

6.189.1 Variable Documentation

6.189.1.1 wnd_funs `float(* wnd_funs[])(float) ( float )`

6.190 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.

6.191 mhachain.cpp File Reference

Classes

- class **mhachain::mhachain_t**

Namespaces

- **mhachain**

6.192 mhafw_lib.cpp File Reference

6.193 mhafw_lib.h File Reference

Classes

- class **io_lib_t**
  Class for loading MHA sound IO module.
- class **fw_vars_t**
- class **fw_t**

6.194 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 *, int err)
- void **IODestroy** (void *handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.194.1 Macro Definition Documentation

6.194.1.1 DBG `#define DBG(`

```
x ) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)
```

6.194.1.2 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.194.1.3 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.194.1.4 ERR_USER `#define ERR_USER -1000`

6.194.1.5 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.194.1.6 IOInit `#define IOInit MHA_STATIC_MHAIOalsa_IOInit`

6.194.1.7 IOPrepare `#define IOPrepare MHA_STATIC_MHAIOalsa_IOPrepare`

6.194.1.8 IOStart `#define IOStart MHA_STATIC_MHAIOalsa_IOStart`

6.194.1.9 IOStop `#define IOStop MHA_STATIC_MHAIOalsa_IOStop`

6.194.1.10 IORelease `#define IORelease MHA_STATIC_MHAIOalsa_IORelease`

6.194.1.11 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOalsa_IOSetVar`

6.194.1.12 IOStrError `#define IOStrError MHA_STATIC_MHAIOalsa_IOStrError`

6.194.1.13 IODestroy `#define IODestroy MHA_STATIC_MHAIOalsa_IODestroy`

6.194.1.14 dummy_interface_test

```
void dummy_interface_test(
    void ) MHA_STATIC_MHAIOalsa_dummy_interface_test
```

6.194.2 Function Documentation

6.194.2.1 IOInit()

```
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.

6.194.2.2 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

IO library prepare function, called after the MHA prepared the processing plugins.

6.194.2.3 IOStart()

```
int IOStart (
    void * handle )
```

6.194.2.4 IOStop() int IOStop (

```
void * handle )
```

6.194.2.5 IORelease() int IORelease (

```
void * handle )
```

6.194.2.6 IOSetVar() int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

6.194.2.7 IOStrError() const char* IOStrError (

```
void * ,
int err )
```

6.194.2.8 IODestroy() void IODestroy (

```
void * handle )
```

6.194.3 Variable Documentation

6.194.3.1 user_err_msg char user_err_msg[MAX_USER_ERR] [static]

6.195 MHAIOAsterisk.cpp File Reference

Classes

- class **io_asterisk_parser_t**
  The parser interface of the IOAsterisk library.
- class **io_asterisk_sound_t**
  Sound data handling of io tcp library.
- class **io_asterisk_fwcb_t**
  TCP sound-io library's interface to the framework callbacks.
- class **io_asterisk_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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.195.1 Macro Definition Documentation

6.195.1.1 ERR_SUCCESS #define ERR_SUCCESS 0

6.195.1.2 ERR_IHANDLE #define ERR_IHANDLE -1

6.195.1.3 ERR_USER #define ERR_USER -1000

6.195.1.4 MAX_USER_ERR #define MAX_USER_ERR 0x2000

6.195.1.5 MHA_ErrorMsg2 #define MHA_ErrorMsg2(

```
x,
y )  MHA_Error(__FILE__,__LINE__,(x),(y))
```

6.195.1.6 MHA_ErrorMsg3 #define MHA_ErrorMsg3(

```
x,
y,
z )  MHA_Error(__FILE__,__LINE__,(x),(y),(z))
```

6.195.1.7 MIN_TCP_PORT #define MIN_TCP_PORT 0

6.195.1.8 MIN_TCP_PORT_STR #define MIN_TCP_PORT_STR "0"

6.195.1.9 MAX_TCP_PORT

```
#define MAX_TCP_PORT 65535
```

6.195.1.10 MAX_TCP_PORT_STR

```
#define MAX_TCP_PORT_STR "65535"
```

6.195.1.11 IOInit

```
#define IOInit MHA_STATIC_MHAIOTCP_IOInit
```

6.195.1.12 IOPrepare

```
#define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare
```

6.195.1.13 IOStart

```
#define IOStart MHA_STATIC_MHAIOTCP_IOStart
```

6.195.1.14 IOStop

```
#define IOStop MHA_STATIC_MHAIOTCP_IOStop
```

6.195.1.15 IORelease

```
#define IORelease MHA_STATIC_MHAIOTCP_IORelease
```

6.195.1.16 IOSetVar

```
#define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar
```

6.195.1.17 IOStrError

```
#define IOStrError MHA_STATIC_MHAIOTCP_IOStrError
```

6.195.1.18 IODestroy `#define IODestroy MHA_STATIC_MHAIOTCP_IODestroy`

6.195.1.19 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIOTCP_dummy_interface_test`

6.195.2 Function Documentation

6.195.2.1 copy_error() `static int copy_error ( MHA_Error & e ) [static]`

6.195.2.2 thread_startup_function() `static void* thread_startup_function ( void * parameter ) [static]`

6.195.2.3 IOInit() `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 )`

6.195.2.4 IOPrepare() `int IOPrepare ( void * handle, int num_inchannels, int num_outchannels )`

6.195.2.5 IOStart() int IOStart (void * *handle*)

6.195.2.6 IOStop() int IOStop (void * *handle*)

6.195.2.7 IORelease() int IORelease (void * *handle*)

6.195.2.8 IOSetVar() int IOSetVar (void * *handle*, const char * *cmd*, char * *retval*, unsigned int *len*)

6.195.2.9 IOStrError() const char* IOStrError (void * *handle*, int *err*)

6.195.2.10 IODestroy() void IODestroy (void * *handle*)

6.195.3 Variable Documentation

6.195.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] [static]

6.196 MHAIODummy.cpp File Reference

Classes

- class **io_dummy_t**
  Dummy sound io library.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIODummy_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIODummy_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIODummy_IOStart
- #define **IOStop** MHA_STATIC_MHAIODummy_IOStop
- #define **IORelease** MHA_STATIC_MHAIODummy_IORelease
- #define **IOSetVar** MHA_STATIC_MHAIODummy_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIODummy_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIODummy_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIODummy_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 ∗, int err)
- void **IODestroy** (void ∗handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

6.196.1 Macro Definition Documentation

6.196.1.1 ERR_SUCCESS

```
#define ERR_SUCCESS 0
```

6.196.1.2 ERR_IHANDLE

```
#define ERR_IHANDLE -1
```

6.196.1.3 ERR_USER

```
#define ERR_USER -1000
```

6.196.1.4 MAX_USER_ERR

```
#define MAX_USER_ERR 0x500
```

6.196.1.5 IOInit

```
#define IOInit MHA_STATIC_MHAIODummy_IOInit
```

6.196.1.6 IOPrepare

```
#define IOPrepare MHA_STATIC_MHAIODummy_IOPrepare
```

6.196.1.7 IOStart

```
#define IOStart MHA_STATIC_MHAIODummy_IOStart
```

6.196.1.8 IOStop

```
#define IOStop MHA_STATIC_MHAIODummy_IOStop
```

6.196.1.9 IORelease

```
#define IORelease MHA_STATIC_MHAIODummy_IORelease
```

6.196.1.10 IOSetVar `#define IOSetVar MHA_STATIC_MHAIODummy_IOSetVar`

6.196.1.11 IOStrError `#define IOStrError MHA_STATIC_MHAIODummy_IOStrError`

6.196.1.12 IODestroy `#define IODestroy MHA_STATIC_MHAIODummy_IODestroy`

6.196.1.13 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIODummy_dummy_interface_test`

6.196.2 Function Documentation

6.196.2.1 IOInit() `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 )
```

6.196.2.2 IOPrepare() `int IOPrepare (`

```
void * handle,
int nch_in,
int nch_out )
```

6.196.2.3 IOStart() int IOStart (

```
void * handle )
```

6.196.2.4 IOStop() int IOStop (

```
void * handle )
```

6.196.2.5 IORelease() int IORelease (

```
void * handle )
```

6.196.2.6 IOSetVar() int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

6.196.2.7 IOStrError() const char* IOStrError (

```
void * ,
int err )
```

6.196.2.8 IODestroy() void IODestroy (

```
void * handle )
```

6.196.3 Variable Documentation

6.196.3.1 user_err_msg char user_err_msg[MAX_USER_ERR] = "" [static]

6.197 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 *, int err)
- void **IODestroy** (void *handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.197.1 Macro Definition Documentation

6.197.1.1 DEBUG

```
#define DEBUG(
            x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " <<
x << std::endl
```

6.197.1.2 ERR_SUCCESS

```
#define ERR_SUCCESS 0
```

6.197.1.3 ERR_IHANDLE

```
#define ERR_IHANDLE -1
```

6.197.1.4 ERR_USER

```
#define ERR_USER -1000
```

6.197.1.5 MAX_USER_ERR

```
#define MAX_USER_ERR 0x500
```

6.197.1.6 IOInit

```
#define IOInit MHA_STATIC_MHAIOFile_IOInit
```

6.197.1.7 IOPrepare

```
#define IOPrepare MHA_STATIC_MHAIOFile_IOPrepare
```

6.197.1.8 IOStart

```
#define IOStart MHA_STATIC_MHAIOFile_IOStart
```

6.197.1.9 IOStop `#define IOStop MHA_STATIC_MHAIOFile_IOStop`

6.197.1.10 IORelease `#define IORelease MHA_STATIC_MHAIOFile_IORelease`

6.197.1.11 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOFile_IOSetVar`

6.197.1.12 IOStrError `#define IOStrError MHA_STATIC_MHAIOFile_IOStrError`

6.197.1.13 IODestroy `#define IODestroy MHA_STATIC_MHAIOFile_IODestroy`

6.197.1.14 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIOFile_dummy_interface_test`

6.197.2 Function Documentation

6.197.2.1 IOInit() `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 )
```

6.197.2.2 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

6.197.2.3 IOStart()

```
int IOStart (
    void * handle )
```

6.197.2.4 IOStop()

```
int IOStop (
    void * handle )
```

6.197.2.5 IORelease()

```
int IORelease (
    void * handle )
```

6.197.2.6 IOSetVar()

```
int IOSetVar (
    void * handle,
    const char * command,
    char * retval,
    unsigned int maxretlen )
```

6.197.2.7 IOStrError()

```
const char* IOStrError (
    void * ,
    int err )
```

6.197.2.8 IODestroy()

```
void IODestroy (
    void * handle )
```

6.197.3 Variable Documentation

6.197.3.1 user_err_msg `char user_err_msg[ MAX_USER_ERR] [static]`

6.198 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 ∗, int err)
- void **IODestroy** (void ∗handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

6.198.1 Macro Definition Documentation

6.198.1.1 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.198.1.2 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.198.1.3 ERR_USER `#define ERR_USER -1000`

6.198.1.4 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.198.1.5 IOInit `#define IOInit MHA_STATIC_MHAIOJack_IOInit`

6.198.1.6 IOPrepare `#define IOPrepare MHA_STATIC_MHAIOJack_IOPrepare`

6.198.1.7 IOStart `#define IOStart MHA_STATIC_MHAIOJack_IOStart`

6.198.1.8 IOStop `#define IOStop MHA_STATIC_MHAIOJack_IOStop`

6.198.1.9 IORelease `#define IORelease MHA_STATIC_MHAIOJack_IORelease`

6.198.1.10 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOJack_IOSetVar`

6.198.1.11 IOStrError `#define IOStrError MHA_STATIC_MHAIOJack_IOStrError`

6.198.1.12 IODestroy `#define IODestroy MHA_STATIC_MHAIOJack_IODestroy`

6.198.1.13 dummy_interface_test `#define dummy_interface_test(`
void) MHA_STATIC_MHAIOJack_dummy_interface_test

6.198.2 Function Documentation

6.198.2.1 IOInit() `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 )
```

6.198.2.2 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

6.198.2.3 IOStart()

```
int IOStart (
    void * handle )
```

6.198.2.4 IOStop()

```
int IOStop (
    void * handle )
```

6.198.2.5 IORelease()

```
int IORelease (
    void * handle )
```

6.198.2.6 IOSetVar()

```
int IOSetVar (
    void * handle,
    const char * command,
    char * retval,
    unsigned int maxretlen )
```

6.198.2.7 IOStrError()

```
const char* IOStrError (
    void * ,
    int err )
```

6.198.2.8 IODestroy()

```
void IODestroy (
    void * handle )
```

6.198.3 Variable Documentation

6.198.3.1 user_err_msg `char user_err_msg[ MAX_USER_ERR] = "" [static]`

6.199 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 ∗, int err)
- void **IODestroy** (void ∗handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

6.199.1 Macro Definition Documentation

6.199.1.1 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.199.1.2 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.199.1.3 ERR_USER `#define ERR_USER -1000`

6.199.1.4 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.199.1.5 IOInit `#define IOInit MHA_STATIC_MHAIOJackdb_IOInit`

6.199.1.6 IOPrepare `#define IOPrepare MHA_STATIC_MHAIOJackdb_IOPrepare`

6.199.1.7 IOStart `#define IOStart MHA_STATIC_MHAIOJackdb_IOStart`

6.199.1.8 IOStop `#define IOStop MHA_STATIC_MHAIOJackdb_IOStop`

6.199.1.9 IORelease `#define IORelease MHA_STATIC_MHAIOJackdb_IORelease`

6.199.1.10 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOJackdb_IOSetVar`

6.199.1.11 IOStrError `#define IOStrError MHA_STATIC_MHAIOJackdb_IOStrError`

6.199.1.12 IODestroy `#define IODestroy MHA_STATIC_MHAIOJackdb_IODestroy`

6.199.1.13 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIOJackdb_dummy_interface_test`

6.199.2 Function Documentation

6.199.2.1 IOInit() `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 )
```

6.199.2.2 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

6.199.2.3 IOStart()

```
int IOStart (
    void * handle )
```

6.199.2.4 IOStop()

```
int IOStop (
    void * handle )
```

6.199.2.5 IORelease()

```
int IORelease (
    void * handle )
```

6.199.2.6 IOSetVar()

```
int IOSetVar (
    void * handle,
    const char * command,
    char * retval,
    unsigned int maxretlen )
```

6.199.2.7 IOStrError()

```
const char* IOStrError (
    void * ,
    int err )
```

6.199.2.8 IODestroy()

```
void IODestroy (
    void * handle )
```

6.199.3 Variable Documentation

6.199.3.1 user_err_msg `char user_err_msg[ MAX_USER_ERR] = "" [static]`

6.200 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, **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 ∗, int err)
- void **IODestroy** (void ∗handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.200.1 Macro Definition Documentation

6.200.1.1 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.200.1.2 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.200.1.3 ERR_USER `#define ERR_USER -1000`

6.200.1.4 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.200.1.5 IOInit `#define IOInit MHA_STATIC_MHAIOParser_IOInit`

6.200.1.6 IOPrepare `#define IOPrepare MHA_STATIC_MHAIOParser_IOPrepare`

6.200.1.7 IOStart `#define IOStart MHA_STATIC_MHAIOParser_IOStart`

6.200.1.8 IOStop `#define IOStop MHA_STATIC_MHAIOParser_IOStop`

6.200.1.9 IORelease `#define IORelease MHA_STATIC_MHAIOParser_IORelease`

6.200.1.10 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOParser_IOSetVar`

6.200.1.11 IOStrError `#define IOStrError MHA_STATIC_MHAIOParser_IOStrError`

6.200.1.12 IODestroy `#define IODestroy MHA_STATIC_MHAIOParser_IODestroy`

6.200.1.13 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIOParser_dummy_interface_test`

6.200.2 Function Documentation

6.200.2.1 IOInit() `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 )
```

6.200.2.2 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

6.200.2.3 IOStart()

```
int IOStart (
    void * handle )
```

6.200.2.4 IOStop()

```
int IOStop (
    void * handle )
```

6.200.2.5 IORelease()

```
int IORelease (
    void * handle )
```

6.200.2.6 IOSetVar()

```
int IOSetVar (
    void * handle,
    const char * command,
    char * retval,
    unsigned int maxretlen )
```

6.200.2.7 IOStrError()

```
const char* IOStrError (
    void * ,
    int err )
```

6.200.2.8 IODestroy()

```
void IODestroy (
    void * handle )
```

6.200.3 Variable Documentation

6.200.3.1 user_err_msg `char user_err_msg[ MAX_USER_ERR] [static]`

6.201 MHAIOPortAudio.cpp File Reference

Classes

- class **MHAIOPortAudio::stream_info_t**
- 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 *userData)
- 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 *, int err)
- void **IODestroy** (void *handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""
- PaStreamCallback **portaudio_callback**

6.201.1 Macro Definition Documentation

6.201.1.1 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.201.1.2 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.201.1.3 ERR_USER `#define ERR_USER -1000`

6.201.1.4 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.201.1.5 IOInit

```
#define IOInit MHA_STATIC_MHAIOPortAudio_IOInit
```

6.201.1.6 IOPrepare

```
#define IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare
```

6.201.1.7 IOStart

```
#define IOStart MHA_STATIC_MHAIOPortAudio_IOStart
```

6.201.1.8 IOStop

```
#define IOStop MHA_STATIC_MHAIOPortAudio_IOStop
```

6.201.1.9 IORelease

```
#define IORelease MHA_STATIC_MHAIOPortAudio_IORelease
```

6.201.1.10 IOSetVar

```
#define IOSetVar MHA_STATIC_MHAIOPortAudio_IOSetVar
```

6.201.1.11 IOStrError

```
#define IOStrError MHA_STATIC_MHAIOPortAudio_IOStrError
```

6.201.1.12 IODestroy

```
#define IODestroy MHA_STATIC_MHAIOPortAudio_IODestroy
```

6.201.1.13 dummy_interface_test

```
#define dummy_interface_test(
    void ) MHA_STATIC_MHAIOPortAudio_dummy_interface_test
```

6.201.2 Function Documentation

6.201.2.1 portaudio_callback()

```
int portaudio_callback (
    const void * input,
    void * output,
    unsigned long frameCount,
    const PaStreamCallbackTimeInfo * timeInfo,
    PaStreamCallbackFlags statusFlags,
    void * userData )
```

6.201.2.2 IOInit()

```
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 )
```

6.201.2.3 IOPrepare()

```
int IOPrepare (
    void * handle,
    int nch_in,
    int nch_out )
```

6.201.2.4 IOStart()

```
int IOStart (
    void * handle )
```

6.201.2.5 IOStop()

```
int IOStop (
    void * handle )
```

6.201.2.6 IORelease() int IORelease (

void * *handle*)

6.201.2.7 IOSetVar() int IOSetVar (

void * *handle*,
const char * *command*,
char * *retval*,
unsigned int *maxretlen*)

6.201.2.8 IOStrError() const char* IOStrError (

void * ,
int *err*)

6.201.2.9 IODestroy() void IODestroy (

void * *handle*)

6.201.3 Variable Documentation

6.201.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] = "" [static]

6.201.3.2 portaudio_callback PaStreamCallback portaudio_callback

6.202 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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

6.202.1 Macro Definition Documentation

6.202.1.1 ERR_SUCCESS `#define ERR_SUCCESS 0`

6.202.1.2 ERR_IHANDLE `#define ERR_IHANDLE -1`

6.202.1.3 ERR_USER `#define ERR_USER -1000`

6.202.1.4 MAX_USER_ERR `#define MAX_USER_ERR 0x2000`

6.202.1.5 MHA_ErrorMsg2 #define MHA_ErrorMsg2(

```
x,
y ) MHA_Error(__FILE__,__LINE__,(x),(y))
```

6.202.1.6 MHA_ErrorMsg3 #define MHA_ErrorMsg3(

```
x,
y,
z ) MHA_Error(__FILE__,__LINE__,(x),(y),(z))
```

6.202.1.7 MIN_TCP_PORT #define MIN_TCP_PORT 0

6.202.1.8 MIN_TCP_PORT_STR #define MIN_TCP_PORT_STR "0"

6.202.1.9 MAX_TCP_PORT #define MAX_TCP_PORT 65535

6.202.1.10 MAX_TCP_PORT_STR #define MAX_TCP_PORT_STR "65535"

6.202.1.11 IOInit #define IOInit MHA_STATIC_MHAIOTCP_IOInit

6.202.1.12 IOPrepare #define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare

6.202.1.13 IOStart `#define IOStart MHA_STATIC_MHAIOTCP_IOStart`

6.202.1.14 IOStop `#define IOStop MHA_STATIC_MHAIOTCP_IOStop`

6.202.1.15 IORelease `#define IORelease MHA_STATIC_MHAIOTCP_IORelease`

6.202.1.16 IOSetVar `#define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar`

6.202.1.17 IOStrError `#define IOStrError MHA_STATIC_MHAIOTCP_IOStrError`

6.202.1.18 IODestroy `#define IODestroy MHA_STATIC_MHAIOTCP_IODestroy`

6.202.1.19 dummy_interface_test `#define dummy_interface_test( void ) MHA_STATIC_MHAIOTCP_dummy_interface_test`

6.202.2 Function Documentation

6.202.2.1 copy_error() `static int copy_error ( MHA_Error & e ) [static]`

6.202.2.2 thread_startup_function()

```
static void* thread_startup_function (
    void * parameter )  [static]
```

6.202.2.3 IOInit()

```
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 )
```

6.202.2.4 IOPrepare()

```
int IOPrepare (
    void * handle,
    int num_inchannels,
    int num_outchannels )
```

6.202.2.5 IOStart()

```
int IOStart (
    void * handle )
```

6.202.2.6 IOStop()

```
int IOStop (
    void * handle )
```

6.202.2.7 IORelease()

```
int IORelease (
    void * handle )
```

6.202.2.8 IOSetVar() int IOSetVar (

```
void * handle,
const char * cmd,
char * retval,
unsigned int len )
```

6.202.2.9 IOStrError() const char* IOStrError (

```
void * handle,
int err )
```

6.202.2.10 IODestroy() void IODestroy (

```
void * handle )
```

6.202.3 Variable Documentation

6.202.3.1 user_err_msg char user_err_msg[MAX_USER_ERR] [static]

6.203 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

6.203.1 Function Documentation

6.203.1.1 jack_error_handler()

```
static void jack_error_handler (
    const char * msg ) [static]
```

6.203.1.2 dummy_jack_proc_cb()

```
static int dummy_jack_proc_cb (
    jack_nframes_t ,
    void * ) [static]
```

6.203.1.3 make_friendly_number()

```
void make_friendly_number (
    jack_default_audio_sample_t & x ) [inline]
```

6.203.2 Variable Documentation

6.203.2.1 last_jack_err_msg

```
char last_jack_err_msg[ MAX_USER_ERR] = ""
```

6.203.2.2 last_jack_err

```
int last_jack_err = 0
```

6.204 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**]

6.204.1 Macro Definition Documentation

6.204.1.1 MHAJACK_FW_STARTED `#define MHAJACK_FW_STARTED 1`

6.204.1.2 MHAJACK_STOPPED `#define MHAJACK_STOPPED 2`

6.204.1.3 MHAJACK_STARTING `#define MHAJACK_STARTING 8`

6.204.1.4 IO_ERROR_JACK `#define IO_ERROR_JACK 11`

6.204.1.5 IO_ERROR_MHAJACKLIB `#define IO_ERROR_MHAJACKLIB 12`

6.204.1.6 MAX_USER_ERR `#define MAX_USER_ERR 0x500`

6.204.2 Variable Documentation

6.204.2.1 last_jack_err_msg `char last_jack_err_msg[ MAX_USER_ERR] [extern]`

6.205 mhamain.cpp File Reference

Classes

- class **mhaserver_t**
  MHA Framework listening on TCP port for commands.
- class **mhaserver_t::tcp_server_t**

Macros

- #define **HELP_TEXT**
- #define **NORELEASE_WARNING**
- #define **VERSION_EXTENSION** "+"
- #define **BUILDHOST_INFO** ""
- #define **GREETING_TEXT**

Functions

- int **mhamain** (int argc, char ∗argv[ ])

6.205.1 Macro Definition Documentation

6.205.1.1 HELP_TEXT `#define HELP_TEXT`

6.205.1.2 NORELEASE_WARNING `#define NORELEASE_WARNING`

6.205.1.3 VERSION_EXTENSION `#define VERSION_EXTENSION "+"`

6.205.1.4 BUILDHOST_INFO `#define BUILDHOST_INFO ""`

6.205.1.5 GREETING_TEXT `#define GREETING_TEXT`

6.205.2 Function Documentation

6.205.2.1 mhamain()

```
int mhamain (
    int argc,
    char * argv[] )
```

6.206 mhapluginloader.cpp File Reference

6.207 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. 996) structures, and report differences as an error.*

6.208 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)

6.208.1 Function Documentation

6.208.1.1 write_wave() void write_wave (

```
const  mha_wave_t & sig,
const char * fname,
const float & srate,
const int & format )
```

6.208.1.2 validator_channels() unsigned int validator_channels (

```
std::vector< int > channel_map,
unsigned int channels )
```

6.208.1.3 validator_length() unsigned int validator_length (

```
unsigned int maxlen,
unsigned int frames,
unsigned int startpos )
```

6.209 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)

6.209.1 Function Documentation

6.209.1.1 write_wave()

```
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 )
```

6.210 multibandcompressor.cpp File Reference

Classes

- class **multibandcompressor::plugin_signals_t**
- class **multibandcompressor::fftfb_plug_t**
- class **multibandcompressor::interface_t**

Namespaces

- **multibandcompressor**

6.211 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)

6.211.1 Macro Definition Documentation

6.211.1.1 NORMALIZATION_TYPES `#define NORMALIZATION_TYPES "[none default sum]"`

6.211.1.2 NORM_NONE `#define NORM_NONE 0`

6.211.1.3 NORM_DEFAULT `#define NORM_DEFAULT 1`

6.211.1.4 NORM_SUM `#define NORM_SUM 2`

6.211.1.5 ESTIMATION_TYPES `#define ESTIMATION_TYPES "[previous current]"`

6.211.1.6 ESTIM_PREV `#define ESTIM_PREV 0`

6.211.1.7 ESTIM_CUR `#define ESTIM_CUR 1`

6.211.2 Function Documentation

6.211.2.1 make_friendly_number_by_limiting() void make_friendly_number_by_limiting (mha_real_t & x) [inline]

6.212 noise.cpp File Reference

Classes

- class **cfg_t**
- class **noise_t**

6.213 noise_psd_estimator.cpp File Reference

Classes

- class **noise_psd_estimator::noise_psd_estimator_t**
- class **noise_psd_estimator::noise_psd_estimator_if_t**

Namespaces

- **noise_psd_estimator**

Macros

- #define **POWSPEC_FACTOR** 0.0025

6.213.1 Macro Definition Documentation

6.213.1.1 POWSPEC_FACTOR #define POWSPEC_FACTOR 0.0025

6.214 osc2ac.cpp File Reference

Classes

- class **osc_variable_t**
  Class for converting messages received at a single osc address to a single AC variable.
- class **osc_server_t**
  OSC receiver implemented using liblo.
- class **osc2ac_t**

6.215 overlapadd.cpp File Reference

Namespaces

- **overlapadd**

6.216 overlapadd.hh File Reference

Classes

- class **overlapadd::overlapadd_t**
- class **overlapadd::overlapadd_if_t**

Namespaces

- **overlapadd**

6.217 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)

6.218 pluginbrowser.cpp File Reference

6.219 pluginbrowser.h File Reference

Classes

- class **plugindescription_t**
- class **pluginloader_t**
- class **pluginbrowser_t**

6.220 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)

6.220.1 Macro Definition Documentation

6.220.1.1 PATCH_VAR

```
#define PATCH_VAR(
    var ) patchbay.connect(&var.valuechanged, this, & prediction_error
::update_cfg)
```

6.220.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
    var ) insert_member(var); PATCH_VAR(var)
```

6.220.2 Function Documentation

6.220.2.1 make_friendly_number_by_limiting() void make_friendly_number_by_limiting (mha_real_t & x) [inline]

6.221 prediction_error.h File Reference

Classes

- class **prediction_error_config**
- class **prediction_error**

6.222 proc_counter.cpp File Reference

Classes

- class **proc_counter_t**

6.223 resampling.cpp File Reference

Classes

- class **MHAPlugin_Resampling::resampling_t**
- class **MHAPlugin_Resampling::resampling_if_t**

Namespaces

- **MHAPlugin_Resampling**

6.224 rmslevel.cpp File Reference

Classes

- class **rmslevel::rmslevel_if_t**
  Rmslevel plugin.

Namespaces

- **rmslevel**

Enumerations

- enum class **rmslevel::UNIT** { **rmslevel::SPL** =0 , **rmslevel::HL** =1 }

6.225 rnn.c File Reference

Functions

- static float **tansig_approx** (float x)
- static float **sigmoid_approx** (float x)
- static float **relu** (float x)
- void **compute_dense** (const **DenseLayer** *layer, float *output, const float *input)
- void **compute_dense_grouped** (const **DenseLayer** *layer, float *output, const float *input)
- void **compute_dconv_1x1_grouped** (const **DConvLayer1x1** *dclayer, float *output, const float *input, const float *input_skip)
- void **add_skip** (int input_size, float *output, const float *input, const float *input_skip)
- void **compute_scale** (const **ScalerLayer** *dclayer, float *output, const float *input)
- void **compute_dconv_1xX_grouped** (const **DConvLayer** *dclayer, **counter** *idx_start, **counter** *idx_write, float *buffer, float *output, const float *input)
- void **compute_gru_grouped** (const **GRULayer** *gru, float *state, const float *input)
- void **compute_rnn** (**RNNState** *rnn, float *filter_b, float *filter_t, const float *input)

6.225.1 Function Documentation

6.225.1.1 tansig_approx() `static float tansig_approx ( float x ) [inline], [static]`

6.225.1.2 sigmoid_approx() `static float sigmoid_approx ( float x ) [inline], [static]`

6.225.1.3 relu() `static float relu ( float x ) [inline], [static]`

6.225.1.4 compute_dense()

```
void compute_dense (
    const  DenseLayer * layer,
    float * output,
    const float * input )
```

6.225.1.5 compute_dense_grouped()

```
void compute_dense_grouped (
    const  DenseLayer * layer,
    float * output,
    const float * input )
```

6.225.1.6 compute_dconv_1x1_grouped()

```
void compute_dconv_1x1_grouped (
    const  DConvLayer1x1 * dclayer,
    float * output,
    const float * input,
    const float * input_skip )
```

6.225.1.7 add_skip()

```
void add_skip (
    int input_size,
    float * output,
    const float * input,
    const float * input_skip )
```

6.225.1.8 compute_scale()

```
void compute_scale (
    const  ScalerLayer * dclayer,
    float * output,
    const float * input )
```

6.225.1.9 compute_dconv_1xX_grouped()

```
void compute_dconv_1xX_grouped (
    const  DConvLayer * dclayer,
     counter * idx_start,
     counter * idx_write,
    float * buffer,
    float * output,
    const float * input )
```

6.225.1.10 compute_gru_grouped()

```
void compute_gru_grouped (
const  GRULayer * gru,
float * state,
const float * input )
```

6.225.1.11 compute_rnn()

```
void compute_rnn (
 RNNState * rnn,
float * filter_b,
float * filter_t,
const float * input )
```

6.226 rnn.c File Reference

Functions

- static float **tansig_approx** (float x)
- static float **sigmoid_approx** (float x)
- static float **relu** (float x)
- void **compute_dense** (const **DenseLayer** *layer, float *output, const float *input)
- void **compute_dense_grouped** (const **DenseLayer** *layer, float *output, const float *input)
- void **compute_dconv_1x1_grouped** (const **DConvLayer1x1** *dclayer, float *output, const float *input, const float *input_skip)
- void **add_skip** (int input_size, float *output, const float *input, const float *input_skip)
- void **compute_scale** (const **ScalerLayer** *dclayer, float *output, const float *input)
- void **compute_dconv_1xX_grouped** (const **DConvLayer** *dclayer, **counter** *idx_start, **counter** *idx_write, float *buffer, float *output, const float *input)
- void **compute_gru_grouped** (const **GRULayer** *gru, float *state, const float *input)
- void **compute_rnn** (**RNNState** *rnn, float *filter_b, float *filter_t, const float *input)

6.226.1 Function Documentation

6.226.1.1 tansig_approx()

```
static float tansig_approx (
float x )  [inline], [static]
```

6.226.1.2 sigmoid_approx()

```
static float sigmoid_approx (
    float x ) [inline], [static]
```

6.226.1.3 relu()

```
static float relu (
    float x ) [inline], [static]
```

6.226.1.4 compute_dense()

```
void compute_dense (
    const DenseLayer * layer,
    float * output,
    const float * input )
```

6.226.1.5 compute_dense_grouped()

```
void compute_dense_grouped (
    const DenseLayer * layer,
    float * output,
    const float * input )
```

6.226.1.6 compute_dconv_1x1_grouped()

```
void compute_dconv_1x1_grouped (
    const DConvLayer1x1 * dclayer,
    float * output,
    const float * input,
    const float * input_skip )
```

6.226.1.7 add_skip()

```
void add_skip (
    int input_size,
    float * output,
    const float * input,
    const float * input_skip )
```

6.226.1.8 compute_scale()

```
void compute_scale (
    const  ScalerLayer * dclayer,
    float * output,
    const float * input )
```

6.226.1.9 compute_dconv_1xX_grouped()

```
void compute_dconv_1xX_grouped (
    const  DConvLayer * dclayer,
     counter * idx_start,
     counter * idx_write,
    float * buffer,
    float * output,
    const float * input )
```

6.226.1.10 compute_gru_grouped()

```
void compute_gru_grouped (
    const  GRULayer * gru,
    float * state,
    const float * input )
```

6.226.1.11 compute_rnn()

```
void compute_rnn (
     RNNState * rnn,
    float * filter_b,
    float * filter_t,
    const float * input )
```

6.227 rnn.h File Reference

Classes

- struct **DenseLayer**
- struct **DConvLayer1x1**
- struct **ScalerLayer**
- struct **DConvLayer**
- struct **GRULayer**

Macros

- #define **WEIGHTS_SCALE** (1.f/128)
- #define **WEIGHTS_SCALE_BIAS** (1.f/32768)
- #define **MAX_NEURONS** 32
- #define **ACTIVATION_TANH** 0
- #define **ACTIVATION_SIGMOID** 1
- #define **ACTIVATION_RELU** 2
- #define **ACTIVATION_LINEAR** 3
- #define **NUM_GROUPS** 8

Typedefs

- typedef int8_t **rnn_weight**
- typedef int16_t **rnn_bias**
- typedef int16_t **counter**
- typedef struct **RNNState** **RNNState**

Functions

- void **compute_dense** (const **DenseLayer** ∗layer, float ∗output, const float ∗input)
- void **compute_dense_grouped** (const **DenseLayer** ∗layer, float ∗output, const float ∗input)
- void **compute_gru_grouped** (const **GRULayer** ∗gru, float ∗state, const float ∗input)
- void **add_skip** (int input_size, float ∗output, const float ∗input, const float ∗input_skip)
- void **compute_rnn** (**RNNState** ∗rnn, float ∗filter_b, float ∗filter_t, const float ∗input)
- void **compute_dconv_1x1_grouped** (const **DConvLayer1x1** ∗dclayer, float ∗output, const float ∗input, const float ∗input_skip)
- void **compute_scale** (const **ScalerLayer** ∗dclayer, float ∗output, const float ∗input)
- void **compute_dconv_1xX_grouped** (const **DConvLayer** ∗dclayer, **counter** ∗idx_start, **counter** ∗idx_write, float ∗buffer, float ∗output, const float ∗input)

6.227.1 Macro Definition Documentation

6.227.1.1 WEIGHTS_SCALE `#define WEIGHTS_SCALE (1.f/128)`

6.227.1.2 WEIGHTS_SCALE_BIAS #define WEIGHTS_SCALE_BIAS (1.f/32768)

6.227.1.3 MAX_NEURONS #define MAX_NEURONS 32

6.227.1.4 ACTIVATION_TANH #define ACTIVATION_TANH 0

6.227.1.5 ACTIVATION_SIGMOID #define ACTIVATION_SIGMOID 1

6.227.1.6 ACTIVATION_RELU #define ACTIVATION_RELU 2

6.227.1.7 ACTIVATION_LINEAR #define ACTIVATION_LINEAR 3

6.227.1.8 NUM_GROUPS #define NUM_GROUPS 8

6.227.2 Typedef Documentation

6.227.2.1 rnn_weight typedef int8_t **rnn_weight**

6.227.2.2 rnn_bias typedef int16_t **rnn_bias**

6.227.2.3 counter typedef int16_t counter

6.227.2.4 RNNState typedef struct RNNState RNNState

6.227.3 Function Documentation

6.227.3.1 compute_dense() void compute_dense (

```
const DenseLayer * layer,
float * output,
const float * input )
```

6.227.3.2 compute_dense_grouped() void compute_dense_grouped (

```
const DenseLayer * layer,
float * output,
const float * input )
```

6.227.3.3 compute_gru_grouped() void compute_gru_grouped (

```
const GRULayer * gru,
float * state,
const float * input )
```

6.227.3.4 add_skip() void add_skip (

```
int input_size,
float * output,
const float * input,
const float * input_skip )
```

6.227.3.5 compute_rnn()

```
void compute_rnn (
    RNNState * rnn,
    float * filter_b,
    float * filter_t,
    const float * input )
```

6.227.3.6 compute_dconv_1x1_grouped()

```
void compute_dconv_1x1_grouped (
    const  DConvLayer1x1 * dclayer,
    float * output,
    const float * input,
    const float * input_skip )
```

6.227.3.7 compute_scale()

```
void compute_scale (
    const  ScalerLayer * dclayer,
    float * output,
    const float * input )
```

6.227.3.8 compute_dconv_1xX_grouped()

```
void compute_dconv_1xX_grouped (
    const  DConvLayer * dclayer,
    counter * idx_start,
    counter * idx_write,
    float * buffer,
    float * output,
    const float * input )
```

6.228 rnn.h File Reference

Classes

- struct **DenseLayer**
- struct **DConvLayer1x1**
- struct **ScalerLayer**
- struct **DConvLayer**
- struct **GRULayer**

Macros

- #define **WEIGHTS_SCALE** (1.f/128)
- #define **WEIGHTS_SCALE_BIAS** (1.f/32768)
- #define **MAX_NEURONS** 32
- #define **ACTIVATION_TANH** 0
- #define **ACTIVATION_SIGMOID** 1
- #define **ACTIVATION_RELU** 2
- #define **ACTIVATION_LINEAR** 3
- #define **NUM_GROUPS** 8

Typedefs

- typedef int8_t **rnn_weight**
- typedef int16_t **rnn_bias**
- typedef int16_t **counter**
- typedef struct **RNNState** **RNNState**

Functions

- void **compute_dense** (const **DenseLayer** ∗layer, float ∗output, const float ∗input)
- void **compute_dense_grouped** (const **DenseLayer** ∗layer, float ∗output, const float ∗input)
- void **compute_gru_grouped** (const **GRULayer** ∗gru, float ∗state, const float ∗input)
- void **add_skip** (int input_size, float ∗output, const float ∗input, const float ∗input_skip)
- void **compute_rnn** (**RNNState** ∗rnn, float ∗filter_b, float ∗filter_t, const float ∗input)
- void **compute_dconv_1x1_grouped** (const **DConvLayer1x1** ∗dclayer, float ∗output, const float ∗input, const float ∗input_skip)
- void **compute_scale** (const **ScalerLayer** ∗dclayer, float ∗output, const float ∗input)
- void **compute_dconv_1xX_grouped** (const **DConvLayer** ∗dclayer, **counter** ∗idx_start, **counter** ∗idx_write, float ∗buffer, float ∗output, const float ∗input)

6.228.1 Macro Definition Documentation

6.228.1.1 WEIGHTS_SCALE `#define WEIGHTS_SCALE (1.f/128)`

6.228.1.2 WEIGHTS_SCALE_BIAS #define WEIGHTS_SCALE_BIAS (1.f/32768)

6.228.1.3 MAX_NEURONS #define MAX_NEURONS 32

6.228.1.4 ACTIVATION_TANH #define ACTIVATION_TANH 0

6.228.1.5 ACTIVATION_SIGMOID #define ACTIVATION_SIGMOID 1

6.228.1.6 ACTIVATION_RELU #define ACTIVATION_RELU 2

6.228.1.7 ACTIVATION_LINEAR #define ACTIVATION_LINEAR 3

6.228.1.8 NUM_GROUPS #define NUM_GROUPS 8

6.228.2 Typedef Documentation

6.228.2.1 rnn_weight typedef int8_t **rnn_weight**

6.228.2.2 rnn_bias typedef int16_t **rnn_bias**

6.228.2.3 counter typedef int16_t counter

6.228.2.4 RNNState typedef struct RNNState RNNState

6.228.3 Function Documentation

6.228.3.1 compute_dense() void compute_dense (

```
const DenseLayer * layer,
float * output,
const float * input )
```

6.228.3.2 compute_dense_grouped() void compute_dense_grouped (

```
const DenseLayer * layer,
float * output,
const float * input )
```

6.228.3.3 compute_gru_grouped() void compute_gru_grouped (

```
const GRULayer * gru,
float * state,
const float * input )
```

6.228.3.4 add_skip() void add_skip (

```
int input_size,
float * output,
const float * input,
const float * input_skip )
```

6.228.3.5 compute_rnn()

```
void compute_rnn (
    RNNState * rnn,
    float * filter_b,
    float * filter_t,
    const float * input )
```

6.228.3.6 compute_dconv_1x1_grouped()

```
void compute_dconv_1x1_grouped (
    const  DConvLayer1x1 * dclayer,
    float * output,
    const float * input,
    const float * input_skip )
```

6.228.3.7 compute_scale()

```
void compute_scale (
    const  ScalerLayer * dclayer,
    float * output,
    const float * input )
```

6.228.3.8 compute_dconv_1xX_grouped()

```
void compute_dconv_1xX_grouped (
    const  DConvLayer * dclayer,
    counter * idx_start,
    counter * idx_write,
    float * buffer,
    float * output,
    const float * input )
```

6.229 rnn_data.c File Reference

Variables

- static const float **scaler_weights** [1]
- static const float **scaler_bias** [1]
- static const **ScalerLayer scaler**
- static const **rnn_weight dense_projection_weights** [66560]
- static const **rnn_bias dense_projection_bias** [128]
- static const **DenseLayer dense_projection**
- static const **rnn_weight dense_map_1_weights** [512]
- static const **rnn_bias dense_map_1_bias** [32]

- static const **DenseLayer dense_map_1**
- static const **rnn_weight dconv_5_weights** [160]
- static const **rnn_bias dconv_5_bias** [32]
- static const **DConvLayer dconv_5**
- static const **rnn_weight dense_dconv_5_weights** [1024]
- static const **rnn_bias dense_dconv_5_bias** [32]
- static const **DenseLayer dense_dconv_5**
- static const **rnn_weight dconv_3_weights** [96]
- static const **rnn_bias dconv_3_bias** [32]
- static const **DConvLayer dconv_3**
- static const **rnn_weight dense_dconv_3_weights** [1024]
- static const **rnn_bias dense_dconv_3_bias** [32]
- static const **DenseLayer dense_dconv_3**
- static const **rnn_weight dconv1x1_skip_1_weights** [32]
- static const **rnn_bias dconv1x1_skip_1_bias** [32]
- static const **DConvLayer1x1 dconv1x1_skip_1**
- static const **rnn_weight dense_tac1_1_weights** [512]
- static const **rnn_bias dense_tac1_1_bias** [16]
- static const **DenseLayer dense_tac1_1**
- static const **rnn_weight dense_tac1_2_weights** [16384]
- static const **rnn_bias dense_tac1_2_bias** [128]
- static const **DenseLayer dense_tac1_2**
- static const **rnn_weight dense_tac1_3_weights** [512]
- static const **rnn_bias dense_tac1_3_bias** [32]
- static const **DenseLayer dense_tac1_3**
- static const **rnn_weight gru_2_1_weights** [3072]
- static const **rnn_weight gru_2_1_recurrent_weights** [3072]
- static const **rnn_bias gru_2_1_bias** [192]
- static const **GRULayer gru_2_1**
- static const **rnn_weight gru_2_2_weights** [3072]
- static const **rnn_weight gru_2_2_recurrent_weights** [3072]
- static const **rnn_bias gru_2_2_bias** [192]
- static const **GRULayer gru_2_2**
- static const **rnn_weight dconv1x1_skip_2_weights** [32]
- static const **rnn_bias dconv1x1_skip_2_bias** [32]
- static const **DConvLayer1x1 dconv1x1_skip_2**
- static const **rnn_weight dense_tac2_1_weights** [512]
- static const **rnn_bias dense_tac2_1_bias** [16]
- static const **DenseLayer dense_tac2_1**
- static const **rnn_weight dense_tac2_2_weights** [16384]
- static const **rnn_bias dense_tac2_2_bias** [128]
- static const **DenseLayer dense_tac2_2**
- static const **rnn_weight dense_tac2_3_weights** [512]
- static const **rnn_bias dense_tac2_3_bias** [32]
- static const **DenseLayer dense_tac2_3**
- static const **rnn_weight dense_map_2_weights** [512]
- static const **rnn_bias dense_map_2_bias** [16]

- static const **DenseLayer dense_map_2**
- static const **rnn_weight dense_filt_b_weights** [33280]
- static const **rnn_bias dense_filt_b_bias** [260]
- static const **DenseLayer dense_filt_b**
- static const float **scaler_b_weights** [1]
- static const float **scaler_b_bias** [1]
- static const **ScalerLayer scaler_b**
- static const **rnn_weight dense_filt_t_weights** [16640]
- static const **rnn_bias dense_filt_t_bias** [130]
- static const **DenseLayer dense_filt_t**
- static const float **scaler_t_weights** [1]
- static const float **scaler_t_bias** [1]
- static const **ScalerLayer scaler_t**
- const struct **RNNModel rnnoise_model_orig**

6.229.1 Variable Documentation

6.229.1.1 scaler_weights `const float scaler_weights[1] [static]`

6.229.1.2 scaler_bias `const float scaler_bias[1] [static]`

6.229.1.3 scaler `const` **`ScalerLayer`** `scaler [static]`

6.229.1.4 dense_projection_weights `const` **`rnn_weight`** `dense_projection_weights[66560] [static]`

6.229.1.5 dense_projection_bias `const` **`rnn_bias`** `dense_projection_bias[128] [static]`

6.229.1.6 dense_projection const **DenseLayer** dense_projection [static]

6.229.1.7 dense_map_1_weights const **rnn_weight** dense_map_1_weights[512] [static]

6.229.1.8 dense_map_1_bias const **rnn_bias** dense_map_1_bias[32] [static]

6.229.1.9 dense_map_1 const **DenseLayer** dense_map_1 [static]

6.229.1.10 dconv_5_weights const **rnn_weight** dconv_5_weights[160] [static]

6.229.1.11 dconv_5_bias const **rnn_bias** dconv_5_bias[32] [static]

6.229.1.12 dconv_5 const **DConvLayer** dconv_5 [static]

6.229.1.13 dense_dconv_5_weights const **rnn_weight** dense_dconv_5_weights[1024] [static]

6.229.1.14 dense_dconv_5_bias const **rnn_bias** dense_dconv_5_bias[32] [static]

6.229.1.15 dense_dconv_5

```
const DenseLayer dense_dconv_5 [static]
```

6.229.1.16 dconv_3_weights

```
const rnn_weight dconv_3_weights[96] [static]
```

6.229.1.17 dconv_3_bias

```
const rnn_bias dconv_3_bias[32] [static]
```

6.229.1.18 dconv_3

```
const DConvLayer dconv_3 [static]
```

6.229.1.19 dense_dconv_3_weights

```
const rnn_weight dense_dconv_3_weights[1024] [static]
```

6.229.1.20 dense_dconv_3_bias

```
const rnn_bias dense_dconv_3_bias[32] [static]
```

6.229.1.21 dense_dconv_3

```
const DenseLayer dense_dconv_3 [static]
```

6.229.1.22 dconv1x1_skip_1_weights

```
const rnn_weight dconv1x1_skip_1_weights[32] [static]
```

6.229.1.23 dconv1x1_skip_1_bias

```
const rnn_bias dconv1x1_skip_1_bias[32] [static]
```

6.229.1.24 dconv1x1_skip_1 const **DConvLayer1x1** dconv1x1_skip_1 [static]

6.229.1.25 dense_tac1_1_weights const **rnn_weight** dense_tac1_1_weights[512] [static]

6.229.1.26 dense_tac1_1_bias const **rnn_bias** dense_tac1_1_bias[16] [static]

6.229.1.27 dense_tac1_1 const **DenseLayer** dense_tac1_1 [static]

6.229.1.28 dense_tac1_2_weights const **rnn_weight** dense_tac1_2_weights[16384] [static]

6.229.1.29 dense_tac1_2_bias const **rnn_bias** dense_tac1_2_bias[128] [static]

6.229.1.30 dense_tac1_2 const **DenseLayer** dense_tac1_2 [static]

6.229.1.31 dense_tac1_3_weights const **rnn_weight** dense_tac1_3_weights[512] [static]

6.229.1.32 dense_tac1_3_bias const **rnn_bias** dense_tac1_3_bias[32] [static]

6.229.1.33 dense_tac1_3 const DenseLayer dense_tac1_3 [static]

6.229.1.34 gru_2_1_weights const rnn_weight gru_2_1_weights[3072] [static]

6.229.1.35 gru_2_1_recurrent_weights const rnn_weight gru_2_1_recurrent_weights[3072] [static]

6.229.1.36 gru_2_1_bias const rnn_bias gru_2_1_bias[192] [static]

6.229.1.37 gru_2_1 const GRULayer gru_2_1 [static]

6.229.1.38 gru_2_2_weights const rnn_weight gru_2_2_weights[3072] [static]

6.229.1.39 gru_2_2_recurrent_weights const rnn_weight gru_2_2_recurrent_weights[3072] [static]

6.229.1.40 gru_2_2_bias const rnn_bias gru_2_2_bias[192] [static]

6.229.1.41 gru_2_2 const GRULayer gru_2_2 [static]

6.229.1.42 dconv1x1_skip_2_weights const **rnn_weight** dconv1x1_skip_2_weights[32] [static]

6.229.1.43 dconv1x1_skip_2_bias const **rnn_bias** dconv1x1_skip_2_bias[32] [static]

6.229.1.44 dconv1x1_skip_2 const **DConvLayer1x1** dconv1x1_skip_2 [static]

6.229.1.45 dense_tac2_1_weights const **rnn_weight** dense_tac2_1_weights[512] [static]

6.229.1.46 dense_tac2_1_bias const **rnn_bias** dense_tac2_1_bias[16] [static]

6.229.1.47 dense_tac2_1 const **DenseLayer** dense_tac2_1 [static]

6.229.1.48 dense_tac2_2_weights const **rnn_weight** dense_tac2_2_weights[16384] [static]

6.229.1.49 dense_tac2_2_bias const **rnn_bias** dense_tac2_2_bias[128] [static]

6.229.1.50 dense_tac2_2 const **DenseLayer** dense_tac2_2 [static]

6.229.1.51 dense_tac2_3_weights

const **rnn_weight** dense_tac2_3_weights[512] [static]

6.229.1.52 dense_tac2_3_bias

const **rnn_bias** dense_tac2_3_bias[32] [static]

6.229.1.53 dense_tac2_3

const **DenseLayer** dense_tac2_3 [static]

6.229.1.54 dense_map_2_weights

const **rnn_weight** dense_map_2_weights[512] [static]

6.229.1.55 dense_map_2_bias

const **rnn_bias** dense_map_2_bias[16] [static]

6.229.1.56 dense_map_2

const **DenseLayer** dense_map_2 [static]

6.229.1.57 dense_filt_b_weights

const **rnn_weight** dense_filt_b_weights[33280] [static]

6.229.1.58 dense_filt_b_bias

const **rnn_bias** dense_filt_b_bias[260] [static]

6.229.1.59 dense_filt_b

const **DenseLayer** dense_filt_b [static]

6.229.1.60 scaler_b_weights const float scaler_b_weights[1] [static]

6.229.1.61 scaler_b_bias const float scaler_b_bias[1] [static]

6.229.1.62 scaler_b const **ScalerLayer** scaler_b [static]

6.229.1.63 dense_filt_t_weights const **rnn_weight** dense_filt_t_weights[16640] [static]

6.229.1.64 dense_filt_t_bias const **rnn_bias** dense_filt_t_bias[130] [static]

6.229.1.65 dense_filt_t const **DenseLayer** dense_filt_t [static]

6.229.1.66 scaler_t_weights const float scaler_t_weights[1] [static]

6.229.1.67 scaler_t_bias const float scaler_t_bias[1] [static]

6.229.1.68 scaler_t const **ScalerLayer** scaler_t [static]

6.229.1.69 rnnoise_model_orig const struct **RNNModel** rnnoise_model_orig

6.230 rnn_data.c File Reference

Variables

- static const float **scaler_weights** [1]
- static const float **scaler_bias** [1]
- static const **ScalerLayer scaler**
- static const **rnn_weight dense_projection_weights** [33280]
- static const **rnn_bias dense_projection_bias** [128]
- static const **DenseLayer dense_projection**
- static const **rnn_weight dense_map_1_weights** [512]
- static const **rnn_bias dense_map_1_bias** [32]
- static const **DenseLayer dense_map_1**
- static const **rnn_weight dconv_5_weights** [160]
- static const **rnn_bias dconv_5_bias** [32]
- static const **DConvLayer dconv_5**
- static const **rnn_weight dense_dconv_5_weights** [1024]
- static const **rnn_bias dense_dconv_5_bias** [32]
- static const **DenseLayer dense_dconv_5**
- static const **rnn_weight dconv_3_weights** [96]
- static const **rnn_bias dconv_3_bias** [32]
- static const **DConvLayer dconv_3**
- static const **rnn_weight dense_dconv_3_weights** [1024]
- static const **rnn_bias dense_dconv_3_bias** [32]
- static const **DenseLayer dense_dconv_3**
- static const **rnn_weight dconv1x1_skip_1_weights** [32]
- static const **rnn_bias dconv1x1_skip_1_bias** [32]
- static const **DConvLayer1x1 dconv1x1_skip_1**
- static const **rnn_weight dense_tac1_1_weights** [512]
- static const **rnn_bias dense_tac1_1_bias** [16]
- static const **DenseLayer dense_tac1_1**
- static const **rnn_weight dense_tac1_2_weights** [16384]
- static const **rnn_bias dense_tac1_2_bias** [128]
- static const **DenseLayer dense_tac1_2**
- static const **rnn_weight dense_tac1_3_weights** [512]
- static const **rnn_bias dense_tac1_3_bias** [32]
- static const **DenseLayer dense_tac1_3**
- static const **rnn_weight gru_2_1_weights** [3072]
- static const **rnn_weight gru_2_1_recurrent_weights** [3072]
- static const **rnn_bias gru_2_1_bias** [192]
- static const **GRULayer gru_2_1**
- static const **rnn_weight gru_2_2_weights** [3072]
- static const **rnn_weight gru_2_2_recurrent_weights** [3072]
- static const **rnn_bias gru_2_2_bias** [192]
- static const **GRULayer gru_2_2**
- static const **rnn_weight dconv1x1_skip_2_weights** [32]

- static const **rnn_bias dconv1x1_skip_2_bias** [32]
- static const **DConvLayer1x1 dconv1x1_skip_2**
- static const **rnn_weight dense_tac2_1_weights** [512]
- static const **rnn_bias dense_tac2_1_bias** [16]
- static const **DenseLayer dense_tac2_1**
- static const **rnn_weight dense_tac2_2_weights** [16384]
- static const **rnn_bias dense_tac2_2_bias** [128]
- static const **DenseLayer dense_tac2_2**
- static const **rnn_weight dense_tac2_3_weights** [512]
- static const **rnn_bias dense_tac2_3_bias** [32]
- static const **DenseLayer dense_tac2_3**
- static const **rnn_weight dense_map_2_weights** [512]
- static const **rnn_bias dense_map_2_bias** [16]
- static const **DenseLayer dense_map_2**
- static const **rnn_weight dense_filt_b_weights** [33280]
- static const **rnn_bias dense_filt_b_bias** [260]
- static const **DenseLayer dense_filt_b**
- static const float **scaler_b_weights** [1]
- static const float **scaler_b_bias** [1]
- static const **ScalerLayer scaler_b**
- static const **rnn_weight dense_filt_t_weights** [16640]
- static const **rnn_bias dense_filt_t_bias** [130]
- static const **DenseLayer dense_filt_t**
- static const float **scaler_t_weights** [1]
- static const float **scaler_t_bias** [1]
- static const **ScalerLayer scaler_t**
- const struct **RNNModel rnnoise_model_orig**

6.230.1 Variable Documentation

6.230.1.1 scaler_weights `const float scaler_weights[1] [static]`

6.230.1.2 scaler_bias `const float scaler_bias[1] [static]`

6.230.1.3 scaler `const` **`ScalerLayer`** `scaler [static]`

6.230.1.4 dense_projection_weights

```
const rnn_weight dense_projection_weights[33280] [static]
```

6.230.1.5 dense_projection_bias

```
const rnn_bias dense_projection_bias[128] [static]
```

6.230.1.6 dense_projection

```
const DenseLayer dense_projection [static]
```

6.230.1.7 dense_map_1_weights

```
const rnn_weight dense_map_1_weights[512] [static]
```

6.230.1.8 dense_map_1_bias

```
const rnn_bias dense_map_1_bias[32] [static]
```

6.230.1.9 dense_map_1

```
const DenseLayer dense_map_1 [static]
```

6.230.1.10 dconv_5_weights

```
const rnn_weight dconv_5_weights[160] [static]
```

6.230.1.11 dconv_5_bias

```
const rnn_bias dconv_5_bias[32] [static]
```

6.230.1.12 dconv_5

```
const DConvLayer dconv_5 [static]
```

6.230.1.13 dense_dconv_5_weights const **rnn_weight** dense_dconv_5_weights[1024] [static]

6.230.1.14 dense_dconv_5_bias const **rnn_bias** dense_dconv_5_bias[32] [static]

6.230.1.15 dense_dconv_5 const **DenseLayer** dense_dconv_5 [static]

6.230.1.16 dconv_3_weights const **rnn_weight** dconv_3_weights[96] [static]

6.230.1.17 dconv_3_bias const **rnn_bias** dconv_3_bias[32] [static]

6.230.1.18 dconv_3 const **DConvLayer** dconv_3 [static]

6.230.1.19 dense_dconv_3_weights const **rnn_weight** dense_dconv_3_weights[1024] [static]

6.230.1.20 dense_dconv_3_bias const **rnn_bias** dense_dconv_3_bias[32] [static]

6.230.1.21 dense_dconv_3 const **DenseLayer** dense_dconv_3 [static]

6.230.1.22 dconv1x1_skip_1_weights

```
const rnn_weight dconv1x1_skip_1_weights[32] [static]
```

6.230.1.23 dconv1x1_skip_1_bias

```
const rnn_bias dconv1x1_skip_1_bias[32] [static]
```

6.230.1.24 dconv1x1_skip_1

```
const DConvLayer1x1 dconv1x1_skip_1 [static]
```

6.230.1.25 dense_tac1_1_weights

```
const rnn_weight dense_tac1_1_weights[512] [static]
```

6.230.1.26 dense_tac1_1_bias

```
const rnn_bias dense_tac1_1_bias[16] [static]
```

6.230.1.27 dense_tac1_1

```
const DenseLayer dense_tac1_1 [static]
```

6.230.1.28 dense_tac1_2_weights

```
const rnn_weight dense_tac1_2_weights[16384] [static]
```

6.230.1.29 dense_tac1_2_bias

```
const rnn_bias dense_tac1_2_bias[128] [static]
```

6.230.1.30 dense_tac1_2

```
const DenseLayer dense_tac1_2 [static]
```

6.230.1.31 dense_tac1_3_weights const rnn_weight dense_tac1_3_weights[512] [static]

6.230.1.32 dense_tac1_3_bias const rnn_bias dense_tac1_3_bias[32] [static]

6.230.1.33 dense_tac1_3 const DenseLayer dense_tac1_3 [static]

6.230.1.34 gru_2_1_weights const rnn_weight gru_2_1_weights[3072] [static]

6.230.1.35 gru_2_1_recurrent_weights const rnn_weight gru_2_1_recurrent_weights[3072] [static]

6.230.1.36 gru_2_1_bias const rnn_bias gru_2_1_bias[192] [static]

6.230.1.37 gru_2_1 const GRULayer gru_2_1 [static]

6.230.1.38 gru_2_2_weights const rnn_weight gru_2_2_weights[3072] [static]

6.230.1.39 gru_2_2_recurrent_weights const rnn_weight gru_2_2_recurrent_weights[3072] [static]

6.230.1.40 gru_2_2_bias

```
const rnn_bias gru_2_2_bias[192] [static]
```

6.230.1.41 gru_2_2

```
const GRULayer gru_2_2 [static]
```

6.230.1.42 dconv1x1_skip_2_weights

```
const rnn_weight dconv1x1_skip_2_weights[32] [static]
```

6.230.1.43 dconv1x1_skip_2_bias

```
const rnn_bias dconv1x1_skip_2_bias[32] [static]
```

6.230.1.44 dconv1x1_skip_2

```
const DConvLayer1x1 dconv1x1_skip_2 [static]
```

6.230.1.45 dense_tac2_1_weights

```
const rnn_weight dense_tac2_1_weights[512] [static]
```

6.230.1.46 dense_tac2_1_bias

```
const rnn_bias dense_tac2_1_bias[16] [static]
```

6.230.1.47 dense_tac2_1

```
const DenseLayer dense_tac2_1 [static]
```

6.230.1.48 dense_tac2_2_weights

```
const rnn_weight dense_tac2_2_weights[16384] [static]
```

6.230.1.49 dense_tac2_2_bias const rnn_bias dense_tac2_2_bias[128] [static]

6.230.1.50 dense_tac2_2 const DenseLayer dense_tac2_2 [static]

6.230.1.51 dense_tac2_3_weights const rnn_weight dense_tac2_3_weights[512] [static]

6.230.1.52 dense_tac2_3_bias const rnn_bias dense_tac2_3_bias[32] [static]

6.230.1.53 dense_tac2_3 const DenseLayer dense_tac2_3 [static]

6.230.1.54 dense_map_2_weights const rnn_weight dense_map_2_weights[512] [static]

6.230.1.55 dense_map_2_bias const rnn_bias dense_map_2_bias[16] [static]

6.230.1.56 dense_map_2 const DenseLayer dense_map_2 [static]

6.230.1.57 dense_filt_b_weights const rnn_weight dense_filt_b_weights[33280] [static]

6.230.1.58 dense_filt_b_bias

```
const rnn_bias dense_filt_b_bias[260] [static]
```

6.230.1.59 dense_filt_b

```
const DenseLayer dense_filt_b [static]
```

6.230.1.60 scaler_b_weights

```
const float scaler_b_weights[1] [static]
```

6.230.1.61 scaler_b_bias

```
const float scaler_b_bias[1] [static]
```

6.230.1.62 scaler_b

```
const ScalerLayer scaler_b [static]
```

6.230.1.63 dense_filt_t_weights

```
const rnn_weight dense_filt_t_weights[16640] [static]
```

6.230.1.64 dense_filt_t_bias

```
const rnn_bias dense_filt_t_bias[130] [static]
```

6.230.1.65 dense_filt_t

```
const DenseLayer dense_filt_t [static]
```

6.230.1.66 scaler_t_weights

```
const float scaler_t_weights[1] [static]
```

6.230.1.67 scaler_t_bias const float scaler_t_bias[1] [static]

6.230.1.68 scaler_t const ScalerLayer scaler_t [static]

6.230.1.69 rnnoise_model_orig const struct RNNModel rnnoise_model_orig

6.231 rnn_data.h File Reference

Classes

- struct **RNNModel**
- struct **RNNState**

6.232 rnn_data.h File Reference

Classes

- struct **RNNModel**
- struct **RNNState**

6.233 rnnoise.h File Reference

Macros

- #define **RNNOISE_EXPORT**

Typedefs

- typedef struct **DenoiseState** **DenoiseState**
- typedef struct **RNNModel** **RNNModel**

Functions

- **RNNOISE_EXPORT** int **rnnoise_get_size** ()
- **RNNOISE_EXPORT** int **rnnoise_init** (**DenoiseState** ∗st)
- **RNNOISE_EXPORT** **DenoiseState** ∗ **rnnoise_create** ()
- **RNNOISE_EXPORT** void **rnnoise_destroy** (**DenoiseState** ∗st)
- **RNNOISE_EXPORT** void **rnnoise_process_frame** (**DenoiseState** ∗st, float ∗real_↩ output, float ∗imag_output, const float ∗real_input, const float ∗imag_input)
- **RNNOISE_EXPORT** **RNNModel** ∗ **rnnoise_model_from_file** (FILE ∗f)
- **RNNOISE_EXPORT** void **rnnoise_model_free** (**RNNModel** ∗model)

6.233.1 Macro Definition Documentation

6.233.1.1 RNNOISE_EXPORT `#define RNNOISE_EXPORT`

6.233.2 Typedef Documentation

6.233.2.1 DenoiseState `typedef struct DenoiseState DenoiseState`

6.233.2.2 RNNModel `typedef struct RNNModel RNNModel`

6.233.3 Function Documentation

6.233.3.1 rnnoise_get_size() `RNNOISE_EXPORT int rnnoise_get_size ( )`

6.233.3.2 rnnoise_init()

```
RNNOISE_EXPORT int rnnoise_init (
    DenoiseState * st )
```

6.233.3.3 rnnoise_create()

```
RNNOISE_EXPORT DenoiseState* rnnoise_create ( )
```

6.233.3.4 rnnoise_destroy()

```
RNNOISE_EXPORT void rnnoise_destroy (
    DenoiseState * st )
```

6.233.3.5 rnnoise_process_frame()

```
RNNOISE_EXPORT void rnnoise_process_frame (
    DenoiseState * st,
    float * real_output,
    float * imag_output,
    const float * real_input,
    const float * imag_input )
```

6.233.3.6 rnnoise_model_from_file()

```
RNNOISE_EXPORT RNNModel* rnnoise_model_from↩
_file (
    FILE * f )
```

6.233.3.7 rnnoise_model_free()

```
RNNOISE_EXPORT void rnnoise_model_free (
    RNNModel * model )
```

6.234 rnnoise.h File Reference

Macros

- #define **RNNOISE_EXPORT**

Typedefs

- typedef struct **DenoiseState** **DenoiseState**
- typedef struct **RNNModel** **RNNModel**

Functions

- **RNNOISE_EXPORT** int **rnnoise_get_size** ()
- **RNNOISE_EXPORT** int **rnnoise_init** (**DenoiseState** ∗st)
- **RNNOISE_EXPORT** **DenoiseState** ∗ **rnnoise_create** ()
- **RNNOISE_EXPORT** void **rnnoise_destroy** (**DenoiseState** ∗st)
- **RNNOISE_EXPORT** void **rnnoise_process_frame** (**DenoiseState** ∗st, float ∗real_↩ output, float ∗imag_output, const float ∗real_input, const float ∗imag_input)
- **RNNOISE_EXPORT** **RNNModel** ∗ **rnnoise_model_from_file** (FILE ∗f)
- **RNNOISE_EXPORT** void **rnnoise_model_free** (**RNNModel** ∗model)

6.234.1 Macro Definition Documentation

6.234.1.1 RNNOISE_EXPORT `#define RNNOISE_EXPORT`

6.234.2 Typedef Documentation

6.234.2.1 DenoiseState `typedef struct` **`DenoiseState`** **`DenoiseState`**

6.234.2.2 RNNModel `typedef struct` **`RNNModel`** **`RNNModel`**

6.234.3 Function Documentation

6.234.3.1 rnnoise_get_size() RNNOISE_EXPORT int rnnoise_get_size ( )

6.234.3.2 rnnoise_init() RNNOISE_EXPORT int rnnoise_init (DenoiseState * st)

6.234.3.3 rnnoise_create() RNNOISE_EXPORT DenoiseState* rnnoise_create ( )

6.234.3.4 rnnoise_destroy() RNNOISE_EXPORT void rnnoise_destroy (DenoiseState * st)

6.234.3.5 rnnoise_process_frame() RNNOISE_EXPORT void rnnoise_process_frame (DenoiseState * st, float * real_output, float * imag_output, const float * real_input, const float * imag_input)

6.234.3.6 rnnoise_model_from_file() RNNOISE_EXPORT RNNModel* rnnoise_model_from↩ _file (FILE * f)

6.234.3.7 rnnoise_model_free() RNNOISE_EXPORT void rnnoise_model_free (RNNModel * model)

6.235 rohBeam.cpp File Reference

Namespaces

- **rohBeam**

Variables

- auto **rohBeam::scalarify** =[ ](auto t){return t(0);}

6.236 rohBeam.hh File Reference

Classes

- struct **rohBeam::configOptions**
- class **rohBeam::rohConfig**
- class **rohBeam::rohBeam**

Namespaces

- **rohBeam**

Macros

- #define **NDEBUG**

Functions

- double **rohBeam::j0** (double x)
  Cylindrical bessel function of the first kind of order 0.

Variables

- constexpr float **rohBeam::CONST_C** = 343.0115f
- constexpr int **rohBeam::refL** = 0
- constexpr int **rohBeam::refR** = 3

6.236.1 Macro Definition Documentation

6.236.1.1 NDEBUG `#define NDEBUG`

6.237 route.cpp File Reference

Classes

- class **route::process_t**
- class **route::interface_t**

Namespaces

- **route**

6.238 save_spec.cpp File Reference

Classes

- class **save_spec_t**

6.239 save_wave.cpp File Reference

Classes

- class **save_wave_t**

6.240 set_rms.cpp File Reference

6.241 set_rms.hh File Reference

Classes

- class **Set_rms_cfg**
  Runtime configuration class for setting the channels of an output signal (e.g.
- class **Set_rms**
  Plugin interface class for seting the channels of an output signal(e.g.

6.242 shadowfilter_begin.cpp File Reference

Classes

- class **shadowfilter_begin::cfg_t**
- class **shadowfilter_begin::shadowfilter_begin_t**

Namespaces

- **shadowfilter_begin**

6.243 shadowfilter_end.cpp File Reference

Classes

- class **shadowfilter_end::cfg_t**
- class **shadowfilter_end::shadowfilter_end_t**

Namespaces

- **shadowfilter_end**

6.244 sine.cpp File Reference

Classes

- struct **sine_cfg_t**
  Runtime configuration of the sine plugin.
- class **sine_t**
  Interface class of plugin `sine`*, a sinusoid generator plugin.*

6.245 smooth_cepstrum.cpp File Reference

Macros

- #define **INSERT_VAR**(var) insert_item(#var, &var)
- #define **PATCH_VAR**(var)
- #define **INSERT_PATCH**(var) **INSERT_VAR**(var); **PATCH_VAR**(var)

6.245.1 Macro Definition Documentation

6.245.1.1 INSERT_VAR `#define INSERT_VAR( var ) insert_item(#var, &var)`

6.245.1.2 PATCH_VAR `#define PATCH_VAR( var )`

6.245.1.3 INSERT_PATCH `#define INSERT_PATCH( var ) INSERT_VAR(var); PATCH_VAR(var)`

6.246 smooth_cepstrum.hh File Reference

Classes

- class **smooth_cepstrum::smooth_params**
- class **smooth_cepstrum::smooth_cepstrum_t**
- class **smooth_cepstrum::smooth_cepstrum_if_t**

Namespaces

- **smooth_cepstrum**

6.247 smoothgains_bridge.cpp File Reference

Classes

- class **smoothgains_bridge::smoothspec_wrap_t**
- class **smoothgains_bridge::overlapadd_if_t**

Namespaces

- **smoothgains_bridge**

6.248 softclip.cpp File Reference

Classes

- class **cfg_t**
- class **softclip_t**

6.249 spec2wave.cpp File Reference

Classes

- class **hanning_ramps_t**
  Class for storing two hanning ramps: a rising ramp a and a falling ramp b.
- class **spec2wave_t**
  Runtime config class for plugin spec2wave, the counterpart of wave2spec.
- class **spec2wave_if_t**

Functions

- unsigned int **max** (unsigned int a, unsigned int b)
- unsigned int **min** (unsigned int a, unsigned int b)

6.249.1 Function Documentation

6.249.1.1 max()

```
unsigned int max (
    unsigned int a,
    unsigned int b ) [inline]
```

6.249.1.2 min()

```
unsigned int min (
    unsigned int a,
    unsigned int b ) [inline]
```

6.250 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)
- 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}

6.250.1 Macro Definition Documentation

6.250.1.1 NUM_ENTR_MHAORIG `#define NUM_ENTR_MHAORIG 76`

6.250.1.2 NUM_ENTR_LTASS `#define NUM_ENTR_LTASS 25`

6.250.1.3 NUM_ENTR_OLNOISE `#define NUM_ENTR_OLNOISE 49`

6.250.2 Function Documentation

6.250.2.1 fhz2bandno() `float fhz2bandno (`
`float` *x*)

6.250.2.2 erb_hz_f_hz() `float erb_hz_f_hz (`
`float` *f_hz*)

6.250.2.3 hz2hz() float hz2hz (

float *x*)

6.250.2.4 bandw_correction() float bandw_correction (

float *f,*

float *ldb*)

6.250.3 Variable Documentation

6.250.3.1 vMHAOrigSpec 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}

6.250.3.2 vMHAOrigFreq 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}

6.250.3.3 vLTASS_freq 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}

6.250.3.4 vLTASS_combined_lev

```
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}
```

6.250.3.5 vLTASS_female_lev

```
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}
```

6.250.3.6 vLTASS_male_lev

```
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}
```

6.250.3.7 vOlnoiseFreq

```
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}
```

6.250.3.8 vOlnoiseLev

```
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}
```

6.251 speechnoise.h File Reference

Classes

- class **speechnoise_t**

6.252 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. 1331) 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 **native_thread_platform_type** posix_threads_t

Enumerations

- enum { **MHAPlugin_Split::INVALID_THREAD_PRIORITY** = 999999999 }
  Invalid thread priority.

6.252.1 Detailed Description

Source code for the split plugin. The split plugin splits the audio signal by channel. The splitted paths execute in parallel.

6.252.2 Macro Definition Documentation

6.252.2.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN `#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.

6.252.2.2 posixthreads `#define posixthreads 1`

6.252.2.3 native_thread_platform_type `#define native_thread_platform_type posix_↩ threads_t`

6.253 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)

6.253.1 Macro Definition Documentation

6.253.1.1 PATCH_VAR `#define PATCH_VAR( var ) patchbay.connect(&var.valuechanged, this, & steerbf::update_cfg)`

6.253.1.2 INSERT_PATCH `#define INSERT_PATCH( var ) insert_member(var); PATCH_VAR(var)`

6.254 steerbf.h File Reference

Classes

- class **parser_int_dyn**
- class **steerbf_config**
- class **steerbf**

6.255 testalsadevice.c File Reference

Functions

- int **main** (int argc, char ∗∗argv)

6.255.1 Function Documentation

6.255.1.1 main() int main (int *argc*, char ** *argv*)

6.256 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**

6.257 transducers.cpp File Reference

Classes

- class **softclipper_variables_t**
  Parser aggregate of all configuration variables for the output soft clipper.
- class **softclipper_t**
  Soft clipper signal processing implementation.
- 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)

6.257.1 Typedef Documentation

6.257.1.1 rmslevelmeter typedef MHAPlugin::config_t< MHASignal::async_rmslevel_t> rmslevelmeter

6.257.1.2 rtcalibrator typedef MHAPlugin::plugin_t< calibrator_runtime_layer_t> rtcalibrator

6.257.2 Function Documentation

6.257.2.1 kw_index2type()

```
speechnoise_t::noise_type_t kw_index2type (
    unsigned int idx )
```

6.257.2.2 vint_0123n1()

```
std::vector<int> vint_0123n1 (
    unsigned int n )
```

6.258 trigger2lsl.cpp File Reference

6.259 trigger2lsl.hh File Reference

Classes

- class **trigger2lsl::trigger2lsl_rt_t**
  *real-time configuration class for **trigger2lsl*** (p. 166) *plugin*
- class **trigger2lsl::trigger2lsl_if_t**
  *Plugin interface class of plugin **trigger2lsl*** (p. 166).

Namespaces

- **trigger2lsl**
  *namespace for **trigger2lsl*** (p. 166) *plugin*

6.260 upsample.cpp File Reference

Classes

- class **us_t**

6.261 wave2lsl.cpp File Reference

Classes

- class **wave2lsl::cfg_t**
  *Runtime configuration class of the **wave2lsl*** (p. 166) *plugin.*
- class **wave2lsl::wave2lsl_t**
  *Plugin class of **wave2lsl*** (p. 166).

Namespaces

- **wave2lsl**
  *All types for the **wave2lsl** (p. 166) plugins live in this namespace.*

6.262 wave2spec.cpp File Reference

6.263 wave2spec.hh File Reference

Classes

- class **wave2spec_t**
  Runtime configuration class for plugin wave2spec.
- class **wave2spec_if_t**
  *Plugin wave2spec interface class, uses **wave2spec_t** (p. 1694) as runtime configuration.*

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.263.1 Macro Definition Documentation

6.263.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN `#define MHAPLUGIN_OVERLOAD_↩ OUTDOMAIN`

6.264 wavrec.cpp File Reference

Classes

- class **wavwriter_t**
- class **wavrec_t**

6.265 windnoise.cpp File Reference

Namespaces

- **windnoise**
  namespace for plugin windnoise which detects and cancels wind noise

Macros

- #define **register_configuration_variable**(v)

6.265.1 Macro Definition Documentation

6.265.1.1 register_configuration_variable `#define register_configuration_variable( v )`

6.266 windnoise.hh File Reference

Classes

- class **windnoise::cfg_t**
  Runtime config class for windnoise plugin.
- class **windnoise::if_t**
  interface class for windnoise plugin

Namespaces

- **windnoise**
  namespace for plugin windnoise which detects and cancels wind noise

6.267 windowselector.cpp File Reference

6.268 windowselector.h File Reference

Classes

- class **windowselector_t**
  A combination of mha parser variables to describe an overalapadd analysis window.

Index

R