

The Open Master Hearing Aid (openMHA)

4.5.6

Plugin Developers' Manual



HörTech

Kompetenzzentrum für
Hörgeräte-Systemtechnik

LICENSE AGREEMENT

This file is part of the HörTech Open Master Hearing Aid (openMHA)

Copyright © 2005 2006 2007 2008 2009 2010 2012 2013 2014 2015 2016 HörTech gGmbH.

Copyright © 2017 2018 HörTech gGmbH.

openMHA is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, version 3 of the License.

openMHA is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License, version 3 for more details.

You should have received a copy of the GNU Affero General Public License, version 3 along with openMHA. If not, see <<http://www.gnu.org/licenses/>>.

Contents

| | | |
|----------|---|-----------|
| 1 | Overview | 1 |
| 1.1 | Structure | 1 |
| 1.2 | Platform Services and Conventions | 2 |
| 2 | Todo List | 4 |
| 3 | Module Documentation | 4 |
| 3.1 | Concept of Variables and Data Exchange in the openMHA | 4 |
| 3.2 | The openMHA Plugins (programming interface) | 6 |
| 3.3 | Writing openMHA Plugins. A step-by-step tutorial | 10 |
| 3.4 | The MHA Framework interface | 26 |
| 3.5 | Communication between algorithms | 27 |
| 3.6 | Error handling in the openMHA | 31 |
| 3.7 | The openMHA configuration language | 33 |
| 3.8 | The openMHA Toolbox library | 34 |
| 3.9 | Vector and matrix processing toolbox | 36 |
| 3.10 | Complex arithmetics in the openMHA | 60 |
| 3.11 | Fast Fourier Transform functions | 70 |
| 4 | Namespace Documentation | 77 |
| 4.1 | acmon Namespace Reference | 77 |
| 4.2 | acsave Namespace Reference | 77 |
| 4.3 | ADM Namespace Reference | 77 |
| 4.4 | AuditoryProfile Namespace Reference | 78 |
| 4.5 | coherence Namespace Reference | 79 |
| 4.6 | dc Namespace Reference | 79 |
| 4.7 | dc_simple Namespace Reference | 80 |
| 4.8 | delay Namespace Reference | 81 |
| 4.9 | delaysum Namespace Reference | 81 |
| 4.10 | DynComp Namespace Reference | 82 |
| 4.11 | fader_wave Namespace Reference | 83 |
| 4.12 | fftfilterbank Namespace Reference | 83 |
| 4.13 | gain Namespace Reference | 84 |
| 4.14 | matrixmixer Namespace Reference | 84 |
| 4.15 | MHA_AC Namespace Reference | 84 |
| 4.16 | mha_error_helpers Namespace Reference | 85 |
| 4.17 | MHA_TCP Namespace Reference | 86 |
| 4.18 | mhachain Namespace Reference | 88 |
| 4.19 | MHAEvents Namespace Reference | 88 |
| 4.20 | MHAFilter Namespace Reference | 89 |
| 4.21 | MHAIOJack Namespace Reference | 93 |
| 4.22 | MHAIOPortAudio Namespace Reference | 93 |
| 4.23 | MHAJack Namespace Reference | 94 |
| 4.24 | MHAKernel Namespace Reference | 96 |
| 4.25 | MHAMultiSrc Namespace Reference | 96 |
| 4.26 | MHAOvIFilter Namespace Reference | 97 |
| 4.27 | MHAOvIFilter::barkscale Namespace Reference | 98 |
| 4.28 | MHAOvIFilter::FreqScaleFun Namespace Reference | 98 |
| 4.29 | MHAOvIFilter::ShapeFun Namespace Reference | 100 |
| 4.30 | MHAParser Namespace Reference | 103 |

| | | |
|----------|--|------------|
| 4.31 | MHAParser::StrCnv Namespace Reference | 107 |
| 4.32 | MHAPLugin Namespace Reference | 112 |
| 4.33 | MHAPLugin_Resampling Namespace Reference | 112 |
| 4.34 | MHAPLugin_Split Namespace Reference | 112 |
| 4.35 | MHASignal Namespace Reference | 113 |
| 4.36 | MHASndFile Namespace Reference | 120 |
| 4.37 | MHATableLookup Namespace Reference | 120 |
| 4.38 | MHAWindow Namespace Reference | 121 |
| 4.39 | multibandcompressor Namespace Reference | 122 |
| 4.40 | noisePowProposedScale Namespace Reference | 122 |
| 4.41 | overlapadd Namespace Reference | 123 |
| 4.42 | PluginLoader Namespace Reference | 123 |
| 4.43 | route Namespace Reference | 124 |
| 4.44 | shadowfilter_begin Namespace Reference | 124 |
| 4.45 | shadowfilter_end Namespace Reference | 124 |
| 4.46 | smoothgains_bridge Namespace Reference | 124 |
| 5 | Class Documentation | 124 |
| 5.1 | ac2wave_if_t Class Reference | 125 |
| 5.2 | ac2wave_t Class Reference | 127 |
| 5.3 | acConcat_wave Class Reference | 128 |
| 5.4 | acConcat_wave_config Class Reference | 131 |
| 5.5 | acmon::ac_monitor_t Class Reference | 132 |
| 5.6 | acmon::acmon_t Class Reference | 135 |
| 5.7 | acPooling_wave Class Reference | 138 |
| 5.8 | acPooling_wave_config Class Reference | 141 |
| 5.9 | acsave::acsave_t Class Reference | 143 |
| 5.10 | acsave::cfg_t Class Reference | 146 |
| 5.11 | acsave::mat4head_t Struct Reference | 147 |
| 5.12 | acsave::save_var_t Class Reference | 148 |
| 5.13 | acSteer Class Reference | 150 |
| 5.14 | acSteer_config Class Reference | 152 |
| 5.15 | acTransform_wave Class Reference | 154 |
| 5.16 | acTransform_wave_config Class Reference | 156 |
| 5.17 | ADM::ADM< F > Class Template Reference | 158 |
| 5.18 | ADM::Delay< F > Class Template Reference | 160 |
| 5.19 | ADM::Linearphase_FIR< F > Class Template Reference | 162 |
| 5.20 | adm_if_t Class Reference | 165 |
| 5.21 | adm_rtconfig_t Class Reference | 167 |
| 5.22 | algo_comm_t Struct Reference | 171 |
| 5.23 | altplugs_t Class Reference | 176 |
| 5.24 | analysepath_t Class Reference | 180 |
| 5.25 | analysispath_if_t Class Reference | 182 |
| 5.26 | AuditoryProfile::fmap_t Class Reference | 184 |
| 5.27 | AuditoryProfile::parser_t Class Reference | 185 |
| 5.28 | AuditoryProfile::parser_t::ear_t Class Reference | 186 |
| 5.29 | AuditoryProfile::parser_t::fmap_t Class Reference | 187 |
| 5.30 | AuditoryProfile::profile_t Class Reference | 189 |
| 5.31 | AuditoryProfile::profile_t::ear_t Class Reference | 190 |
| 5.32 | bbcalib_interface_t Class Reference | 191 |
| 5.33 | calibrator_runtime_layer_t Class Reference | 192 |
| 5.34 | calibrator_t Class Reference | 194 |

| | | |
|------|--|-----|
| 5.35 | calibrator_variables_t Class Reference | 196 |
| 5.36 | cfg_t Class Reference | 198 |
| 5.37 | coherence::cohflt_if_t Class Reference | 200 |
| 5.38 | coherence::cohflt_t Class Reference | 202 |
| 5.39 | coherence::vars_t Class Reference | 204 |
| 5.40 | combc_if_t Class Reference | 206 |
| 5.41 | combc_t Class Reference | 207 |
| 5.42 | comm_var_t Struct Reference | 209 |
| 5.43 | cpuload_t Class Reference | 210 |
| 5.44 | db_if_t Class Reference | 212 |
| 5.45 | db_t Class Reference | 214 |
| 5.46 | dc::dc_if_t Class Reference | 216 |
| 5.47 | dc::dc_t Class Reference | 218 |
| 5.48 | dc::dc_vars_t Class Reference | 220 |
| 5.49 | dc::dc_vars_validator_t Class Reference | 223 |
| 5.50 | dc::wb_inhib_cfg_t Class Reference | 223 |
| 5.51 | dc::wideband_inhib_vars_t Class Reference | 224 |
| 5.52 | dc_simple::dc_if_t Class Reference | 226 |
| 5.53 | dc_simple::dc_t Class Reference | 229 |
| 5.54 | dc_simple::dc_t::line_t Class Reference | 231 |
| 5.55 | dc_simple::dc_vars_t Class Reference | 232 |
| 5.56 | dc_simple::dc_vars_validator_t Class Reference | 233 |
| 5.57 | dc_simple::level_smoother_t Class Reference | 234 |
| 5.58 | delay::interface_t Class Reference | 236 |
| 5.59 | delaysum::delaysum_if_t Class Reference | 237 |
| 5.60 | delaysum::delaysum_t Class Reference | 240 |
| 5.61 | doasvm_classification Class Reference | 242 |
| 5.62 | doasvm_classification_config Class Reference | 244 |
| 5.63 | doasvm_feature_extraction Class Reference | 246 |
| 5.64 | doasvm_feature_extraction_config Class Reference | 248 |
| 5.65 | droptect_t Class Reference | 250 |
| 5.66 | ds_t Class Reference | 253 |
| 5.67 | dynamiclib_t Class Reference | 254 |
| 5.68 | DynComp::dc_afterburn_rt_t Class Reference | 255 |
| 5.69 | DynComp::dc_afterburn_t Class Reference | 257 |
| 5.70 | DynComp::dc_afterburn_vars_t Class Reference | 259 |
| 5.71 | DynComp::gaintable_t Class Reference | 261 |
| 5.72 | example1_t Class Reference | 266 |
| 5.73 | example2_t Class Reference | 268 |
| 5.74 | example3_t Class Reference | 271 |
| 5.75 | example4_t Class Reference | 275 |
| 5.76 | example5_t Class Reference | 278 |
| 5.77 | example6_t Class Reference | 279 |
| 5.78 | expression_t Class Reference | 281 |
| 5.79 | fader_if_t Class Reference | 281 |
| 5.80 | fader_wave::fader_wave_if_t Class Reference | 283 |
| 5.81 | fader_wave::level_adapt_t Class Reference | 285 |
| 5.82 | fftfilterbank::fftfb_interface_t Class Reference | 286 |
| 5.83 | fftfilterbank::fftfb_plug_t Class Reference | 290 |
| 5.84 | frequency_translator_t Class Reference | 292 |
| 5.85 | fw_t Class Reference | 294 |

| | | |
|-------|--|-----|
| 5.86 | <code>fw_vars_t</code> Class Reference | 299 |
| 5.87 | <code>gain::gain_if_t</code> Class Reference | 301 |
| 5.88 | <code>gain::scaler_t</code> Class Reference | 303 |
| 5.89 | <code>hanning_ramps_t</code> Class Reference | 303 |
| 5.90 | <code>hilbert_shifter_t</code> Class Reference | 305 |
| 5.91 | <code>identity_t</code> Class Reference | 307 |
| 5.92 | <code>iirfilter_t</code> Class Reference | 308 |
| 5.93 | <code>io_file_t</code> Class Reference | 309 |
| 5.94 | <code>io_lib_t</code> Class Reference | 313 |
| 5.95 | <code>io_parser_t</code> Class Reference | 317 |
| 5.96 | <code>io_tcp_fwcb_t</code> Class Reference | 320 |
| 5.97 | <code>io_tcp_parser_t</code> Class Reference | 323 |
| 5.98 | <code>io_tcp_sound_t</code> Class Reference | 329 |
| 5.99 | <code>io_tcp_sound_t::float_union</code> Union Reference | 333 |
| 5.100 | <code>io_tcp_t</code> Class Reference | 334 |
| 5.101 | <code>latex_doc_t</code> Class Reference | 336 |
| 5.102 | <code>lpc</code> Class Reference | 338 |
| 5.103 | <code>lpc_bl_predictor</code> Class Reference | 341 |
| 5.104 | <code>lpc_bl_predictor_config</code> Class Reference | 343 |
| 5.105 | <code>lpc_burglattice</code> Class Reference | 345 |
| 5.106 | <code>lpc_burglattice_config</code> Class Reference | 348 |
| 5.107 | <code>lpc_config</code> Class Reference | 349 |
| 5.108 | <code>matrixmixer::cfg_t</code> Class Reference | 351 |
| 5.109 | <code>matrixmixer::matmix_t</code> Class Reference | 353 |
| 5.110 | <code>MHA_AC::ac2matrix_helper_t</code> Class Reference | 355 |
| 5.111 | <code>MHA_AC::ac2matrix_t</code> Class Reference | 356 |
| 5.112 | <code>MHA_AC::acspace2matrix_t</code> Class Reference | 358 |
| 5.113 | <code>MHA_AC::double_t</code> Class Reference | 361 |
| 5.114 | <code>MHA_AC::float_t</code> Class Reference | 363 |
| 5.115 | <code>MHA_AC::int_t</code> Class Reference | 364 |
| 5.116 | <code>MHA_AC::spectrum_t</code> Class Reference | 365 |
| 5.117 | <code>MHA_AC::stat_t</code> Class Reference | 367 |
| 5.118 | <code>MHA_AC::waveform_t</code> Class Reference | 368 |
| 5.119 | <code>mha_audio_descriptor_t</code> Struct Reference | 370 |
| 5.120 | <code>mha_audio_t</code> Struct Reference | 371 |
| 5.121 | <code>mha_channel_info_t</code> Struct Reference | 372 |
| 5.122 | <code>mha_complex_t</code> Struct Reference | 374 |
| 5.123 | <code>mha_dblbuf_t< FIFO ></code> Class Template Reference | 374 |
| 5.124 | <code>mha_direction_t</code> Struct Reference | 380 |
| 5.125 | <code>mha_drifter_fifo_t< T ></code> Class Template Reference | 381 |
| 5.126 | <code>MHA_Error</code> Class Reference | 387 |
| 5.127 | <code>mha_fifo_lw_t< T ></code> Class Template Reference | 389 |
| 5.128 | <code>mha_fifo_posix_threads_t</code> Class Reference | 392 |
| 5.129 | <code>mha_fifo_t< T ></code> Class Template Reference | 394 |
| 5.130 | <code>mha_fifo_thread_guard_t</code> Class Reference | 398 |
| 5.131 | <code>mha_fifo_thread_platform_t</code> Class Reference | 399 |
| 5.132 | <code>mha_rt_fifo_element_t< T ></code> Class Template Reference | 402 |
| 5.133 | <code>mha_rt_fifo_t< T ></code> Class Template Reference | 403 |
| 5.134 | <code>mha_spec_t</code> Struct Reference | 406 |
| 5.135 | <code>MHA_TCP::Async_Notify</code> Class Reference | 408 |
| 5.136 | <code>MHA_TCP::Client</code> Class Reference | 409 |

| | |
|---|-----|
| 5.137 MHA_TCP::Connection Class Reference | 411 |
| 5.138 MHA_TCP::Event_Watcher Class Reference | 418 |
| 5.139 MHA_TCP::OS_EVENT_TYPE Struct Reference | 420 |
| 5.140 MHA_TCP::Server Class Reference | 421 |
| 5.141 MHA_TCP::Sockaccept_Event Class Reference | 424 |
| 5.142 MHA_TCP::Sockread_Event Class Reference | 424 |
| 5.143 MHA_TCP::Sockwrite_Event Class Reference | 426 |
| 5.144 MHA_TCP::Thread Class Reference | 426 |
| 5.145 MHA_TCP::Timeout_Event Class Reference | 430 |
| 5.146 MHA_TCP::Timeout_Watcher Class Reference | 431 |
| 5.147 MHA_TCP::Wakeup_Event Class Reference | 433 |
| 5.148 mha_tictoc_t Struct Reference | 435 |
| 5.149 mha_wave_t Struct Reference | 436 |
| 5.150 mhachain::chain_base_t Class Reference | 437 |
| 5.151 mhachain::mhachain_t Class Reference | 440 |
| 5.152 mhachain::plugs_t Class Reference | 441 |
| 5.153 mhaconfig_t Struct Reference | 444 |
| 5.154 MHAEvents::connector_base_t Class Reference | 446 |
| 5.155 MHAEvents::connector_t< receiver_t > Class Template Reference | 448 |
| 5.156 MHAEvents::emitter_t Class Reference | 450 |
| 5.157 MHAEvents::patchbay_t< receiver_t > Class Template Reference | 452 |
| 5.158 MHAFilter::adapt_filter_param_t Class Reference | 453 |
| 5.159 MHAFilter::adapt_filter_state_t Class Reference | 454 |
| 5.160 MHAFilter::adapt_filter_t Class Reference | 455 |
| 5.161 MHAFilter::blockprocessing_polyphase_resampling_t Class Reference | 457 |
| 5.162 MHAFilter::complex_bandpass_t Class Reference | 460 |
| 5.163 MHAFilter::diff_t Class Reference | 462 |
| 5.164 MHAFilter::fftfiler_t Class Reference | 463 |
| 5.165 MHAFilter::fftfilerbank_t Class Reference | 467 |
| 5.166 MHAFilter::filter_t Class Reference | 471 |
| 5.167 MHAFilter::gammaflt_t Class Reference | 475 |
| 5.168 MHAFilter::iir_filter_state_t Class Reference | 478 |
| 5.169 MHAFilter::iir_filter_t Class Reference | 479 |
| 5.170 MHAFilter::iir_ord1_real_t Class Reference | 482 |
| 5.171 MHAFilter::o1_ar_filter_t Class Reference | 485 |
| 5.172 MHAFilter::o1flt_lowpass_t Class Reference | 488 |
| 5.173 MHAFilter::o1flt_maxtrack_t Class Reference | 491 |
| 5.174 MHAFilter::o1flt_mintrack_t Class Reference | 492 |
| 5.175 MHAFilter::partitioned_convolution_t Class Reference | 494 |
| 5.176 MHAFilter::partitioned_convolution_t::index_t Struct Reference | 498 |
| 5.177 MHAFilter::polyphase_resampling_t Class Reference | 499 |
| 5.178 MHAFilter::resampling_filter_t Class Reference | 504 |
| 5.179 MHAFilter::smoothspec_t Class Reference | 505 |
| 5.180 MHAFilter::thirdoctave_analyzer_t Class Reference | 508 |
| 5.181 MHAFilter::transfer_function_t Struct Reference | 510 |
| 5.182 MHAFilter::transfer_matrix_t Struct Reference | 513 |
| 5.183 MHAIOJack::io_jack_t Class Reference | 514 |
| 5.184 MHAIOPortAudio::device_info_t Class Reference | 518 |
| 5.185 MHAIOPortAudio::io_portaudio_t Class Reference | 520 |
| 5.186 MHAJack::client_avg_t Class Reference | 523 |
| 5.187 MHAJack::client_noncont_t Class Reference | 526 |

| | |
|---|-----|
| 5.188 MHAJack::client_t Class Reference | 528 |
| 5.189 MHAJack::port_t Class Reference | 534 |
| 5.190 MHAKernel::algo_comm_class_t Class Reference | 537 |
| 5.191 MHAKernel::comm_var_map_t Class Reference | 541 |
| 5.192 MHAMultiSrc::base_t Class Reference | 541 |
| 5.193 MHAMultiSrc::channel_t Class Reference | 543 |
| 5.194 MHAMultiSrc::channels_t Class Reference | 543 |
| 5.195 MHAMultiSrc::spectrum_t Class Reference | 544 |
| 5.196 MHAMultiSrc::waveform_t Class Reference | 545 |
| 5.197 MHAOvFilter::band_descriptor_t Class Reference | 546 |
| 5.198 MHAOvFilter::barkscale::bark2hz_t Class Reference | 547 |
| 5.199 MHAOvFilter::barkscale::hz2bark_t Class Reference | 548 |
| 5.200 MHAOvFilter::fftfb_ac_info_t Class Reference | 549 |
| 5.201 MHAOvFilter::fftfb_t Class Reference | 550 |
| 5.202 MHAOvFilter::fftfb_vars_t Class Reference | 553 |
| 5.203 MHAOvFilter::fscale_bw_t Class Reference | 555 |
| 5.204 MHAOvFilter::fscale_t Class Reference | 557 |
| 5.205 MHAOvFilter::fspacing_t Class Reference | 558 |
| 5.206 MHAOvFilter::overlap_save_filterbank_analytic_t Class Reference | 561 |
| 5.207 MHAOvFilter::overlap_save_filterbank_t Class Reference | 562 |
| 5.208 MHAOvFilter::overlap_save_filterbank_t::vars_t Class Reference | 564 |
| 5.209 MHAOvFilter::scale_var_t Class Reference | 566 |
| 5.210 MHAParser::base_t Class Reference | 568 |
| 5.211 MHAParser::base_t::replace_t Class Reference | 577 |
| 5.212 MHAParser::bool_mon_t Class Reference | 578 |
| 5.213 MHAParser::bool_t Class Reference | 579 |
| 5.214 MHAParser::c_ifc_parser_t Class Reference | 582 |
| 5.215 MHAParser::commit_t< receiver_t > Class Template Reference | 584 |
| 5.216 MHAParser::complex_mon_t Class Reference | 586 |
| 5.217 MHAParser::complex_t Class Reference | 588 |
| 5.218 MHAParser::entry_t Class Reference | 590 |
| 5.219 MHAParser::expression_t Class Reference | 591 |
| 5.220 MHAParser::float_mon_t Class Reference | 592 |
| 5.221 MHAParser::float_t Class Reference | 593 |
| 5.222 MHAParser::int_mon_t Class Reference | 596 |
| 5.223 MHAParser::int_t Class Reference | 598 |
| 5.224 MHAParser::keyword_list_t Class Reference | 600 |
| 5.225 MHAParser::kw_t Class Reference | 603 |
| 5.226 MHAParser::mcomplex_mon_t Class Reference | 606 |
| 5.227 MHAParser::mcomplex_t Class Reference | 608 |
| 5.228 MHAParser::mfloat_mon_t Class Reference | 610 |
| 5.229 MHAParser::mfloat_t Class Reference | 611 |
| 5.230 MHAParser::mhaconfig_mon_t Class Reference | 614 |
| 5.231 MHAParser::mhapluginloader_t Class Reference | 616 |
| 5.232 MHAParser::monitor_t Class Reference | 619 |
| 5.233 MHAParser::parser_t Class Reference | 620 |
| 5.234 MHAParser::range_var_t Class Reference | 625 |
| 5.235 MHAParser::string_mon_t Class Reference | 628 |
| 5.236 MHAParser::string_t Class Reference | 630 |
| 5.237 MHAParser::variable_t Class Reference | 632 |
| 5.238 MHAParser::vcomplex_mon_t Class Reference | 634 |

| | | |
|-------|--|-----|
| 5.239 | MHAParser::vcomplex_t Class Reference | 636 |
| 5.240 | MHAParser::vfloat_mon_t Class Reference | 638 |
| 5.241 | MHAParser::vfloat_t Class Reference | 640 |
| 5.242 | MHAParser::vint_mon_t Class Reference | 642 |
| 5.243 | MHAParser::vint_t Class Reference | 644 |
| 5.244 | MHAParser::vstring_mon_t Class Reference | 646 |
| 5.245 | MHAParser::vstring_t Class Reference | 648 |
| 5.246 | MHAParser::window_t Class Reference | 650 |
| 5.247 | mhaplug_cfg_t Class Reference | 653 |
| 5.248 | MHAPLugin::cfg_chain_t< runtime_cfg_t > Class Template Reference | 653 |
| 5.249 | MHAPLugin::config_t< runtime_cfg_t > Class Template Reference | 654 |
| 5.250 | MHAPLugin::plugin_t< runtime_cfg_t > Class Template Reference | 659 |
| 5.251 | MHAPLugin_Resampling::resampling_if_t Class Reference | 664 |
| 5.252 | MHAPLugin_Resampling::resampling_t Class Reference | 666 |
| 5.253 | MHAPLugin_Split::domain_handler_t Class Reference | 667 |
| 5.254 | MHAPLugin_Split::dummy_threads_t Class Reference | 672 |
| 5.255 | MHAPLugin_Split::posix_threads_t Class Reference | 674 |
| 5.256 | MHAPLugin_Split::split_t Class Reference | 678 |
| 5.257 | MHAPLugin_Split::splitted_part_t Class Reference | 682 |
| 5.258 | MHAPLugin_Split::thread_platform_t Class Reference | 687 |
| 5.259 | MHAPLugin_Split::uni_processor_t Class Reference | 690 |
| 5.260 | mhaserver_t Class Reference | 691 |
| 5.261 | MHASignal::async_rmslevel_t Class Reference | 694 |
| 5.262 | MHASignal::delay_spec_t Class Reference | 696 |
| 5.263 | MHASignal::delay_t Class Reference | 697 |
| 5.264 | MHASignal::delay_wave_t Class Reference | 699 |
| 5.265 | MHASignal::doublebuffer_t Class Reference | 700 |
| 5.266 | MHASignal::fft_t Class Reference | 703 |
| 5.267 | MHASignal::hilbert_fftw_t Class Reference | 706 |
| 5.268 | MHASignal::hilbert_t Class Reference | 707 |
| 5.269 | MHASignal::loop_wavefragment_t Class Reference | 709 |
| 5.270 | MHASignal::matrix_t Class Reference | 713 |
| 5.271 | MHASignal::minphase_t Class Reference | 721 |
| 5.272 | MHASignal::quantizer_t Class Reference | 723 |
| 5.273 | MHASignal::ringbuffer_t Class Reference | 724 |
| 5.274 | MHASignal::schroeder_t Class Reference | 728 |
| 5.275 | MHASignal::spectrum_t Class Reference | 731 |
| 5.276 | MHASignal::stat_t Class Reference | 736 |
| 5.277 | MHASignal::subsample_delay_t Class Reference | 737 |
| 5.278 | MHASignal::uint_vector_t Class Reference | 740 |
| 5.279 | MHASignal::waveform_t Class Reference | 743 |
| 5.280 | MHASndFile::sf_t Class Reference | 753 |
| 5.281 | MHASndFile::sf_wave_t Class Reference | 754 |
| 5.282 | MHATableLookup::linear_table_t Class Reference | 755 |
| 5.283 | MHATableLookup::table_t Class Reference | 759 |
| 5.284 | MHATableLookup::xy_table_t Class Reference | 761 |
| 5.285 | MHAWindow::bartlett_t Class Reference | 765 |
| 5.286 | MHAWindow::base_t Class Reference | 766 |
| 5.287 | MHAWindow::blackman_t Class Reference | 768 |
| 5.288 | MHAWindow::fun_t Class Reference | 769 |
| 5.289 | MHAWindow::hamming_t Class Reference | 770 |

| | |
|--|-----|
| 5.290 MHAWindow::hanning_t Class Reference | 771 |
| 5.291 MHAWindow::rect_t Class Reference | 772 |
| 5.292 MHAWindow::user_t Class Reference | 773 |
| 5.293 mon_t Class Reference | 775 |
| 5.294 multibandcompressor::fftfb_plug_t Class Reference | 776 |
| 5.295 multibandcompressor::interface_t Class Reference | 778 |
| 5.296 multibandcompressor::plugin_signals_t Class Reference | 780 |
| 5.297 nlms_t Class Reference | 781 |
| 5.298 noise_t Class Reference | 784 |
| 5.299 noisePowProposedScale::interface_t Class Reference | 786 |
| 5.300 noisePowProposedScale::noisePowProposed Class Reference | 788 |
| 5.301 overlapadd::overlapadd_if_t Class Reference | 790 |
| 5.302 overlapadd::overlapadd_t Class Reference | 792 |
| 5.303 parser_int_dyn Class Reference | 795 |
| 5.304 plug_t Class Reference | 796 |
| 5.305 plugin_interface_t Class Reference | 797 |
| 5.306 pluginbrowser_t Class Reference | 799 |
| 5.307 plugindescription_t Class Reference | 800 |
| 5.308 PluginLoader::config_file_splitter_t Class Reference | 801 |
| 5.309 PluginLoader::fourway_processor_t Class Reference | 802 |
| 5.310 PluginLoader::mhapluginloader_t Class Reference | 805 |
| 5.311 pluginloader_t Class Reference | 810 |
| 5.312 prediction_error Class Reference | 811 |
| 5.313 prediction_error_config Class Reference | 814 |
| 5.314 rmslevel_if_t Class Reference | 817 |
| 5.315 rmslevel_t Class Reference | 819 |
| 5.316 route::interface_t Class Reference | 820 |
| 5.317 route::process_t Class Reference | 822 |
| 5.318 rt_nlms_t Class Reference | 823 |
| 5.319 save_spec_t Class Reference | 826 |
| 5.320 save_wave_t Class Reference | 827 |
| 5.321 shadowfilter_begin::cfg_t Class Reference | 828 |
| 5.322 shadowfilter_begin::shadowfilter_begin_t Class Reference | 830 |
| 5.323 shadowfilter_end::cfg_t Class Reference | 831 |
| 5.324 shadowfilter_end::shadowfilter_end_t Class Reference | 833 |
| 5.325 sine_cfg_t Struct Reference | 834 |
| 5.326 sine_t Class Reference | 835 |
| 5.327 smoothgains_bridge::overlapadd_if_t Class Reference | 837 |
| 5.328 smoothgains_bridge::smoothspec_wrap_t Class Reference | 839 |
| 5.329 softclip_t Class Reference | 841 |
| 5.330 softclipper_t Class Reference | 842 |
| 5.331 softclipper_variables_t Class Reference | 844 |
| 5.332 spec2wave_if_t Class Reference | 845 |
| 5.333 spec2wave_t Class Reference | 847 |
| 5.334 spec_fader_t Class Reference | 848 |
| 5.335 speechnoise_t Class Reference | 849 |
| 5.336 steerbf Class Reference | 851 |
| 5.337 steerbf_config Class Reference | 853 |
| 5.338 timo_AC Class Reference | 854 |
| 5.339 timo_params Class Reference | 856 |
| 5.340 timoConfig Class Reference | 858 |

| | | |
|----------|--|------------|
| 5.341 | timoSmooth Class Reference | 862 |
| 5.342 | us_t Class Reference | 865 |
| 5.343 | wave2spec_if_t Class Reference | 867 |
| 5.344 | wave2spec_t Class Reference | 869 |
| 5.345 | wavrec_t Class Reference | 871 |
| 5.346 | wavwriter_t Class Reference | 873 |
| 5.347 | windowselector_t Class Reference | 875 |
| 6 | File Documentation | 878 |
| 6.1 | ac2wave.cpp File Reference | 878 |
| 6.2 | ac_monitor_type.cpp File Reference | 878 |
| 6.3 | ac_monitor_type.hh File Reference | 878 |
| 6.4 | acConcat_wave.cpp File Reference | 878 |
| 6.5 | acConcat_wave.h File Reference | 878 |
| 6.6 | acmon.cpp File Reference | 879 |
| 6.7 | acPooling_wave.cpp File Reference | 879 |
| 6.8 | acPooling_wave.h File Reference | 879 |
| 6.9 | acsave.cpp File Reference | 879 |
| 6.10 | acSteer.cpp File Reference | 880 |
| 6.11 | acSteer.h File Reference | 880 |
| 6.12 | acTransform_wave.cpp File Reference | 881 |
| 6.13 | acTransform_wave.h File Reference | 881 |
| 6.14 | adm.cpp File Reference | 881 |
| 6.15 | adm.hh File Reference | 882 |
| 6.16 | altplugs.cpp File Reference | 883 |
| 6.17 | analysemhaplugin.cpp File Reference | 883 |
| 6.18 | analysispath.cpp File Reference | 883 |
| 6.19 | auditory_profile.cpp File Reference | 884 |
| 6.20 | auditory_profile.h File Reference | 884 |
| 6.21 | browsemhaplugins.cpp File Reference | 884 |
| 6.22 | coherence.cpp File Reference | 885 |
| 6.23 | combinechannels.cpp File Reference | 885 |
| 6.24 | complex_filter.cpp File Reference | 885 |
| 6.25 | complex_filter.h File Reference | 885 |
| 6.26 | cpupload.cpp File Reference | 886 |
| 6.27 | db.cpp File Reference | 886 |
| 6.28 | dc.cpp File Reference | 886 |
| 6.29 | dc_afterburn.cpp File Reference | 887 |
| 6.30 | dc_afterburn.h File Reference | 887 |
| 6.31 | dc_simple.cpp File Reference | 888 |
| 6.32 | delay.cpp File Reference | 888 |
| 6.33 | delaysum.cpp File Reference | 889 |
| 6.34 | doasvm_classification.cpp File Reference | 889 |
| 6.35 | doasvm_classification.h File Reference | 889 |
| 6.36 | doasvm_feature_extraction.cpp File Reference | 889 |
| 6.37 | doasvm_feature_extraction.h File Reference | 890 |
| 6.38 | doc_appendix.h File Reference | 890 |
| 6.39 | doc_examples.h File Reference | 890 |
| 6.40 | doc_frameworks.h File Reference | 890 |
| 6.41 | doc_general.h File Reference | 890 |
| 6.42 | doc_kernel.h File Reference | 890 |
| 6.43 | doc_matlab.h File Reference | 890 |

| | | |
|------|---|-----|
| 6.44 | doc_mhamain.h File Reference | 890 |
| 6.45 | doc_parser.h File Reference | 890 |
| 6.46 | doc_pluginif.cpp File Reference | 890 |
| 6.47 | doc_plugins.h File Reference | 890 |
| 6.48 | doc_system.h File Reference | 890 |
| 6.49 | doc_toolbox.h File Reference | 890 |
| 6.50 | downsample.cpp File Reference | 890 |
| 6.51 | droptect.cpp File Reference | 891 |
| 6.52 | example1.cpp File Reference | 891 |
| 6.53 | example2.cpp File Reference | 891 |
| 6.54 | example3.cpp File Reference | 891 |
| 6.55 | example4.cpp File Reference | 891 |
| 6.56 | example5.cpp File Reference | 891 |
| 6.57 | example6.cpp File Reference | 892 |
| 6.58 | fader_spec.cpp File Reference | 892 |
| 6.59 | fader_wave.cpp File Reference | 892 |
| 6.60 | fftfilterbank.cpp File Reference | 893 |
| 6.61 | fshift_hilbert.cpp File Reference | 893 |
| 6.62 | gain.cpp File Reference | 893 |
| 6.63 | gaintable.cpp File Reference | 894 |
| 6.64 | gaintable.h File Reference | 894 |
| 6.65 | generatemhaplugindoc.cpp File Reference | 895 |
| 6.66 | hann.cpp File Reference | 895 |
| 6.67 | hann.h File Reference | 896 |
| 6.68 | identity.cpp File Reference | 896 |
| 6.69 | ifftshift.cpp File Reference | 896 |
| 6.70 | ifftshift.h File Reference | 897 |
| 6.71 | iirfilter.cpp File Reference | 897 |
| 6.72 | lpc.cpp File Reference | 897 |
| 6.73 | lpc.h File Reference | 898 |
| 6.74 | lpc_bl_predictor.cpp File Reference | 898 |
| 6.75 | lpc_bl_predictor.h File Reference | 898 |
| 6.76 | lpc_burg-lattice.cpp File Reference | 899 |
| 6.77 | lpc_burg-lattice.h File Reference | 899 |
| 6.78 | matrixmixer.cpp File Reference | 899 |
| 6.79 | mha.cpp File Reference | 900 |
| 6.80 | mha.h File Reference | 900 |
| 6.81 | mha_algo_comm.cpp File Reference | 905 |
| 6.82 | mha_algo_comm.h File Reference | 905 |
| 6.83 | mha_algo_comm.hh File Reference | 907 |
| 6.84 | mha_defs.h File Reference | 907 |
| 6.85 | mha_errno.c File Reference | 909 |
| 6.86 | mha_errno.h File Reference | 910 |
| 6.87 | mha_error.cpp File Reference | 911 |
| 6.88 | mha_error.hh File Reference | 912 |
| 6.89 | mha_event_emitter.h File Reference | 913 |
| 6.90 | mha_events.cpp File Reference | 913 |
| 6.91 | mha_events.h File Reference | 913 |
| 6.92 | mha_fftfb.cpp File Reference | 913 |
| 6.93 | mha_fftfb.hh File Reference | 915 |
| 6.94 | mha_fifo.cpp File Reference | 916 |

| | | |
|-------|--|-----|
| 6.95 | mha_fifo.h File Reference | 916 |
| 6.96 | mha_filter.cpp File Reference | 916 |
| 6.97 | mha_filter.hh File Reference | 917 |
| 6.98 | mha_generic_chain.cpp File Reference | 918 |
| 6.99 | mha_generic_chain.h File Reference | 919 |
| 6.100 | mha_io_ifc.h File Reference | 919 |
| 6.101 | mha_multisrc.cpp File Reference | 920 |
| 6.102 | mha_multisrc.h File Reference | 921 |
| 6.103 | mha_os.cpp File Reference | 921 |
| 6.104 | mha_os.h File Reference | 921 |
| 6.105 | mha_parser.cpp File Reference | 924 |
| 6.106 | mha_parser.hh File Reference | 925 |
| 6.107 | mha_plugin.hh File Reference | 929 |
| 6.108 | mha_profiling.c File Reference | 932 |
| 6.109 | mha_profiling.h File Reference | 932 |
| 6.110 | mha_ruby.cpp File Reference | 933 |
| 6.111 | mha_signal.cpp File Reference | 933 |
| 6.112 | mha_signal.hh File Reference | 936 |
| 6.113 | mha_signal_fft.h File Reference | 946 |
| 6.114 | mha_tablelookup.cpp File Reference | 946 |
| 6.115 | mha_tablelookup.hh File Reference | 946 |
| 6.116 | mha_tcp.cpp File Reference | 947 |
| 6.117 | mha_tcp.hh File Reference | 948 |
| 6.118 | mha_toolbox.h File Reference | 950 |
| 6.119 | mha_windowparser.cpp File Reference | 950 |
| 6.120 | mha_windowparser.h File Reference | 950 |
| 6.121 | mhachain.cpp File Reference | 951 |
| 6.122 | mhafw_lib.cpp File Reference | 951 |
| 6.123 | mhafw_lib.h File Reference | 951 |
| 6.124 | MHAIOFile.cpp File Reference | 952 |
| 6.125 | MHAIOJack.cpp File Reference | 954 |
| 6.126 | MHAIOParser.cpp File Reference | 957 |
| 6.127 | MHAIOPortAudio.cpp File Reference | 960 |
| 6.128 | MHAIOTCP.cpp File Reference | 962 |
| 6.129 | mhajack.cpp File Reference | 966 |
| 6.130 | mhajack.h File Reference | 966 |
| 6.131 | mhamain.cpp File Reference | 968 |
| 6.132 | mhapluginloader.cpp File Reference | 969 |
| 6.133 | mhapluginloader.h File Reference | 969 |
| 6.134 | mhasndfile.cpp File Reference | 970 |
| 6.135 | mhasndfile.h File Reference | 970 |
| 6.136 | multibandcompressor.cpp File Reference | 971 |
| 6.137 | nlms_wave.cpp File Reference | 971 |
| 6.138 | noise.cpp File Reference | 972 |
| 6.139 | noisePowProposedScale.cpp File Reference | 972 |
| 6.140 | overlapadd.cpp File Reference | 973 |
| 6.141 | pluginbrowser.cpp File Reference | 973 |
| 6.142 | pluginbrowser.h File Reference | 973 |
| 6.143 | prediction_error.cpp File Reference | 973 |
| 6.144 | prediction_error.h File Reference | 974 |
| 6.145 | resampling.cpp File Reference | 974 |

| | |
|---|-----|
| 6.146 rmslevel.cpp File Reference | 974 |
| 6.147 route.cpp File Reference | 975 |
| 6.148 save_spec.cpp File Reference | 975 |
| 6.149 save_wave.cpp File Reference | 975 |
| 6.150 shadowfilter_begin.cpp File Reference | 975 |
| 6.151 shadowfilter_end.cpp File Reference | 975 |
| 6.152 sine.cpp File Reference | 976 |
| 6.153 smoothgains_bridge.cpp File Reference | 976 |
| 6.154 softclip.cpp File Reference | 976 |
| 6.155 spec2wave.cpp File Reference | 976 |
| 6.156 speechnoise.cpp File Reference | 977 |
| 6.157 speechnoise.h File Reference | 980 |
| 6.158 split.cpp File Reference | 980 |
| 6.159 steerbf.cpp File Reference | 981 |
| 6.160 steerbf.h File Reference | 981 |
| 6.161 testalsadevice.c File Reference | 982 |
| 6.162 timoconfig.cpp File Reference | 982 |
| 6.163 timoconfig.h File Reference | 982 |
| 6.164 timoSmooth.cpp File Reference | 983 |
| 6.165 timosmooth.h File Reference | 983 |
| 6.166 transducers.cpp File Reference | 983 |
| 6.167 upsample.cpp File Reference | 984 |
| 6.168 wave2spec.cpp File Reference | 984 |
| 6.169 wavrec.cpp File Reference | 984 |
| 6.170 windowselector.cpp File Reference | 985 |
| 6.171 windowselector.h File Reference | 985 |

Index**987**

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. 33)
- **Signal processing plugins** (p. 6)
- Audio input-output (IO) plugins (see `io_file_t` (p. 309), `MHAIOJack` (p. 93), `io_parser_t` (p. 317), `io_tcp_parser_t` (p. 323))
- **The openMHA toolbox library** (p. 34)

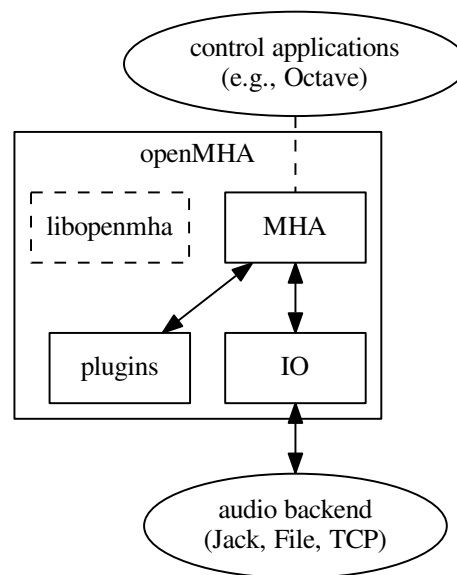


Figure 1 openMHA structure

The openMHA command line application (MHA) (p. 33) 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. 93) plugin (see plugins' manual) is used, which provides an interface to the Jack Audio Connection Kit (JACK). Other IO plugins provide audio file access or TCP/IP-based processing.

openMHA plugins (p. 6) 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. 27) 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 ($1\text{Pa} = 1\text{N/m}^2$). With this convention in place, all plugins can determine the absolute physical sound pressure level from the sound samples that they process. A derived convention is employed in the spectral domain for STFT signals. Due to the dependency of the calibration on the hardware used, it is the responsibility of the user of the openMHA to perform calibration measurements and adapt the openMHA settings to make sure that this calibration convention is met. We provide the plugin `transducers` which can be configured to perform the necessary signal adjustments.

2 Todo List

Class AuditoryProfile::profile_t (p. 189)

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

Class mhaconfig_t (p. 444)

Add information on number of bands and on center frequencies, or replace by **mha_audio↔_descriptor_t** (p. 370).

Class MHAFilter::filter_t (p. 471)

Implement a more robust filter form.

Member MHAFilter::polyphase_resampling_t::now_index (p. 503)

Index into what? What is the meaning of now?

Class MHAPlugin::plugin_t< runtime_cfg_t > (p. 659)

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

3 Module Documentation

3.1 Concept of Variables and Data Exchange in the openMHA

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

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

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

- **Configuration variables** : Read and write accesses are possible through the openM↔HA 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. 27)): 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. 20) and references therein).

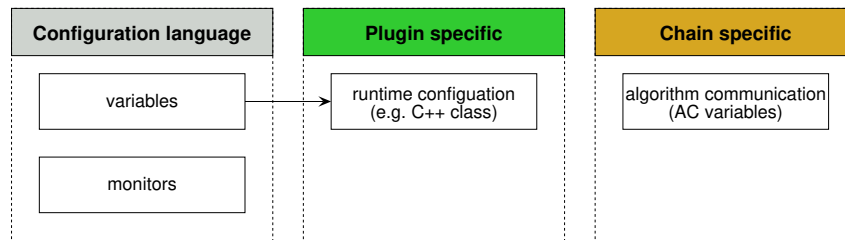


Figure 2 Variable types in the openMHA

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

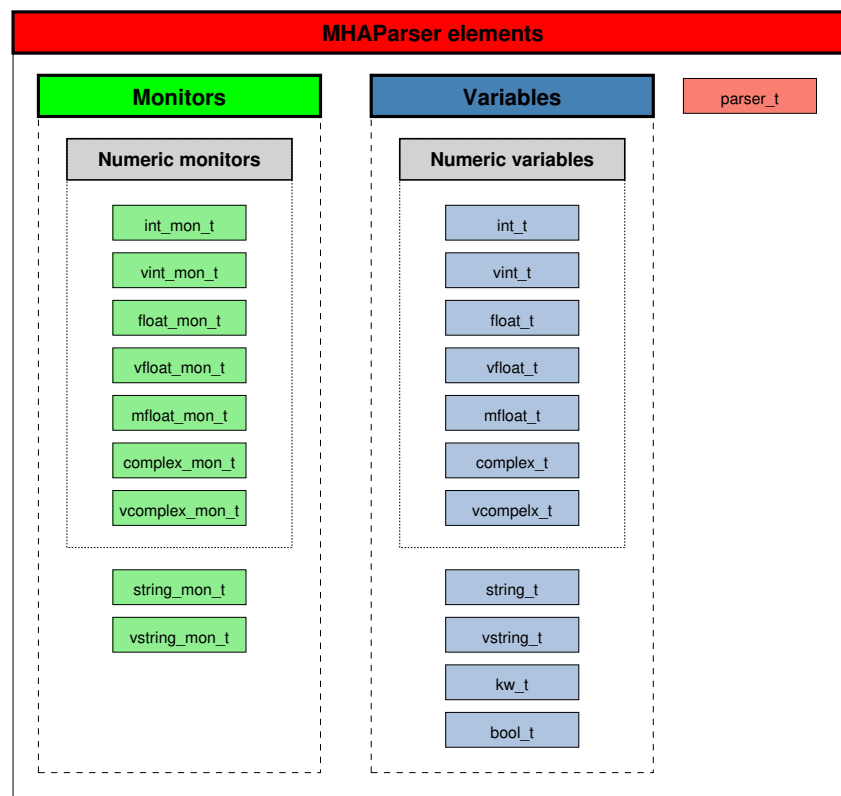


Figure 3 MHAParser elements

3.2 The openMHA Plugins (programming interface)

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

Classes

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

Macros

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

3.2.1 Detailed Description

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

openMHA plugins can be combined into processing chains. One of the configured chains can be selected for output which allows direct comparison of single algorithms or complex signal processing configurations. Algorithms within one chain can communicate with each other by sharing some of their variables, see section **Communication between algorithms** (p. 27).

The openMHA plugins can use the openMHA configuration language for their configuration. If they do so, the configuration can be changed through the framework even at run time. A description of this language can be found in section **The openMHA configuration language** (p. 33). If the algorithms should make use of the openMHA configuration language, they need to be written in C++ rather than pure C.

In the openMHA package a set of example plugins is included. These examples are the base of a step by step tutorial on how to write an openMHA plugin. See section **Writing openMHA Plugins. A step-by-step tutorial** (p. 10) for details.

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

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

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

- **mha_wave_t** (p. 436) * process(**mha_wave_t** (p. 436) *): pure waveform processing
- **mha_spec_t** (p. 406) * process(**mha_spec_t** (p. 406) *): pure spectral processing

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

- **mha_spec_t** (p. 406) * process(**mha_wave_t** (p. 436) *): Domain transformation from waveform to spectrum
- **mha_wave_t** (p. 436) * process(**mha_spec_t** (p. 406) *): Domain transformation from spectrum to waveform

For preparation and release of a plugin, the methods

- void prepare(**mhaconfig_t** (p. 444) &) and
- void release(void)

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

For configuration purposes, the plugin class has to export a method called `parse()` which implements the openMHA configuration language. We strongly recommend that you do not implement this method yourself, but by inheriting from the class **MHAParser::parser_t** (p. 620) from the openMHA toolbox, directly or indirectly (inheriting from a class that itself inherits from **MHAParser::parser_t** (p. 620)).

3.2.2 Connecting the C++ class with the C Interface

A C++ class which provides the appropriate methods can be used as an openMHA Plugin by connecting it to the C interface using the **MHAPLUGIN_CALLBACKS** (p. 9) macro.

The openMHA Toolbox library provides a base class **MHAPLugin::plugin_t** (p. 659)<T> (a template class) which can be used as the base class for a plugin class. This base class implements some necessary features for openMHA plugin developers like integration into the openMHA configuration language environment (it inherits from **MHAParser::parser_t** (p. 620)) and thread-safe runtime configuration update.

3.2.3 Error reporting

When your plugin detects a situation that it cannot handle, like input signal of the wrong signal domain at preparation time, unsupported number of input channels at preparation time, unsupported combinations of values in the plugin's variables during configuration, it should throw a C++ exception. The exception should be of type `MHAError`. Exceptions of this type are caught by the **MHAPLUGIN_CALLBACKS** (p. 9) macro for further error Reporting.

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

3.2.4 Contents of the openMHA Plugin programming interface

3.2.5 Macro Definition Documentation

3.2.5.1 #define MHAPLUGIN_CALLBACKS_PREFIX(

```
prefix,
classname,
indom,
outdom )
```

C++ wrapper macro for the plugin interface.

Parameters

| | |
|------------------|--|
| <i>classname</i> | The name of the plugin class |
| <i>indom</i> | Input domain (<i>wave</i> or <i>spec</i>) |
| <i>outdom</i> | Output domain (<i>wave</i> or <i>spec</i>) |

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 appropriate member functions have to be defined in the class. It is suggested to make usage of the **MHAPlugin::plugin_t** (p. 659) template class. Exceptions of type **MHA_Error** (p. 387) are caught and transformed into appropriate error codes with their corresponding error messages.

```

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

```

C++ wrapper macro for the plugin interface.

Parameters

| | |
|------------------|---|
| <i>plugname</i> | The file name of the plugin without the .so or .dll extension |
| <i>classname</i> | The name of the plugin class |
| <i>indom</i> | Input domain (<i>wave</i> or <i>spec</i>) |
| <i>outdom</i> | Output domain (<i>wave</i> or <i>spec</i>) |

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 appropriate member functions have to be defined in the class. It is suggested to make usage of the `MHAPLugin::plugin_t` (p. 659) template class. Exceptions of type `MHA_Error` (p. 387) are caught and transformed into appropriate error codes with their corresponding error messages.

```

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

```

Wrapper macro for the plugin documentation interface.

Parameters

| | |
|---------------|--|
| <i>plugin</i> | The file name of the plugin without the .so or .dll extension |
| <i>cat</i> | Space separated list of categories to which belong the plugin (as const char*) |
| <i>doc</i> | Documentation of the plugin (as const char*) |

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

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

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

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

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

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

3.3.1 example1.cpp

The example plugin file **example1.cpp** (p. 891) 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. 659) exports several methods, but only two of them need a non-empty implementation: `prepare()` method is a pure virtual function and `process()` is called when signal processing starts.

```
#include "mha_plugin.hh"

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

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

Every plugin implementation should include the '**mha_plugin.hh** (p. 929)' 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. 929).

The class `plugin1_t` inherits from the class **MHAPlugin::plugin_t** (p. 659), which then inherits from **MHAParser::parser_t** (p. 620) – the configuration language interface in the method "parse". Our plugin class therefore exports the working "parse" method inherited from **MHAParser::parser_t** (p. 620), and the plugin is visible in the openMHA configuration tree.

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

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

3.3.1.1 The prepare method

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

Parameters

| | |
|--------------------|--|
| <i>signal_info</i> | Contains information about the input signal's parameters, see mhaconfig_t (p. 444). |
|--------------------|--|

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

3.3.1.2 The signal processing method

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

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

Parameters

| | |
|---------------|---|
| <i>signal</i> | Pointer to the input signal structure mha_wave_t (p. 436). |
|---------------|---|

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. 436) of the process method's parameter). It scales the first channel by a factor of 0.1. The output signal reuses the structure that previously contained the input signal (in-place processing).

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

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

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

This macro takes care of accessing the C++ class from the C functions required as the plugin's interface. It implements the C functions and calls the corresponding C++ instance methods. Plugin classes should be derived from the template class **MHAPlugin::plugin_t** (p. 659) to be compatible with the C interface wrapper.

This macro also catches C++ exceptions of type **MHA_Error** (p. 387), 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. 387) 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. 387). This is important, because: (1) C++ exceptions cannot cross the plugin interface, which is in C, and (2) there is no error handling code for your exception classes in the openMHA framework anyways.

3.3.2 example2.cpp

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

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

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

    void prepare(mhaconfig_t & signal_info);

    void release(void);

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

Parameters

| | |
|-----------------|------------------------------------|
| <i>scale_ch</i> | – the channel number to be scaled |
| <i>factor</i> | – the scale factor of the scaling. |

This class again inherits from the template class **MHAPlugin::plugin_t** (p. 659) for integration with the openMHA configuration language. The two data members serve as externally visible configuration variables. All methods of this class have a non-empty implementation.

3.3.2.1 Constructor

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

```

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

```

The constructor invokes the superclass constructor with a string parameter. This string parameter serves as the help text that describes the functionality of the plugin. The constructor registers configuration variables with the openMHA configuration tree and sets their default values and permitted ranges. The minimum permitted value for both variables is zero, and there is no maximum limit (apart from the limitations of the underlying C data type). The configuration variables have to be registered with the parser node instance using the **MHAParser::parser←_t::insert_item** (p. 622) method.

3.3.2.2 The prepare method

```

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

```

Parameters

| | |
|--------------------|--|
| <i>signal_info</i> | – contains information about the input signal's parameters, see mhaconfig_t (p. 444). |
|--------------------|--|

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. $0 \leq \text{scale_ch.data} < \text{signal_info.channels}$. The prepare method does not have to check that $0 \leq \text{scale_ch.data}$, 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 concatenates a string of the form "[0,n[". `n` is the number of channels in the input signal, and is used here as an exclusive upper boundary. To convert the number of channels into a string, a helper function for string conversion from the openMHA Toolbox is used. This function is overloaded and works for several data types.

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

3.3.2.3 The release method

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

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

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

3.3.2.4 The signal processing method

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

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

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

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

Consistency

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

Fortunately, for some data types on PC systems, case (c) cannot happen. These are 32bit wide data types with a 4-byte alignment. Therefore, the values in **MHAParser::int_t** (p. 598) and **MHAParser::float_t** (p. 593) are always consistent, but this is not the case for vectors, strings, or complex values. With these, you can get a mixture of the bit patterns of old and new values, or you can even cause a memory access violation in case a vector or string grows and has to be reallocated to a different memory address.

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

3.3.3 example3.cpp

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

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

writeaccess

- triggered on write access to a configuration variable.

valuechanged

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

readaccess

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

prereadaccess

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

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

3.3.3.1 Data member declarations

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

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

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

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

3.3.3.2 Method declarations

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

    void prepare(mhaconfig_t & signal_info);

    void release(void);

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

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

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

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

3.3.3.3 Example 3 constructor

```

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

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

```

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

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

3.3.3.4 The prepare method

```

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

    // bookkeeping
    prepared.data = 1;
}

```

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

3.3.3.5 The release method

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

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

3.3.3.6 The signal processing method

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

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

3.3.3.7 The callback methods

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

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

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

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

3.3.4 example4.cpp

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

3.3.4.1 The Prepare method

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

    // bookkeeping
    prepared.data = 1;
}
```

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

3.3.4.2 The signal processing method

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

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

The **mha_spec_t** (p. 406) instance stores the complex (**mha_complex_t** (p. 374)) spectral signal for positive frequencies only (since the waveform signal is always real). The `num_frames` member of **mha_spec_t** (p. 406) actually denotes the number of STFT bins.

Please note that different from **mha_wave_t** (p. 436), a multichannel signal in **mha_spec_t** (p. 406) is stored non-interleaved in the signal buffer.

Some arithmetic operations are defined on struct **mha_complex_t** (p. 374) to facilitate efficient complex computations. The `*=` operator used here (defined for real and for complex arguments) is one of them.

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

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

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

3.3.5 example5.cpp

Many algorithms use complex operations to transform the user space variables into run time configurations. If this takes a noticeable time (e.g. more than 100-500 μ sec), the update of the runtime configuration can not take place in the real time processing thread. Furthermore, the parallel access to complex structures may cause unpredictable results if variables are read while only parts of them are written to memory (cf. section **Consistency** (p. 15)). To handle these situations, a special C++ template class **MHAPLugin::plugin_t** (p. 659) 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. 278) is the parameter of the template class **MHAPLugin::plugin_t** (p. 659). 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. 659), 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. 452) instance.

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

The constructor of the runtime configuration analyses and validates the user variables. If the configuration is invalid, an exception of type **MHA_Error** (p. 387) 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. 387) exception.

```

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

```

In this example, the run time configuration class `example5_t` (p. 278) 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. 452) instance.

```

plugin_interface_t::plugin_interface_t(
    const algo_comm_t& iac,
    const std::string&, const std::string&)
: MHAPlugin::plugin_t<example5_t>("example plugin configuration structure", iac),
  /* initializing variable 'scale_ch' with MHAParser::int_t(char* name, .... ) */
  scale_ch("channel number to be scaled", "0", "[0, [",
  /* initializing 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 function 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 configuration is invalid, then an exception of type **MHA_Error** (p. 387) 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. 658) stores the configuration in a FiFo queue of runtime configurations. Once they are inserted in the FiFo, the **MHAPLugin::plugin_t** (p. 659) 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. 9) defines all ANSI-C interface functions and passes them to the corresponding C++ class member functions (partly defined by the **MHAPLugin::plugin_t** (p. 659) template class). All exceptions of type **MHA_Error** (p. 387) are caught and transformed into an appropriate error code and error message.

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

3.3.6 example6.cpp

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

In the constructor of the plugin class the variable `rmsdb` is registered under the name `example6_rmslev` as a one-dimensional AC variable of type float. For registration of other types, read access and other detailed informations please see **Communication between algorithms** (p. 27).

```
example6_t::example6_t(const algo_comm_t& iac,
                      const std::string&, const std::string&)
: MHAPPlugin::plugin_t<cfg_t>("example plugin configuration structure", iac),
  /* initializing 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 function 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, then it is necessary to use different names for this
     * variable (i.e. prefixing the name with the algorithm name
     * passed to MHAInit).
     */
    ac.insert_var_float( ac.handle, "example6_rmslev", &rmsdb );
}
```

3.3.7 Debugging openMHA plugins

Suppose you would want to step through the code of your openMHA plugin with a debugger. This example details how to use the linux gdb debugger to inspect the `example6_t::prepare()` (p. 280) and `example6_t::process()` (p. 280) routines of `example6.cpp` (p. 23) 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. 280)) 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) 2005-2017 HoerTech gGmbH, D-26129 Oldenburg, Germany
```

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

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

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

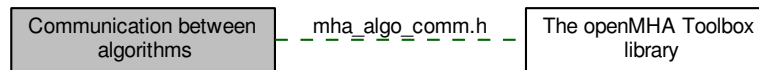
```
Breakpoint 2, example6_t::process (this=0x7ffff6a06c0d, wave=0x10a8b550)
    at example6.cpp:162
162     {
(gdb) n
163         poll_config();
(gdb)
```


3.4 The MHA Framework interface

3.5 Communication between algorithms

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

Collaboration diagram for Communication between algorithms:



Files

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

Namespaces

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

Classes

- class **MHA_AC::spectrum_t**
*Insert a **MHASignal::spectrum_t** (p. 731) class into the AC space.*
- class **MHA_AC::waveform_t**
*Insert a **MHASignal::waveform_t** (p. 743) class into the AC space.*
- class **MHA_AC::int_t**
Insert a integer variable into the AC space.
- class **MHA_AC::float_t**
Insert a float point variable into the AC space.
- class **MHA_AC::double_t**
Insert a double precision floating point variable into the AC space.
- class **MHA_AC::ac2matrix_t**
Copy AC variable to a matrix.
- class **MHA_AC::acspace2matrix_t**
Copy all or a subset of all numeric AC variables into an array of matrixes.
- struct **algo_comm_t**
A reference handle for algorithm communication variables.
- struct **comm_var_t**
Algorithm communication variable structure.

Functions

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

3.5.1 Detailed Description

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

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

An algorithm communication handle (**algo_comm_t** (p. 171)) is passed at initialisation time to the constructor of each plugin class **constructor** (p. 659). This handle contains a reference handle, **algo_comm_t::handle** (p. 171), and a number of function pointers, **algo_comm_t::insert_var** (p. 171) etc.. An algorithm communication variable is an object of type **comm_var_t** (p. 209).

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

3.5.2 Function Documentation

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

Convert an AC variable into a spectrum.

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

Parameters

| | |
|-------------|----------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of the variable |

Returns

Spectrum structure

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

Convert an AC variable into a waveform.

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

Parameters

| | |
|-------------|----------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of the variable |

Returns

waveform structure

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

Return value of an integer scalar AC variable.

Parameters

| | |
|-------------|----------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of the variable |

Returns

Variable value

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

Return value of an floating point scalar AC variable.

Parameters

| | |
|-------------|----------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of the variable |

Returns

Variable value

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

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

Parameters

| | |
|-------------|----------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of the variable |

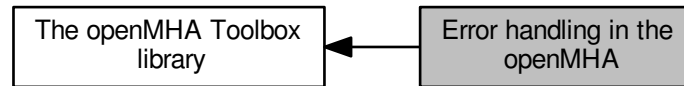
Returns

Variable value

3.6 Error handling in the openMHA

Errors are reported to the user via the **MHA_Error** (p. 387) 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. 387).*
- #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. 387) with the values.*

Functions

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

3.6.1 Detailed Description

Errors are reported to the user via the **MHA_Error** (p. 387) exception.

3.6.2 Macro Definition Documentation

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

Throw an openMHA error with a text message.

Parameters

| | |
|----------|---------------|
| <i>x</i> | Text message. |
|----------|---------------|

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

Assertion macro, which throws an **MHA_Error** (p. 387).

Parameters

| | |
|----------|--|
| <i>x</i> | Boolean expression which should be true. |
|----------|--|

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

Equality assertion macro, which throws an **MHA_Error** (p. 387) with the values.

Parameters

| | |
|----------|---|
| <i>a</i> | Numeric expression which can be converted to double (for printing). |
| <i>b</i> | Numeric expression which should be equal to <i>a</i> |

3.6.3 Function Documentation

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

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

3.7 The openMHA configuration language

openMHA Plugins that should use the openMHA configuration language for their configuration have to be implemented in C++ and need to include **mha_parser.hh** (p. 925).

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. 925).

All required classes and functions for parser access are declared in the namespace **MHAParser** (p. 103). The plugin class should be derived from the class **MHAParser::parser_t** (p. 620) (or **MHAParser::plugin_t** (p. 659)), 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. 622).

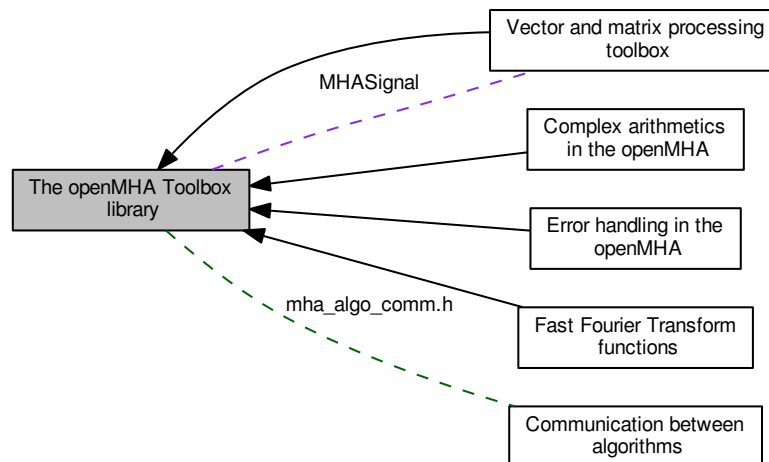
The openMHA Plugin template class **MHAParser::plugin_t** (p. 659) together with the Plugin macro **MHAPLUGIN_CALLBACKS** (p. 9) provide the callback mappings and correct inheritance. If your plugin is based on that template class, you simply have to use the **insert_item** command to give access to your variables, everything else is managed internally.

A complete list of all openMHA script items is given in the description of the **MHAParser** (p. 103) namespace.

3.8 The openMHA Toolbox library

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

Collaboration diagram for The openMHA Toolbox library:



Modules

- **Error handling in the openMHA**

*Errors are reported to the user via the **MHA_Error** (p. 387) exception.*

- **Vector and matrix processing toolbox**

*The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 113), and many functions and operators for use with the structures **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406).*

- **Complex arithmetics in the openMHA**

- **Fast Fourier Transform functions**

Files

- file **mha_algo_comm.h**

Header file for Algorithm Communication.

- file **mha_filter.hh**

Header file for IIR filter classes.

- file **mha_signal.hh**

Header file for audio signal handling and processing classes.

- file **mha_tablelookup.hh**

Header file for table lookup classes.

Namespaces

- **MHAOvIFilter**
Namespace for overlapping FFT based filter bank classes and functions.
- **MHAFilter**
Namespace for IIR and FIR filter classes.
- **MHAParser**
Name space for the openMHA-Parser configuration language.
- **MHASignal**
Namespace for audio signal handling and processing classes.
- **MHATableLookup**
Namespace for table lookup classes.

3.8.1 Detailed Description

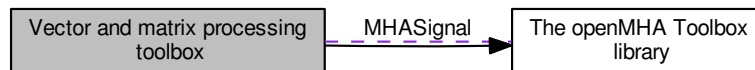
The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins.

It contains the openMHA script language classes.

3.9 Vector and matrix processing toolbox

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 113), and many functions and operators for use with the structures **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406).

Collaboration diagram for Vector and matrix processing toolbox:



Namespaces

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

Classes

- struct **mha_wave_t**
Waveform signal structure.
- struct **mha_spec_t**
Spectrum signal structure.
- struct **mha_audio_descriptor_t**
*Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 444)).*
- struct **mha_audio_t**
*An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406)).*
- class **MHASignal::spectrum_t**
*a signal processing class for spectral data (based on **mha_spec_t** (p. 406))*
- class **MHASignal::waveform_t**
*signal processing class for waveform data (based on **mha_wave_t** (p. 436))*
- class **MHASignal::doublebuffer_t**
Double-buffering class.
- class **MHASignal::hilbert_t**
Hilbert transformation of a waveform segment.
- class **MHASignal::minphase_t**
Minimal phase function.
- class **MHASignal::uint_vector_t**

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

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

Typedefs

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

Functions

- **mha_wave_t range** (**mha_wave_t** s, unsigned int k0, unsigned int len)
Return a time interval from a waveform chunk.
- **mha_spec_t channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)
Return a channel interval from a spectrum.
- void **MHASignal::for_each** (**mha_wave_t** *s, **mha_real_t**(*fun)(**mha_real_t**))
*Apply a function to each element of a **mha_wave_t** (p. 436).*
- **mha_real_t MHASignal::lin2db** (**mha_real_t** x)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t MHASignal::db2lin** (**mha_real_t** x)
Conversion from dB scale to linear (no SPL reference)
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t MHASignal::pa2dbspl** (**mha_real_t** x, **mha_real_t** eps=1e-20f)
Conversion from squared Pascal scale to dB SPL.
- **mha_real_t MHASignal::dbspl2pa** (**mha_real_t** x)
Conversion from dB SPL to linear Pascal scale.
- **mha_real_t MHASignal::smp2sec** (**mha_real_t** n, **mha_real_t** srate)
conversion from samples to seconds
- **mha_real_t MHASignal::sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
conversion from seconds to samples
- **mha_real_t MHASignal::bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)
conversion from fft bin index to frequency
- **mha_real_t MHASignal::freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)
conversion from frequency to fft bin index
- **mha_real_t MHASignal::smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
conversion from delay in samples to phase shift

- **mha_real_t MHASignal::rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)
conversion from phase shift to delay in samples
- template<class elem_type >
std::vector< elem_type > **MHASignal::dupvec** (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size.
- template<class elem_type >
std::vector< elem_type > **MHASignal::dupvec_chk** (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size, check for dimension.
- bool **equal_dim** (const **mha_wave_t** &a, const **mha_wave_t** &b)
Test for equal dimension of waveform structures.
- bool **equal_dim** (const **mha_wave_t** &a, const **mhaconfig_t** &b)
Test for match of waveform dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_spec_t** &b)
Test for equal dimension of spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mhaconfig_t** &b)
Test for match of spectrum dimension with mhaconfig structure.
- bool **equal_dim** (const **mha_wave_t** &a, const **mha_spec_t** &b)
Test for equal dimension of waveform/spectrum structures.
- bool **equal_dim** (const **mha_spec_t** &a, const **mha_wave_t** &b)
Test for equal dimension of waveform/spectrum structures.
- void **integrate** (**mha_wave_t** &s)
Numeric integration of a signal vector (real values)
- void **integrate** (**mha_spec_t** &s)
Numeric integration of a signal vector (complex values)
- unsigned int **size** (const **mha_wave_t** &s)
Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** &s)
Return size of a spectrum structure.
- unsigned int **size** (const **mha_wave_t** *s)
Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** *s)
Return size of a spectrum structure.
- void **clear** (**mha_wave_t** &s)
Set all values of waveform to zero.
- void **clear** (**mha_wave_t** *s)
Set all values of waveform to zero.
- void **clear** (**mha_spec_t** &s)
Set all values of spectrum to zero.
- void **clear** (**mha_spec_t** *s)
Set all values of spectrum to zero.
- void **assign** (**mha_wave_t** self, **mha_real_t** val)
Set all values of waveform 'self' to 'val'.

- void **assign** (**mha_wave_t** self, const **mha_wave_t** &val)
Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_spec_t** self, const **mha_spec_t** &val)
Set all values of spectrum 'self' to 'val'.
- void **timeshift** (**mha_wave_t** &self, int shift)
Time shift of waveform chunk.
- **mha_real_t** & **value** (**mha_wave_t** *s, unsigned int fr, unsigned int ch)
Access an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** *s, unsigned int fr, unsigned int ch)
Constant access to an element of a waveform structure.
- **mha_complex_t** & **value** (**mha_spec_t** *s, unsigned int fr, unsigned int ch)
Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** *s, unsigned int fr, unsigned int ch)
Constant access to an element of a spectrum.
- **mha_real_t** & **value** (**mha_wave_t** &s, unsigned int fr, unsigned int ch)
Access to an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** &s, unsigned int fr, unsigned int ch)
Constant access to an element of a waveform structure.
- **mha_complex_t** & **value** (**mha_spec_t** &s, unsigned int fr, unsigned int ch)
Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** &s, unsigned int fr, unsigned int ch)
Constant access to an element of a spectrum.
- std::vector< float > **std_vector_float** (const **mha_wave_t** &)
*Converts a **mha_wave_t** (p. 436) 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. 436) 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. 406) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).*
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_real_t** &)
Addition operator.
- **mha_wave_t** & **operator+=** (**mha_wave_t** &, const **mha_wave_t** &)
Addition operator.
- **mha_wave_t** & **operator-=** (**mha_wave_t** &, const **mha_wave_t** &)
Subtraction operator.
- **mha_spec_t** & **operator-=** (**mha_spec_t** &, const **mha_spec_t** &)
Subtraction operator.
- **mha_wave_t** & **operator*=** (**mha_wave_t** &, const **mha_real_t** &)
Element-wise multiplication operator.
- **mha_wave_t** & **operator*=** (**mha_wave_t** &, const **mha_wave_t** &)
Element-wise multiplication operator.
- **mha_spec_t** & **operator*=** (**mha_spec_t** &, const **mha_real_t** &)
Element-wise multiplication operator.

- **mha_spec_t & operator*= (mha_spec_t &, const mha_wave_t &)**
Element-wise multiplication operator.
- **mha_spec_t & operator*= (mha_spec_t &, const mha_spec_t &)**
Element-wise multiplication operator.
- **mha_spec_t & operator/= (mha_spec_t &, const mha_spec_t &)**
Element-wise division operator.
- **mha_wave_t & operator/= (mha_wave_t &, const mha_wave_t &)**
Element-wise division operator.
- **mha_spec_t & operator+= (mha_spec_t &, const mha_spec_t &)**
Addition operator.
- **mha_spec_t & operator+= (mha_spec_t &, const mha_real_t &)**
Addition operator.
- **mha_wave_t & operator^= (mha_wave_t &self, const mha_real_t &arg)**
Exponent operator.
- **void MHASignal::copy_channel (mha_spec_t &self, const mha_spec_t &src, unsigned sch, unsigned dch)**
Copy one channel of a source signal.
- **void MHASignal::copy_channel (mha_wave_t &self, const mha_wave_t &src, unsigned src_channel, unsigned dest_channel)**
Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel (const mha_spec_t &s, unsigned int channel, unsigned int fftlen)**
Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity (const mha_spec_t &s, unsigned int channel, unsigned int fftlen, mha_real_t sqfreq_response[])**
Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs (const mha_spec_t &s, unsigned int channel)**
Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel (const mha_wave_t &s, unsigned int channel)**
Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs (const mha_wave_t &s, unsigned int channel)**
Find maximal absolute value.
- **mha_real_t MHASignal::maxabs (const mha_wave_t &s)**
Find maximal absolute value.
- **mha_real_t MHASignal::max (const mha_wave_t &s)**
Find maximal value.
- **mha_real_t MHASignal::min (const mha_wave_t &s)**
Find minimal value.
- **mha_real_t MHASignal::sumsq_channel (const mha_wave_t &s, unsigned int channel)**
Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsq_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. 406) with its conjugate.*

3.9.1 Detailed Description

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 113), and many functions and operators for use with the structures **mha_↔_wave_t** (p. 436) and **mha_spec_t** (p. 406).

3.9.2 Typedef Documentation

3.9.2.1 typedef float mha_real_t

openMHA type for real numbers

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

3.9.3 Function Documentation

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

Return a time interval from a waveform chunk.

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

Parameters

| | |
|------------|--------------------------------|
| <i>s</i> | Waveform structure |
| <i>k0</i> | Index of first value in output |
| <i>len</i> | Number of frames in output |

Returns

Waveform structure representing the sub-interval.

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

Return a channel interval from a spectrum.

Parameters

| | |
|-----------------|----------------------------------|
| <i>s</i> | Input spectrum |
| <i>ch_start</i> | Index of first channel in output |
| <i>nch</i> | Number of channels in output |

Returns

Spectrum structure representing the sub-interval.

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

Apply a function to each element of a **mha_wave_t** (p. 436).

Parameters

| | |
|------------|---|
| <i>s</i> | Pointer to a mha_wave_t (p. 436) structure |
| <i>fun</i> | Function to be applied (one argument) |

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

Conversion from linear scale to dB (no SPL reference)

Parameters

| | |
|----------|---------------|
| <i>x</i> | Linear input. |
|----------|---------------|

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

Conversion from dB scale to linear (no SPL reference)

Parameters

| | |
|----------|-----------|
| <i>x</i> | dB input. |
|----------|-----------|

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

Conversion from linear Pascal scale to dB SPL.

Parameters

| | |
|----------|---------------|
| <i>x</i> | Linear input. |
|----------|---------------|

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

Conversion from squared Pascal scale to dB SPL.

Parameters

| | |
|------------|------------------------------|
| <i>x</i> | squared pascal input |
| <i>eps</i> | minimum squared-pascal value |

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

Conversion from dB SPL to linear Pascal scale.

Parameters

| | |
|----------|---------------|
| <i>x</i> | Linear input. |
|----------|---------------|

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

conversion from samples to seconds

Parameters

| | |
|--------------|--------------------|
| <i>n</i> | number of samples |
| <i>srate</i> | sampling rate / Hz |

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

conversion from seconds to samples

Parameters

| | |
|--------------|--------------------|
| <i>sec</i> | time in seconds |
| <i>srate</i> | sampling rate / Hz |

Returns

number of samples, generally has non-zero fractional part

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

conversion from fft bin index to frequency

Parameters

| | |
|---------------|----------------------------------|
| <i>bin</i> | index of fft bin, index 0 has dc |
| <i>fftlen</i> | FFT length |
| <i>srate</i> | sampling frequency / Hz |

Returns

frequency of fft bin / Hz

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

conversion from frequency to fft bin index

Parameters

| | |
|---------------|-------------------------|
| <i>freq</i> | frequency / Hz |
| <i>fftlen</i> | FFT length |
| <i>srate</i> | sampling frequency / Hz |

Returns

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

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

conversion from delay in samples to phase shift

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

Parameters

| | |
|----------------|--|
| <i>samples</i> | delay in samples. Positive delay: shift current signal to future. |
| <i>bin</i> | index of fft bin, index 0 has dc (index 0 and nyquist bin cannot be delayed) |
| <i>fftlen</i> | 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 π should not be used. If bin is 0 or nyquist, returns 0 phase shift.

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

conversion from phase shift to delay in samples

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

Parameters

| | |
|--------------------|--|
| <i>phase_shift</i> | phase shift in radiant |
| <i>bin</i> | index of fft bin, index 0 has dc (index 0 and nyquist bin cannot be delayed) |
| <i>fftlen</i> | FFT length |

Returns

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

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

Duplicate last vector element to match desired size.

Parameters

| | |
|------------|----------------------------|
| <i>vec</i> | Input vector. |
| <i>n</i> | Target number of elements. |

Return values

| | |
|----------------|---------|
| <i>Resized</i> | vector. |
|----------------|---------|

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

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

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

Parameters

| | |
|------------|----------------------------|
| <i>vec</i> | Input vector. |
| <i>n</i> | Target number of elements. |

Return values

| | |
|----------------|---------|
| <i>Resized</i> | vector. |
|----------------|---------|

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

Test for equal dimension of waveform structures.

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

Test for match of waveform dimension with mhaconfig structure.

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

Test for equal dimension of spectrum structures.

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

Test for match of spectrum dimension with mhaconfig structure.

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

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 436) use interleaved data order, while spectrum structures **mha_spec_t** (p. 406) use non-interleaved.

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

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 436) use interleaved data order, while spectrum structures **mha_spec_t** (p. 406) use non-interleaved.

```
3.9.3.23 void integrate (
    mha_wave_t & s )
```

Numeric integration of a signal vector (real values)

Parameters

| | |
|---|---------------------|
| s | Input signal vector |
|---|---------------------|

```
3.9.3.24 void integrate (
    mha_spec_t & s )
```

Numeric integration of a signal vector (complex values)

Parameters

| | |
|---|---------------------|
| s | Input signal vector |
|---|---------------------|

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

3.9.3.29 void clear (
 mha_wave_t & s) [inline]

Set all values of waveform to zero.

3.9.3.30 void clear (
 mha_wave_t * s) [inline]

Set all values of waveform to zero.

3.9.3.31 void clear (
 mha_spec_t & s) [inline]

Set all values of spectrum to zero.

3.9.3.32 void clear (
 mha_spec_t * s) [inline]

Set all values of spectrum to zero.

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

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

Parameters

| | |
|-------------|--|
| <i>self</i> | Waveform to be modified. |
| <i>val</i> | Value to be assigned to all entries of waveform. |

```
3.9.3.34 void assign (
            mha_wave_t self,
            const mha_wave_t & val )
```

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

Parameters

| | |
|-------------|----------------------------|
| <i>self</i> | Waveform to be modified. |
| <i>val</i> | Source waveform structure. |

```
3.9.3.35 void assign (
            mha_spec_t self,
            const mha_spec_t & val )
```

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

Parameters

| | |
|-------------|--------------------------|
| <i>self</i> | Spectrum to be modified. |
| <i>val</i> | Source spectrum. |

```
3.9.3.36 void timeshift (
            mha_wave_t & self,
            int shift )
```

Time shift of waveform chunk.

Shifted areas are filled with zeros.

Parameters

| | |
|--------------|--|
| <i>self</i> | Waveform chunk to be shifted |
| <i>shift</i> | Shift amount, positive values shift to later times |

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

Access an element of a waveform structure.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Waveform structure |
| <i>fr</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Constant access to an element of a waveform structure.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Waveform structure |
| <i>fr</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Access to an element of a spectrum.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Spectrum structure |
| <i>fr</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Constant access to an element of a spectrum.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Spectrum structure |
| <i>fr</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Access to an element of a waveform structure.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Waveform structure |
| <i>fr</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Constant access to an element of a waveform structure.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Waveform structure |
| <i>fr</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Access to an element of a spectrum.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Spectrum structure |
| <i>fr</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Constant access to an element of a spectrum.

Parameters

| | |
|-----------|--------------------|
| <i>s</i> | Spectrum structure |
| <i>fr</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

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

Converts a **mha_wave_t** (p. 436) structure into a `std::vector<float>` (interleaved order).

Warning

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

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

Converts a **mha_wave_t** (p. 436) structure into a `std::vector< std::vector<float> >` (outer vector represents channels).

Warning

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

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

Converts a **mha_spec_t** (p. 406) structure into a `std::vector< std::vector<mha_complex_t> >` (outer vector represents channels).

Warning

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

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

Addition operator.

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

Addition operator.

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

Subtraction operator.

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

Subtraction operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

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

Element-wise division operator.

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

Element-wise division operator.

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

Addition operator.

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

Addition operator.

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

Exponent operator.

Warning

This overwrites the xor operator!

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

Copy one channel of a source signal.

Parameters

| | |
|-------------|----------------------------|
| <i>self</i> | Destination. |
| <i>src</i> | Source |
| <i>sch</i> | Source channel number |
| <i>dch</i> | Destination channel number |

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

Copy one channel of a source signal.

Parameters

| | |
|---------------------|----------------------------|
| <i>self</i> | Destination. |
| <i>src</i> | Source |
| <i>src_channel</i> | Source channel number |
| <i>dest_channel</i> | Destination channel number |

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

Return RMS level of a spectrum channel.

Parameters

| | |
|----------------|--|
| <i>s</i> | Input spectrum |
| <i>channel</i> | Channel number to be tested |
| <i>fftlen</i> | FFT length (to correctly count the level of the Nyquist bin) |

Returns

RMS level in Pa

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

Colored spectrum intensity.

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

Parameters

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

Returns

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

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

Find maximal absolute value.

Parameters

| | |
|----------------|----------------------|
| <i>s</i> | Input signal |
| <i>channel</i> | Channel to be tested |

Returns

maximum absolute value

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

Return RMS level of a waveform channel.

Parameters

| | |
|----------------|-----------------------------|
| <i>s</i> | Input waveform signal |
| <i>channel</i> | Channel number to be tested |

Returns

RMS level in Pa

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

Find maximal absolute value.

Parameters

| | |
|----------------|----------------------|
| <i>s</i> | Input signal |
| <i>channel</i> | Channel to be tested |

Returns

maximum absolute value

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

Find maximal absolute value.

Parameters

| | |
|----------|--------------|
| <i>s</i> | Input signal |
|----------|--------------|

Returns

maximum absolute value

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

Find maximal value.

Parameters

| | |
|----------|--------------|
| <i>s</i> | Input signal |
|----------|--------------|

Returns

maximum absolute value

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

Find minimal value.

Parameters

| | |
|----------|--------------|
| <i>s</i> | Input signal |
|----------|--------------|

Returns

maximum absolute value

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

Calculate sum of squared values in one channel.

Parameters

| | |
|----------------|--------------|
| <i>s</i> | Input signal |
| <i>channel</i> | Channel |

Returns

$$\sum x^2$$

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

Calculate sum over all channels of squared values.

Parameters

| | |
|--------------|--------------|
| <i>s</i> | Input signal |
| <i>frame</i> | Frame number |

Returns

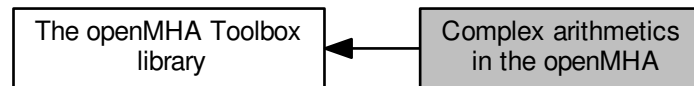
$$\sum x^2$$

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

Replace (!) the value of this **mha_spec_t** (p. 406) with its conjugate.

3.10 Complex arithmetics in the openMHA

Collaboration diagram for Complex arithmetics in the openMHA:



Classes

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

Functions

- **mha_complex_t & set (mha_complex_t &self, mha_real_t real, mha_real_t imag=0)**
*Assign real and imaginary parts to a **mha_complex_t** (p. 374) variable.*
- **mha_complex_t mha_complex (mha_real_t real, mha_real_t imag=0)**
*Create a new **mha_complex_t** (p. 374) 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. 374) 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. 374).*
- **mha_complex_t & expi (mha_complex_t &self, mha_real_t angle)**
*replaces the value of the given **mha_complex_t** (p. 374) 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. 374) 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. 374) with its conjugate.*
- **mha_complex_t _conjugate** (const **mha_complex_t** &self)
Compute the conjugate of this complex value.
- **void reciprocal** (**mha_complex_t** &self)

Replace the value of this complex with its reciprocal.

- **mha_complex_t _reciprocal** (const **mha_complex_t** &self)
compute the reciprocal of this complex value.
- void **normalize** (**mha_complex_t** &self)
Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle).
- void **normalize** (**mha_complex_t** &self, **mha_real_t** margin)
Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle), with a safety margin.
- bool **almost** (const **mha_complex_t** &self, const **mha_complex_t** &other, **mha_real_t** times_epsilon=1e2)
Compare two complex numbers for equality except for a small relative error.
- bool **operator<** (const **mha_complex_t** &x, const **mha_complex_t** &y)
Compares the absolute values of two complex numbers.

3.10.1 Detailed Description

3.10.2 Function Documentation

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

Assign real and imaginary parts to a **mha_complex_t** (p. 374) variable.

Parameters

| | |
|-------------|--|
| <i>self</i> | The mha_complex_t (p. 374) variable whose value is about to change. |
| <i>real</i> | The new real part. |
| <i>imag</i> | The new imaginary part. |

Returns

A reference to the changed variable.

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

Create a new **mha_complex_t** (p. 374) with specified real and imaginary parts.

Parameters

| | |
|-------------|---------------------|
| <i>real</i> | The real part. |
| <i>imag</i> | The imaginary part. |

Returns

The new value.

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

Assign a **mha_complex_t** (p. 374) variable from a `std::complex`.

Parameters

| | |
|-------------------|--|
| <i>self</i> | The mha_complex_t (p. 374) variable whose value is about to change. |
| <i>stdcomplex</i> | The new complex value. |

Returns

A reference to the changed variable.

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

Create a `std::complex` from **mha_complex_t** (p. 374).

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

replaces the value of the given **mha_complex_t** (p. 374) with $\exp(i*b)$.

Parameters

| | |
|--------------|--|
| <i>self</i> | The mha_complex_t (p. 374) variable whose value is about to change. |
| <i>angle</i> | The angle in the complex plane [rad]. |

Returns

A reference to the changed variable.

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

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

Parameters

| | |
|-------------|---|
| <i>self</i> | The complex number whose angle is needed. |
|-------------|---|

Returns

The angle of a complex number in the complex plane.

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

Addition of two complex numbers, overwriting the first.

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

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

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

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

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

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

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

Subtraction of two complex numbers, overwriting the first.

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

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

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

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

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

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

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

Multiplication of two complex numbers, overwriting the first.

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

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

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

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

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

replaces (!) the value of the given **mha_complex_t** (p. 374) with $a * \exp(i*b)$

Parameters

| | |
|---------------|--|
| <i>self</i> | The mha_complex_t (p. 374) variable whose value is about to change. |
| <i>angle</i> | The imaginary exponent. |
| <i>factor</i> | The absolute value of the result. |

Returns

A reference to the changed variable.

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

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

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

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

Returns

The square of the absolute value of self.

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

Compute the absolute value of a complex value.

Returns

The absolute value of self.

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

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

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

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

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

Safe division of two complex numbers, overwriting the first.

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

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

Division of two complex numbers, overwriting the first.

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

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

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

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

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

Compare two complex numbers for equality.

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

Compare two complex numbers for inequality.

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

Replace (!) the value of this **mha_complex_t** (p. [374](#)) with its conjugate.

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

Compute the conjugate of this complex value.

Returns

A temporary object holding the conjugate value.

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

Replace the value of this complex with its reciprocal.

3.10.2.33 `mha_complex_t_reciprocal (`
`const mha_complex_t & self) [inline]`

compute the reciprocal of this complex value.

Returns

A temporary object holding the reciprocal value.

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

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

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

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

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

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

Parameters

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

Returns

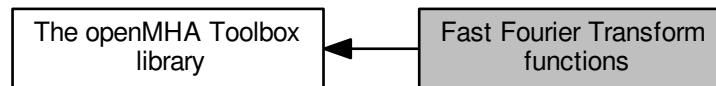
true if the relative difference is below `times_epsilon * std::numeric_limits<mha_real_t>::epsilon`

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

Compares the absolute values of two complex numbers.

3.11 Fast Fourier Transform functions

Collaboration diagram for Fast Fourier Transform functions:



Typedefs

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

Functions

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

3.11.1 Detailed Description

3.11.2 Typedef Documentation

3.11.2.1 typedef void* mha_fft_t

Handle for an FFT object.

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

3.11.3 Function Documentation

3.11.3.1 mha_fft_t mha_fft_new (unsigned int *n*)

Create a new FFT handle.

Parameters

| | |
|-----|-------------|
| n | FFT length. |
|-----|-------------|

Create a new FFT handle.

Parameters

| | |
|-----|------------|
| n | FFT length |
|-----|------------|

Return values

| | |
|------------|--------|
| <i>FFT</i> | object |
|------------|--------|

```
3.11.3.2 void mha_fft_free (
                mha_fft_t h )
```

Destroy an FFT handle.

Parameters

| | |
|-----|-------------------------|
| h | Handle to be destroyed. |
|-----|-------------------------|

Destroy an FFT handle.

Parameters

| | |
|----------|--------------------------|
| <i>h</i> | FFT object to be removed |
|----------|--------------------------|

```

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

```

Tranform waveform segment into spectrum.

Parameters

| | |
|------------|-------------------------|
| <i>h</i> | FFT handle. |
| <i>in</i> | Input waveform segment. |
| <i>out</i> | Output spectrum. |

Tranform waveform segment into spectrum.

Parameters

| | |
|------------|---|
| <i>h</i> | FFT object handle |
| <i>in</i> | pointer to input waveform signal |
| <i>out</i> | pointer to output spectrum signal (has to be allocated) |

```

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

```

Tranform waveform segment into spectrum.

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

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

Parameters

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

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

Tranform spectrum into waveform segment.

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

Parameters

| | |
|------------|--------------------------|
| <i>h</i> | FFT handle. |
| <i>in</i> | Input spectrum. |
| <i>out</i> | Output waveform segment. |

Tranform spectrum into waveform segment.

Parameters

| | |
|------------|---|
| <i>h</i> | FFT object handle |
| <i>in</i> | pointer to input spectrum |
| <i>out</i> | pointer to output waveform signal (has to be allocated) |

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

Tranform spectrum into waveform segment.

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

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

Parameters

| | |
|---------------|------------------------------|
| <i>h</i> | FFT handle. |
| <i>in</i> | Input spectrum. |
| <i>out</i> | Output waveform segment. |
| <i>offset</i> | Offset into iFFT wave buffer |

Transform spectrum into waveform segment.

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

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 of the complete iFFT.

Parameters

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

```
3.11.3.7 void mha_fft_forward (
    mha_fft_t h,
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

Complex to complex FFT (forward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 406) 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

| | |
|-------------|------------------|
| <i>h</i> | FFT handle. |
| <i>sIn</i> | Input spectrum. |
| <i>sOut</i> | Output spectrum. |

```
3.11.3.8 void mha_fft_backward (
    mha_fft_t h,
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

Complex to complex FFT (backward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 406) 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

| | |
|-------------|------------------|
| <i>h</i> | FFT handle. |
| <i>sIn</i> | Input spectrum. |
| <i>sOut</i> | Output spectrum. |

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

Complex to complex FFT (forward).

sIn and *sOut* need to have *nfft* bins (please note that **mha_spec_t** (p. 406) 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

| | |
|-------------|------------------|
| <i>h</i> | FFT handle. |
| <i>sIn</i> | Input spectrum. |
| <i>sOut</i> | Output spectrum. |

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

Complex to complex FFT (backward).

sIn and *sOut* need to have *nfft* bins (please note that **mha_spec_t** (p. 406) 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

| | |
|-------------|------------------|
| <i>h</i> | FFT handle. |
| <i>sIn</i> | Input spectrum. |
| <i>sOut</i> | Output spectrum. |

```

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

```

Tranform waveform segment into spectrum.

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

Parameters

| | |
|------------|-------------------------|
| <i>h</i> | FFT handle. |
| <i>in</i> | Input waveform segment. |
| <i>out</i> | Output spectrum. |

```

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

```

Tranform spectrum into waveform segment.

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

Parameters

| | |
|------------|--------------------------|
| <i>h</i> | FFT handle. |
| <i>in</i> | Input spectrum. |
| <i>out</i> | Output waveform segment. |

4 Namespace Documentation

4.1 acmon Namespace Reference

Namespace for displaying ac variables as parser monitors.

Classes

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

4.1.1 Detailed Description

Namespace for displaying ac variables as parser monitors.

4.2 acsave Namespace Reference

Classes

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

4.3 ADM Namespace Reference

Classes

- class **ADM**
Adaptive differential microphone, working for speech frequency range.
- class **Delay**
A delay-line class which can also do subsample-delays for a limited frequency range below $f_s/4$.
- class **Linearphase_FIR**
An efficient linear-phase fir filter implementation.

Functions

- static double **subsampledelay_coeff** (double samples, double f_design, double fs=1.0)
compute IIR coefficient for subsample delay

Variables

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

4.3.1 Function Documentation

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

compute IIR coefficient for subsample delay

Parameters

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

Returns

IIR coefficient for subsample delay

4.3.2 Variable Documentation

4.3.2.1 const double ADM::PI = 3.14159265358979312

4.3.2.2 const double ADM::C = 340

4.3.2.3 const double ADM::DELAY_FREQ = 2000

4.3.2.4 const double ADM::START_BETA = 0.5

4.4 AuditoryProfile Namespace Reference

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

Classes

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

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

4.5 coherence Namespace Reference

Classes

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

Functions

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

4.5.1 Function Documentation

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

4.6 dc Namespace Reference

Classes

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

Functions

- unsigned int **get_audiochannels** (unsigned int totalchannels, std::string acname, **algo_←_comm_t** ac)

4.6.1 Function Documentation

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

4.7 dc_simple Namespace Reference

Classes

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

Typedefs

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

Functions

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

4.7.1 Typedef Documentation

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

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

4.7.2 Function Documentation

4.7.2.1 void dc_simple::test_fail (

const std::vector< float > & v,

unsigned int s,

const std::string & name)

4.7.2.2 std::vector<float> dc_simple::force_resize (

const std::vector< float > & v,

unsigned int s,

const std::string & name)

4.7.2.3 mha_real_t dc_simple::not_zero (

mha_real_t x,

const std::string & comment = " ")

4.8 delay Namespace Reference

Classes

- class **interface_t**

4.9 delaysum Namespace Reference

This namespace contains the delaysum plugin.

Classes

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

4.9.1 Detailed Description

This namespace contains the delaysum plugin.

4.10 DynComp Namespace Reference

dynamic compression related classes and functions

Classes

- class **dc_afterburn_rt_t**
Real-time class for after burn effect.
- class **dc_afterburn_t**
Afterburn class, to be defined as a member of compressors.
- class **dc_afterburn_vars_t**
*Variables for **dc_afterburn_t** (p. 257) class.*
- class **gaintable_t**
Gain table class.

Functions

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

4.10.1 Detailed Description

dynamic compression related classes and functions

4.10.2 Function Documentation

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

One-dimensional linear interpolation.

Parameters

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

Return values

| | |
|---------------------|---------------------------|
| <i>Interpolated</i> | value Y(X) at position X. |
|---------------------|---------------------------|

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

Linear interpolation in a two-dimensional field.

Parameters

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

Return values

| | |
|---------------------|-------------------------------|
| <i>Interpolated</i> | value Z(X,Y) at position X,Y. |
|---------------------|-------------------------------|

4.11 fader_wave Namespace Reference

Classes

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

Typedefs

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

4.11.1 Typedef Documentation

4.11.1.1 typedef MHAPLugin::plugin_t<level_adapt_t> fader_wave::level_adaptor

4.12 fftfilterbank Namespace Reference

Classes

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

4.13 gain Namespace Reference

Classes

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

4.14 matrixmixer Namespace Reference

Classes

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

4.15 MHA_AC Namespace Reference

Functions and classes for Algorithm Communication (AC) support.

Classes

- class **ac2matrix_helper_t**
- class **ac2matrix_t**
Copy AC variable to a matrix.
- class **acspace2matrix_t**
Copy all or a subset of all numeric AC variables into an array of matrixes.
- class **double_t**
Insert a double precision floating point variable into the AC space.
- class **float_t**
Insert a float point variable into the AC space.
- class **int_t**
Insert a integer variable into the AC space.
- class **spectrum_t**
*Insert a **MHASignal::spectrum_t** (p. 731) class into the AC space.*
- class **stat_t**
- class **waveform_t**
*Insert a **MHASignal::waveform_t** (p. 743) class into the AC space.*

Functions

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

4.15.1 Detailed Description

Functions and classes for Algorithm Communication (AC) support.

4.16 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.

4.16.1 Function Documentation

4.16.1.1 unsigned mha_error_helpers::digits (
unsigned n)

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

Parameters

| | |
|----------|---|
| <i>n</i> | The unsigned integer that we want to know the number of required decimal digits for. return The number of decimal digits in <i>n</i> . |
|----------|---|

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

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

Parameters

| | |
|---------------------|---|
| <i>formatstring</i> | The format string with standard printf formatstring |
|---------------------|---|

Returns

the number of bytes required by printf without the terminating nul

4.17 MHA_TCP Namespace Reference

A Namespace for TCP helper classes.

Classes

- class **Async_Notify**
 Portable Multiplexable cross-thread notification.
- class **Client**
 A portable class for a tcp client connections.
- class **Connection**
 Connection (p. 411) handles Communication between client and server, is used on both sides.
- class **Event_Watcher**
 OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.
- struct **OS_EVENT_TYPE**
- class **Server**
- class **Sockaccept_Event**
- class **Sockread_Event**
 Watch socket for incoming data.
- class **Sockwrite_Event**
- class **Thread**
 A very simple class for portable threads.
- class **Timeout_Event**
- class **Timeout_Watcher**
 OS-independent event watcher with internal fixed-end-time timeout.
- class **Wakeup_Event**
 A base class for asynchronous wakeup events.

Typedefs

- typedef int **SOCKET**

Functions

- std::string **STRERROR** (int err)
Portable conversion from error number to error string.
- std::string **HSTRERROR** (int err)
Portable conversion from hostname error number to error string.
- int **N_ERRNO** ()
Portable access to last network error number.
- int **H_ERRNO** ()
Portable access to last hostname error number.
- int **G_ERRNO** ()
Portable access to last non-network error number.
- double **dtime** ()
Time access function for system's high resolution time, retrieve current time as double.
- double **dtime** (const struct timeval &tv)
Time access function for unix' high resolution time, converts struct timeval to double.
- struct timeval **stime** (double d)
Time access function for unix' high resolution time, converts time from double to struct timeval.

4.17.1 Detailed Description

A Namespace for TCP helper classes.

4.17.2 Typedef Documentation

4.17.2.1 typedef int MHA_TCP::SOCKET

4.17.3 Function Documentation

4.17.3.1 std::string MHA_TCP::STRERROR (int err)

Portable conversion from error number to error string.

4.17.3.2 std::string MHA_TCP::HSTRERROR (int err)

Portable conversion from hostname error number to error string.

4.17.3.3 `int MHA_TCP::N_ERRNO ()`

Portable access to last network error number.

4.17.3.4 `int MHA_TCP::H_ERRNO ()`

Portable access to last hostname error number.

4.17.3.5 `int MHA_TCP::G_ERRNO ()`

Portable access to last non-network error number.

4.17.3.6 `double MHA_TCP::dtime ()`

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

4.17.3.7 `double MHA_TCP::dtime (` `const struct timeval & tv)`

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

4.17.3.8 `struct timeval MHA_TCP::stime (` `double d)`

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

4.18 `mhachain` Namespace Reference

Classes

- class `chain_base_t`
- class `mhachain_t`
- class `plugs_t`

4.19 `MHAEvents` Namespace Reference

Collection of event handling classes.

Classes

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

4.19.1 Detailed Description

Collection of event handling classes.

4.20 MHAFilter Namespace Reference

Namespace for IIR and FIR filter classes.

Classes

- class **adapt_filter_param_t**
- class **adapt_filter_state_t**
- class **adapt_filter_t**
Adaptive filter.
- class **blockprocessing_polyphase_resampling_t**
A class that does polyphase resampling and takes into account block processing.
- class **complex_bandpass_t**
Complex bandpass filter.
- class **diff_t**
Differentiator class (non-normalized)
- class **fftfilter_t**
FFT based FIR filter implementation.
- class **fftfilterbank_t**
FFT based FIR filterbank implementation.
- class **filter_t**
Generic IIR filter class.
- class **gammaflt_t**
Class for gammatone filter.
- class **iir_filter_state_t**
- class **iir_filter_t**
IIR filter class wrapper for integration into parser structure.
- class **iir_ord1_real_t**
First order recursive filter.
- class **o1_ar_filter_t**

- First order attack-release lowpass filter.*
- class **o1flt_lowpass_t**
First order low pass filter.
- class **o1flt_maxtrack_t**
First order maximum tracker.
- class **o1flt_mintrack_t**
First order minimum tracker.
- class **partitioned_convolution_t**
A filter class for partitioned convolution.
- class **polyphase_resampling_t**
A class that performs polyphase resampling.
- class **resampling_filter_t**
Hann shaped low pass filter for resampling.
- class **smoothspec_t**
Smooth spectral gains, create a windowed impulse response.
- class **thirdoctave_analyzer_t**
- struct **transfer_function_t**
a structure containing a source channel number, a target channel number, and an impulse response.
- struct **transfer_matrix_t**
A sparse matrix of transfer function partitionss.

Functions

- void **make_friendly_number** (**mha_real_t** &x)
- void **make_friendly_number** (**mha_complex_t** &x)
- void **make_friendly_number** (double &x)
- void **o1_lp_coeffs** (const **mha_real_t** tau, const **mha_real_t** fs, **mha_real_t** &c1, **mha_real_t** &c2)
Set first order filter coefficients from time constant and sampling rate.
- void **butter_stop_ord1** (double *A, double *B, double f1, double f2, double fs)
Setup a first order butterworth band stop filter.
- **MHASignal::waveform_t** * **spec2fir** (const **mha_spec_t** *spec, const unsigned int fftlen, const **MHAWindow::base_t** &window, const bool minphase)
Create a windowed impulse response/FIR filter coefficients from a spectrum.
- unsigned **gcd** (unsigned a, unsigned b)
greatest common divisor
- double **sinc** (double x)
sin(x)/x function, coping with x=0.
- std::pair< unsigned, unsigned > **resampling_factors** (float source_sampling_rate, float target_sampling_rate, float factor=1.0f)
Computes rational resampling factor from two sampling rates.

4.20.1 Detailed Description

Namespace for IIR and FIR filter classes.

4.20.2 Function Documentation

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

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

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

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

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

Parameters

| | |
|------------|---------------|
| <i>tau</i> | Time constant |
| <i>fs</i> | Sampling rate |

Return values

| | |
|-----------|----------------------------------|
| <i>c1</i> | Recursive filter coefficient |
| <i>c2</i> | Non-recursive filter coefficient |

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

Setup a first order butterworth band stop filter.

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

Return values

| | |
|----------|-----------------------------------|
| <i>A</i> | recursive filter coefficients |
| <i>B</i> | non recursive filter coefficients |

Parameters

| | |
|-----------|------------------|
| <i>f1</i> | lower frequency |
| <i>f2</i> | upper frequency |
| <i>fs</i> | sample frequency |

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

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

Parameters

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

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

greatest common divisor

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

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

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

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

Computes rational resampling factor from two sampling rates.

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

Parameters

| | |
|-----------------------------|---|
| <i>source_sampling_rate</i> | The original sampling rate |
| <i>target_sampling_rate</i> | The desired sampling rate |
| <i>factor</i> | 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

| | |
|---|--|
| <i>MHA_Error</i> (p. 387) | if no rational resampling factor can be found. |
|---|--|

4.21 MHAIOJack Namespace Reference

JACK IO.

Classes

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

4.21.1 Detailed Description

JACK IO.

4.22 MHAIOPortAudio Namespace Reference

Classes

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

Functions

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

4.22.1 Function Documentation

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

4.23 MHAJack Namespace Reference

Classes and functions for openMHA and JACK interaction.

Classes

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

Functions

- void **io** (**mha_wave_t** *s_out, **mha_wave_t** *s_in, const std::string &name, const std::vector< std::string > &p_out, const std::vector< std::string > &p_in, float *srate=NULL, unsigned int *fragsize=NULL, bool use_jack_transport=false)
Functional form of generic client for synchronous playback and recording of waveform fragments.
- std::vector< unsigned int > **get_port_capture_latency** (const std::vector< std::string > &ports)
Return the JACK port latency of ports.
- std::vector< int > **get_port_capture_latency_int** (const std::vector< std::string > &ports)
Return the JACK port latency of ports.
- std::vector< unsigned int > **get_port_playback_latency** (const std::vector< std::string > &ports)
Return the JACK port latency of ports.
- std::vector< int > **get_port_playback_latency_int** (const std::vector< std::string > &ports)

4.23.1 Detailed Description

Classes and functions for openMHA and JACK interaction.

4.23.2 Function Documentation

4.23.2.1 void MHAJack::io (

```

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

```

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

4.23.2.2 std::vector< unsigned int > MHAJack::get_port_capture_latency (

```

    const std::vector< std::string > & ports )

```

Return the JACK port latency of ports.

Parameters

| | |
|--------------|--------------------|
| <i>ports</i> | Ports to be tested |
|--------------|--------------------|

Returns

Latency vector (one entry for each port)

4.23.2.3 std::vector< int > MHAJack::get_port_capture_latency_int (

```

    const std::vector< std::string > & ports )

```

Return the JACK port latency of ports.

Parameters

| | |
|--------------|--------------------|
| <i>ports</i> | Ports to be tested |
|--------------|--------------------|

Returns

Latency vector (one entry for each port)

4.23.2.4 std::vector< unsigned int > MHAJack::get_port_playback_latency (

```

    const std::vector< std::string > & ports )

```

Return the JACK port latency of ports.

Parameters

| | |
|--------------|--------------------|
| <i>ports</i> | Ports to be tested |
|--------------|--------------------|

Returns

Latency vector (one entry for each port)

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

4.24 MHAKernel Namespace Reference**Classes**

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

Functions

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

4.24.1 Function Documentation

4.24.1.1 `MHAKernel::algo_comm_class_t * MHAKernel::algo_comm_safe_cast (`
`void * h)`

4.25 MHAMultiSrc Namespace Reference

Collection of classes for selecting audio chunks from multiple sources.

Classes

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

4.25.1 Detailed Description

Collection of classes for selecting audio chunks from multiple sources.

4.26 MHAOvIFilter Namespace Reference

Namespace for overlapping FFT based filter bank classes and functions.

Namespaces

- **barkscale**
- **FreqScaleFun**
Transform functions from linear scale in Hz to new frequency scales.
- **ShapeFun**
Shape functions for overlapping filters.

Classes

- class **band_descriptor_t**
- class **fftfb_ac_info_t**
- class **fftfb_t**
FFT based overlapping filter bank.
- class **fftfb_vars_t**
Set of configuration variables for FFT-based overlapping filters.
- class **fscale_bw_t**
- class **fscale_t**
- class **fspacing_t**
Class for frequency spacing, used by filterbank shape generator class.
- class **overlap_save_filterbank_analytic_t**
- class **overlap_save_filterbank_t**
*A time-domain minimal phase filter bank with frequency shapes from **MHAOvIFilter::fftfb_t** (p. 550).*
- class **scale_var_t**

Typedefs

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

4.26.1 Detailed Description

Namespace for overlapping FFT based filter bank classes and functions.

4.26.2 Typedef Documentation

4.26.2.1 typedef mha_real_t(MHAOvfFilter::scale_fun_t)(mha_real_t)

4.27 MHAOvfFilter::barkscale Namespace Reference

Classes

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

Variables

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

4.27.1 Variable Documentation

4.27.1.1 mha_real_t MHAOvfFilter::barkscale::vfreq

4.27.1.2 mha_real_t MHAOvfFilter::barkscale::vbark

4.28 MHAOvfFilter::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))

4.28.1 Detailed Description

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

4.28.2 Function Documentation

4.28.2.1 `mha_real_t MHAOvFilter::FreqScaleFun::hz2hz (mha_real_t x)`

Dummy scale transformation Hz to Hz.

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

Parameters

| | |
|----------------|-----------------------|
| <code>x</code> | Input frequency in Hz |
|----------------|-----------------------|

Returns

Frequency in Hz

4.28.2.2 `mha_real_t MHAOvFilter::FreqScaleFun::hz2khz (mha_real_t x)`

4.28.2.3 `mha_real_t MHAOvFilter::FreqScaleFun::hz2octave (mha_real_t x)`

4.28.2.4 `mha_real_t MHAOvFilter::FreqScaleFun::hz2third_octave (mha_real_t x)`

4.28.2.5 `mha_real_t MHAOvFilter::FreqScaleFun::hz2bark (mha_real_t x)`

Transformation to bark scale.

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

Parameters

| | |
|----------------|-----------------------|
| <code>x</code> | Input frequency in Hz |
|----------------|-----------------------|

Returns

Critical band rate in Bark

4.28.2.6 `mha_real_t MHAOvFilter::FreqScaleFun::hz2bark_analytic (mha_real_t x)`

4.28.2.7 `mha_real_t MHAOvFilter::FreqScaleFun::hz2erb (mha_real_t x)`

4.28.2.8 `mha_real_t MHAOvFilter::FreqScaleFun::hz2erb_glasberg1990 (mha_real_t x)`

4.28.2.9 `mha_real_t MHAOvFilter::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

| | |
|----------------|-----------------|
| <code>x</code> | Frequency in Hz |
|----------------|-----------------|

Returns

Third octaves relative to 1000 Hz

4.28.2.10 `mha_real_t MHAOvFilter::FreqScaleFun::inv_scale (mha_real_t y, mha_real_t(*) (mha_real_t) fun)`

4.29 MHAOvFilter::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)`

4.29.1 Detailed Description

Shape functions for overlapping filters.

4.29.2 Function Documentation

4.29.2.1 `mha_real_t MHAOvFilter::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

| | |
|----------------|----------------------------------|
| <code>x</code> | Input value in the range [-1,1]. |
|----------------|----------------------------------|

Returns

Weight function in the range [0,1]

4.29.2.2 `mha_real_t MHAOvFilter::ShapeFun::linear (mha_real_t x)`

Filter shape function for sawtooth filters.

This function creates sawtooth filter shapes. They rise linearly from 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

| | |
|----------------|----------------------------------|
| <code>x</code> | Input value in the range [-1,1]. |
|----------------|----------------------------------|

Returns

Weight function in the range [0,1]

4.29.2.3 `mha_real_t MHAOvFilter::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

Weight function in the range $[0,1]$

4.29.2.4 `mha_real_t MHAOvFilter::ShapeFun::expflt (`
`mha_real_t x)`

4.29.2.5 `mha_real_t MHAOvFilter::ShapeFun::gauss (`
`mha_real_t x)`

4.30 MHAParser Namespace Reference

Name space for the openMHA-Parser configuration language.

Namespaces

- **StrCnv**
String converter namespace.

Classes

- class **base_t**
Base class for all parser items.
- class **bool_mon_t**
Monitor with string value.
- class **bool_t**
Variable with a boolean value ("yes"/"no")
- class **c_ifc_parser_t**
- class **commit_t**
Parser variable with event-emission functionality.
- class **complex_mon_t**
Monitor with complex value.
- class **complex_t**
Variable with complex value.
- class **entry_t**
- class **expression_t**
- class **float_mon_t**
Monitor with float value.
- class **float_t**

- Variable with float value.*
- class **int_mon_t**
 - Monitor variable with int value.*
- class **int_t**
 - Variable with integer value.*
- class **keyword_list_t**
 - Keyword list class.*
- class **kw_t**
 - Variable with keyword list value.*
- class **mcomplex_mon_t**
 - Matrix of complex numbers monitor.*
- class **mcomplex_t**
 - Matrix variable with complex value.*
- class **mfloat_mon_t**
 - Matrix of floats monitor.*
- class **mfloat_t**
 - Matrix variable with float value.*
- class **mhaconfig_mon_t**
- class **mhapluginloader_t**
 - Class to create a plugin loader in a parser, including the load logic.*
- class **monitor_t**
 - Base class for monitors and variable nodes.*
- class **parser_t**
 - Parser node class.*
- class **range_var_t**
 - Base class for all variables with a numeric value range.*
- class **string_mon_t**
 - Monitor with string value.*
- class **string_t**
 - Variable with a string value.*
- class **variable_t**
 - Base class for variable nodes.*
- class **vcomplex_mon_t**
 - Monitor with vector of complex values.*
- class **vcomplex_t**
 - Vector variable with complex value.*
- class **vfloat_mon_t**
 - Vector of floats monitor.*
- class **vfloat_t**
 - Vector variable with float value.*
- class **vint_mon_t**
 - Vector of ints monitor.*
- class **vint_t**
 - Variable with vector<int> value.*
- class **vstring_mon_t**

Vector of monitors with string value.

- class **vstring_t**

Vector variable with string values.

- class **window_t**

MHA configuration interface for a window function generator.

Typedefs

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

Functions

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

4.30.1 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. 33).

4.30.2 List of valid MHAParser items

- **Sub-parser:** `parser_t` (p. 620)
- **Variables:**
 Numeric variables: `int_t` (p. 598), `vint_t` (p. 644), `float_t` (p. 593), `vfloat_t` (p. 640),
`mfloat_t` (p. 611)
 Other variables: `string_t` (p. 630), `vstring_t` (p. 648), `kw_t` (p. 603), `bool_t` (p. 579)
- **Monitors:**
 Numeric monitors: `int_mon_t` (p. 596), `vint_mon_t` (p. 642), `float_mon_t` (p. 592),
`vfloat_mon_t` (p. 638)
`mfloat_mon_t` (p. 610)
`mcomplex_mon_t` (p. 606)
 Other monitors: `bool_mon_t` (p. 578), `string_mon_t` (p. 628), `vstring_mon_t` (p. 646)

Members can be inserted into the configuration namespace by using `MHAParser::insert_item()` or the `insert_member()` (p. 929) macro.

4.30.3 Typedef Documentation

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

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

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

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

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

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

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

4.30.4 Function Documentation

4.30.4.1 `int MHAParser::get_precision ()`

4.30.4.2 `std::string MHAParser::commentate (`
`const std::string & s)`

4.30.4.3 `void MHAParser::trim (`
`std::string & s)`

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

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

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

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

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

4.30.4.9 `void MHAParser::strreplace (`
 `std::string & s,`
 `const std::string & arg,`
 `const std::string & rep)`

string replace function

Parameters

| | |
|------------|-----------------|
| <i>s</i> | target string |
| <i>arg</i> | search pattern |
| <i>rep</i> | replace pattern |

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

4.31 MHAParser::StrCnv Namespace Reference

String converter namespace.

Functions

- `int num_brackets (const std::string &s)`

Return number of brackets at beginning and end of string.

- int **bracket_balance** (const std::string &s)
- void **str2val** (const std::string &, bool &)

Convert from string.
- void **str2val** (const std::string &, float &)

Convert from string.
- void **str2val** (const std::string &, **mha_complex_t** &)

Convert from string.
- void **str2val** (const std::string &, int &)

Convert from string.
- void **str2val** (const std::string &, **keyword_list_t** &)

Convert from string.
- void **str2val** (const std::string &, std::string &)

Convert from string.
- template<class arg_t >

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

Converter for vector types.
- template<>

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

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

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

Converter for matrix types.
- std::string **val2str** (const bool &)

Convert to string.
- std::string **val2str** (const float &)

Convert to string.
- std::string **val2str** (const **mha_complex_t** &)

Convert to string.
- std::string **val2str** (const int &)

Convert to string.
- std::string **val2str** (const **keyword_list_t** &)

Convert to string.
- std::string **val2str** (const std::string &)

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

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

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

Convert to string.
- std::string **val2str** (const std::vector< std::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.

4.31.1 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.

4.31.2 Function Documentation

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

Return number of brackets at beginning and end of string.

Parameters

| | |
|----------------|--------|
| <code>s</code> | String |
|----------------|--------|

Returns

Number of brackets, or -1 for empty string

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

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

Convert from string.

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

Convert from string.

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

Convert from string.

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

Convert from string.

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

Convert from string.

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

Convert from string.

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

Converter for vector types.

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

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

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

Converter for matrix types.

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

Convert to string.

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

Convert to string.

4.31.2.14 `std::string MHAParser::StrCnv::val2str (`
 `const mha_complex_t & v)`

Convert to string.

4.31.2.15 `std::string MHAParser::StrCnv::val2str (`
 `const int & v)`

Convert to string.

4.31.2.16 `std::string MHAParser::StrCnv::val2str (`
 `const keyword_list_t & v)`

Convert to string.

4.31.2.17 `std::string MHAParser::StrCnv::val2str (`
 `const std::string & v)`

Convert to string.

4.31.2.18 `std::string MHAParser::StrCnv::val2str (`
 `const std::vector< float > & v)`

Convert to string.

4.31.2.19 `std::string MHAParser::StrCnv::val2str (`
 `const std::vector< mha_complex_t > & v)`

Convert to string.

4.31.2.20 `std::string MHAParser::StrCnv::val2str (`
 `const std::vector< int > & v)`

Convert to string.

4.31.2.21 `std::string MHAParser::StrCnv::val2str (`
 `const std::vector< std::string > & v)`

Convert to string.

4.31.2.22 `std::string MHAParser::StrCnv::val2str (`
 `const std::vector< std::vector< float > > & v)`

Convert to string.

4.31.2.23 `std::string MHAParser::StrCnv::val2str (`
`const std::vector< std::vector< mha_complex_t > > & v)`

Convert to string.

4.32 MHAPlugin Namespace Reference

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

Classes

- class **cfg_chain_t**
- class **config_t**
Template class for thread safe configuration.
- class **plugin_t**
The template class for C++ openMHA plugins.

4.32.1 Detailed Description

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

4.33 MHAPlugin_Resampling Namespace Reference

Classes

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

4.34 MHAPlugin_Split Namespace Reference

Classes

- class **domain_handler_t**
Handles domain-specific partial input and output signal.
- class **dummy_threads_t**
Dummy specification of a thread platform: This class implements everything in a single thread.
- class **posix_threads_t**
Posix threads specification of thread platform.
- class **split_t**
Implements split plugin.
- class **splitted_part_t**
*The **splitted_part_t** (p. [682](#)) instance manages the plugin that performs processing on the reduced set of channels.*
- class **thread_platform_t**
Basic interface for encapsulating thread creation, thread priority setting, and synchronization on any threading platform (i.e., pthreads or win32threads).
- class **uni_processor_t**
An interface to a class that sports a process method with no parameters and no return value.

Enumerations

4.34.1 Detailed Description

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

4.34.2 Enumeration Type Documentation

4.34.2.1 anonymous enum

Invalid thread priority.

Enumerator

INVALID_THREAD_PRIORITY

4.35 MHASignal Namespace Reference

Namespace for audio signal handling and processing classes.

Classes

- class **async_rmslevel_t**
Class for asynchronous level metering.
- class **delay_spec_t**
- class **delay_t**
Class to realize a simple delay of waveform streams.
- class **delay_wave_t**
Delayline containing wave fragments.
- class **doublebuffer_t**
Double-buffering class.
- class **fft_t**
- class **hilbert_fftw_t**
- class **hilbert_t**
Hilbert transformation of a waveform segment.
- class **loop_wavefragment_t**
Copy a fixed waveform fragment to a series of waveform fragments of other size.
- class **matrix_t**
n-dimensional matrix with real or complex floating point values.
- class **minphase_t**
Minimal phase function.
- class **quantizer_t**

Simple simulation of fixpoint quantization.

- class **ringbuffer_t**
A ringbuffer class for time domain audio signal, which makes no assumptions with respect to fragment size.
- class **schroeder_t**
Schroeder tone complex class.
- class **spectrum_t**
*a signal processing class for spectral data (based on **mha_spec_t** (p. 406))*
- class **stat_t**
- class **subsample_delay_t**
implements subsample delay in spectral domain.
- class **uint_vector_t**
Vector of unsigned values, used for size and index description of n-dimensional matrixes.
- class **waveform_t**
*signal processing class for waveform data (based on **mha_wave_t** (p. 436))*

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. 436).*
- **mha_real_t** **lin2db** (**mha_real_t** x)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t** **db2lin** (**mha_real_t** x)
Conversion from dB scale to linear (no SPL reference)
- **mha_real_t** **pa2dbspl** (**mha_real_t** x)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t** **pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=1e-20f)
Conversion from squared Pascal scale to dB SPL.
- **mha_real_t** **dbspl2pa** (**mha_real_t** x)
Conversion from dB SPL to linear Pascal scale.
- **mha_real_t** **smp2sec** (**mha_real_t** n, **mha_real_t** srate)
conversion from samples to seconds
- **mha_real_t** **sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
conversion from seconds to samples
- **mha_real_t** **bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)
conversion from fft bin index to frequency
- **mha_real_t** **freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)
conversion from frequency to fft bin index
- **mha_real_t** **smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
conversion from delay in samples to phase shift
- **mha_real_t** **rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)
conversion from phase shift to delay in samples
- template<class elem_type >
std::vector< elem_type > **dupvec** (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size.

- `template<class elem_type >`
`std::vector< elem_type > dupvec_chk (std::vector< elem_type > vec, unsigned n)`
Duplicate last vector element to match desired size, check for dimension.
- `void copy_channel (mha_spec_t &self, const mha_spec_t &src, unsigned sch, unsigned dch)`
Copy one channel of a source signal.
- `void copy_channel (mha_wave_t &self, const mha_wave_t &src, unsigned src_channel, unsigned dest_channel)`
Copy one channel of a source signal.
- `mha_real_t rmslevel (const mha_spec_t &s, unsigned int channel, unsigned int fftlen)`
Return RMS level of a spectrum channel.
- `mha_real_t colored_intensity (const mha_spec_t &s, unsigned int channel, unsigned int fftlen, mha_real_t sqfreq_response[])`
Colored spectrum intensity.
- `mha_real_t maxabs (const mha_spec_t &s, unsigned int channel)`
Find maximal absolute value.
- `mha_real_t rmslevel (const mha_wave_t &s, unsigned int channel)`
Return RMS level of a waveform channel.
- `mha_real_t maxabs (const mha_wave_t &s, unsigned int channel)`
Find maximal absolute value.
- `mha_real_t maxabs (const mha_wave_t &s)`
Find maximal absolute value.
- `mha_real_t max (const mha_wave_t &s)`
Find maximal value.
- `mha_real_t min (const mha_wave_t &s)`
Find minimal value.
- `mha_real_t sumsqr_channel (const mha_wave_t &s, unsigned int channel)`
Calculate sum of squared values in one channel.
- `mha_real_t sumsqr_frame (const mha_wave_t &s, unsigned int frame)`
Calculate sum over all channels of squared values.
- `void scale (mha_spec_t *dest, const mha_wave_t *src)`
- `void limit (mha_wave_t &s, const mha_real_t &min, const mha_real_t &max)`
Limit the signal 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.

4.35.1 Detailed Description

Namespace for audio signal handling and processing classes.

4.35.2 Function Documentation

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

4.35.2.2 void MHASignal::limit (
 mha_wave_t & s,
 const **mha_real_t** & min,
 const **mha_real_t** & max)

Limit the signal in the waveform buffer to the range [min, max].

Parameters

| | |
|------------|--|
| <i>s</i> | The signal to limit. The signal in this wave buffer is modified. |
| <i>min</i> | lower limit |
| <i>max</i> | upper limit |

```
4.35.2.3  template<class elem_type > elem_type MHASignal::kth_smallest (
            elem_type array[],
            unsigned n,
            unsigned k )
```

Fast search for the kth smallest element of an array.

The order of elements is altered, but not completely sorted. Using the algorithm from N. Wirth, published in "Algorithms + data structures = programs", Prentice-Hall, 1976

Parameters

| | |
|--------------|---------------|
| <i>array</i> | Element array |
|--------------|---------------|

Postcondition

The order of elements in the array is altered. `array[k]` then holds the result.

Parameters

| | |
|----------|-----------------------------|
| <i>n</i> | number of elements in array |
|----------|-----------------------------|

Precondition

$n \geq 1$

Parameters

| | |
|----------|---|
| <i>k</i> | The k'th smallest 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

```
4.35.2.4  template<class elem_type > elem_type MHASignal::median (
            elem_type array[],
            unsigned n ) [inline]
```

Fast median search.

The order of elements is altered, but not completely sorted.

Parameters

| | |
|--------------|---------------|
| <i>array</i> | Element array |
|--------------|---------------|

Postcondition

The order of elements in the array is altered. `array[(n-1)/2]` then holds the median.

Parameters

| | |
|----------|-----------------------------|
| <i>n</i> | number of elements in array |
|----------|-----------------------------|

Precondition

`n >= 1`

Returns

The median of the array elements

```
4.35.2.5  template<class elem_type > elem_type MHASignal::mean (
            const std::vector< elem_type > & data,
            elem_type start_val ) [inline]
```

Calculate average of elements in a vector.

Parameters

| | |
|------------------|--|
| <i>data</i> | Input vector |
| <i>start_val</i> | Value for initialization of the return value before sum. |

Returns

The average of the vector elements

```
4.35.2.6  template<class elem_type > std::vector<elem_type> MHASignal::quantile (
            std::vector< elem_type > data,
            const std::vector< elem_type > & p ) [inline]
```

Calculate quantile of elements in a vector.

Parameters

| | |
|-------------|-------------------------------|
| <i>data</i> | Input vector |
| <i>p</i> | Vector of probability values. |

Returns

Vector of quantiles of input data, one entry for each probability value.

```
4.35.2.7 void MHASignal::saveas_mat4 (
    const mha_spec_t & data,
    const std::string & varname,
    FILE * fh )
```

Save a openMHA spectrum as a variable in a Matlab4 file.

Parameters

| | |
|----------------|---|
| <i>data</i> | openMHA spectrum to be saved. |
| <i>varname</i> | Matlab variable name (Matlab4 limitations on maximal length are not checked). |
| <i>fh</i> | File handle to Matlab4 file. |

```
4.35.2.8 void MHASignal::saveas_mat4 (
    const mha_wave_t & data,
    const std::string & varname,
    FILE * fh )
```

Save a openMHA waveform as a variable in a Matlab4 file.

Parameters

| | |
|----------------|---|
| <i>data</i> | openMHA waveform to be saved. |
| <i>varname</i> | Matlab variable name (Matlab4 limitations on maximal length are not checked). |
| <i>fh</i> | File handle to Matlab4 file. |

```
4.35.2.9 void MHASignal::saveas_mat4 (
    const std::vector< mha_real_t > & data,
    const std::string & varname,
    FILE * fh )
```

Save a float vector as a variable in a Matlab4 file.

Parameters

| | |
|----------------|---|
| <i>data</i> | Float vector to be saved. |
| <i>varname</i> | Matlab variable name (Matlab4 limitations on maximal length are not checked). |
| <i>fh</i> | File handle to Matlab4 file. |

4.35.2.10 void MHASignal::copy_permuted (
 mha_wave_t * *dest*,
 const mha_wave_t * *src*)

Copy contents of a waveform to a permuted waveform.

Parameters

| | |
|-------------|----------------------|
| <i>dest</i> | Destination waveform |
| <i>src</i> | Source waveform |

The total size of *src* and *dest* must be the same, *num_frames* and *num_channels* must be exchanged in *dest*.

4.35.3 Variable Documentation

4.35.3.1 unsigned long int MHASignal::signal_counter = 0

Signal counter to produce signal ID strings.

4.36 MHASndFile Namespace Reference**Classes**

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

4.37 MHATableLookup Namespace Reference

Namespace for table lookup classes.

Classes

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

4.37.1 Detailed Description

Namespace for table lookup classes.

4.38 MHAWindow Namespace Reference

Collection of Window types.

Classes

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

Functions

- float **rect** (float)
Rectangular window function.
- float **bartlett** (float)
Bartlett window function.
- float **hanning** (float)
Hanning window function.
- float **hamming** (float)
Hamming window function.
- float **blackman** (float)
Blackman window function.

4.38.1 Detailed Description

Collection of Window types.

4.38.2 Function Documentation

4.38.2.1 float MHAWindow::rect (float x)

Rectangular window function.

4.38.2.2 float MHAWindow::bartlett (float x)

Bartlett window function.

4.38.2.3 float MHAWindow::hanning (float x)

Hanning window function.

4.38.2.4 float MHAWindow::hamming (float x)

Hamming window function.

4.38.2.5 float MHAWindow::blackman (float x)

Blackman window function.

4.39 multibandcompressor Namespace Reference

Classes

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

4.40 noisePowProposedScale Namespace Reference

Classes

- class **interface_t**
- class **noisePowProposed**

4.41 overlapadd Namespace Reference

Classes

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

4.42 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. 444) structures, and report differences as an error.*

4.42.1 Function Documentation

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

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

Compare two **mhaconfig_t** (p. 444) structures, and report differences as an error.

Parameters

| | |
|--------------|---|
| <i>req</i> | Expected mhaconfig_t (p. 444) structure |
| <i>avail</i> | Available mhaconfig_t (p. 444) structure |
| <i>pref</i> | Prefix for error messages |

4.43 route Namespace Reference

Classes

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

4.44 shadowfilter_begin Namespace Reference

Classes

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

4.45 shadowfilter_end Namespace Reference

Classes

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

4.46 smoothgains_bridge Namespace Reference

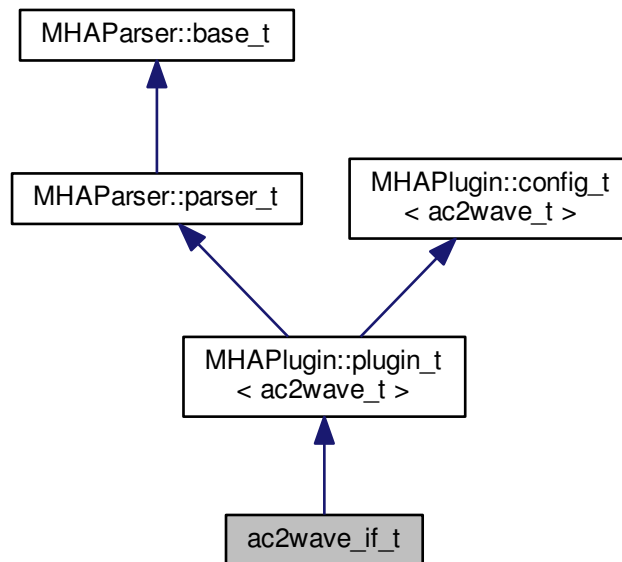
Classes

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

5 Class Documentation

5.1 ac2wave_if_t Class Reference

Inheritance diagram for ac2wave_if_t:



Public Member Functions

- **ac2wave_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::string_t** **name**
- **MHAParser::float_t** **gain_in**
- **MHAParser::float_t** **gain_ac**
- **MHAParser::int_t** **delay_in**
- **MHAParser::int_t** **delay_ac**
- **MHASignal::waveform_t** * **zeros**
- bool **prepared**
- **MHAEvents::patchbay_t** < **ac2wave_if_t** > **patchbay**

Additional Inherited Members

5.1.1 Constructor & Destructor Documentation

5.1.1.1 `ac2wave_if_t::ac2wave_if_t (`
 `const algo_comm_t & iac,`
 `const std::string & ith,`
 `const std::string & ial)`

5.1.2 Member Function Documentation

5.1.2.1 `mha_wave_t * ac2wave_if_t::process (`
 `mha_spec_t *)`

5.1.2.2 `mha_wave_t * ac2wave_if_t::process (`
 `mha_wave_t * s)`

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

Implements **MHAPLugin::plugin_t< ac2wave_t >** (p. 661).

5.1.2.4 `void ac2wave_if_t::release () [virtual]`

Reimplemented from **MHAPLugin::plugin_t< ac2wave_t >** (p. 661).

5.1.2.5 `void ac2wave_if_t::update () [private]`

5.1.3 Member Data Documentation

5.1.3.1 `MHAParser::string_t ac2wave_if_t::name [private]`

5.1.3.2 `MHAParser::float_t ac2wave_if_t::gain_in [private]`

5.1.3.3 `MHAParser::float_t ac2wave_if_t::gain_ac [private]`

5.1.3.4 `MHAParser::int_t ac2wave_if_t::delay_in [private]`

5.1.3.5 `MHAParser::int_t ac2wave_if_t::delay_ac [private]`

5.1.3.6 `MHASignal::waveform_t* ac2wave_if_t::zeros [private]`

5.1.3.7 `bool ac2wave_if_t::prepared [private]`

5.1.3.8 `MHAEvents::patchbay_t<ac2wave_if_t> ac2wave_if_t::patchbay [private]`

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

- **ac2wave.cpp**

5.2 ac2wave_t Class Reference

Public Member Functions

- **ac2wave_t** (unsigned int frames_, unsigned int channels_, **algo_comm_t** ac_, std::string name_, float gain_in_, float gain_ac_, unsigned int delay_in_, unsigned int delay_ac_)
- **mha_wave_t * process** (**mha_wave_t ***)

Private Attributes

- unsigned int **frames**
- unsigned int **channels**
- **mha_wave_t w**
- **algo_comm_t ac**
- std::string **name**
- **MHASignal::delay_wave_t delay_in**
- **MHASignal::delay_wave_t delay_ac**
- **mha_real_t gain_in**
- **mha_real_t gain_ac**

5.2.1 Constructor & Destructor Documentation

5.2.1.1 ac2wave_t::ac2wave_t (
 unsigned int *frames_*,
 unsigned int *channels_*,
 algo_comm_t *ac_*,
 std::string *name_*,
 float *gain_in_*,
 float *gain_ac_*,
 unsigned int *delay_in_*,
 unsigned int *delay_ac_*)

5.2.2 Member Function Documentation

5.2.2.1 **mha_wave_t * ac2wave_t::process** (
 mha_wave_t * s)

5.2.3 Member Data Documentation

5.2.3.1 unsigned int ac2wave_t::frames [private]

5.2.3.2 unsigned int ac2wave_t::channels [private]

5.2.3.3 **mha_wave_t** ac2wave_t::w [private]

5.2.3.4 `algo_comm_t ac2wave_t::ac` [private]

5.2.3.5 `std::string ac2wave_t::name` [private]

5.2.3.6 `MHASignal::delay_wave_t ac2wave_t::delay_in` [private]

5.2.3.7 `MHASignal::delay_wave_t ac2wave_t::delay_ac` [private]

5.2.3.8 `mha_real_t ac2wave_t::gain_in` [private]

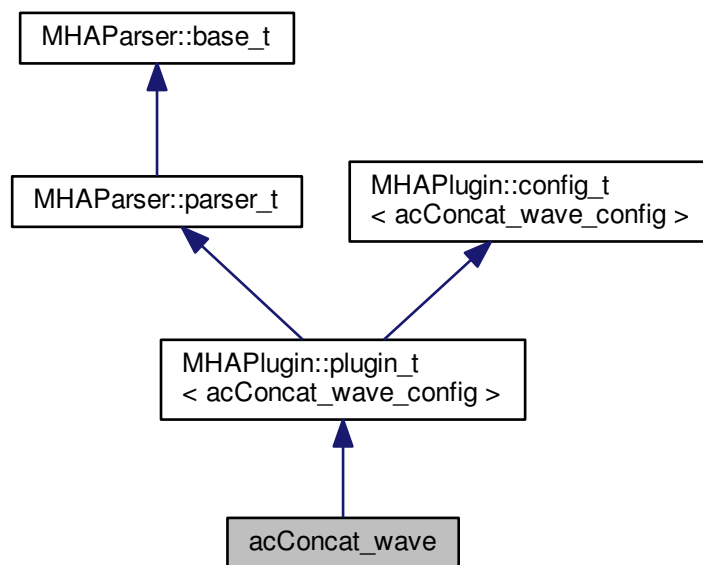
5.2.3.9 `mha_real_t ac2wave_t::gain_ac` [private]

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

- `ac2wave.cpp`

5.3 `acConcat_wave` Class Reference

Inheritance diagram for `acConcat_wave`:



Public Member Functions

- **acConcat_wave** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~acConcat_wave** ()
- **mha_wave_t * process** (**mha_wave_t ***)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t num_AC**
- **MHAParser::string_t prefix_names_AC**
- **MHAParser::vint_t samples_AC**
- **MHAParser::string_t name_conAC**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< acConcat_wave > patchbay**

Additional Inherited Members

5.3.1 Constructor & Destructor Documentation

5.3.1.1 acConcat_wave::acConcat_wave (
 algo_comm_t &ac,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.3.1.2 acConcat_wave::~~acConcat_wave ()

5.3.2 Member Function Documentation

5.3.2.1 mha_wave_t * acConcat_wave::process (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.3.2.2 void acConcat_wave::prepare (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. 661).

```
5.3.2.3 void acConcat_wave::release (
        void ) [inline], [virtual]
```

Reimplemented from **MHAPLugin::plugin_t< acConcat_wave_config >** (p. 661).

```
5.3.2.4 void acConcat_wave::update_cfg ( ) [private]
```

5.3.3 Member Data Documentation

5.3.3.1 **MHAParser::int_t acConcat_wave::num_AC**

5.3.3.2 **MHAParser::string_t acConcat_wave::prefix_names_AC**

5.3.3.3 **MHAParser::vint_t acConcat_wave::samples_AC**

5.3.3.4 **MHAParser::string_t acConcat_wave::name_conAC**

5.3.3.5 **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**

5.4 acConcat_wave_config Class Reference

Public Member Functions

- **acConcat_wave_config (algo_comm_t &ac, const mhaconfig_t in_cfg, acConcat_wave * _concat)**
- **~acConcat_wave_config ()**
- **mha_wave_t * process (mha_wave_t *)**

Public Attributes

- **algo_comm_t & ac**
- **std::vector< std::string > strNames_AC**
- **std::vector< int > numSamples_AC**
- **mha_wave_t vGCC**
- **MHA_AC::waveform_t * vGCC_con**

5.4.1 Constructor & Destructor Documentation

5.4.1.1 **acConcat_wave_config::acConcat_wave_config (**
 algo_comm_t & ac,
 const mhaconfig_t in_cfg,
 acConcat_wave * _concat)

5.4.1.2 **acConcat_wave_config::~~acConcat_wave_config ()**

5.4.2 Member Function Documentation

5.4.2.1 **mha_wave_t * acConcat_wave_config::process (**
 mha_wave_t * wave)

5.4.3 Member Data Documentation

5.4.3.1 **algo_comm_t& acConcat_wave_config::ac**

5.4.3.2 **std::vector<std::string> acConcat_wave_config::strNames_AC**

5.4.3.3 **std::vector<int> acConcat_wave_config::numSamples_AC**

5.4.3.4 **mha_wave_t acConcat_wave_config::vGCC**

5.4.3.5 **MHA_AC::waveform_t* acConcat_wave_config::vGCC_con**

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

- **acConcat_wave.h**
- **acConcat_wave.cpp**

5.5 acmon::ac_monitor_t Class Reference

A class for converting AC variables to Parser monitors of correct type.

Public Member Functions

- **ac_monitor_t** (MHAParser::parser_t &parent, const std::string &name_, algo_comm_t ac, bool use_matrix)
Converts AC variable to parser monitor.
- void **getvar** (algo_comm_t ac)
Update values of monitor.

Public Attributes

- std::string **name**
name of AC variable and parser monitor
- std::string **dimstr**
columns x rows
- MHAParser::vfloat_mon_t **mon**
Monitor used for real vectors.
- MHAParser::mfloat_mon_t **mon_mat**
Monitor used for real matrices.
- MHAParser::vcomplex_mon_t **mon_complex**
monitor used for complex vectors
- MHAParser::mcomplex_mon_t **mon_mat_complex**
monitor used for complex matrices
- MHAParser::parser_t & **p_parser**
parent parser to insert monitor into

Private Attributes

- bool **use_mat**
if true, use matrix monitor, else use vector monitor

5.5.1 Detailed Description

A class for converting AC variables to Parser monitors of correct type.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 acmon::ac_monitor_t::ac_monitor_t (
 MHAParser::parser_t & parent,
 const std::string & name_,
 algo_comm_t ac,
 bool use_matrix)

Converts AC variable to parser monitor.

Parameters

| | |
|-------------------|--|
| <i>parent</i> | The parser to insert a monitor into |
| <i>name_</i> | The name of the AC variable and the monitor variable |
| <i>ac</i> | Handle to algorithm communication space |
| <i>use_matrix</i> | Indicates if a matrix monitor type should be used. |

5.5.3 Member Function Documentation**5.5.3.1 void acmon::ac_monitor_t::getvar (algo_comm_t ac)**

Update values of monitor.

Parameters

| | |
|-----------|---|
| <i>ac</i> | Handle to algorithm communication space |
|-----------|---|

5.5.4 Member Data Documentation**5.5.4.1 std::string acmon::ac_monitor_t::name**

name of AC variable and parser monitor

5.5.4.2 std::string acmon::ac_monitor_t::dimstr

columns x rows

5.5.4.3 MHAParser::vfloat_mon_t acmon::ac_monitor_t::mon

Monitor used for real vectors.

5.5.4.4 MHAParser::mfloat_mon_t acmon::ac_monitor_t::mon_mat

Monitor used for real matrices.

5.5.4.5 MHAParser::vcomplex_mon_t acmon::ac_monitor_t::mon_complex

monitor used for complex vectors

5.5.4.6 MHAParser::mcomplex_mon_t acmon::ac_monitor_t::mon_mat_complex

monitor used for complex matrices

5.5.4.7 MHAParser::parser_t& acmon::ac_monitor_t::p_parser

parent parser to insert monitor into

5.5.4.8 bool acmon::ac_monitor_t::use_mat [private]

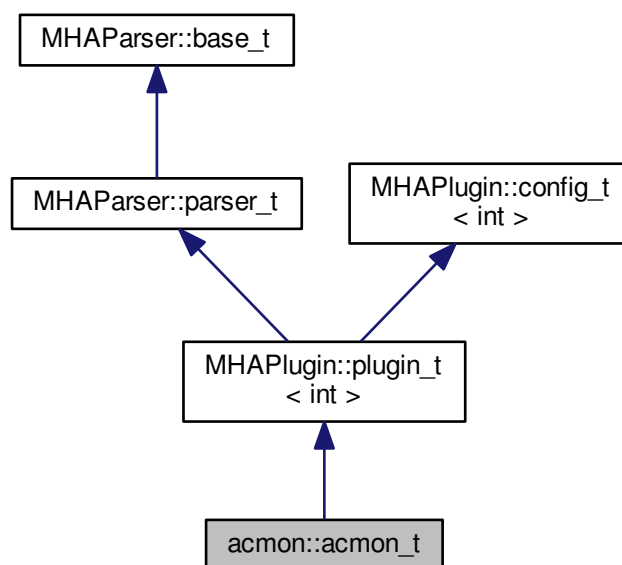
if true, use matrix monitor, else use vector monitor

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

- **ac_monitor_type.hh**
- **ac_monitor_type.cpp**

5.6 acmon::acmon_t Class Reference

Inheritance diagram for acmon::acmon_t:



Public Member Functions

- **acmon_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **~acmon_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **save_vars** ()
- void **update_recmode** ()

Private Attributes

- **algo_comm_t** **ac**
- **MHAParser::vstring_mon_t** **varlist**
- **MHAParser::vstring_mon_t** **dimensions**
- **MHAParser::kw_t** **dispmode**
- **MHAParser::kw_t** **recmode**
- std::vector< **ac_monitor_t** * > **vars**
- **MHAEvents::patchbay_t**< **acmon_t** > **patchbay**
- std::string **chain**
- std::string **algo**
- bool **b_cont**
- bool **b_snapshot**

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 **acmon::acmon_t::acmon_t** (
 const **algo_comm_t** & *iac*,
 const std::string & *ith*,
 const std::string & *ial*)

5.6.1.2 **acmon::acmon_t::~~acmon_t** (
 void)

5.6.2 Member Function Documentation

5.6.2.1 void **acmon::acmon_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Implements **MHAPLugin::plugin_t**< int > (p. 661).

5.6.2.2 void acmon::acmon_t::release (
void) [inline], [virtual]

Reimplemented from **MHAPLugin::plugin_t< int >** (p. 661).

5.6.2.3 mha_spec_t* acmon::acmon_t::process (
mha_spec_t* s)

5.6.2.4 mha_wave_t* acmon::acmon_t::process (
mha_wave_t* s)

5.6.2.5 void acmon::acmon_t::save_vars () [private]

5.6.2.6 void acmon::acmon_t::update_recmode () [private]

5.6.3 Member Data Documentation

5.6.3.1 algo_comm_t acmon::acmon_t::ac [private]

5.6.3.2 MHAParser::vstring_mon_t acmon::acmon_t::varlist [private]

5.6.3.3 MHAParser::vstring_mon_t acmon::acmon_t::dimensions [private]

5.6.3.4 MHAParser::kw_t acmon::acmon_t::dispmode [private]

5.6.3.5 MHAParser::kw_t acmon::acmon_t::recmode [private]

5.6.3.6 std::vector<ac_monitor_t*> acmon::acmon_t::vars [private]

5.6.3.7 MHAEvents::patchbay_t<acmon_t> acmon::acmon_t::patchbay [private]

5.6.3.8 std::string acmon::acmon_t::chain [private]

5.6.3.9 std::string acmon::acmon_t::algo [private]

5.6.3.10 bool acmon::acmon_t::b_cont [private]

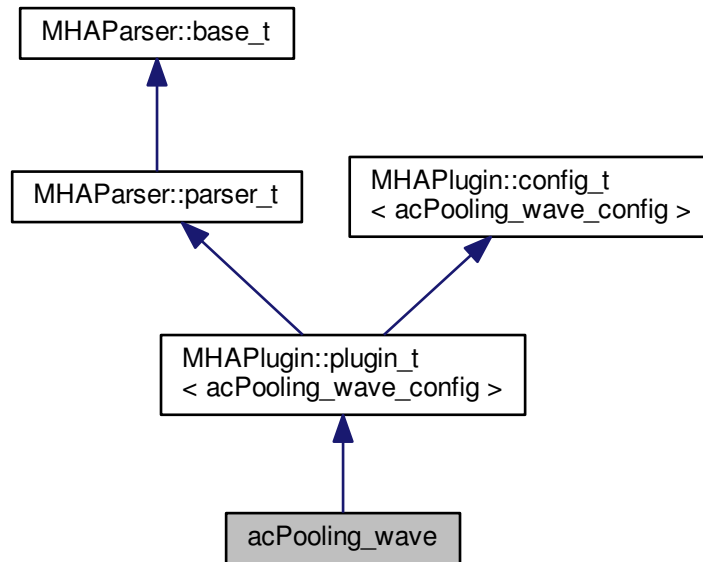
5.6.3.11 bool acmon::acmon_t::b_snapshot [private]

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

- **acmon.cpp**

5.7 acPooling_wave Class Reference

Inheritance diagram for acPooling_wave:



Public Member Functions

- **acPooling_wave** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~acPooling_wave** ()
- **mha_wave_t * process** (**mha_wave_t ***)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t** numsamples
- **MHAParser::int_t** pooling_wndlen
- **MHAParser::kw_t** pooling_type
- **MHAParser::float_t** upper_threshold
- **MHAParser::float_t** lower_threshold
- **MHAParser::int_t** neighbourhood

- **MHAParser::float_t** alpha
- **MHAParser::string_t** p_name
- **MHAParser::string_t** pool_name
- **MHAParser::string_t** max_pool_ind_name
- **MHAParser::string_t** like_ratio_name

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **acPooling_wave** > patchbay

Additional Inherited Members

5.7.1 Constructor & Destructor Documentation

5.7.1.1 **acPooling_wave::acPooling_wave** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.7.1.2 **acPooling_wave::~~acPooling_wave** ()

5.7.2 Member Function Documentation

5.7.2.1 **mha_wave_t** * **acPooling_wave::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.7.2.2 void **acPooling_wave::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate). |
|--------------------|---|

Implements **MHAParser::plugin_t< acPooling_wave_config >** (p. 661).

```
5.7.2.3 void acPooling_wave::release (
        void ) [inline],[virtual]
```

Reimplemented from **MHAParser::plugin_t< acPooling_wave_config >** (p. 661).

```
5.7.2.4 void acPooling_wave::update_cfg ( ) [private]
```

5.7.3 Member Data Documentation

5.7.3.1 **MHAParser::int_t acPooling_wave::numsamples**

5.7.3.2 **MHAParser::int_t acPooling_wave::pooling_wndlen**

5.7.3.3 **MHAParser::kw_t acPooling_wave::pooling_type**

5.7.3.4 **MHAParser::float_t acPooling_wave::upper_threshold**

5.7.3.5 **MHAParser::float_t acPooling_wave::lower_threshold**

5.7.3.6 **MHAParser::int_t acPooling_wave::neighbourhood**

5.7.3.7 **MHAParser::float_t acPooling_wave::alpha**

5.7.3.8 **MHAParser::string_t acPooling_wave::p_name**

5.7.3.9 **MHAParser::string_t acPooling_wave::pool_name**

5.7.3.10 **MHAParser::string_t acPooling_wave::max_pool_ind_name**

5.7.3.11 **MHAParser::string_t acPooling_wave::like_ratio_name**

5.7.3.12 **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**

5.8 acPooling_wave_config Class Reference

Public Member Functions

- **acPooling_wave_config** (**algo_comm_t** &**ac**, const **mhaconfig_t** **in_cfg**, **acPooling_wave** ***_pooling**)
- **~acPooling_wave_config** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **insert** ()

Public Attributes

- **algo_comm_t** & **ac**
- std::string **raw_p_name**
- **MHA_AC::waveform_t** **p**
- **MHA_AC::waveform_t** **p_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**

5.8.1 Constructor & Destructor Documentation

5.8.1.1 **acPooling_wave_config::acPooling_wave_config** (
 algo_comm_t &**ac**,
 const **mhaconfig_t** **in_cfg**,
 acPooling_wave * **_pooling**)

5.8.1.2 **acPooling_wave_config::~acPooling_wave_config** ()

5.8.2 Member Function Documentation

5.8.2.1 **mha_wave_t** * **acPooling_wave_config::process** (
 mha_wave_t * **wave**)

5.8.2.2 void **acPooling_wave_config::insert** ()

5.8.3 Member Data Documentation

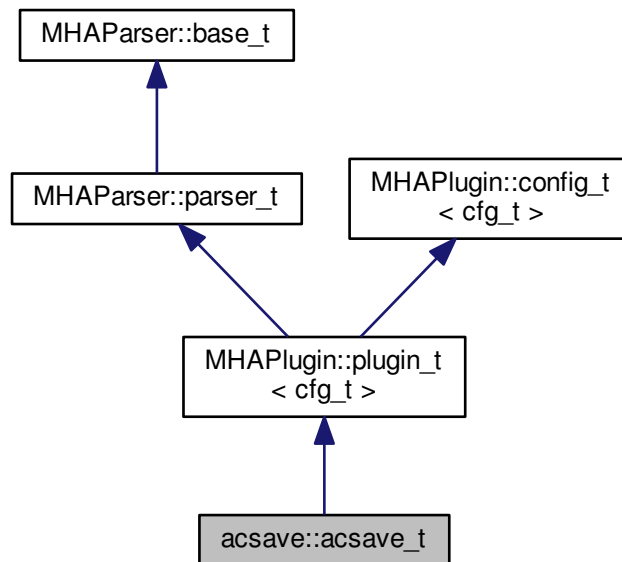
- 5.8.3.1 `algo_comm_t& acPooling_wave_config::ac`
- 5.8.3.2 `std::string acPooling_wave_config::raw_p_name`
- 5.8.3.3 `MHA_AC::waveform_t acPooling_wave_config::p`
- 5.8.3.4 `MHA_AC::waveform_t acPooling_wave_config::p_max`
- 5.8.3.5 `MHA_AC::waveform_t acPooling_wave_config::like_ratio`
- 5.8.3.6 `mha_wave_t acPooling_wave_config::c`
- 5.8.3.7 `unsigned int acPooling_wave_config::pooling_ind`
- 5.8.3.8 `unsigned int acPooling_wave_config::pooling_option`
- 5.8.3.9 `unsigned int acPooling_wave_config::pooling_size`
- 5.8.3.10 `float acPooling_wave_config::up_thresh`
- 5.8.3.11 `float acPooling_wave_config::low_thresh`
- 5.8.3.12 `int acPooling_wave_config::neigh`
- 5.8.3.13 `float acPooling_wave_config::alpha`
- 5.8.3.14 `MHASignal::waveform_t acPooling_wave_config::pool`

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

- `acPooling_wave.h`
- `acPooling_wave.cpp`

5.9 acsave::acsave_t Class Reference

Inheritance diagram for acsave::acsave_t:



Public Member Functions

- **acsave_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **event_start_recording** ()
- void **event_stop_and_flush** ()

Private Types

- typedef std::vector< **save_var_t** * > **varlist_t**

Private Member Functions

- void **process** ()

Private Attributes

- **MHAParser::bool_t bflush**
- **MHAParser::kw_t fileformat**
- **MHAParser::string_t fname**
- **MHAParser::float_t reclen**
- **MHAParser::vstring_t variables**
- **varlist_t varlist**
- **std::string chain**
- **std::string algo**
- **bool b_prepared**
- **bool b_flushed**
- **MHAEvents::patchbay_t< acsave_t > patchbay**

Additional Inherited Members

5.9.1 Member Typedef Documentation

5.9.1.1 `typedef std::vector<save_var_t*> acsave::acsave_t::varlist_t` [private]

5.9.2 Constructor & Destructor Documentation

5.9.2.1 `acsave::acsave_t::acsave_t (`
 `const algo_comm_t & iac,`
 `const std::string & ith,`
 `const std::string & ial)`

5.9.3 Member Function Documentation

5.9.3.1 `void acsave::acsave_t::prepare (`
 `mhaconfig_t & tf)` [virtual]

Implements **MHAPLugin::plugin_t< cfg_t >** (p. 661).

5.9.3.2 `void acsave::acsave_t::release (`
 `void)` [virtual]

Reimplemented from **MHAPLugin::plugin_t< cfg_t >** (p. 661).

5.9.3.3 mha_spec_t * acsave::acsave_t::process (
mha_spec_t * s)

5.9.3.4 mha_wave_t * acsave::acsave_t::process (
mha_wave_t * s)

5.9.3.5 void acsave::acsave_t::event_start_recording ()

5.9.3.6 void acsave::acsave_t::event_stop_and_flush ()

5.9.3.7 void acsave::acsave_t::process () [private]

5.9.4 Member Data Documentation

5.9.4.1 MHAParser::bool_t acsave::acsave_t::bflush [private]

5.9.4.2 MHAParser::kw_t acsave::acsave_t::fileformat [private]

5.9.4.3 MHAParser::string_t acsave::acsave_t::fname [private]

5.9.4.4 MHAParser::float_t acsave::acsave_t::reclen [private]

5.9.4.5 MHAParser::vstring_t acsave::acsave_t::variables [private]

5.9.4.6 varlist_t acsave::acsave_t::varlist [private]

5.9.4.7 std::string acsave::acsave_t::chain [private]

5.9.4.8 std::string acsave::acsave_t::algo [private]

5.9.4.9 bool acsave::acsave_t::b_prepared [private]

5.9.4.10 bool acsave::acsave_t::b_flushed [private]

5.9.4.11 MHAEvents::patchbay_t<acsave_t> acsave::acsave_t::patchbay [private]

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

- **acsave.cpp**

5.10 acsave::cfg_t Class Reference

Public Member Functions

- **cfg_t** (const **algo_comm_t** &iac, unsigned int imax_frames, std::vector< std::string > &var_names)
- **~cfg_t** ()
- void **store_frame** ()
- void **flush_data** (const std::string &, unsigned int)

Private Attributes

- **algo_comm_t** ac
- unsigned int **nvars**
- **save_var_t** ** varlist
- unsigned int **rec_frames**
- unsigned int **max_frames**

5.10.1 Constructor & Destructor Documentation

5.10.1.1 **cfg_t::cfg_t** (
 const **algo_comm_t** & iac,
 unsigned int *imax_frames*,
 std::vector< std::string > & *var_names*)

5.10.1.2 **cfg_t::~~cfg_t** ()

5.10.2 Member Function Documentation

5.10.2.1 void **cfg_t::store_frame** ()

This function is called in the processing thread.

5.10.2.2 void **cfg_t::flush_data** (
 const std::string & *filename*,
 unsigned int *fmt*)

This function is called in the configuration thread.

Parameters

| | |
|-----------------|--------------------|
| <i>filename</i> | Output file name |
| <i>fmt</i> | Output file format |

5.10.3 Member Data Documentation

5.10.3.1 `algo_comm_t` `acsave::cfg_t::ac` [private]

5.10.3.2 `unsigned int` `acsave::cfg_t::nvars` [private]

5.10.3.3 `save_var_t**` `acsave::cfg_t::varlist` [private]

5.10.3.4 `unsigned int` `acsave::cfg_t::rec_frames` [private]

5.10.3.5 `unsigned int` `acsave::cfg_t::max_frames` [private]

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

- `acsave.cpp`

5.11 `acsave::mat4head_t` Struct Reference

Public Attributes

- `int32_t` `t`
- `int32_t` `rows`
- `int32_t` `cols`
- `int32_t` `imag`
- `int32_t` `namelen`

5.11.1 Member Data Documentation

5.11.1.1 `int32_t` `acsave::mat4head_t::t`

5.11.1.2 `int32_t` `acsave::mat4head_t::rows`

5.11.1.3 `int32_t` `acsave::mat4head_t::cols`

5.11.1.4 `int32_t` `acsave::mat4head_t::imag`

5.11.1.5 `int32_t` `acsave::mat4head_t::namelen`

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

- `acsave.cpp`

5.12 acsave::save_var_t Class Reference

Public Member Functions

- **save_var_t** (const std::string &, int, const **algo_comm_t** &)
- **~save_var_t** ()
- void **store_frame** ()
- void **save_txt** (FILE *, unsigned int)
- void **save_mat4** (FILE *, unsigned int)
- void **save_m** (FILE *, unsigned int)

Public Attributes

- double * **data**

Private Attributes

- std::string **name**
- unsigned int **nframes**
- unsigned int **ndim**
- unsigned int **maxframe**
- **algo_comm_t** **ac**
- unsigned int **framecnt**
- bool **b_complex**

5.12.1 Constructor & Destructor Documentation

5.12.1.1 **acsave::save_var_t::save_var_t** (
 const std::string & *nm*,
 int *n*,
 const **algo_comm_t** & *iac*)

5.12.1.2 **acsave::save_var_t::~~save_var_t** ()

5.12.2 Member Function Documentation

5.12.2.1 void **acsave::save_var_t::store_frame** ()

5.12.2.2 void **acsave::save_var_t::save_txt** (
 FILE * *fh*,
 unsigned int *writeframes*)

5.12.2.3 void acsave::save_var_t::save_mat4 (
FILE * *fh*,
unsigned int *writeframes*)

5.12.2.4 void acsave::save_var_t::save_m (
FILE * *fh*,
unsigned int *writeframes*)

5.12.3 Member Data Documentation

5.12.3.1 double* acsave::save_var_t::data

5.12.3.2 std::string acsave::save_var_t::name [private]

5.12.3.3 unsigned int acsave::save_var_t::nframes [private]

5.12.3.4 unsigned int acsave::save_var_t::ndim [private]

5.12.3.5 unsigned int acsave::save_var_t::maxframe [private]

5.12.3.6 algo_comm_t acsave::save_var_t::ac [private]

5.12.3.7 unsigned int acsave::save_var_t::framecnt [private]

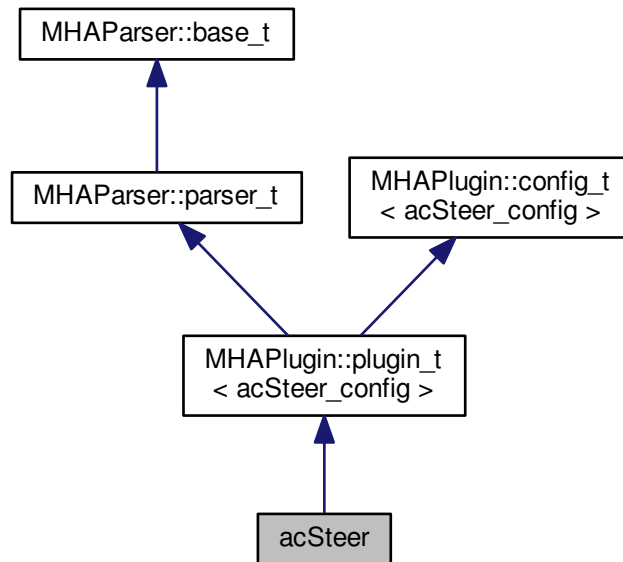
5.12.3.8 bool acsave::save_var_t::b_complex [private]

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

- **acsave.cpp**

5.13 acSteer Class Reference

Inheritance diagram for acSteer:



Public Member Functions

- **acSteer** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~acSteer** ()
- **mha_spec_t * process** (**mha_spec_t ***)
This 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

5.13.1 Constructor & Destructor Documentation

5.13.1.1 **acSteer::acSteer** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.13.1.2 **acSteer::~~acSteer** ()

5.13.2 Member Function Documentation

5.13.2.1 **mha_spec_t** * **acSteer::process** (
 mha_spec_t * *signal*)

This method is a NOOP.

5.13.2.2 void **acSteer::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. 661).

5.13.2.3 `void acSteer::release (`
 `void) [inline],[virtual]`

Reimplemented from **MHAParser::plugin_t< acSteer_config >** (p. 661).

5.13.2.4 `void acSteer::update_cfg () [private]`

5.13.3 Member Data Documentation

5.13.3.1 **MHAParser::string_t acSteer::steerFile**

5.13.3.2 **MHAParser::string_t acSteer::acSteerName1**

5.13.3.3 **MHAParser::string_t acSteer::acSteerName2**

5.13.3.4 **MHAParser::int_t acSteer::nsteerchan**

5.13.3.5 **MHAParser::int_t acSteer::nrefmic**

5.13.3.6 **MHAEvents::patchbay_t<acSteer> acSteer::patchbay [private]**

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

- **acSteer.h**
- **acSteer.cpp**

5.14 acSteer_config Class Reference

Public Member Functions

- **acSteer_config (algo_comm_t &ac, const mhaconfig_t in_cfg, acSteer *acSteer)**
- **~acSteer_config ()**
- **void insert ()**

Public Attributes

- unsigned int **nchan**
- unsigned int **nfreq**
- unsigned int **nsteerchan**
- unsigned int **nrefmic**
- unsigned int **nangle**
- **MHA_AC::spectrum_t specSteer1**
- **MHA_AC::spectrum_t specSteer2**

5.14.1 Constructor & Destructor Documentation

5.14.1.1 `acSteer_config::acSteer_config (`
 `algo_comm_t & ac,`
 `const mhaconfig_t in_cfg,`
 `acSteer * acSteer)`

5.14.1.2 `acSteer_config::~~acSteer_config ()`

5.14.2 Member Function Documentation

5.14.2.1 `void acSteer_config::insert ()`

5.14.3 Member Data Documentation

5.14.3.1 `unsigned int acSteer_config::nchan`

5.14.3.2 `unsigned int acSteer_config::nfreq`

5.14.3.3 `unsigned int acSteer_config::nsteerchan`

5.14.3.4 `unsigned int acSteer_config::nrefmic`

5.14.3.5 `unsigned int acSteer_config::nangle`

5.14.3.6 `MHA_AC::spectrum_t acSteer_config::specSteer1`

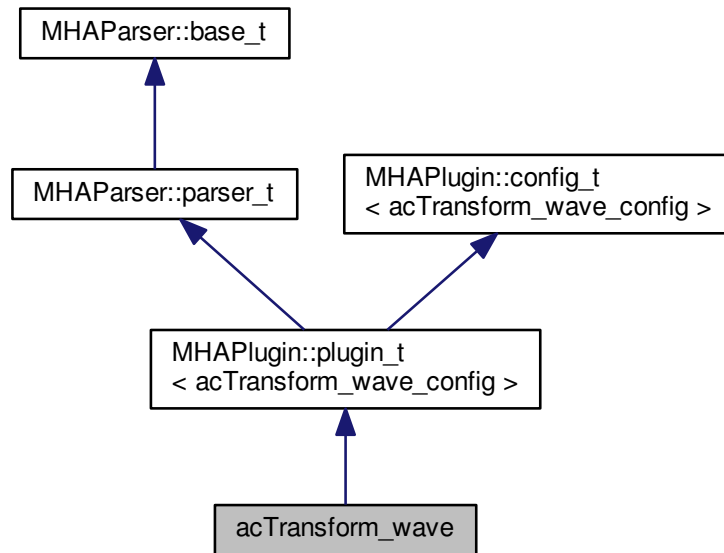
5.14.3.7 `MHA_AC::spectrum_t acSteer_config::specSteer2`

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

- **acSteer.h**
- **acSteer.cpp**

5.15 acTransform_wave Class Reference

Inheritance diagram for acTransform_wave:



Public Member Functions

- **acTransform_wave** (algo_comm_t &ac, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~acTransform_wave** ()
- **mha_wave_t * process** (mha_wave_t *)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (mhaconfig_t &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::string_t** ang_name
- **MHAParser::string_t** raw_p_name
- **MHAParser::string_t** raw_p_max_name
- **MHAParser::string_t** rotated_p_name
- **MHAParser::string_t** rotated_p_max_name
- **MHAParser::int_t** numsamples
- **MHAParser::bool_t** to_from

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t** < **acTransform_wave** > **patchbay**

Additional Inherited Members

5.15.1 Constructor & Destructor Documentation

5.15.1.1 **acTransform_wave::acTransform_wave** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.15.1.2 **acTransform_wave::~acTransform_wave** ()

5.15.2 Member Function Documentation

5.15.2.1 **mha_wave_t** * **acTransform_wave::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.15.2.2 void **acTransform_wave::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. 661).

5.15.2.3 `void acTransform_wave::release (`
`void) [inline],[virtual]`

Reimplemented from **MHAPLugin::plugin_t< acTransform_wave_config >** (p. 661).

5.15.2.4 `void acTransform_wave::update_cfg () [private]`

5.15.3 Member Data Documentation

5.15.3.1 **MHAParser::string_t acTransform_wave::ang_name**

5.15.3.2 **MHAParser::string_t acTransform_wave::raw_p_name**

5.15.3.3 **MHAParser::string_t acTransform_wave::raw_p_max_name**

5.15.3.4 **MHAParser::string_t acTransform_wave::rotated_p_name**

5.15.3.5 **MHAParser::string_t acTransform_wave::rotated_p_max_name**

5.15.3.6 **MHAParser::int_t acTransform_wave::numsamples**

5.15.3.7 **MHAParser::bool_t acTransform_wave::to_from**

5.15.3.8 **MHAEvents::patchbay_t<acTransform_wave> acTransform_wave::patchbay**
`[private]`

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

5.16 acTransform_wave_config Class Reference

Public Member Functions

- **acTransform_wave_config (algo_comm_t &ac, const mhaconfig_t in_cfg, ac↔
Transform_wave *_transform)**
- **~acTransform_wave_config ()**
- **mha_wave_t * process (mha_wave_t *)**

Public Attributes

- **algo_comm_t** & **ac**
- std::string **ang_name**
- std::string **raw_p_name**
- std::string **raw_p_max_name**
- **MHA_AC::waveform_t** **rotated_p**
- **MHA_AC::int_t** **rotated_i**
- unsigned int **offset**
- unsigned int **resolution**
- unsigned int **to_from**

5.16.1 Constructor & Destructor Documentation

5.16.1.1 **acTransform_wave_config::acTransform_wave_config** (
 algo_comm_t & *ac*,
 const **mhaconfig_t** *in_cfg*,
 acTransform_wave * *_transform*)

5.16.1.2 **acTransform_wave_config::~~acTransform_wave_config** ()

5.16.2 Member Function Documentation

5.16.2.1 **mha_wave_t** * **acTransform_wave_config::process** (
 mha_wave_t * *wave*)

5.16.3 Member Data Documentation

5.16.3.1 **algo_comm_t** & **acTransform_wave_config::ac**

5.16.3.2 std::string **acTransform_wave_config::ang_name**

5.16.3.3 std::string **acTransform_wave_config::raw_p_name**

5.16.3.4 std::string **acTransform_wave_config::raw_p_max_name**

5.16.3.5 **MHA_AC::waveform_t** **acTransform_wave_config::rotated_p**

5.16.3.6 **MHA_AC::int_t** **acTransform_wave_config::rotated_i**

5.16.3.7 unsigned int **acTransform_wave_config::offset**

5.16.3.8 unsigned int **acTransform_wave_config::resolution**

5.16.3.9 unsigned int **acTransform_wave_config::to_from**

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

5.17 ADM::ADM< F > Class Template Reference

Adaptive differential microphone, working for speech frequency range.

Public Member Functions

- **ADM** (F fs, F dist, unsigned lp_order, const F *lp_alphas, unsigned decomb_order, const F *decomb_alphas, F tau_beta=F(50e-3), F mu_beta=F(1e-4))
Create Adaptive Differential Microphone.
- F **process** (const F &front, const F &back, const F &external_beta=F(-1))
ADM (p. 158) 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**

5.17.1 Detailed Description

```
template<class F>
class ADM::ADM< F >
```

Adaptive differential microphone, working for speech frequency range.

5.17.2 Constructor & Destructor Documentation

```
5.17.2.1 template<class F > ADM::ADM< F >::ADM (
    F fs,
    F dist,
    unsigned lp_order,
    const F * lp_alphas,
    unsigned decomb_order,
    const F * decomb_alphas,
    F tau_beta = F ( 50e-3 ) ,
    F mu_beta = F ( 1e-4 ) )
```

Create Adaptive Differential Microphone.

Parameters

| | |
|----------------------|---|
| <i>fs</i> | Sampling rate / Hz |
| <i>dist</i> | Distance between physical microphones / m |
| <i>lp_order</i> | Filter order of FIR lowpass filter used for adaptation |
| <i>lp_alphas</i> | Pointer to array of alpha coefficients for the lowpass filter used for adaptation. Since this class uses linear phase FIR filters only, only the first half ($\text{order}/2 + 1$) of the coefficients will be read (coefficients for linear-phase FIR filters are symmetric). |
| <i>decomb_order</i> | Filter order of FIR compensation filter (compensates for comb filter characteristic) |
| <i>decomb_alphas</i> | Pointer to array of alpha coefficients for the compensation filter used to compensate for the comb filter characteristic. Since this class uses linear phase FIR filters only, only the first half ($\text{order}/2 + 1$) of the coefficients will be read (coefficients for linear-phase FIR filters are symmetric). |
| <i>tau_beta</i> | Time constant of the lowpass filter used for averaging the power of the output signal |
| <i>mu_beta</i> | adaption speed |

5.17.3 Member Function Documentation

5.17.3.1 `template<class F> F ADM::ADM< F >::process (`
`const F & front,`
`const F & back,`
`const F & external_beta = F(-1)) [inline]`

ADM (p. 158) processes one frame.

Parameters

| | |
|----------------------|--|
| <i>front</i> | The current front input signal sample |
| <i>back</i> | The current rear input signal sample |
| <i>external_beta</i> | If ≥ 0 , this is used as the "beta" parameter for direction to filter out. Else, the beta parameter is adapted to filtered out a direction so that best reduction of signal intensity from the back hemisphere is achieved. |

Returns

The computed output sample

5.17.3.2 `template<class F> F ADM::ADM< F >::beta () const [inline]`

5.17.4 Member Data Documentation

- 5.17.4.1 `template<class F> Delay<F> ADM::ADM<F>::m_delay_front` [private]
- 5.17.4.2 `template<class F> Delay<F> ADM::ADM<F>::m_delay_back` [private]
- 5.17.4.3 `template<class F> Linearphase_FIR<F> ADM::ADM<F>::m_lp_bf` [private]
- 5.17.4.4 `template<class F> Linearphase_FIR<F> ADM::ADM<F>::m_lp_result` [private]
- 5.17.4.5 `template<class F> Linearphase_FIR<F> ADM::ADM<F>::m_decomb` [private]
- 5.17.4.6 `template<class F> F ADM::ADM<F>::m_beta` [private]
- 5.17.4.7 `template<class F> F ADM::ADM<F>::m_mu_beta` [private]
- 5.17.4.8 `template<class F> F ADM::ADM<F>::m_powerfilter_coeff` [private]
- 5.17.4.9 `template<class F> F ADM::ADM<F>::m_powerfilter_norm` [private]
- 5.17.4.10 `template<class F> F ADM::ADM<F>::m_powerfilter_state` [private]

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

- **adm.hh**

5.18 ADM::Delay<F> Class Template Reference

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

Public Member Functions

- **Delay** (F samples, F f_{design} , F f_s)
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

5.18.1 Detailed Description

```
template<class F>
class ADM::Delay< F >
```

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

5.18.2 Constructor & Destructor Documentation

```
5.18.2.1 template<class F > ADM::Delay< F >::Delay (
                F samples,
                F f_design,
                F fs )
```

Create a signal delay object.

Parameters

| | |
|-----------------|---|
| <i>samples</i> | number of samples to delay (may be non-integer) |
| <i>f_design</i> | subsampledelay is exact for this frequency |
| <i>fs</i> | sampling frequency |

```
5.18.2.2 template<class F > ADM::Delay< F >::~~Delay ( )
```

5.18.3 Member Function Documentation

```
5.18.3.1 template<class F > F ADM::Delay< F >::process (
                const F & in_sample ) [inline]
```

Apply delay to signal.

Parameters

| | |
|------------------|---------------------------------|
| <i>in_sample</i> | The current input signal sample |
|------------------|---------------------------------|

Returns

The computed output sample

5.18.4 Member Data Documentation**5.18.4.1 `template<class F> unsigned ADM::Delay<F>::m_fullsamples` [private]**

Integer part of delay.

5.18.4.2 `template<class F> F ADM::Delay<F>::m_coeff` [private]

coefficient for 1st order IIR lowpass filter which does the subsample delay

5.18.4.3 `template<class F> F ADM::Delay<F>::m_norm` [private]

normalization for the IIR subsample delay filter

5.18.4.4 `template<class F> F* ADM::Delay<F>::m_state` [private]

Ringbuffer: Delayline.

5.18.4.5 `template<class F> unsigned ADM::Delay<F>::m_now_in` [private]

current position for inserting new samples into m_state ringbuffer

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

- **adm.hh**

5.19 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

5.19.1 Detailed Description

```
template<class F>
class ADM::Linearphase_FIR< F >
```

An efficient linear-phase fir filter implementation.

5.19.2 Constructor & Destructor Documentation

```
5.19.2.1 template<class F > ADM::Linearphase_FIR< F >::Linearphase_FIR (
    unsigned order,
    const F * alphas )
```

Create linear-phase FIR filter.

Parameters

| | |
|---------------|---|
| <i>order</i> | filter order of this FIR filter. restriction: must be even. |
| <i>alphas</i> | 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.) |

5.19.2.2 `template<class F> ADM::Linearphase_FIR<F>::~~Linearphase_FIR ()`

5.19.3 Member Function Documentation

5.19.3.1 `template<class F> F ADM::Linearphase_FIR<F>::process (const F & in_sample) [inline]`

Filter one sample with this linear-phase FIR filter.

Parameters

| | |
|------------------|--------------------------|
| <i>in_sample</i> | the current input sample |
|------------------|--------------------------|

Returns

the computed output sample

5.19.4 Member Data Documentation

5.19.4.1 `template<class F> unsigned ADM::Linearphase_FIR<F>::m_order [private]`

The filter order of this linear-phase FIR filter.

5.19.4.2 `template<class F> F* ADM::Linearphase_FIR<F>::m_alphas [private]`

FIR filter coefficients.

Only $m_order / 2 + 1$ coefficients need to be stored since coefficients of linear-phase FIR filters are symmetric

5.19.4.3 `template<class F> F* ADM::Linearphase_FIR<F>::m_output [private]`

Ringbuffer for building future output.

5.19.4.4 `template<class F> unsigned ADM::Linearphase_FIR<F>::m_now [private]`

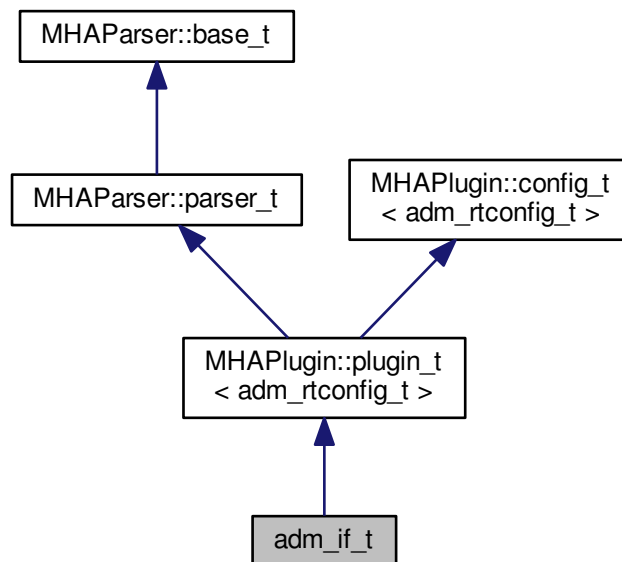
current start of ringbuffer

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

- **adm.hh**

5.20 adm_if_t Class Reference

Inheritance diagram for adm_if_t:



Public Member Functions

- **adm_if_t** (const **algo_comm_t** &ac, const std::string &thread_name, const std::string &algo_name)
- **mha_wave_t * process** (mha_wave_t *in)
- virtual void **prepare** (mhaconfig_t &)
- virtual void **release** ()

Private Member Functions

- void **update** ()
- bool **is_prepared** ()

Private Attributes

- **MHASignal::waveform_t * out**
- **MHAParser::vint_t front_channels**
- **MHAParser::vint_t rear_channels**
- **MHAParser::vfloat_t distances**

- **MHAParser::int_t lp_order**
- **MHAParser::int_t decomb_order**
- **MHAParser::int_t bypass**
- **MHAParser::float_t beta**
- **MHAParser::vfloat_t mu_beta**
- **MHAParser::vfloat_t tau_beta**
- **MHAParser::vfloat_mon_t coeff_lp**
- **MHAParser::vfloat_mon_t coeff_decomb**
- unsigned **input_channels**
- **mha_real_t srate**
- **MHAEvents::patchbay_t< adm_if_t > patchbay**

Additional Inherited Members

5.20.1 Constructor & Destructor Documentation

5.20.1.1 **adm_if_t::adm_if_t (**
 const algo_comm_t & ac,
 const std::string & thread_name,
 const std::string & algo_name)

5.20.2 Member Function Documentation

5.20.2.1 **mha_wave_t * adm_if_t::process (**
 mha_wave_t * in)

5.20.2.2 **void adm_if_t::prepare (**
 mhaconfig_t & cfg) [virtual]

Implements **MHAPLugin::plugin_t< adm_rtconfig_t >** (p. [661](#)).

5.20.2.3 **void adm_if_t::release (**
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t< adm_rtconfig_t >** (p. [661](#)).

5.20.2.4 void adm_if_t::update () [private]

5.20.2.5 bool adm_if_t::is_prepared () [inline], [private]

5.20.3 Member Data Documentation

5.20.3.1 MHASignal::waveform_t* adm_if_t::out [private]

5.20.3.2 MHAParser::vint_t adm_if_t::front_channels [private]

5.20.3.3 MHAParser::vint_t adm_if_t::rear_channels [private]

5.20.3.4 MHAParser::vfloat_t adm_if_t::distances [private]

5.20.3.5 MHAParser::int_t adm_if_t::lp_order [private]

5.20.3.6 MHAParser::int_t adm_if_t::decomb_order [private]

5.20.3.7 MHAParser::int_t adm_if_t::bypass [private]

5.20.3.8 MHAParser::float_t adm_if_t::beta [private]

5.20.3.9 MHAParser::vfloat_t adm_if_t::mu_beta [private]

5.20.3.10 MHAParser::vfloat_t adm_if_t::tau_beta [private]

5.20.3.11 MHAParser::vfloat_mon_t adm_if_t::coeff_lp [private]

5.20.3.12 MHAParser::vfloat_mon_t adm_if_t::coeff_decomb [private]

5.20.3.13 unsigned adm_if_t::input_channels [private]

5.20.3.14 mha_real_t adm_if_t::srate [private]

5.20.3.15 MHAEvents::patchbay_t<adm_if_t> adm_if_t::patchbay [private]

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

- **adm.cpp**

5.21 adm_rtconfig_t Class Reference

Public Types

- typedef **ADM::ADM**< mha_real_t > **adm_t**

Public Member Functions

- **adm_rtconfig_t** (unsigned nchannels_in, unsigned nchannels_out, const std::vector< int > &**front_channels**, const std::vector< int > &**rear_channels**, const **mha_real_t** fs, const std::vector< **mha_real_t** > &distances, const int lp_order, const int decomb_order, const std::vector< **mha_real_t** > &tau_beta, const std::vector< **mha_real_t** > &mu_beta)
 - Construct new ADMs.*
- virtual ~**adm_rtconfig_t** ()
- size_t **num_adms** () const
- **adm_t** & **adm** (unsigned index)
 - Returns adm object number index.*
- int **front_channel** (unsigned index) const
 - Returns index of front channel for adm number index.*
- int **rear_channel** (unsigned index) const
 - Returns index of rear channel for adm number index.*

Private Member Functions

- void **check_index** (unsigned index) const
 - Index checking for all internal arrays.*

Private Attributes

- std::vector< int > **front_channels**
 - Indices of channels containing the signals from the front microphones.*
- std::vector< int > **rear_channels**
 - Indices of channels containing the signals from the rear microphones.*
- **MHASignal::waveform_t** * **lp_coeffs**
 - Lowpass filter coefficients.*
- std::vector< **MHASignal::waveform_t** * > **decomb_coeffs**
 - Decomb-Filter coefficients.*
- std::vector< **adm_t** * > **adms**
 - ADMs.*

5.21.1 Member Typedef Documentation

5.21.1.1 typedef ADM::ADM<mha_real_t> adm_rtconfig_t::adm_t

5.21.2 Constructor & Destructor Documentation

```

5.21.2.1 adm_rtconfig_t::adm_rtconfig_t (
    unsigned nchannels_in,
    unsigned nchannels_out,
    const std::vector< int > & front_channels,
    const std::vector< int > & rear_channels,
    const mha_real_t fs,
    const std::vector< mha_real_t > & distances,
    const int lp_order,
    const int decomb_order,
    const std::vector< mha_real_t > & tau_beta,
    const std::vector< mha_real_t > & mu_beta )

```

Construct new ADMs.

Used when configuration changes.

Parameters

| | |
|-----------------------|--|
| <i>nchannels_in</i> | Number of input channels |
| <i>nchannels_out</i> | Number of output channels |
| <i>front_channels</i> | Parser's front_channels setting |
| <i>rear_channels</i> | Parser's front_channels setting |
| <i>fs</i> | Sampling rate / Hz |
| <i>distances</i> | Distances between microphones / m |
| <i>lp_order</i> | Filter order of FIR lowpass filter for adaptation |
| <i>decomb_order</i> | Filter order of FIR compensation filter (compensates for comb filter characteristic) |
| <i>tau_beta</i> | Time constants of the lowpass filter used for averaging the power of the output signal used for adaptation |
| <i>mu_beta</i> | Adaptation step sizes |

```

5.21.2.2 adm_rtconfig_t::~~adm_rtconfig_t ( ) [virtual]

```

5.21.3 Member Function Documentation

```

5.21.3.1 void adm_rtconfig_t::check_index (
    unsigned index ) const [inline], [private]

```

Index checking for all internal arrays.

Exceptions

| | |
|---|------------------------|
| <i>MHA_Error</i> (p. 387) | if index out of range. |
|---|------------------------|

5.21.3.2 `size_t adm_rtconfig_t::num_adms () const [inline]`

5.21.3.3 `adm_t& adm_rtconfig_t::adm (unsigned index) [inline]`

Returns adm object number index.

5.21.3.4 `int adm_rtconfig_t::front_channel (unsigned index) const [inline]`

Returns index of front channel for adm number index.

5.21.3.5 `int adm_rtconfig_t::rear_channel (unsigned index) const [inline]`

Returns index of rear channel for adm number index.

5.21.4 Member Data Documentation

5.21.4.1 `std::vector<int> adm_rtconfig_t::front_channels [private]`

Indices of channels containing the signals from the front microphones.

5.21.4.2 `std::vector<int> adm_rtconfig_t::rear_channels [private]`

Indices of channels containing the signals from the rear microphones.

5.21.4.3 `MHASignal::waveform_t* adm_rtconfig_t::lp_coeffs [private]`

Lowpass filter coefficients.

5.21.4.4 `std::vector<MHASignal::waveform_t*> adm_rtconfig_t::decomb_coeffs [private]`

Decomb-Filter coefficients.

5.21.4.5 `std::vector<adm_t*> adm_rtconfig_t::adms [private]`

ADMs.

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

- **adm.cpp**

5.22 algo_comm_t Struct Reference

A reference handle for algorithm communication variables.

Public Attributes

- void * **handle**
AC variable control handle.
- int(* **insert_var**)(void *, const char *, **comm_var_t**)
Register an AC variable.
- int(* **insert_var_int**)(void *, const char *, int *)
Register an int as an AC variable.
- int(* **insert_var_float**)(void *, const char *, float *)
Register a float as an AC variable.
- int(* **remove_var**)(void *, const char *)
Remove an AC variable.
- int(* **remove_ref**)(void *, void *)
Remove all AC variable which refer to address.
- int(* **is_var**)(void *, const char *)
Test if an AC variable exists.
- int(* **get_var**)(void *, const char *, **comm_var_t** *)
Get the variable handle of an AC variable.
- int(* **get_var_int**)(void *, const char *, int *)
Get the value of an int AC variable.
- int(* **get_var_float**)(void *, const char *, float *)
Get the value of a float AC variable.
- int(* **get_entries**)(void *, char *, unsigned int)
Return a space separated list of all variable names.
- const char *(* **get_error**)(int)
Convert AC error codes into human readable error messages.

5.22.1 Detailed Description

A reference handle for algorithm communication variables.

This structure contains a coontrol handle and a set of function pointers for sharing variables within one processing chain. See also section **Communication between algorithms** (p. 27).

5.22.2 Member Data Documentation

5.22.2.1 algo_comm_t::handle

AC variable control handle.

5.22.2.2 algo_comm_t::insert_var

Register an AC variable.

This function can register a variable to be shared within one chain. If a variable of this name exists it will be overwritten.

Parameters

| | |
|----------|---|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable. May not be empty. Must not contain space character. The name is copied, therefore it is allowed that the char array pointed to gets invalid after return. |
| <i>v</i> | variable handle of type comm_var_t (p. 209) |

Returns

Error code or zero on success

5.22.2.3 algo_comm_t::insert_var_int

Register an int as an AC variable.

This function can register an int variable to be shared with other algorithms. It behaves similar to `ac.insert_var`.

Parameters

| | |
|----------|-------------------------|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |
| <i>v</i> | pointer on the variable |

Returns

Error code or zero on success

5.22.2.4 algo_comm_t::insert_var_float

Register a float as an AC variable.

This function can register a float variable to be shared with other algorithms. It behaves similar to `ac.insert_var`.

Parameters

| | |
|----------|-------------------------|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |
| <i>v</i> | pointer on the variable |

Returns

Error code or zero on success

5.22.2.5 algo_comm_t::remove_var

Remove an AC variable.

Remove (unregister) an AC variable. After calling this function, the variable is not available to `ac.is_var` or `ac.get_var`. The data pointer is not affected.

Parameters

| | |
|----------|--------------------------------|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable to be removed |

Returns

Error code or zero on success

5.22.2.6 algo_comm_t::remove_ref

Remove all AC variable which refer to address.

This function removes all AC variables whos data field points to the given address.

Parameters

| | |
|----------|--|
| <i>h</i> | AC handle |
| <i>p</i> | address which should not be referred to any more |

Returns

Error code or zero on success

5.22.2.7 algo_comm_t::is_var

Test if an AC variable exists.

This function tests if an AC variable of a given name exists. Use `ac.get_var` to get information about the variables type and dimension.

Parameters

| | |
|----------|------------------|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |

Returns

1 if the variable exists, 0 otherwise

5.22.2.8 algo_comm_t::get_var

Get the variable handle of an AC variable.

This function returns the variable handle **comm_var_t** (p. 209) of a variable of the given name. If no variable of that name exists, an error code is returned.

Parameters

| | |
|----------|---------------------------------|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |
| <i>v</i> | pointer to a AC variable object |

Returns

Error code or zero on success

5.22.2.9 algo_comm_t::get_var_int

Get the value of an int AC variable.

This function returns the value of an int AC variable of the given name. If no variable exists, the variable type is mismatching or more than one entry is registered, a corresponding error code is returned. This is a special version of `ac.get_var`.

Parameters

| | |
|----------|--|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |
| <i>v</i> | pointer on an int variable to store the result |

Returns

Error code or zero on success

5.22.2.10 algo_comm_t::get_var_float

Get the value of a float AC variable.

This function returns the value of a float AC variable of the given name. If no variable exists, the variable type is mismatching or more than one entry is registered, a corresponding error code is returned. This is a special version of `ac.get_var`.

Parameters

| | |
|----------|---|
| <i>h</i> | AC handle |
| <i>n</i> | name of variable |
| <i>v</i> | pointer on a float variable to store the result |

Returns

Error code or zero on success

5.22.2.11 algo_comm_t::get_entries

Return a space separated list of all variable names.

This function returns the names of all registered variables, separated by a single space.

Parameters

| | |
|----------|-----------|
| <i>h</i> | AC handle |
|----------|-----------|

Return values

| | |
|------------|-----------------------------------|
| <i>ret</i> | Character buffer for return value |
|------------|-----------------------------------|

Parameters

| | |
|------------|----------------------------|
| <i>len</i> | length of character buffer |
|------------|----------------------------|

Returns

Error code or zero on success. -1: invalid ac handle. -3: not enough room in character buffer to store all variable names.

5.22.2.12 algo_comm_t::get_error

Convert AC error codes into human readable error messages.

Parameters

| | |
|----------|------------|
| <i>e</i> | Error code |
|----------|------------|

Returns

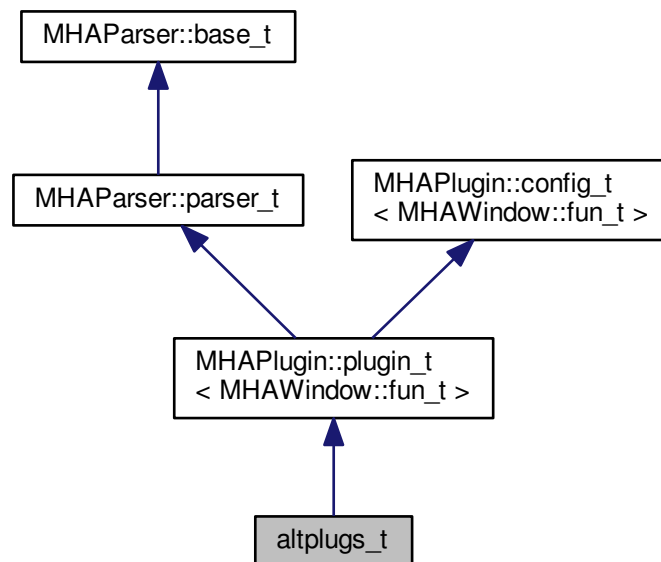
Error message

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

- **mha.h**
- **mha_algo_comm.cpp**

5.23 altplugins_t Class Reference

Inheritance diagram for altplugins_t:



Public Member Functions

- **altplugins_t** (**algo_comm_t** iac, const char *chain, const char *algo)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** (**mha_wave_t** *, **mha_wave_t** **)
- void **process** (**mha_spec_t** *, **mha_wave_t** **)
- void **process** (**mha_wave_t** *, **mha_spec_t** **)
- void **process** (**mha_spec_t** *, **mha_spec_t** **)
- virtual std::string **parse** (const std::string &arg)
- virtual void **parse** (const char *a1, char *a2, unsigned int a3)

Private Member Functions

- void **event_set_plugs** ()
- void **event_add_plug** ()
- void **event_delete_plug** ()
- void **event_select_plug** ()
- void **update_selector_list** ()
- void **update_ramplen** ()
- void **proc_ramp** (mha_wave_t *s)

Private Attributes

- **MHAParser::bool_t** use_own_ac
- **MHAParser::vstring_t** parser_plugs
- **MHAParser::string_t** add_plug
- **MHAParser::string_t** delete_plug
- **MHAParser::float_t** ramplen
- **MHAParser::kw_t** select_plug
- **MHAParser::parser_t** current
- **MHAParser::vstring_mon_t** nondefault_labels
- **std::vector< mhaplug_cfg_t * >** plugs
- **mhaplug_cfg_t *** selected_plug
- **MHAEvents::patchbay_t< altplugins_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

5.23.1 Constructor & Destructor Documentation

5.23.1.1 altplugins_t::altplugins_t (
 algo_comm_t iac,
 const char * chain,
 const char * algo)

5.23.2 Member Function Documentation

5.23.2.1 void altplugins_t::prepare (
 mhaconfig_t & cf) [virtual]

Implements **MHAPugin::plugin_t< MHAWindow::fun_t >** (p. 661).

5.23.2.2 void altplugins_t::release (
void) [virtual]

Reimplemented from **MHAPugin::plugin_t** < **MHAWindow::fun_t** > (p. [661](#)).

5.23.2.3 void altplugins_t::process (
mha_wave_t * *sIn*,
mha_wave_t ** *sOut*)

5.23.2.4 void altplugins_t::process (
mha_spec_t * *sIn*,
mha_wave_t ** *sOut*)

5.23.2.5 void altplugins_t::process (
mha_wave_t * *sIn*,
mha_spec_t ** *sOut*)

5.23.2.6 void altplugins_t::process (
mha_spec_t * *sIn*,
mha_spec_t ** *sOut*)

5.23.2.7 std::string altplugins_t::parse (
const std::string & *arg*) [virtual]

Reimplemented from **MHAParser::base_t** (p. [571](#)).

5.23.2.8 virtual void altplugins_t::parse (
const char * *a1*,
char * *a2*,
unsigned int *a3*) [inline],[virtual]

Reimplemented from **MHAParser::base_t** (p. [572](#)).

5.23.2.9 void altplugins_t::event_set_plugs () [private]

5.23.2.10 void altplugins_t::event_add_plug () [private]

5.23.2.11 void altplugins_t::event_delete_plug () [private]

5.23.2.12 void altplugins_t::event_select_plug () [private]

5.23.2.13 void altplugins_t::update_selector_list () [private]

5.23.2.14 void altplugins_t::update_ramplen () [private]

5.23.2.15 void altplugins_t::proc_ramp (
 mha_wave_t* s) [private]

5.23.3 Member Data Documentation

5.23.3.1 MHAParser::bool_t altplugins_t::use_own_ac [private]

5.23.3.2 MHAParser::vstring_t altplugins_t::parser_plugs [private]

5.23.3.3 MHAParser::string_t altplugins_t::add_plug [private]

5.23.3.4 MHAParser::string_t altplugins_t::delete_plug [private]

5.23.3.5 MHAParser::float_t altplugins_t::ramplen [private]

5.23.3.6 MHAParser::kw_t altplugins_t::select_plug [private]

5.23.3.7 MHAParser::parser_t altplugins_t::current [private]

5.23.3.8 MHAParser::vstring_mon_t altplugins_t::nondefault_labels [private]

5.23.3.9 std::vector<mhaplug_cfg_t*> altplugins_t::plugs [private]

5.23.3.10 mhaplug_cfg_t* altplugins_t::selected_plug [private]

5.23.3.11 MHAEvents::patchbay_t<altplugins_t> altplugins_t::patchbay [private]

5.23.3.12 MHASignal::waveform_t* altplugins_t::fallback_wave [private]

5.23.3.13 MHASignal::spectrum_t* altplugins_t::fallback_spec [private]

5.23.3.14 mhaconfig_t altplugins_t::cfin [private]

5.23.3.15 mhaconfig_t altplugins_t::cfout [private]

5.23.3.16 bool altplugins_t::prepared [private]

5.23.3.17 bool altplugins_t::added_via_plugs [private]

5.23.3.18 unsigned int altplugins_t::ramp_counter [private]

5.23.3.19 unsigned int altplugins_t::ramp_len [private]

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

- altplugins.cpp

5.24 analysepath_t Class Reference

Public Member Functions

- **analysepath_t** (unsigned int *nchannels_in*, unsigned int *outer_fragsize*, unsigned int *inner_fragsize*, int **priority**, **MHAProc_wave2wave_t** *inner_proc_wave2wave*, **MHAProc_wave2spec_t** *inner_proc_wave2spec*, void **ilibdata*, **algo_comm_t** *outer_ac*, const **MHA_AC::acspace2matrix_t** &*acspace_template*, **mha_domain_t** *inner_out_domain*, unsigned int *fifo_len_blocks*)
- virtual **~analysepath_t** ()
- void **rt_process** (**mha_wave_t** *)
- virtual int **svc** ()

Private Attributes

- **MHAProc_wave2wave_t** *inner_process_wave2wave*
- **MHAProc_wave2spec_t** *inner_process_wave2spec*
- **MHASignal::waveform_t** *inner_input*
- void * *libdata*
- **mha_fifo_t** < **mha_real_t** > *wave_fifo*
- **mha_fifo_t** < **MHA_AC::acspace2matrix_t** > *ac_fifo*
- **MHA_AC::acspace2matrix_t** *inner_ac_copy*
- **MHA_AC::acspace2matrix_t** *outer_ac_copy*
- **algo_comm_t** *outer_ac*
- **mha_domain_t** *inner_out_domain*
- **MHA_Error** *inner_error*
- bool *has_inner_error*
- bool *flag_terminate_inner_thread*
- int *input_to_process*
- **pthread_mutex_t** *ProcessMutex*
- **pthread_attr_t** *attr*
- struct sched_param *priority*
- int *scheduler*
- **pthread_t** *thread*
- **pthread_cond_t** *cond_to_process*

5.24.1 Constructor & Destructor Documentation

5.24.1.1 **analysepath_t::analysepath_t** (
 unsigned int *nchannels_in*,
 unsigned int *outer_fragsize*,
 unsigned int *inner_fragsize*,
 int *priority*,
 MHAProc_wave2wave_t *inner_proc_wave2wave*,
 MHAProc_wave2spec_t *inner_proc_wave2spec*,
 void * *ilibdata*,
 algo_comm_t *outer_ac*,
 const **MHA_AC::acspace2matrix_t** & *acspace_template*,
 mha_domain_t *inner_out_domain*,
 unsigned int *fifo_len_blocks*)

5.24.1.2 analysepath_t::~analysepath_t () [virtual]

5.24.2 Member Function Documentation

5.24.2.1 void analysepath_t::rt_process (
 mha_wave_t * outer_input)

5.24.2.2 int analysepath_t::svc () [virtual]

5.24.3 Member Data Documentation

5.24.3.1 MHAProc_wave2wave_t analysepath_t::inner_process_wave2wave [private]

5.24.3.2 MHAProc_wave2spec_t analysepath_t::inner_process_wave2spec [private]

5.24.3.3 MHASignal::waveform_t analysepath_t::inner_input [private]

5.24.3.4 void* analysepath_t::libdata [private]

5.24.3.5 mha_fifo_t<mha_real_t> analysepath_t::wave_fifo [private]

5.24.3.6 mha_fifo_t<MHA_AC::acspace2matrix_t> analysepath_t::ac_fifo [private]

5.24.3.7 MHA_AC::acspace2matrix_t analysepath_t::inner_ac_copy [private]

5.24.3.8 MHA_AC::acspace2matrix_t analysepath_t::outer_ac_copy [private]

5.24.3.9 algo_comm_t analysepath_t::outer_ac [private]

5.24.3.10 mha_domain_t analysepath_t::inner_out_domain [private]

5.24.3.11 MHA_Error analysepath_t::inner_error [private]

5.24.3.12 bool analysepath_t::has_inner_error [private]

5.24.3.13 bool analysepath_t::flag_terminate_inner_thread [private]

5.24.3.14 int analysepath_t::input_to_process [private]

5.24.3.15 pthread_mutex_t analysepath_t::ProcessMutex [private]

5.24.3.16 pthread_attr_t analysepath_t::attr [private]

5.24.3.17 struct sched_param analysepath_t::priority [private]

5.24.3.18 int analysepath_t::scheduler [private]

5.24.3.19 pthread_t analysepath_t::thread [private]

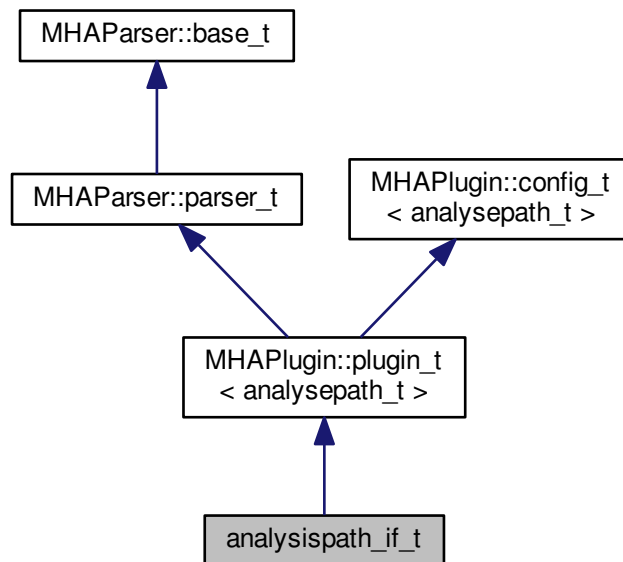
5.24.3.20 pthread_cond_t analysepath_t::cond_to_process [private]

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

- **analysispath.cpp**

5.25 analysispath_if_t Class Reference

Inheritance diagram for analysispath_if_t:



Public Member Functions

- **analysispath_if_t** (algo_comm_t, std::string, std::string)
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()
- **~analysispath_if_t** ()

Private Member Functions

- void **loadlib** ()

Private Attributes

- **MHAEvents::patchbay_t< analysispath_if_t > patchbay**
- **MHAParser::string_t libname**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t fifolen**
- **MHAParser::int_t priority**
- **MHAParser::vstring_t vars**
- **plug_t * plug**
- std::string **chain**
- std::string **algo**
- **MHA_AC::acspace2matrix_t * acspace_template**

Additional Inherited Members

5.25.1 Constructor & Destructor Documentation

5.25.1.1 `analysispath_if_t::analysispath_if_t (`
 `algo_comm_t iac,`
 `std::string th,`
 `std::string al)`

5.25.1.2 `analysispath_if_t::~~analysispath_if_t ()`

5.25.2 Member Function Documentation

5.25.2.1 `mha_wave_t * analysispath_if_t::process (`
 `mha_wave_t * s)`

5.25.2.2 `void analysispath_if_t::prepare (`
 `mhaconfig_t & conf) [virtual]`

Implements **MHAPlugin::plugin_t< analysepath_t >** (p. 661).

5.25.2.3 `void analysispath_if_t::release (`
 `void) [virtual]`

Reimplemented from **MHAPlugin::plugin_t< analysepath_t >** (p. 661).

5.25.2.4 `void analysispath_if_t::loadlib () [private]`

5.25.3 Member Data Documentation

5.25.3.1 **MHAEvents::patchbay_t< analysispath_if_t > analysispath_if_t::patchbay**
 [private]

5.25.3.2 **MHAParser::string_t analysispath_if_t::libname** [private]

5.25.3.3 **MHAParser::int_t analysispath_if_t::fragsize** [private]

5.25.3.4 **MHAParser::int_t analysispath_if_t::fifolen** [private]

5.25.3.5 **MHAParser::int_t analysispath_if_t::priority** [private]

5.25.3.6 **MHAParser::vstring_t analysispath_if_t::vars** [private]

5.25.3.7 **plug_t* analysispath_if_t::plug** [private]

5.25.3.8 **std::string analysispath_if_t::chain** [private]

5.25.3.9 **std::string analysispath_if_t::algo** [private]

5.25.3.10 **MHA_AC::acspace2matrix_t* analysispath_if_t::acspace_template** [private]

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

- **analysispath.cpp**

5.26 AuditoryProfile::fmap_t Class Reference

A class to store frequency dependent data (e.g., HTL and UCL).

Inherits `map< mha_real_t, mha_real_t >`.

Public Member Functions

- `std::vector< mha_real_t > get_frequencies () const`
Return configured frequencies.
- `std::vector< mha_real_t > get_values () const`
Return stored values corresponding to the frequencies.
- `bool isempty () const`

5.26.1 Detailed Description

A class to store frequency dependent data (e.g., HTL and UCL).

5.26.2 Member Function Documentation

5.26.2.1 `std::vector< mha_real_t > AuditoryProfile::fmap_t::get_frequencies () const`

Return configured frequencies.

5.26.2.2 `std::vector< mha_real_t > AuditoryProfile::fmap_t::get_values () const`

Return stored values corresponding to the frequencies.

5.26.2.3 `bool AuditoryProfile::fmap_t::isempty () const` `[inline]`

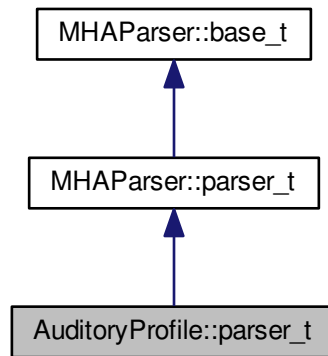
The documentation for this class was generated from the following files:

- `auditory_profile.h`
- `auditory_profile.cpp`

5.27 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

5.27.1 Detailed Description

Class to make the auditory profile accessible through the parser interface.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 AuditoryProfile::parser_t::parser_t ()

5.27.3 Member Function Documentation

5.27.3.1 AuditoryProfile::profile_t AuditoryProfile::parser_t::get_current_profile ()

5.27.4 Member Data Documentation

5.27.4.1 AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::L [private]

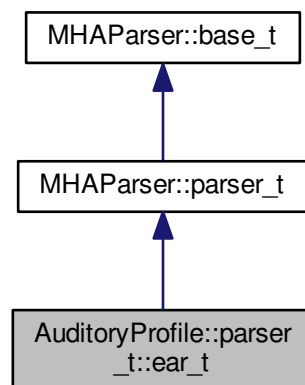
5.27.4.2 AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::R [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.28 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

5.28.1 Constructor & Destructor Documentation

5.28.1.1 AuditoryProfile::parser_t::ear_t::ear_t ()

5.28.2 Member Function Documentation

5.28.2.1 AuditoryProfile::profile_t::ear_t AuditoryProfile::parser_t::ear_t::get_ear () const

5.28.3 Member Data Documentation

5.28.3.1 AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::HTL [private]

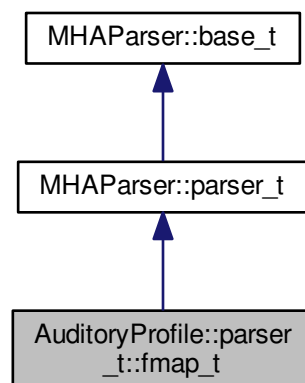
5.28.3.2 AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::UCL [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.29 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

5.29.1 Constructor & Destructor Documentation

5.29.1.1 **AuditoryProfile::parser_t::fmap_t::fmap_t** (
 const std::string & *name*,
 const std::string & *help*)

5.29.2 Member Function Documentation

5.29.2.1 **AuditoryProfile::fmap_t AuditoryProfile::parser_t::fmap_t::get_fmap** () const

5.29.2.2 **void AuditoryProfile::parser_t::fmap_t::validate** () [private]

5.29.3 Member Data Documentation

5.29.3.1 **MHAEvents::patchbay_t**<**AuditoryProfile::parser_t::fmap_t**>
AuditoryProfile::parser_t::fmap_t::patchbay [private]

5.29.3.2 **MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::f** [private]

5.29.3.3 **MHAParser::vfloat_t AuditoryProfile::parser_t::fmap_t::value** [private]

5.29.3.4 **std::string AuditoryProfile::parser_t::fmap_t::name_** [private]

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.30 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.

5.30.1 Detailed Description

The Auditory Profile class.

See definition of auditory profile

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

Currently only the audiogram data is stored.

5.30.2 Member Function Documentation

5.30.2.1 AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::get_ear (unsigned int *channel*) const `[inline]`

Return ear information of channel number.

5.30.3 Member Data Documentation

5.30.3.1 **AuditoryProfile::profile_t::ear_t** AuditoryProfile::profile_t::L

Left ear data.

5.30.3.2 **AuditoryProfile::profile_t::ear_t** AuditoryProfile::profile_t::R

Right ear data.

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

- **auditory_profile.h**

5.31 **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

5.31.1 Detailed Description

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

5.31.2 Member Function Documentation

5.31.2.1 void **AuditoryProfile::profile_t::ear_t::convert_empty2normal** ()

5.31.3 Member Data Documentation

5.31.3.1 **AuditoryProfile::fmap_t** AuditoryProfile::profile_t::ear_t::HTL

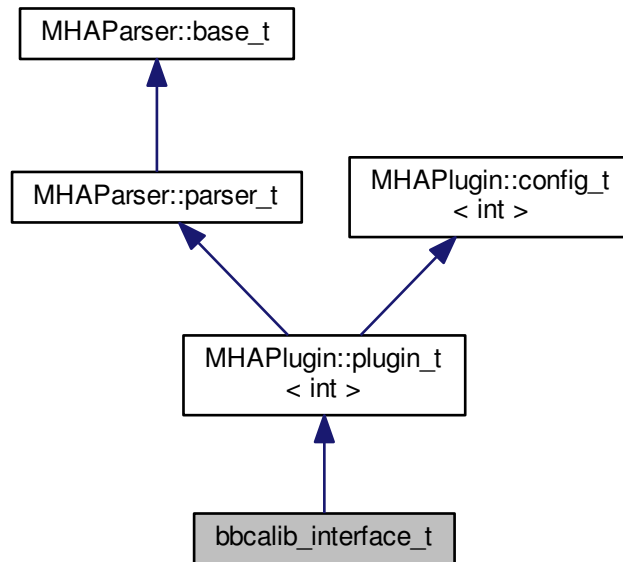
5.31.3.2 **AuditoryProfile::fmap_t** AuditoryProfile::profile_t::ear_t::UCL

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

- **auditory_profile.h**
- **auditory_profile.cpp**

5.32 bbcalib_interface_t Class Reference

Inheritance diagram for bbcalib_interface_t:



Public Member Functions

- **bbcalib_interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **~bbcalib_interface_t** ()
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Attributes

- **calibrator_t** **calib_in**
- **calibrator_t** **calib_out**
- **MHAParser::mhapuginloader_t** **plugloader**

Additional Inherited Members

5.32.1 Constructor & Destructor Documentation

5.32.1.1 `bbcalib_interface_t::bbcalib_interface_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

5.32.1.2 `bbcalib_interface_t::~~bbcalib_interface_t ()`

5.32.2 Member Function Documentation

5.32.2.1 `mha_wave_t * bbcalib_interface_t::process (`
 `mha_wave_t * s)`

5.32.2.2 `void bbcalib_interface_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPLugin::plugin_t< int >** (p. 661).

5.32.2.3 `void bbcalib_interface_t::release (`
 `void) [virtual]`

Reimplemented from **MHAPLugin::plugin_t< int >** (p. 661).

5.32.3 Member Data Documentation

5.32.3.1 `calibrator_t bbcalib_interface_t::calib_in [private]`

5.32.3.2 `calibrator_t bbcalib_interface_t::calib_out [private]`

5.32.3.3 `MHAParser::mhapluginloader_t bbcalib_interface_t::plugloader [private]`

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

- **transducers.cpp**

5.33 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 **firfir2ftlen** (unsigned int, const std::vector< std::vector< float > > &)

Private Attributes

- **MHAFilter::fftfilter_t** fir
- **MHASignal::quantizer_t** quant
- **MHASignal::waveform_t** gain
- **softclipper_t** softclip
- bool **b_is_input**
- bool **b_use_fir**
- bool **b_use_clipping**
- **MHASignal::loop_wavefragment_t** speechnoise
- **MHASignal::loop_wavefragment_t::playback_mode_t** pmode

5.33.1 Constructor & Destructor Documentation

5.33.1.1 calibrator_runtime_layer_t::calibrator_runtime_layer_t (
 bool *is_input*,
 const mhaconfig_t & *tf*,
 calibrator_variables_t & *vars*)

5.33.2 Member Function Documentation

5.33.2.1 mha_real_t calibrator_runtime_layer_t::process (
 mha_wave_t ** *s*)

5.33.2.2 unsigned int calibrator_runtime_layer_t::firfirlen (
 const std::vector< std::vector< float > > & *fir*) [static], [private]

5.33.2.3 unsigned int calibrator_runtime_layer_t::firfir2ftlen (
 unsigned int *fragsize*,
 const std::vector< std::vector< float > > & *fir*) [static], [private]

5.33.3 Member Data Documentation

5.33.3.1 MHAFilter::fftfilter_t calibrator_runtime_layer_t::fir [private]

5.33.3.2 MHASignal::quantizer_t calibrator_runtime_layer_t::quant [private]

5.33.3.3 MHASignal::waveform_t calibrator_runtime_layer_t::gain [private]

5.33.3.4 **softclipper_t** **calibrator_runtime_layer_t::softclip** [private]

5.33.3.5 **bool** **calibrator_runtime_layer_t::b_is_input** [private]

5.33.3.6 **bool** **calibrator_runtime_layer_t::b_use_fir** [private]

5.33.3.7 **bool** **calibrator_runtime_layer_t::b_use_clipping** [private]

5.33.3.8 **MHASignal::loop_wavefragment_t** **calibrator_runtime_layer_t::speechnoise**
[private]

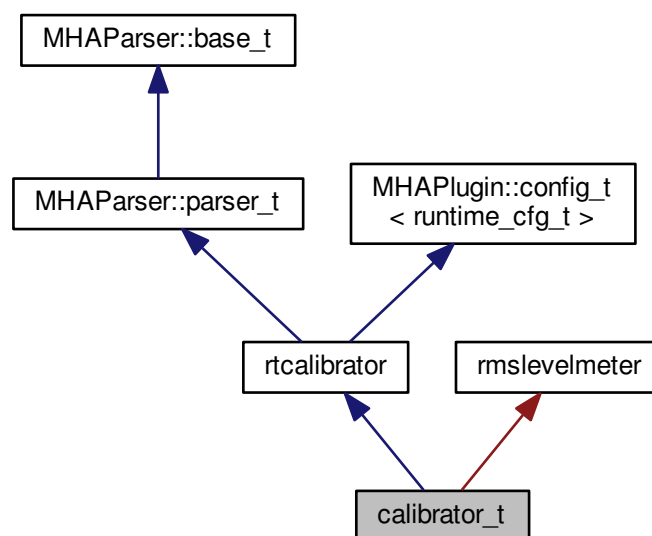
5.33.3.9 **MHASignal::loop_wavefragment_t::playback_mode_t** **calibrator_runtime_layer_t::pmode** [private]

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

- **transducers.cpp**

5.34 calibrator_t Class Reference

Inheritance diagram for **calibrator_t**:



Public Member Functions

- **calibrator_t** (**algo_comm_t**, bool *is_input*)
- void **prepare** (**mhaconfig_t** &*tf*)
- void **release** ()
- **mha_wave_t** * **process** (**mha_wave_t** **s*)

Private Member Functions

- void **update** ()
- void **update_tau_level** ()
- void **read_levels** ()

Private Attributes

- bool **b_is_input**
- **MHAEvents::patchbay_t** < **calibrator_t** > **patchbay**
- **calibrator_variables_t** **vars**
- bool **prepared**

Additional Inherited Members

5.34.1 Constructor & Destructor Documentation

5.34.1.1 **calibrator_t::calibrator_t** (
 algo_comm_t *iac*,
 bool *is_input*)

5.34.2 Member Function Documentation

5.34.2.1 void **calibrator_t::prepare** (
 mhaconfig_t & *tf*) [inline], [virtual]

Implements **MHAPlugin::plugin_t** < **runtime_cfg_t** > (p. 661).

5.34.2.2 void **calibrator_t::release** (
 void) [inline], [virtual]

Reimplemented from **MHAPlugin::plugin_t** < **runtime_cfg_t** > (p. 661).

5.34.2.3 `mha_wave_t* calibrator_t::process (`
 `mha_wave_t* s)`

5.34.2.4 `void calibrator_t::update () [private]`

5.34.2.5 `void calibrator_t::update_tau_level () [private]`

5.34.2.6 `void calibrator_t::read_levels () [private]`

5.34.3 Member Data Documentation

5.34.3.1 `bool calibrator_t::b_is_input [private]`

5.34.3.2 `MHAEvents::patchbay_t<calibrator_t> calibrator_t::patchbay [private]`

5.34.3.3 `calibrator_variables_t calibrator_t::vars [private]`

5.34.3.4 `bool calibrator_t::prepared [private]`

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

- **transducers.cpp**

5.35 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

5.35.1 Constructor & Destructor Documentation

5.35.1.1 calibrator_variables_t::calibrator_variables_t (
 bool *is_input*,
 MHAParser::parser_t & *parent*)

5.35.2 Member Data Documentation

5.35.2.1 MHAParser::vfloat_t calibrator_variables_t::peaklevel

5.35.2.2 MHAParser::mfloat_t calibrator_variables_t::fir

5.35.2.3 MHAParser::int_t calibrator_variables_t::nbits

5.35.2.4 MHAParser::float_t calibrator_variables_t::tau_level

5.35.2.5 MHAParser::kw_t calibrator_variables_t::spnoise_mode

5.35.2.6 MHAParser::vint_t calibrator_variables_t::spnoise_channels

5.35.2.7 MHAParser::float_t calibrator_variables_t::spnoise_level

5.35.2.8 MHAParser::vfloat_mon_t calibrator_variables_t::rmslevel

5.35.2.9 MHAParser::parser_t calibrator_variables_t::spnoise_parser

5.35.2.10 MHAParser::float_mon_t calibrator_variables_t::srate

5.35.2.11 MHAParser::int_mon_t calibrator_variables_t::fragsize

5.35.2.12 MHAParser::int_mon_t calibrator_variables_t::num_channels

5.35.2.13 MHAParser::parser_t calibrator_variables_t::config_parser

5.35.2.14 softclipper_variables_t calibrator_variables_t::softclip

5.35.2.15 MHAParser::bool_t calibrator_variables_t::do_clipping

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

- **transducers.cpp**

5.36 `cfg_t` Class Reference

Public Member Functions

- **`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*)
- **`cfg_t`** (unsigned int, unsigned int)
- **`cfg_t`** (**`mhaconfig_t`** *chcfg*, **`mha_real_t`** *newlev*, bool *replace*, **`mha_real_t`** *len*)
- void **`process`** (**`mha_wave_t`** *)
- void **`process`** (**`mha_spec_t`** *)

Public Attributes

- **`mha_real_t`** *start_lin*
- **`mha_real_t`** *alpha*
- **`MHAFilter::o1flt_lowpass_t`** *attack*
- **`MHAFilter::o1flt_maxtrack_t`** *decay*
- unsigned int *channel*

Private Attributes

- **`mha_real_t`** *gain_wave_*
- **`mha_real_t`** *gain_spec_*
- bool *replace_*
- bool *use_frozen_*
- **`MHASignal::waveform_t`** *frozen_noise_*
- unsigned int *pos*

5.36.1 Constructor & Destructor Documentation

5.36.1.1 `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 )
```

5.36.1.2 `cfg_t::cfg_t` (

```
unsigned int ichannel,
unsigned int numchannels )
```

5.36.1.3 `cfg_t::cfg_t (`
 `mhaconfig_t chcfg,`
 `mha_real_t newlev,`
 `bool replace,`
 `mha_real_t len)`

5.36.2 Member Function Documentation

5.36.2.1 `void cfg_t::process (`
 `mha_wave_t * s)` `[inline]`

5.36.2.2 `void cfg_t::process (`
 `mha_spec_t * s)` `[inline]`

5.36.3 Member Data Documentation

5.36.3.1 `mha_real_t cfg_t::start_lin`

5.36.3.2 `mha_real_t cfg_t::alpha`

5.36.3.3 `MHAFilter::o1flt_lowpass_t cfg_t::attack`

5.36.3.4 `MHAFilter::o1flt_maxtrack_t cfg_t::decay`

5.36.3.5 `unsigned int cfg_t::channel`

5.36.3.6 `mha_real_t cfg_t::gain_wave_` `[private]`

5.36.3.7 `mha_real_t cfg_t::gain_spec_` `[private]`

5.36.3.8 `bool cfg_t::replace_` `[private]`

5.36.3.9 `bool cfg_t::use_frozen_` `[private]`

5.36.3.10 `MHASignal::waveform_t cfg_t::frozen_noise_` `[private]`

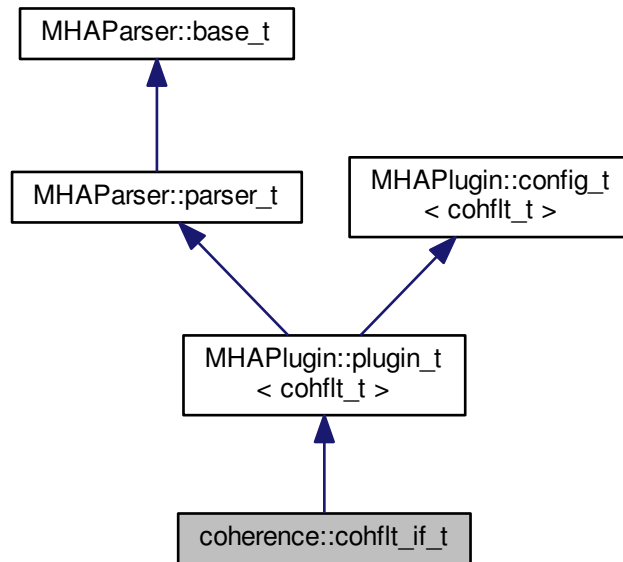
5.36.3.11 `unsigned int cfg_t::pos` `[private]`

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

- `softclip.cpp`
- `example6.cpp`
- `noise.cpp`

5.37 coherence::cohflt_if_t Class Reference

Inheritance diagram for coherence::cohflt_if_t:



Public Member Functions

- **cohflt_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t** < **cohflt_if_t** > **patchbay**
- **vars_t** **vars**
- const std::string **algo**

Additional Inherited Members

5.37.1 Constructor & Destructor Documentation

5.37.1.1 coherence::cohflt_if_t::cohflt_if_t (
 const algo_comm_t & ac,
 const std::string & th,
 const std::string & al)

5.37.2 Member Function Documentation

5.37.2.1 void coherence::cohflt_if_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< cohflt_t >** (p. 661).

5.37.2.2 void coherence::cohflt_if_t::release (
 void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< cohflt_t >** (p. 661).

5.37.2.3 mha_spec_t * coherence::cohflt_if_t::process (
 mha_spec_t * s)

5.37.2.4 void coherence::cohflt_if_t::update () [private]

5.37.3 Member Data Documentation

5.37.3.1 MHAEvents::patchbay_t<cohflt_if_t> coherence::cohflt_if_t::patchbay [private]

5.37.3.2 vars_t coherence::cohflt_if_t::vars [private]

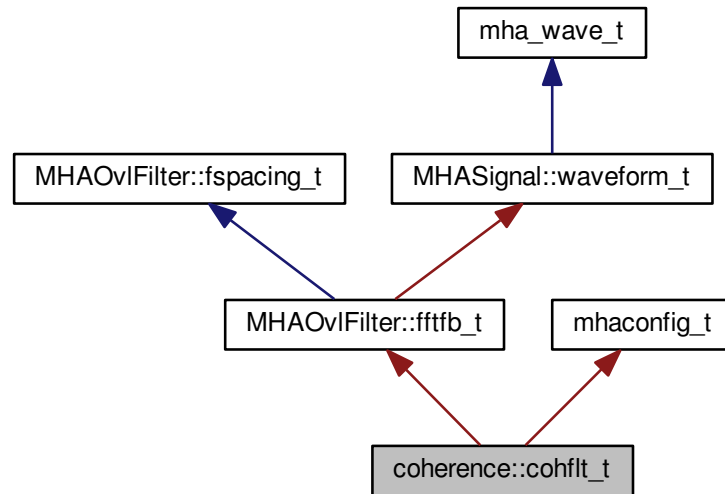
5.37.3.3 const std::string coherence::cohflt_if_t::algo [private]

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

- coherence.cpp

5.38 coherence::cohflt_t Class Reference

Inheritance diagram for coherence::cohflt_t:



Public Member Functions

- **cohflt_t** (**vars_t** &v, const **mhaconfig_t** &icf, **algo_comm_t** iac, const std::string &name)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **insert** ()

Private Attributes

- unsigned int **nbands**
- bool **avg_ipd**
- **mha_complex_t** **cg**
- float **g**
- float **c_scale**
- float **c_min**
- **MHASignal::waveform_t** **alpha**
- float **limit**
- **MHAFilter::o1flt_lowpass_t** **lp1r**
- **MHAFilter::o1flt_lowpass_t** **lp1i**
- **MHA_AC::spectrum_t** **coh_c**
- **MHA_AC::waveform_t** **coh_rlp**
- **MHASignal::waveform_t** **gain**
- **MHASignal::delay_wave_t** **gain_delay**

- **MHASignal::spectrum_t s_out**
- bool **blinvert**
- **MHAFilter::o1flt_lowpass_t lp1ltg**
- bool **b_ltg**
- std::vector< float > **staticgain**

Additional Inherited Members

5.38.1 Constructor & Destructor Documentation

5.38.1.1 coherence::cohflt_t::cohflt_t (
vars_t & v,
const mhaconfig_t & icf,
algo_comm_t iac,
const std::string & name)

5.38.2 Member Function Documentation

5.38.2.1 mha_spec_t * coherence::cohflt_t::process (
mha_spec_t * s)

5.38.2.2 void coherence::cohflt_t::insert ()

5.38.3 Member Data Documentation

5.38.3.1 unsigned int coherence::cohflt_t::nbands [private]

5.38.3.2 bool coherence::cohflt_t::avg_ipd [private]

5.38.3.3 mha_complex_t coherence::cohflt_t::cg [private]

5.38.3.4 float coherence::cohflt_t::g [private]

5.38.3.5 float coherence::cohflt_t::c_scale [private]

5.38.3.6 float coherence::cohflt_t::c_min [private]

5.38.3.7 MHASignal::waveform_t coherence::cohflt_t::alpha [private]

5.38.3.8 float coherence::cohflt_t::limit [private]

5.38.3.9 MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1r [private]

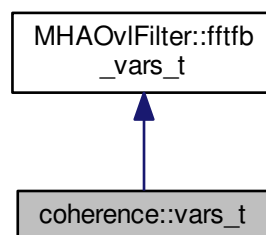
- 5.38.3.10 `MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1i` [private]
- 5.38.3.11 `MHA_AC::spectrum_t coherence::cohflt_t::coh_c` [private]
- 5.38.3.12 `MHA_AC::waveform_t coherence::cohflt_t::coh_rlp` [private]
- 5.38.3.13 `MHASignal::waveform_t coherence::cohflt_t::gain` [private]
- 5.38.3.14 `MHASignal::delay_wave_t coherence::cohflt_t::gain_delay` [private]
- 5.38.3.15 `MHASignal::spectrum_t coherence::cohflt_t::s_out` [private]
- 5.38.3.16 `bool coherence::cohflt_t::blinvert` [private]
- 5.38.3.17 `MHAFilter::o1flt_lowpass_t coherence::cohflt_t::lp1ltg` [private]
- 5.38.3.18 `bool coherence::cohflt_t::b_ltg` [private]
- 5.38.3.19 `std::vector<float> coherence::cohflt_t::staticgain` [private]

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

- **coherence.cpp**

5.39 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

5.39.1 Constructor & Destructor Documentation

5.39.1.1 coherence::vars_t::vars_t (
 MHAParser::parser_t * *p*)

5.39.2 Member Data Documentation

5.39.2.1 **MHAParser::kw_t** coherence::vars_t::tau_unit

5.39.2.2 **MHAParser::vfloat_t** coherence::vars_t::tau

5.39.2.3 **MHAParser::vfloat_t** coherence::vars_t::alpha

5.39.2.4 **MHAParser::float_t** coherence::vars_t::limit

5.39.2.5 **MHAParser::vfloat_t** coherence::vars_t::mapping

5.39.2.6 **MHAParser::kw_t** coherence::vars_t::average

5.39.2.7 **MHAParser::bool_t** coherence::vars_t::invert

5.39.2.8 **MHAParser::bool_t** coherence::vars_t::ltgcomp

5.39.2.9 **MHAParser::vfloat_t** coherence::vars_t::ltgtau

5.39.2.10 **MHAParser::vfloat_t** coherence::vars_t::staticgain

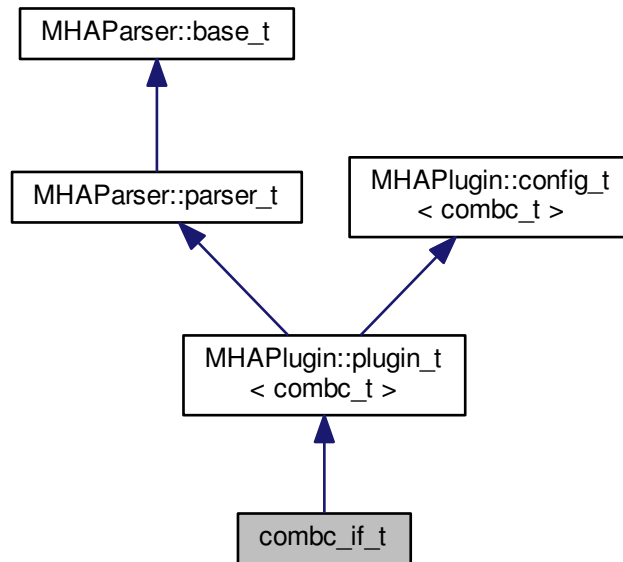
5.39.2.11 **MHAParser::int_t** coherence::vars_t::delay

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

- **coherence.cpp**

5.40 combc_if_t Class Reference

Inheritance diagram for combc_if_t:



Public Member Functions

- **combc_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHAParser::int_t** outchannels
- **MHAParser::bool_t** interleaved
- **MHAParser::string_t** channel_gain_name
- **MHAParser::string_t** element_gain_name

Additional Inherited Members

5.40.1 Constructor & Destructor Documentation

5.40.1.1 **combc_if_t::combc_if_t** (
 const **algo_comm_t** & *iac*,
 const std::string & ,
 const std::string &)

5.40.2 Member Function Documentation

5.40.2.1 void **combc_if_t::prepare** (
 mhaconfig_t & *chcfg*) [virtual]

Implements **MHAParser::plugin_t**< **combc_t** > (p. 661).

5.40.2.2 **mha_wave_t** * **combc_if_t::process** (
 mha_wave_t * *s*)

5.40.2.3 **mha_spec_t** * **combc_if_t::process** (
 mha_spec_t * *s*)

5.40.3 Member Data Documentation

5.40.3.1 **MHAParser::int_t** **combc_if_t::outchannels** [private]

5.40.3.2 **MHAParser::bool_t** **combc_if_t::interleaved** [private]

5.40.3.3 **MHAParser::string_t** **combc_if_t::channel_gain_name** [private]

5.40.3.4 **MHAParser::string_t** **combc_if_t::element_gain_name** [private]

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

- **combinechannels.cpp**

5.41 **combc_t** Class Reference

Public Member Functions

- **combc_t** (**algo_comm_t** ac, **mhaconfig_t** cfg_input, **mhaconfig_t** cfg_output, std::vector< float > channel_gains, const std::string & element_gain_name, bool interleaved)
- **mha_wave_t** * **process** (**mha_wave_t** *s)
- **mha_spec_t** * **process** (**mha_spec_t** *s)

Private Attributes

- **algo_comm_t** *ac_*
- **bool** *interleaved_*
- **unsigned int** *nbands*
- **MHASignal::waveform_t** *w_out*
- **MHASignal::spectrum_t** *s_out*
- **std::vector< mha_real_t >** *channel_gains_*
- **std::string** *element_gain_name_*

5.41.1 Constructor & Destructor Documentation

5.41.1.1 **combc_t::combc_t** (
 algo_comm_t *ac*,
 mhaconfig_t *cfg_input*,
 mhaconfig_t *cfg_output*,
 std::vector< float > *channel_gains*,
 const std::string & *element_gain_name*,
 bool *interleaved*)

5.41.2 Member Function Documentation

5.41.2.1 **mha_wave_t * combc_t::process** (
 mha_wave_t * s)

5.41.2.2 **mha_spec_t * combc_t::process** (
 mha_spec_t * s)

5.41.3 Member Data Documentation

5.41.3.1 **algo_comm_t** *combc_t::ac_* [private]

5.41.3.2 **bool** *combc_t::interleaved_* [private]

5.41.3.3 **unsigned int** *combc_t::nbands* [private]

5.41.3.4 **MHASignal::waveform_t** *combc_t::w_out* [private]

5.41.3.5 **MHASignal::spectrum_t** *combc_t::s_out* [private]

5.41.3.6 **std::vector<mha_real_t>** *combc_t::channel_gains_* [private]

5.41.3.7 **std::string** *combc_t::element_gain_name_* [private]

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

- **combinechannels.cpp**

5.42 comm_var_t Struct Reference

Algorithm communication variable structure.

Public Attributes

- unsigned int **data_type**
Type of data.
- unsigned int **num_entries**
Number of entries.
- unsigned int **stride**
length of one row (C interpretation) or of one column (Fortran interpretation)
- void * **data**
Pointer to variable data.

5.42.1 Detailed Description

Algorithm communication variable structure.

Algorithm communication variables (AC variables) are objects of this type. The member data is a pointer to the variable 'data'. This pointer has to be valid for the lifetime of this AC variable. The member 'data_type' can be one of the predefined types or any user defined type. The member 'num_entries' describes the number of elements of this base type stored at the pointer address.

```
An AC variable can be registered with the \ref  
algo_comm_t::insert_var "insert_var" function.
```

5.42.2 Member Data Documentation

5.42.2.1 comm_var_t::data_type

Type of data.

This can be one of the predefined types

- MHA_AC_CHAR
- MHA_AC_INT
- MHA_AC_MHAREAL
- MHA_AC_FLOAT
- MHA_AC_DOUBLE
- MHA_AC_MHACOMPLEX
- MHA_AC_VEC_FLOAT or any user defined type with a value greater than
- MHA_AC_USER

5.42.2.2 `comm_var_t::num_entries`

Number of entries.

5.42.2.3 `comm_var_t::stride`

length of one row (C interpretation) or of one column (Fortran interpretation)

5.42.2.4 `comm_var_t::data`

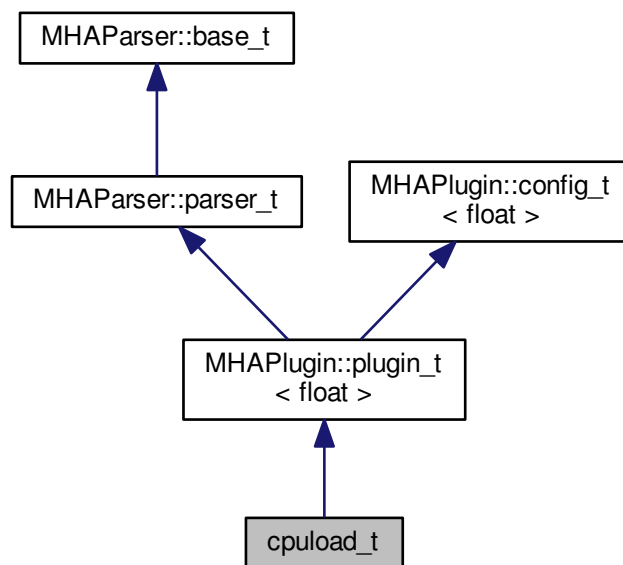
Pointer to variable data.

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

- **mha.h**
- **mha_algo_comm.cpp**

5.43 `cpuload_t` Class Reference

Inheritance diagram for `cpuload_t`:



Public Member Functions

- **cpuload_t** (**algo_comm_t**, const char *, const char *)
- **mha_spec_t * process** (**mha_spec_t** *)
- **mha_wave_t * process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **compute_something** ()
- void **compute_something_else** ()

Private Attributes

- **MHAParser::float_t factor**
- **MHAParser::bool_t use_sine**
- float **phase**
- volatile float **result**
- std::vector< float > **table**

Additional Inherited Members

5.43.1 Constructor & Destructor Documentation

5.43.1.1 **cpuload_t::cpuload_t** (
 algo_comm_t iac,
 const char *,
 const char *)

5.43.2 Member Function Documentation

5.43.2.1 **mha_spec_t * cpuload_t::process** (
 mha_spec_t * s)

5.43.2.2 **mha_wave_t * cpuload_t::process** (
 mha_wave_t * s)

5.43.2.3 **void cpuload_t::prepare** (
 mhaconfig_t & cf) [virtual]

Implements **MHAPLugin::plugin_t< float >** (p. 661).

5.43.2.4 void cpuload_t::compute_something () [inline],[private]

5.43.2.5 void cpuload_t::compute_something_else () [inline],[private]

5.43.3 Member Data Documentation

5.43.3.1 MHAParser::float_t cpuload_t::factor [private]

5.43.3.2 MHAParser::bool_t cpuload_t::use_sine [private]

5.43.3.3 float cpuload_t::phase [private]

5.43.3.4 volatile float cpuload_t::result [private]

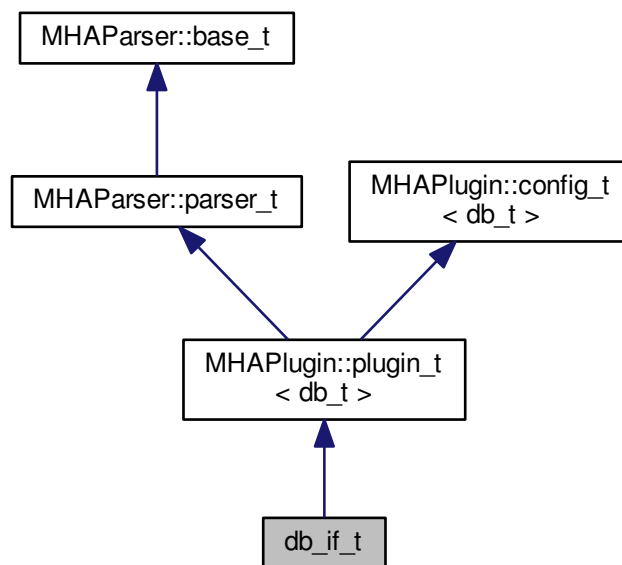
5.43.3.5 std::vector<float> cpuload_t::table [private]

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

- cpuload.cpp

5.44 db_if_t Class Reference

Inheritance diagram for db_if_t:



Public Member Functions

- **db_if_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t * process** (**mha_wave_t ***)
- void **prepare** (**mhaconfig_t &**)
- void **release** ()
- **~db_if_t** ()

Private Attributes

- **MHAEvents::patchbay_t< db_if_t > patchbay**
- **MHAParser::int_t fragsize**
- **MHAParser::mhapluginloader_t plugloader**
- std::string **chain**
- std::string **algo**
- bool **bypass**

Additional Inherited Members

5.44.1 Constructor & Destructor Documentation

5.44.1.1 **db_if_t::db_if_t** (
 algo_comm_t iac,
 std::string **th**,
 std::string **al**)

5.44.1.2 **db_if_t::~~db_if_t** ()

5.44.2 Member Function Documentation

5.44.2.1 **mha_wave_t * db_if_t::process** (
 mha_wave_t * s)

5.44.2.2 void **db_if_t::prepare** (
 mhaconfig_t & conf) [virtual]

Implements **MHAPlugin::plugin_t< db_t >** (p. 661).

5.44.2.3 void **db_if_t::release** (
 void) [virtual]

Reimplemented from **MHAPlugin::plugin_t< db_t >** (p. 661).

5.44.3 Member Data Documentation

5.44.3.1 **MHAEvents::patchbay_t** < **db_if_t** > **db_if_t::patchbay** [private]

5.44.3.2 **MHAParser::int_t** **db_if_t::fragsize** [private]

5.44.3.3 **MHAParser::mhapluginloader_t** **db_if_t::plugloader** [private]

5.44.3.4 **std::string** **db_if_t::chain** [private]

5.44.3.5 **std::string** **db_if_t::algo** [private]

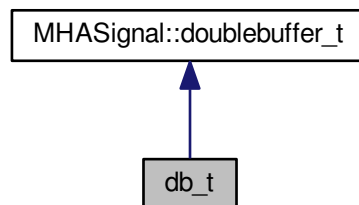
5.44.3.6 **bool** **db_if_t::bypass** [private]

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

- **db.cpp**

5.45 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

5.45.1 Constructor & Destructor Documentation

5.45.1.1 db_t::db_t (
 unsigned int *outer_fragsize*,
 unsigned int *inner_fragsize*,
 unsigned int *nch_in*,
 unsigned int *nch_out*,
 MHAParser::mhapluginloader_t & *plug*)

5.45.2 Member Function Documentation

5.45.2.1 mha_wave_t* db_t::inner_process (
 mha_wave_t* *s*) [virtual]

Implements **MHASignal::doublebuffer_t** (p. [702](#)).

5.45.3 Member Data Documentation

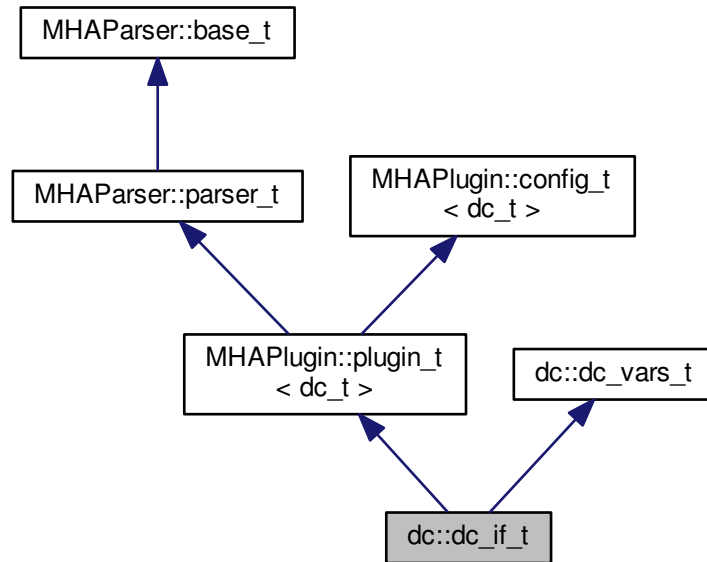
5.45.3.1 MHAParser::mhapluginloader_t& db_t::plugloader [private]

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

- **db.cpp**

5.46 dc::dc_if_t Class Reference

Inheritance diagram for dc::dc_if_t:



Public Member Functions

- **dc_if_t** (const **algo_comm_t** &ac_, const std::string &th_, const std::string &al_)
- void **prepare** (**mhaconfig_t** &tf)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update_monitors** ()
Called from within the processing routines: updates the monitor variables.
- void **update** ()
Called by MHA configuration change event mechanism: creates new runtime configuration.

Private Attributes

- std::string **algo**
- **wideband_inhib_vars_t** **wbinhib**
- **MHAEvents::patchbay_t** < **dc_if_t** > **patchbay**

Additional Inherited Members

5.46.1 Constructor & Destructor Documentation

5.46.1.1 `dc::dc_if_t::dc_if_t (`
 `const algo_comm_t & ac_,`
 `const std::string & th_,`
 `const std::string & al_)`

5.46.2 Member Function Documentation

5.46.2.1 `void dc::dc_if_t::prepare (`
 `mhaconfig_t & tf)` `[virtual]`

Implements **MHAPlugin::plugin_t< dc_t >** (p. 661).

5.46.2.2 `mha_wave_t * dc::dc_if_t::process (`
 `mha_wave_t * s_in)`

5.46.2.3 `mha_spec_t * dc::dc_if_t::process (`
 `mha_spec_t * s_in)`

5.46.2.4 `void dc::dc_if_t::update_monitors ()` `[private]`

Called from within the processing routines: updates the monitor variables.

5.46.2.5 `void dc::dc_if_t::update ()` `[private]`

Called by MHA configuration change event mechanism: creates new runtime configuration.

5.46.3 Member Data Documentation

5.46.3.1 `std::string dc::dc_if_t::algo` `[private]`

5.46.3.2 `wideband_inhib_vars_t dc::dc_if_t::wbinhib` `[private]`

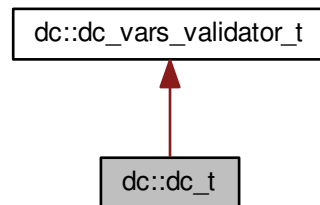
5.46.3.3 `MHAEvents::patchbay_t<dc_if_t> dc::dc_if_t::patchbay` `[private]`

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

- **dc.cpp**

5.47 dc::dc_t Class Reference

Inheritance diagram for dc::dc_t:



Public Member Functions

- **dc_t** (**dc_vars_t** vars, **mha_real_t** filter_rate, unsigned int nch, **algo_comm_t** ac, **mha_↔_domain_t** domain, unsigned int fftlen, std::string algo)
- **mha_wave_t * process** (**mha_wave_t ***)
- **mha_spec_t * process** (**mha_spec_t ***, **wb_inhib_cfg_t** *wb_inhib)
- void **explicit_insert** ()
- unsigned **get_nbands** () const
Number of frequency bands accessor.
- const **MHASignal::waveform_t & get_level_in_db** () const
- const **MHASignal::waveform_t & get_level_in_db_adjusted** () const

Private Attributes

- std::vector< **MHATableLookup::linear_table_t** > **gt**
- **MHAFilter::o1flt_lowpass_t** **rmslevel**
- **MHAFilter::o1flt_lowpass_t** **attack**
- **MHAFilter::o1flt_maxtrack_t** **decay**
- bool **powersum**
- bool **bypass**
- unsigned int **naudiochannels**
- unsigned int **nbands**
- **MHA_AC::waveform_t** **level_in_db**
- **MHA_AC::waveform_t** **level_in_db_adjusted**
- **MHA_AC::waveform_t** **inhib_gain**
- **MHASignal::waveform_t** **max_level_difference**
- unsigned int **k_nyquist**

Additional Inherited Members

5.47.1 Constructor & Destructor Documentation

5.47.1.1 `dc::dc_t::dc_t(`
 `dc_vars_t vars,`
 `mha_real_t filter_rate,`
 `unsigned int nch,`
 `algo_comm_t ac,`
 `mha_domain_t domain,`
 `unsigned int fftlen,`
 `std::string algo)`

5.47.2 Member Function Documentation

5.47.2.1 `mha_wave_t * dc::dc_t::process (`
 `mha_wave_t * s)`

5.47.2.2 `mha_spec_t * dc::dc_t::process (`
 `mha_spec_t * s,`
 `wb_inhib_cfg_t * wbinhib)`

5.47.2.3 `void dc::dc_t::explicit_insert ()`

5.47.2.4 `unsigned dc::dc_t::get_nbands () const` `[inline]`

Number of frequency bands accessor.

5.47.2.5 `const MHASignal::waveform_t& dc::dc_t::get_level_in_db () const` `[inline]`

5.47.2.6 `const MHASignal::waveform_t& dc::dc_t::get_level_in_db_adjusted () const`
`[inline]`

5.47.3 Member Data Documentation

5.47.3.1 `std::vector<MHATableLookup::linear_table_t> dc::dc_t::gt` `[private]`

5.47.3.2 `MHAFilter::o1flt_lowpass_t dc::dc_t::rmslevel` `[private]`

5.47.3.3 `MHAFilter::o1flt_lowpass_t dc::dc_t::attack` `[private]`

5.47.3.4 `MHAFilter::o1flt_maxtrack_t dc::dc_t::decay` `[private]`

5.47.3.5 `bool dc::dc_t::powersum` `[private]`

5.47.3.6 `bool dc::dc_t::bypass` `[private]`

5.47.3.7 `unsigned int dc::dc_t::naudiochannels` `[private]`

5.47.3.8 `unsigned int dc::dc_t::nbands` `[private]`

5.47.3.9 `MHA_AC::waveform_t dc::dc_t::level_in_db` `[private]`

5.47.3.10 `MHA_AC::waveform_t dc::dc_t::level_in_db_adjusted` `[private]`

5.47.3.11 `MHA_AC::waveform_t dc::dc_t::inhib_gain` `[private]`

5.47.3.12 `MHASignal::waveform_t dc::dc_t::max_level_difference` `[private]`

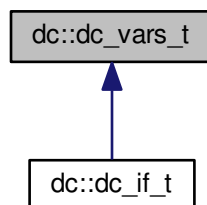
5.47.3.13 `unsigned int dc::dc_t::k_nyquist` `[private]`

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

- **dc.cpp**

5.48 `dc::dc_vars_t` Class Reference

Inheritance diagram for `dc::dc_vars_t`:



Public Member Functions

- **`dc_vars_t` (`MHAParser::parser_t &`)**

Public Attributes

- **MHAParser::bool_t powersum**
- **MHAParser::mfloat_t gtdata**
- **MHAParser::vfloat_t gtmin**
- **MHAParser::vfloat_t gtstep**
- **MHAParser::vfloat_t taurmslevel**
- **MHAParser::vfloat_t tauattack**
- **MHAParser::vfloat_t taudecay**
- **MHAParser::string_t filterbank**
- **std::string cf_name**
- **std::string ef_name**
- **std::string bw_name**
- **MHAParser::string_t chname**
- **MHAParser::bool_t bypass**
- **MHAParser::string_t clientid**
- **MHAParser::string_t gainrule**
- **MHAParser::string_t preset**
- **MHAParser::int_mon_t modified**
- **MHAParser::mfloat_t max_level_difference**
- **MHAParser::vfloat_mon_t input_level**
- **MHAParser::vfloat_mon_t filtered_level**
- **MHAParser::vfloat_mon_t center_frequencies**
- **MHAParser::vfloat_mon_t edge_frequencies**
- **MHAParser::vfloat_mon_t band_weights**
- **MHAParser::bool_t use_wbinhib**

5.48.1 Constructor & Destructor Documentation

5.48.1.1 **dc::dc_vars_t::dc_vars_t (**
 MHAParser::parser_t & p)

5.48.2 Member Data Documentation

5.48.2.1 **MHAParser::bool_t dc::dc_vars_t::powersum**

5.48.2.2 **MHAParser::mfloat_t dc::dc_vars_t::gtdata**

5.48.2.3 **MHAParser::vfloat_t dc::dc_vars_t::gtmin**

5.48.2.4 **MHAParser::vfloat_t dc::dc_vars_t::gtstep**

5.48.2.5 **MHAParser::vfloat_t dc::dc_vars_t::taurmslevel**

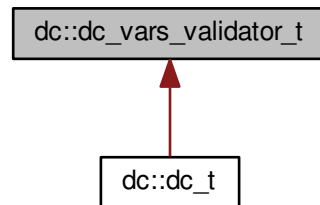
- 5.48.2.6 **MHAParser::vfloat_t dc::dc_vars_t::tauattack**
- 5.48.2.7 **MHAParser::vfloat_t dc::dc_vars_t::taudecay**
- 5.48.2.8 **MHAParser::string_t dc::dc_vars_t::filterbank**
- 5.48.2.9 **std::string dc::dc_vars_t::cf_name**
- 5.48.2.10 **std::string dc::dc_vars_t::ef_name**
- 5.48.2.11 **std::string dc::dc_vars_t::bw_name**
- 5.48.2.12 **MHAParser::string_t dc::dc_vars_t::chname**
- 5.48.2.13 **MHAParser::bool_t dc::dc_vars_t::bypass**
- 5.48.2.14 **MHAParser::string_t dc::dc_vars_t::clientid**
- 5.48.2.15 **MHAParser::string_t dc::dc_vars_t::gainrule**
- 5.48.2.16 **MHAParser::string_t dc::dc_vars_t::preset**
- 5.48.2.17 **MHAParser::int_mon_t dc::dc_vars_t::modified**
- 5.48.2.18 **MHAParser::mfloat_t dc::dc_vars_t::max_level_difference**
- 5.48.2.19 **MHAParser::vfloat_mon_t dc::dc_vars_t::input_level**
- 5.48.2.20 **MHAParser::vfloat_mon_t dc::dc_vars_t::filtered_level**
- 5.48.2.21 **MHAParser::vfloat_mon_t dc::dc_vars_t::center_frequencies**
- 5.48.2.22 **MHAParser::vfloat_mon_t dc::dc_vars_t::edge_frequencies**
- 5.48.2.23 **MHAParser::vfloat_mon_t dc::dc_vars_t::band_weights**
- 5.48.2.24 **MHAParser::bool_t dc::dc_vars_t::use_wbinhib**

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

- **dc.cpp**

5.49 dc::dc_vars_validator_t Class Reference

Inheritance diagram for dc::dc_vars_validator_t:



Public Member Functions

- **dc_vars_validator_t** (**dc_vars_t** &*v*, unsigned int *s*, **mha_domain_t** *domain*)

5.49.1 Constructor & Destructor Documentation

5.49.1.1 `dc::dc_vars_validator_t::dc_vars_validator_t(`
 dc_vars_t & *v*,
 unsigned int *s*,
 mha_domain_t *domain*)

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

- **dc.cpp**

5.50 dc::wb_inhib_cfg_t Class Reference

Public Member Functions

- **wb_inhib_cfg_t** (const **wideband_inhib_vars_t** &*vars*)

Public Attributes

- std::vector< float > **weights**
- float **dl_map_min**
- float **dl_map_max**
- float **dl_diff**
- float **l_min**
- std::vector< std::vector< float > > **g_scale**

5.50.1 Constructor & Destructor Documentation

5.50.1.1 `dc::wb_inhib_cfg_t::wb_inhib_cfg_t (`
`const wideband_inhib_vars_t & vars)`

5.50.2 Member Data Documentation

5.50.2.1 `std::vector<float> dc::wb_inhib_cfg_t::weights`

5.50.2.2 `float dc::wb_inhib_cfg_t::dl_map_min`

5.50.2.3 `float dc::wb_inhib_cfg_t::dl_map_max`

5.50.2.4 `float dc::wb_inhib_cfg_t::dl_diff`

5.50.2.5 `float dc::wb_inhib_cfg_t::l_min`

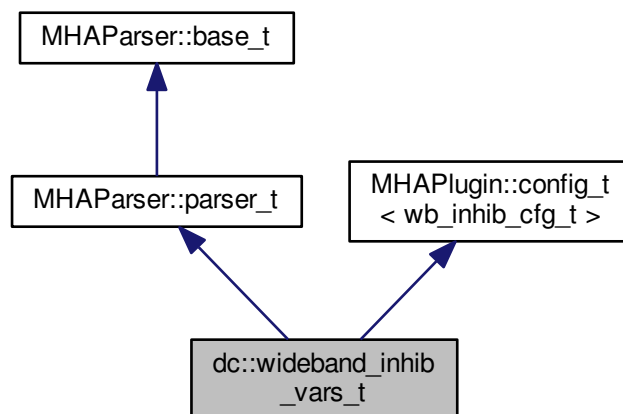
5.50.2.6 `std::vector<std::vector<float> > dc::wb_inhib_cfg_t::g_scale`

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

- **dc.cpp**

5.51 dc::wideband_inhib_vars_t Class Reference

Inheritance diagram for `dc::wideband_inhib_vars_t`:



Public Member Functions

- **wideband_inhib_vars_t** ()
- void **setchannels** (unsigned int *ch*, unsigned int *bnds*)
- **wb_inhib_cfg_t** * **current** ()
- void **update** ()

Public Attributes

- **MHAParser::vfloat_t** **weights**
- **MHAParser::float_t** **dl_map_min**
- **MHAParser::float_t** **dl_map_max**
- **MHAParser::float_t** **l_min**
- **MHAParser::mfloat_t** **g_scale**
- **MHAEvents::patchbay_t** < **wideband_inhib_vars_t** > **patchbay**
- unsigned int **channels**
- unsigned int **bands**

Additional Inherited Members

5.51.1 Constructor & Destructor Documentation

5.51.1.1 dc::wideband_inhib_vars_t::wideband_inhib_vars_t ()

5.51.2 Member Function Documentation

5.51.2.1 void dc::wideband_inhib_vars_t::setchannels (
 unsigned int *ch*,
 unsigned int *bnds*) [inline]

5.51.2.2 **wb_inhib_cfg_t*** dc::wideband_inhib_vars_t::current () [inline]

5.51.2.3 void dc::wideband_inhib_vars_t::update ()

5.51.3 Member Data Documentation

5.51.3.1 **MHAParser::vfloat_t** dc::wideband_inhib_vars_t::weights

5.51.3.2 **MHAParser::float_t** dc::wideband_inhib_vars_t::dl_map_min

5.51.3.3 **MHAParser::float_t** dc::wideband_inhib_vars_t::dl_map_max

5.51.3.4 **MHAParser::float_t** dc::wideband_inhib_vars_t::l_min

5.51.3.5 **MHAParser::mfloat_t** dc::wideband_inhib_vars_t::g_scale

5.51.3.6 **MHAEvents::patchbay_t**<wideband_inhib_vars_t> dc::wideband_inhib_vars_t↔
::patchbay

5.51.3.7 **unsigned int** dc::wideband_inhib_vars_t::channels

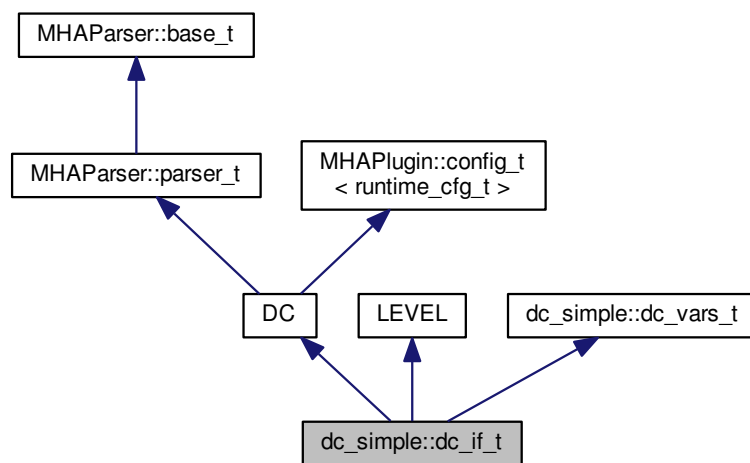
5.51.3.8 **unsigned int** dc::wideband_inhib_vars_t::bands

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

- **dc.cpp**

5.52 dc_simple::dc_if_t Class Reference

Inheritance diagram for dc_simple::dc_if_t:



Public Member Functions

- **dc_if_t** (const **algo_comm_t** &ac_, const std::string &th_, const std::string &al_)
- void **prepare** (**mhaconfig_t** &tf)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update_dc** ()
- void **update_level** ()
- void **has_been_modified** ()
- void **read_modified** ()
- void **update_level_mon** ()
- void **update_gain_mon** ()

Private Attributes

- **MHAParser::string_t** clientid
- **MHAParser::string_t** gainrule
- **MHAParser::string_t** preset
- **MHAParser::int_mon_t** modified
- **MHAParser::vfloat_mon_t** mon_l
- **MHAParser::vfloat_mon_t** mon_g
- **MHAParser::string_t** filterbank
- **MHAParser::vfloat_mon_t** center_frequencies
- **MHAParser::vfloat_mon_t** edge_frequencies
- **MHAEvents::patchbay_t**< **dc_if_t** > patchbay
- bool **prepared**

Additional Inherited Members

5.52.1 Constructor & Destructor Documentation

5.52.1.1 **dc_simple::dc_if_t::dc_if_t** (
 const **algo_comm_t** & *ac*_,
 const std::string & *th*_,
 const std::string & *al*_)

5.52.2 Member Function Documentation

5.52.2.1 void **dc_simple::dc_if_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Implements **MHAPLugin::plugin_t**< **runtime_cfg_t** > (p. 661).

5.52.2.2 void **dc_simple::dc_if_t::release** (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t**< **runtime_cfg_t** > (p. 661).

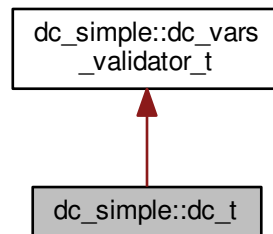
- 5.52.2.3 `mha_spec_t * dc_simple::dc_if_t::process (mha_spec_t * s)`
- 5.52.2.4 `mha_wave_t * dc_simple::dc_if_t::process (mha_wave_t * s)`
- 5.52.2.5 `void dc_simple::dc_if_t::update_dc () [private]`
- 5.52.2.6 `void dc_simple::dc_if_t::update_level () [private]`
- 5.52.2.7 `void dc_simple::dc_if_t::has_been_modified () [inline],[private]`
- 5.52.2.8 `void dc_simple::dc_if_t::read_modified () [inline],[private]`
- 5.52.2.9 `void dc_simple::dc_if_t::update_level_mon () [private]`
- 5.52.2.10 `void dc_simple::dc_if_t::update_gain_mon () [private]`
- 5.52.3 Member Data Documentation
- 5.52.3.1 `MHAParser::string_t dc_simple::dc_if_t::clientid [private]`
- 5.52.3.2 `MHAParser::string_t dc_simple::dc_if_t::gainrule [private]`
- 5.52.3.3 `MHAParser::string_t dc_simple::dc_if_t::preset [private]`
- 5.52.3.4 `MHAParser::int_mon_t dc_simple::dc_if_t::modified [private]`
- 5.52.3.5 `MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_l [private]`
- 5.52.3.6 `MHAParser::vfloat_mon_t dc_simple::dc_if_t::mon_g [private]`
- 5.52.3.7 `MHAParser::string_t dc_simple::dc_if_t::filterbank [private]`
- 5.52.3.8 `MHAParser::vfloat_mon_t dc_simple::dc_if_t::center_frequencies [private]`
- 5.52.3.9 `MHAParser::vfloat_mon_t dc_simple::dc_if_t::edge_frequencies [private]`
- 5.52.3.10 `MHAEvents::patchbay_t<dc_if_t> dc_simple::dc_if_t::patchbay [private]`
- 5.52.3.11 `bool dc_simple::dc_if_t::prepared [private]`

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

- `dc_simple.cpp`

5.53 dc_simple::dc_t Class Reference

Inheritance diagram for dc_simple::dc_t:



Classes

- class **line_t**

Public Member Functions

- **dc_t** (const **dc_vars_t** &vars, **mha_real_t** filter_rate, unsigned int nch, unsigned int fftlen)
- **mha_spec_t** * **process** (**mha_spec_t** *, **mha_wave_t** *level_db)
- **mha_wave_t** * **process** (**mha_wave_t** *, **mha_wave_t** *level_db)

Public Attributes

- std::vector< float > **mon_l**
- std::vector< float > **mon_g**

Private Attributes

- std::vector< **mha_real_t** > **expansion_threshold**
- std::vector< **mha_real_t** > **limiter_threshold**
- std::vector< **line_t** > **compression**
- std::vector< **line_t** > **expansion**
- std::vector< **line_t** > **limiter**
- std::vector< **mha_real_t** > **maxgain**
- unsigned int **nbands**

Additional Inherited Members

5.53.1 Constructor & Destructor Documentation

5.53.1.1 `dc_simple::dc_t::dc_t (`
 `const dc_vars_t & vars,`
 `mha_real_t filter_rate,`
 `unsigned int nch,`
 `unsigned int fftlen)`

5.53.2 Member Function Documentation

5.53.2.1 `mha_spec_t * dc_simple::dc_t::process (`
 `mha_spec_t * s,`
 `mha_wave_t * level_db)`

5.53.2.2 `mha_wave_t * dc_simple::dc_t::process (`
 `mha_wave_t * s,`
 `mha_wave_t * level_db)`

5.53.3 Member Data Documentation

5.53.3.1 `std::vector<mha_real_t> dc_simple::dc_t::expansion_threshold` [private]

5.53.3.2 `std::vector<mha_real_t> dc_simple::dc_t::limiter_threshold` [private]

5.53.3.3 `std::vector<line_t> dc_simple::dc_t::compression` [private]

5.53.3.4 `std::vector<line_t> dc_simple::dc_t::expansion` [private]

5.53.3.5 `std::vector<line_t> dc_simple::dc_t::limiter` [private]

5.53.3.6 `std::vector<mha_real_t> dc_simple::dc_t::maxgain` [private]

5.53.3.7 `unsigned int dc_simple::dc_t::nbands` [private]

5.53.3.8 `std::vector<float> dc_simple::dc_t::mon_l`

5.53.3.9 `std::vector<float> dc_simple::dc_t::mon_g`

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

- **dc_simple.cpp**

5.54 dc_simple::dc_t::line_t Class Reference

Public Member Functions

- **line_t** (mha_real_t x1, mha_real_t y1, mha_real_t x2, mha_real_t y2)
- **line_t** (mha_real_t x1, mha_real_t y1, mha_real_t slope)
- **mha_real_t operator()** (mha_real_t x)

Private Attributes

- **mha_real_t m**
- **mha_real_t y0**

5.54.1 Constructor & Destructor Documentation

5.54.1.1 dc_simple::dc_t::line_t::line_t (
 mha_real_t x1,
 mha_real_t y1,
 mha_real_t x2,
 mha_real_t y2)

5.54.1.2 dc_simple::dc_t::line_t::line_t (
 mha_real_t x1,
 mha_real_t y1,
 mha_real_t *slope*)

5.54.2 Member Function Documentation

5.54.2.1 mha_real_t dc_simple::dc_t::line_t::operator() (
 mha_real_t x) [inline]

5.54.3 Member Data Documentation

5.54.3.1 mha_real_t dc_simple::dc_t::line_t::m [private]

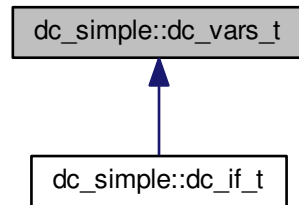
5.54.3.2 mha_real_t dc_simple::dc_t::line_t::y0 [private]

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

- **dc_simple.cpp**

5.55 dc_simple::dc_vars_t Class Reference

Inheritance diagram for dc_simple::dc_vars_t:



Public Member Functions

- **dc_vars_t (MHAParser::parser_t &)**

Public Attributes

- **MHAParser::vfloat_t g50**
- **MHAParser::vfloat_t g80**
- **MHAParser::vfloat_t maxgain**
- **MHAParser::vfloat_t expansion_threshold**
- **MHAParser::vfloat_t expansion_slope**
- **MHAParser::vfloat_t limiter_threshold**
- **MHAParser::vfloat_t tauattack**
- **MHAParser::vfloat_t taudecay**
- **MHAParser::bool_t bypass**

5.55.1 Constructor & Destructor Documentation

5.55.1.1 `dc_simple::dc_vars_t::dc_vars_t (MHAParser::parser_t & p)`

5.55.2 Member Data Documentation

5.55.2.1 `MHAParser::vfloat_t dc_simple::dc_vars_t::g50`

5.55.2.2 `MHAParser::vfloat_t dc_simple::dc_vars_t::g80`

5.55.2.3 MHAParser::vfloat_t dc_simple::dc_vars_t::maxgain

5.55.2.4 MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_threshold

5.55.2.5 MHAParser::vfloat_t dc_simple::dc_vars_t::expansion_slope

5.55.2.6 MHAParser::vfloat_t dc_simple::dc_vars_t::limiter_threshold

5.55.2.7 MHAParser::vfloat_t dc_simple::dc_vars_t::tauattack

5.55.2.8 MHAParser::vfloat_t dc_simple::dc_vars_t::taudecay

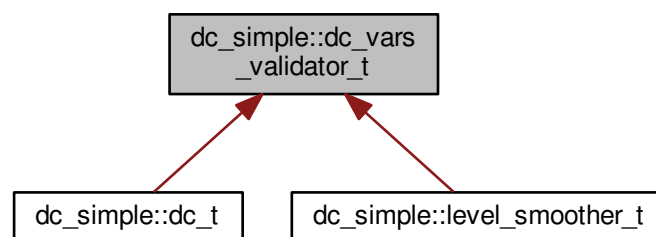
5.55.2.9 MHAParser::bool_t dc_simple::dc_vars_t::bypass

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

- **dc_simple.cpp**

5.56 dc_simple::dc_vars_validator_t Class Reference

Inheritance diagram for dc_simple::dc_vars_validator_t:



Public Member Functions

- **dc_vars_validator_t** (const **dc_vars_t** &v, unsigned int s)

5.56.1 Constructor & Destructor Documentation

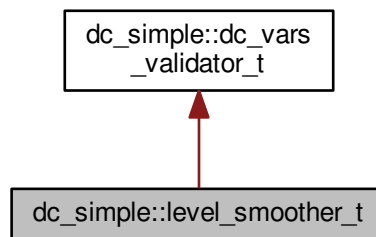
5.56.1.1 `dc_simple::dc_vars_validator_t::dc_vars_validator_t (`
`const dc_vars_t & v,`
`unsigned int s)`

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

- **dc_simple.cpp**

5.57 `dc_simple::level_smoother_t` Class Reference

Inheritance diagram for `dc_simple::level_smoother_t`:



Public Member Functions

- **level_smoother_t** (const **dc_vars_t** &vars, **mha_real_t** filter_rate, **mhaconfig_t** buscfg)
- **mha_wave_t * process** (**mha_spec_t** *)
- **mha_wave_t * process** (**mha_wave_t** *)

Private Attributes

- **MHAFilter::o1flt_lowpass_t** attack
- **MHAFilter::o1flt_maxtrack_t** decay
- unsigned int **nbands**
- unsigned int **fftl**
- **MHASignal::waveform_t** level_wave
- **MHASignal::waveform_t** level_spec

Additional Inherited Members

5.57.1 Constructor & Destructor Documentation

5.57.1.1 dc_simple::level_smoother_t::level_smoother_t (
 const dc_vars_t & vars,
 mha_real_t filter_rate,
 mhaconfig_t buscfg)

5.57.2 Member Function Documentation

5.57.2.1 mha_wave_t * dc_simple::level_smoother_t::process (
 mha_spec_t * s)

5.57.2.2 mha_wave_t * dc_simple::level_smoother_t::process (
 mha_wave_t * s)

5.57.3 Member Data Documentation

5.57.3.1 MHAFilter::o1flt_lowpass_t dc_simple::level_smoother_t::attack [private]

5.57.3.2 MHAFilter::o1flt_maxtrack_t dc_simple::level_smoother_t::decay [private]

5.57.3.3 unsigned int dc_simple::level_smoother_t::nbands [private]

5.57.3.4 unsigned int dc_simple::level_smoother_t::fftlens [private]

5.57.3.5 MHASignal::waveform_t dc_simple::level_smoother_t::level_wave [private]

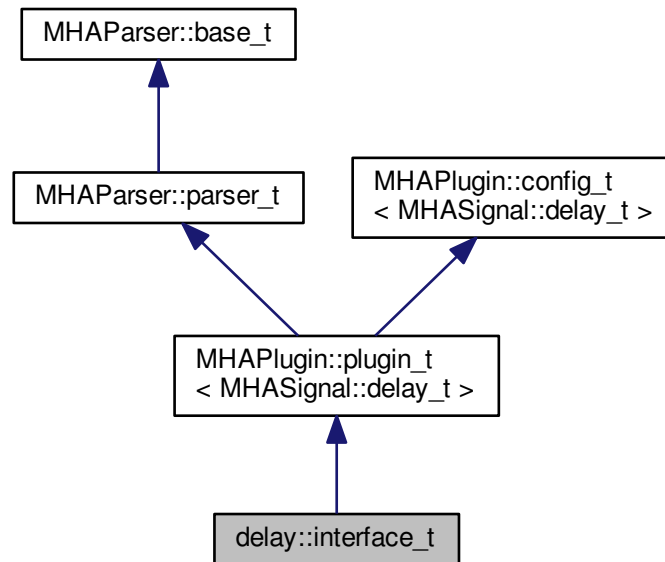
5.57.3.6 MHASignal::waveform_t dc_simple::level_smoother_t::level_spec [private]

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

- dc_simple.cpp

5.58 delay::interface_t Class Reference

Inheritance diagram for delay::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::vint_t** **delays**
- **MHAEvents::patchbay_t** < **interface_t** > **patchbay**

Additional Inherited Members

5.58.1 Constructor & Destructor Documentation

5.58.1.1 `delay::interface_t::interface_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

5.58.2 Member Function Documentation

5.58.2.1 `void delay::interface_t::prepare (`
 `mhaconfig_t & tf)` [virtual]

Implements **MHAParser::plugin_t**< **MHASignal::delay_t** > (p. 661).

5.58.2.2 `mha_wave_t * delay::interface_t::process (`
 `mha_wave_t * s)`

5.58.2.3 `void delay::interface_t::update ()` [private]

5.58.3 Member Data Documentation

5.58.3.1 `MHAParser::vint_t delay::interface_t::delays` [private]

5.58.3.2 `MHAEvents::patchbay_t<interface_t> delay::interface_t::patchbay` [private]

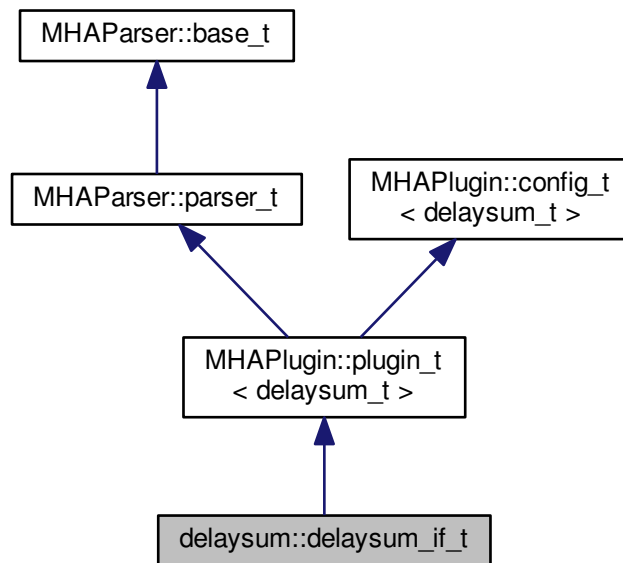
The documentation for this class was generated from the following file:

- **delay.cpp**

5.59 delaysum::delaysum_if_t Class Reference

Interface class for the delaysum plugin.

Inheritance diagram for `delaysum::delaysum_if_t`:



Public Member Functions

- **delaysum_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::vfloat_t weights**
Linear weights to be multiplied with the audio signal, one factor for each channel.
- **MHAParser::vint_t delay**
vector of channel-specific delays, in samples.
- **MHAEvents::patchbay_t< delaysum_if_t > patchbay**
The patchbay to react to config changes.

Additional Inherited Members

5.59.1 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.

5.59.2 Constructor & Destructor Documentation

5.59.2.1 `delaysum::delaysum_if_t::delaysum_if_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

5.59.3 Member Function Documentation

5.59.3.1 `mha_wave_t * delaysum::delaysum_if_t::process (`
 `mha_wave_t * wave)`

5.59.3.2 `void delaysum::delaysum_if_t::prepare (`
 `mhaconfig_t & tcfg)` [virtual]

Implements **MHAPLugin::plugin_t< delaysum_t >** (p. 661).

5.59.3.3 `void delaysum::delaysum_if_t::release (`
 `void)` [virtual]

Reimplemented from **MHAPLugin::plugin_t< delaysum_t >** (p. 661).

5.59.3.4 `void delaysum::delaysum_if_t::update_cfg (`
 `void)` [private]

5.59.4 Member Data Documentation

5.59.4.1 `MHAParser::vfloat_t delaysum::delaysum_if_t::weights` [private]

Linear weights to be multiplied with the audio signal, one factor for each channel.

Order is [chan0, chan1, ...]

5.59.4.2 MHAParser::vint_t delaysum::delaysum_if_t::delay [private]

vector of channel-specific delays, in samples.

5.59.4.3 MHAEvents::patchbay_t<delaysum_if_t> delaysum::delaysum_if_t::patchbay [private]

The patchbay to react to config changes.

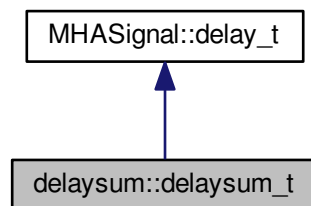
The documentation for this class was generated from the following file:

- **delaysum.cpp**

5.60 delaysum::delaysum_t Class Reference

Runtime configuration of the delaysum plugin.

Inheritance diagram for delaysum::delaysum_t:



Public Member Functions

- **delaysum_t** (unsigned int nch, unsigned int fragsize, const std::vector< **mha_real_t** > &weights_, const std::vector< int > &delays_)
Constructor of the runtime configuration.
- **mha_wave_t * process** (**mha_wave_t ***)

Private Attributes

- std::vector< **mha_real_t** > **weights**
Relative weights for each channel. Order is [chan0, chan1, ...].
- **MHASignal::waveform_t out**
Output waveform.

5.60.1 Detailed Description

Runtime configuration of the delaysum plugin.

Inherits from the already present `delay_t` class. The constructor initializes and validates the runtime configuration and forwards the delay vector to the `delay_t` class. The process function first calls `delay_t::process` and then multiplies every output channel with its weight and adds them into the output channel.

5.60.2 Constructor & Destructor Documentation

5.60.2.1 `delaysum::delaysum_t::delaysum_t (`
 `unsigned int nch,`
 `unsigned int fragsize,`
 `const std::vector< mha_real_t > & weights_,`
 `const std::vector< int > & delays_)`

Constructor of the runtime configuration.

Parameters

| | |
|------------------------|--|
| <i>nch</i> | Number of input channels. |
| <i>fragsize</i> | Size of one input fragment in frames. |
| <i>weights_</i> ↔ — | Vector of weights for each channel. |
| <i>delays_</i> ↔ — | Vector of delays, one entry per channel. |

5.60.3 Member Function Documentation

5.60.3.1 `mha_wave_t * delaysum::delaysum_t::process (`
 `mha_wave_t * signal)`

5.60.4 Member Data Documentation

5.60.4.1 `std::vector<mha_real_t> delaysum::delaysum_t::weights` [private]

Relative weights for each channel. Order is [chan0, chan1, ...].

5.60.4.2 `MHASignal::waveform_t delaysum::delaysum_t::out` [private]

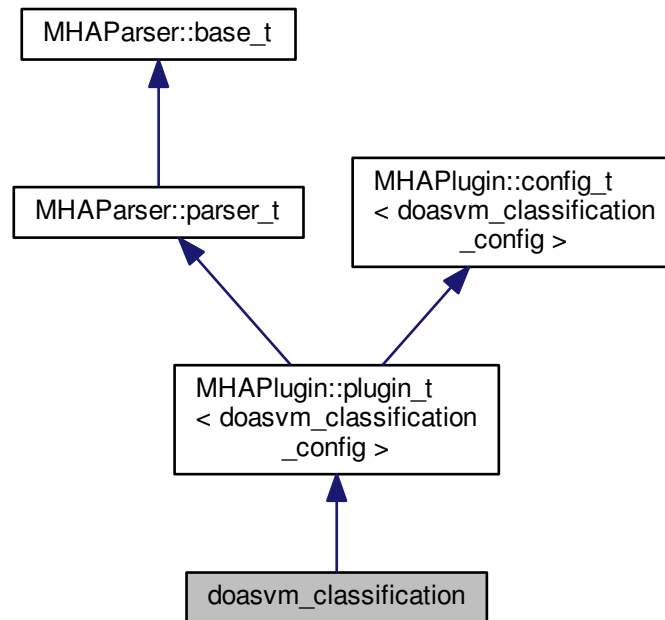
Output waveform.

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

- **delaysum.cpp**

5.61 doasvm_classification Class Reference

Inheritance diagram for doasvm_classification:



Public Member Functions

- **doasvm_classification** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~doasvm_classification** ()
- **mha_wave_t * process** (**mha_wave_t ***)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::vfloat_t** angles
- **MHAParser::mfloat_t** w
- **MHAParser::vfloat_t** b
- **MHAParser::vfloat_t** x
- **MHAParser::vfloat_t** y
- **MHAParser::string_t** p_name
- **MHAParser::string_t** max_p_ind_name
- **MHAParser::string_t** vGCC_name

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **doasvm_classification** > **patchbay**

Additional Inherited Members

5.61.1 Constructor & Destructor Documentation

5.61.1.1 **doasvm_classification::doasvm_classification** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.61.1.2 **doasvm_classification::~~doasvm_classification** ()

5.61.2 Member Function Documentation

5.61.2.1 **mha_wave_t** * **doasvm_classification::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.61.2.2 void **doasvm_classification::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

5.61.2.3 `void doasvm_classification::release (`
`void) [inline],[virtual]`

Reimplemented from **MHAPLugin::plugin_t< doasvm_classification_config >** (p. 661).

5.61.2.4 `void doasvm_classification::update_cfg () [private]`

5.61.3 Member Data Documentation

5.61.3.1 **MHAParser::vfloat_t doasvm_classification::angles**

5.61.3.2 **MHAParser::mfloat_t doasvm_classification::w**

5.61.3.3 **MHAParser::vfloat_t doasvm_classification::b**

5.61.3.4 **MHAParser::vfloat_t doasvm_classification::x**

5.61.3.5 **MHAParser::vfloat_t doasvm_classification::y**

5.61.3.6 **MHAParser::string_t doasvm_classification::p_name**

5.61.3.7 **MHAParser::string_t doasvm_classification::max_p_ind_name**

5.61.3.8 **MHAParser::string_t doasvm_classification::vGCC_name**

5.61.3.9 **MHAEvents::patchbay_t<doasvm_classification> doasvm_classification::patchbay**
`[private]`

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

5.62 doasvm_classification_config Class Reference

Public Member Functions

- **doasvm_classification_config (algo_comm_t &ac, const mhaconfig_t in_cfg, doasvm_classification *_doasvm)**
- **~doasvm_classification_config ()**
- **mha_wave_t * process (mha_wave_t *)**

Public Attributes

- **algo_comm_t & ac**
- **doasvm_classification * doasvm**
- **MHA_AC::waveform_t p**
- **MHA_AC::int_t p_max**
- **mha_wave_t c**

5.62.1 Constructor & Destructor Documentation

5.62.1.1 **doasvm_classification_config::doasvm_classification_config (
 algo_comm_t & ac,
 const mhaconfig_t in_cfg,
 doasvm_classification * _doasvm)**

5.62.1.2 **doasvm_classification_config::~doasvm_classification_config ()**

5.62.2 Member Function Documentation

5.62.2.1 **mha_wave_t * doasvm_classification_config::process (
 mha_wave_t * wave)**

5.62.3 Member Data Documentation

5.62.3.1 **algo_comm_t& doasvm_classification_config::ac**

5.62.3.2 **doasvm_classification* doasvm_classification_config::doasvm**

5.62.3.3 **MHA_AC::waveform_t doasvm_classification_config::p**

5.62.3.4 **MHA_AC::int_t doasvm_classification_config::p_max**

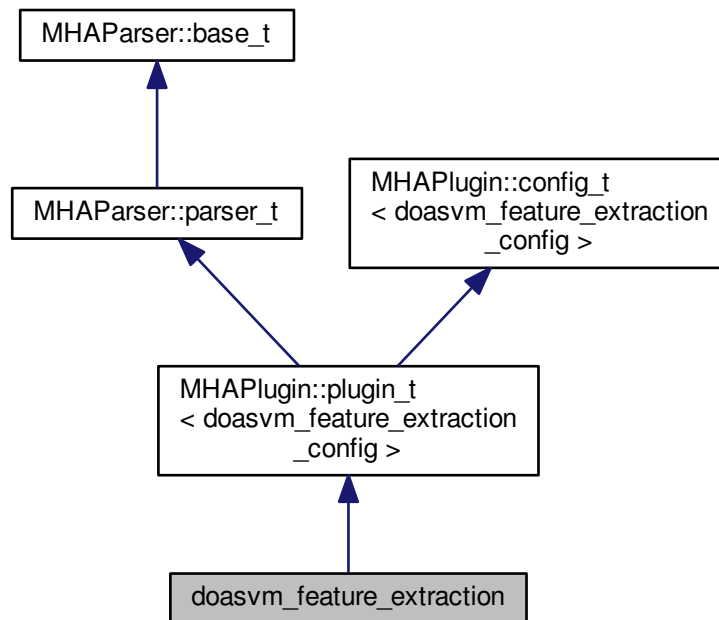
5.62.3.5 **mha_wave_t doasvm_classification_config::c**

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

5.63 doasvm_feature_extraction Class Reference

Inheritance diagram for doasvm_feature_extraction:



Public Member Functions

- **doasvm_feature_extraction** (algo_comm_t &ac, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~doasvm_feature_extraction** ()
- **mha_wave_t * process** (mha_wave_t *)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (mhaconfig_t &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t** fftlen
- **MHAParser::int_t** max_lag
- **MHAParser::int_t** nupsample
- **MHAParser::string_t** vGCC_name

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **doasvm_feature_extraction** > **patchbay**

Additional Inherited Members

5.63.1 Constructor & Destructor Documentation

5.63.1.1 **doasvm_feature_extraction::doasvm_feature_extraction** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.63.1.2 **doasvm_feature_extraction::~~doasvm_feature_extraction** ()

5.63.2 Member Function Documentation

5.63.2.1 **mha_wave_t** * **doasvm_feature_extraction::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.63.2.2 void **doasvm_feature_extraction::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

5.63.2.3 void doasvm_feature_extraction::release (
void) [inline],[virtual]

Reimplemented from **MHAPLugin::plugin_t< doasvm_feature_extraction_config >**
(p. 661).

5.63.2.4 void doasvm_feature_extraction::update_cfg () [private]

5.63.3 Member Data Documentation

5.63.3.1 **MHAParser::int_t doasvm_feature_extraction::fftl**

5.63.3.2 **MHAParser::int_t doasvm_feature_extraction::max_lag**

5.63.3.3 **MHAParser::int_t doasvm_feature_extraction::nupsample**

5.63.3.4 **MHAParser::string_t doasvm_feature_extraction::vGCC_name**

5.63.3.5 **MHAEvents::patchbay_t<doasvm_feature_extraction>**
doasvm_feature_extraction::patchbay [private]

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

- **doasvm_feature_extraction.h**
- **doasvm_feature_extraction.cpp**

5.64 doasvm_feature_extraction_config Class Reference

Public Member Functions

- **doasvm_feature_extraction_config** (algo_comm_t &ac, const mhaconfig_t in_cfg, doasvm_feature_extraction *_doagcc)
- **~doasvm_feature_extraction_config** ()
- **mha_wave_t * process** (mha_wave_t *)

Public Attributes

- **doasvm_feature_extraction * doagcc**
- unsigned int **wndlen**
- unsigned int **fftl**
- 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**

5.64.1 Constructor & Destructor Documentation

5.64.1.1 `doasvm_feature_extraction_config::doasvm_feature_extraction_config (
 algo_comm_t & ac,
 const mhaconfig_t in_cfg,
 doasvm_feature_extraction * _doagcc)`

5.64.1.2 `doasvm_feature_extraction_config::~doasvm_feature_extraction_config ()`

5.64.2 Member Function Documentation

5.64.2.1 `mha_wave_t * doasvm_feature_extraction_config::process (
 mha_wave_t * wave)`

5.64.3 Member Data Documentation

5.64.3.1 `doasvm_feature_extraction* doasvm_feature_extraction_config::doagcc`

5.64.3.2 `unsigned int doasvm_feature_extraction_config::wndlen`

5.64.3.3 `unsigned int doasvm_feature_extraction_config::fftlens`

5.64.3.4 `unsigned int doasvm_feature_extraction_config::G_length`

5.64.3.5 `unsigned int doasvm_feature_extraction_config::GCC_start`

5.64.3.6 `unsigned int doasvm_feature_extraction_config::GCC_end`

5.64.3.7 `MHA_AC::waveform_t doasvm_feature_extraction_config::vGCC_ac`

5.64.3.8 `mha_fft_t doasvm_feature_extraction_config::fft`

5.64.3.9 `mha_fft_t doasvm_feature_extraction_config::ifft`

5.64.3.10 `double doasvm_feature_extraction_config::hifftwin_sum`

5.64.3.11 `MHASignal::waveform_t doasvm_feature_extraction_config::proc_wave`

5.64.3.12 `MHASignal::waveform_t doasvm_feature_extraction_config::hwin`

5.64.3.13 `MHASignal::waveform_t doasvm_feature_extraction_config::hifftwin`

5.64.3.14 `MHASignal::waveform_t doasvm_feature_extraction_config::vGCC`

5.64.3.15 `MHASignal::spectrum_t doasvm_feature_extraction_config::in_spec`

5.64.3.16 `MHASignal::spectrum_t doasvm_feature_extraction_config::G`

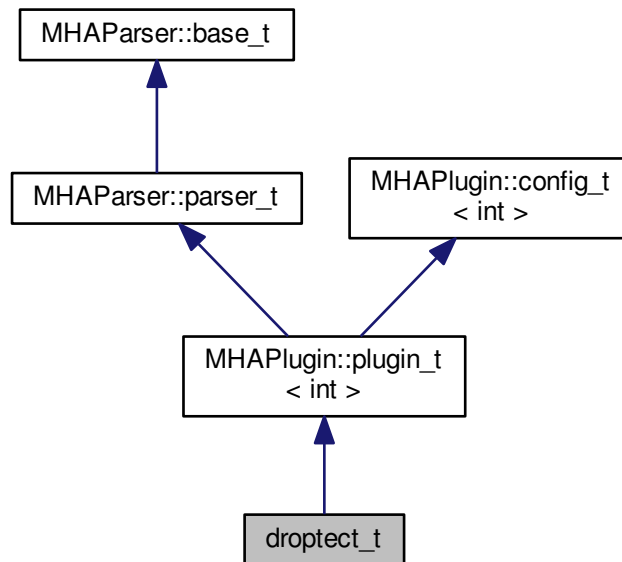
The documentation for this class was generated from the following files:

- `doasvm_feature_extraction.h`
- `doasvm_feature_extraction.cpp`

5.65 droptect_t Class Reference

Detect dropouts in a signal with a constant spectrum.

Inheritance diagram for droptect_t:



Public Member Functions

- **droptect_t** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
- void **release** (void)
- **mha_spec_t** * **process** (**mha_spec_t** *signal)

Private Attributes

- MHAParser::vint_mon_t dropouts
- MHAParser::vint_mon_t consecutive_dropouts
- MHAParser::int_mon_t blocks
- MHAParser::bool_t reset
- MHAParser::float_t threshold
- MHASignal::waveform_t * current_powspec

- **MHASignal::waveform_t** * **filtered_powspec**
- **MHAParser::float_t** **tau**
- **std::vector< bool >** **filter_activated**
- **float** **period**
The period of the process callback.
- **MHAParser::mfloat_mon_t** **filtered_powspec_mon**
User access to filtered spectrum.
- **MHAParser::float_mon_t** **level_mon**

Additional Inherited Members

5.65.1 Detailed Description

Detect dropouts in a signal with a constant spectrum.

5.65.2 Constructor & Destructor Documentation

5.65.2.1 **droptect_t::droptect_t** (
 algo_comm_t & *ac*,
 const **std::string** & *chain_name*,
 const **std::string** & *algo_name*)

This constructor initializes the configuration language variables and inserts them into the MHA configuration tree.

5.65.3 Member Function Documentation

5.65.3.1 **void droptect_t::prepare** (
 mhaconfig_t & *signal_info*) [**virtual**]

Implements **MHAPlugin::plugin_t< int >** (p. 661).

5.65.3.2 **void droptect_t::release** (
 void) [**virtual**]

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 661).

5.65.3.3 `mha_spec_t * droptect_t::process (`
 `mha_spec_t * signal)`

5.65.4 Member Data Documentation

5.65.4.1 `MHAParser::vint_mon_t droptect_t::dropouts` [private]

5.65.4.2 `MHAParser::vint_mon_t droptect_t::consecutive_dropouts` [private]

5.65.4.3 `MHAParser::int_mon_t droptect_t::blocks` [private]

5.65.4.4 `MHAParser::bool_t droptect_t::reset` [private]

5.65.4.5 `MHAParser::float_t droptect_t::threshold` [private]

5.65.4.6 `MHASignal::waveform_t* droptect_t::current_powspec` [private]

5.65.4.7 `MHASignal::waveform_t* droptect_t::filtered_powspec` [private]

5.65.4.8 `MHAParser::float_t droptect_t::tau` [private]

5.65.4.9 `std::vector<bool> droptect_t::filter_activated` [private]

5.65.4.10 `float droptect_t::period` [private]

The period of the process callback.

5.65.4.11 `MHAParser::mfloat_mon_t droptect_t::filtered_powspec_mon` [private]

User access to filtered spectrum.

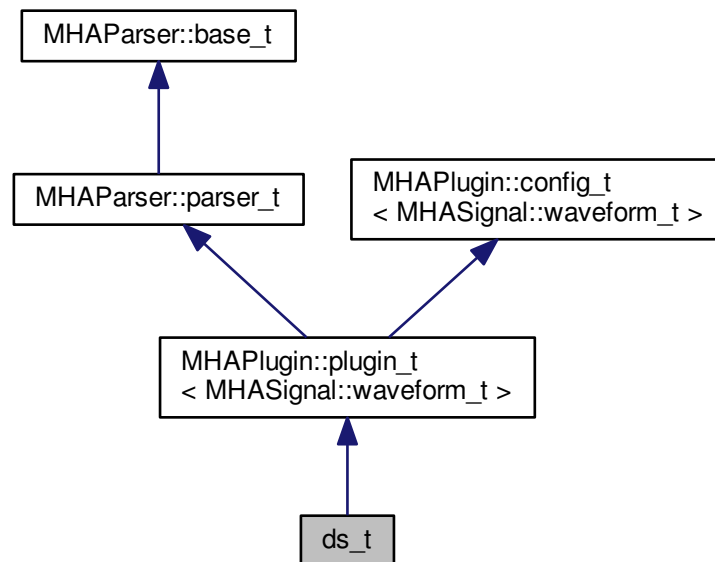
5.65.4.12 `MHAParser::float_mon_t droptect_t::level_mon` [private]

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

- **droptect.cpp**

5.66 ds_t Class Reference

Inheritance diagram for ds_t:



Public Member Functions

- **ds_t** (algo_comm_t, std::string, std::string)
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Attributes

- **MHAParser::int_t ratio**
- **MHAFilter::iir_filter_t antialias**

Additional Inherited Members

5.66.1 Constructor & Destructor Documentation

```
5.66.1.1 ds_t::ds_t (
            algo_comm_t iac,
            std::string ,
            std::string )
```

5.66.2 Member Function Documentation

```
5.66.2.1 mha_wave_t * ds_t::process (
            mha_wave_t * s )
```

```
5.66.2.2 void ds_t::prepare (
            mhaconfig_t & cf ) [virtual]
```

Implements **MHAPLugin::plugin_t**< **MHASignal::waveform_t** > (p. [661](#)).

```
5.66.2.3 void ds_t::release (
            void ) [virtual]
```

Reimplemented from **MHAPLugin::plugin_t**< **MHASignal::waveform_t** > (p. [661](#)).

5.66.3 Member Data Documentation

```
5.66.3.1 MHAParser::int_t ds_t::ratio [private]
```

```
5.66.3.2 MHAFilter::iir_filter_t ds_t::antialias [private]
```

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

- **downsample.cpp**

5.67 dynamiclib_t Class Reference

Public Member Functions

- **dynamiclib_t** (const std::string &)
- void * **resolve** (const std::string &)
- void * **resolve_checked** (const std::string &)
- **~dynamiclib_t** ()
- const std::string & **getmodulename** () const
- const std::string & **getname** () const

Private Attributes

- `std::string fullname`
- `std::string modulename`
- `mha_libhandle_t h`

5.67.1 Constructor & Destructor Documentation

5.67.1.1 `dynamiclib_t::dynamiclib_t (`
 `const std::string & n)`

5.67.1.2 `dynamiclib_t::~~dynamiclib_t ()`

5.67.2 Member Function Documentation

5.67.2.1 `void * dynamiclib_t::resolve (`
 `const std::string & n)`

5.67.2.2 `void * dynamiclib_t::resolve_checked (`
 `const std::string & n)`

5.67.2.3 `const std::string& dynamiclib_t::getmodulename () const` `[inline]`

5.67.2.4 `const std::string& dynamiclib_t::getname () const` `[inline]`

5.67.3 Member Data Documentation

5.67.3.1 `std::string dynamiclib_t::fullname` `[private]`

5.67.3.2 `std::string dynamiclib_t::modulename` `[private]`

5.67.3.3 `mha_libhandle_t dynamiclib_t::h` `[private]`

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

- `mha_os.h`
- `mha_os.cpp`

5.68 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**

5.68.1 Detailed Description

Real-time class for after burn effect.

The constructor processes the parameters and creates pre-processed variables for efficient realtime processing.

5.68.2 Constructor & Destructor Documentation

5.68.2.1 DynComp::dc_afterburn_rt_t::dc_afterburn_rt_t (
const std::vector< float > &cf,
unsigned int *channels*,
float *srate*,
const **dc_afterburn_vars_t** &vars)

5.68.3 Member Function Documentation

5.68.3.1 void DynComp::dc_afterburn_rt_t::burn (
float & *Gin*,
float *Lin*,
unsigned int *band*,
unsigned int *channel*) [inline]

gain modifier method (afterburn).

Parameters

| | |
|----------------|-----------------------|
| <i>Gin</i> | Linear gain. |
| <i>Lin</i> | Input level (Pascal). |
| <i>band</i> | Filter band number. |
| <i>channel</i> | Channel number. |

Output level for MPO is estimated by $Gin * Lin$.

5.68.4 Member Data Documentation

5.68.4.1 `std::vector<float> DynComp::dc_afterburn_rt_t::drain_inv` [private]

5.68.4.2 `std::vector<float> DynComp::dc_afterburn_rt_t::conflux` [private]

5.68.4.3 `std::vector<float> DynComp::dc_afterburn_rt_t::maxgain` [private]

5.68.4.4 `std::vector<float> DynComp::dc_afterburn_rt_t::mpo_inv` [private]

5.68.4.5 `std::vector<MHAFilter::o1flt_lowpass_t> DynComp::dc_afterburn_rt_t::lp` [private]

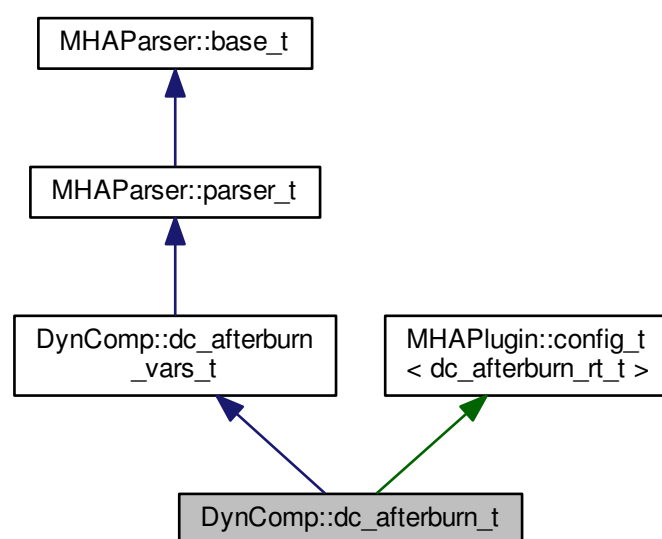
The documentation for this class was generated from the following files:

- `dc_afterburn.h`
- `dc_afterburn.cpp`

5.69 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

5.69.1 Detailed Description

Afterburn class, to be defined as a member of compressors.

5.69.2 Constructor & Destructor Documentation

5.69.2.1 DynComp::dc_afterburn_t::dc_afterburn_t ()

5.69.3 Member Function Documentation

5.69.3.1 void DynComp::dc_afterburn_t::set_fb_pars (const std::vector< float > &cf, unsigned int *channels*, float *srate*)

5.69.3.2 void DynComp::dc_afterburn_t::unset_fb_pars ()

5.69.3.3 void DynComp::dc_afterburn_t::update_burner () [inline]

5.69.3.4 void DynComp::dc_afterburn_t::burn (
float & *Gin*,
float *Lin*,
unsigned int *band*,
unsigned int *channel*) [inline]

5.69.3.5 void DynComp::dc_afterburn_t::update () [private]

5.69.4 Member Data Documentation

5.69.4.1 MHAEvents::patchbay_t<dc_afterburn_t> DynComp::dc_afterburn_t::patchbay
[private]

5.69.4.2 std::vector<float> DynComp::dc_afterburn_t::_cf [private]

5.69.4.3 unsigned int DynComp::dc_afterburn_t::_channels [private]

5.69.4.4 float DynComp::dc_afterburn_t::_srate [private]

5.69.4.5 bool DynComp::dc_afterburn_t::commit_pending [private]

5.69.4.6 bool DynComp::dc_afterburn_t::fb_pars_configured [private]

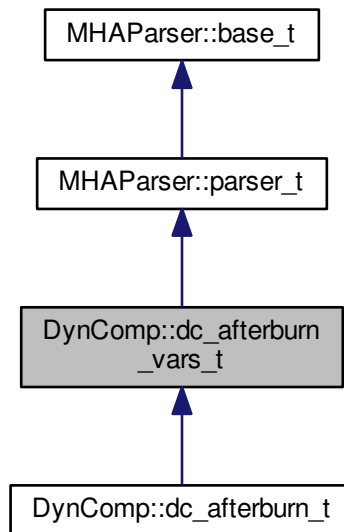
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.70 DynComp::dc_afterburn_vars_t Class Reference

Variables for **dc_afterburn_t** (p. [257](#)) 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

5.70.1 Detailed Description

Variables for **dc_afterburn_t** (p. [257](#)) class.

5.70.2 Constructor & Destructor Documentation

5.70.2.1 DynComp::dc_afterburn_vars_t::dc_afterburn_vars_t()

5.70.3 Member Data Documentation

5.70.3.1 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::f

5.70.3.2 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::drain

5.70.3.3 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::conflux

5.70.3.4 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::maxgain

5.70.3.5 MHAParser::vfloat_t DynComp::dc_afterburn_vars_t::mpo

5.70.3.6 MHAParser::float_t DynComp::dc_afterburn_vars_t::taugain

5.70.3.7 MHAParser::kw_t DynComp::dc_afterburn_vars_t::commit

5.70.3.8 MHAParser::bool_t DynComp::dc_afterburn_vars_t::bypass

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

- **dc_afterburn.h**
- **dc_afterburn.cpp**

5.71 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**

5.71.1 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. 264) member function via a gain prescription rule from an auditory profile.

5.71.2 Constructor & Destructor Documentation

5.71.2.1 `gaintable_t::gaintable_t (`
 `const std::vector< mha_real_t > & LInput,`
 `const std::vector< mha_real_t > & FCenter,`
 `unsigned int channels)`

Constructor.

Parameters

| | |
|-----------------|---|
| <i>LInput</i> | Input level samples, in equivalent LTASS_combined dB SPL. |
| <i>FCenter</i> | Frequency samples in Hz (e.g., center frequencies of filterbank). |
| <i>channels</i> | Number of audio channels (typically 2). |

5.71.2.2 gaintable_t::~~gaintable_t ()**5.71.3 Member Function Documentation****5.71.3.1 void gaintable_t::update (
std::vector< std::vector< std::vector< mha_real_t > > > newGain)**

Update gains from an external table.

Parameters

| | |
|----------------|-------------------------|
| <i>newGain</i> | New gain table entries. |
|----------------|-------------------------|

Dimension change is not allowed. The number of entries are checked.

**5.71.3.2 mha_real_t gaintable_t::get_gain (
mha_real_t Lin,
mha_real_t Fin,
unsigned int channel)**

Read Gain from gain table.

Parameters

| | |
|----------------|-------------------------------------|
| <i>Lin</i> | Input level |
| <i>Fin</i> | Input frequency (no match required) |
| <i>channel</i> | Audio channel |

**5.71.3.3 mha_real_t gaintable_t::get_gain (
mha_real_t Lin,
unsigned int band,
unsigned int channel)**

Read Gain from gain table.

Parameters

| | |
|------------|-------------|
| <i>Lin</i> | Input level |
|------------|-------------|

Parameters

| | |
|----------------|----------------------|
| <i>band</i> | Input frequency band |
| <i>channel</i> | Audio channel |

5.71.3.4 void `gaintable_t::get_gain (`
 `const mha_wave_t & Lin,`
 `mha_wave_t & Gain)`

Read Gains from gain table.

Parameters

| | |
|-------------|---------------|
| <i>Lin</i> | Input levels. |
| <i>Gain</i> | Output gain. |

The number of channels in Lin and Gain must match the number of bands times number of channels in the gaintable.

5.71.3.5 unsigned int `DynComp::gaintable_t::nbands () const` [inline]

Return number of frequency bands.

5.71.3.6 unsigned int `DynComp::gaintable_t::nchannels () const` [inline]

Return number of audio channels.

5.71.3.7 std::vector< std::vector< mha_real_t > > `gaintable_t::get_iofun () const`

Return current input-output function.

5.71.3.8 std::vector<mha_real_t> `DynComp::gaintable_t::get_vL () const` [inline]

5.71.3.9 std::vector<mha_real_t> `DynComp::gaintable_t::get_vF () const` [inline]

5.71.4 Member Data Documentation

5.71.4.1 unsigned int `DynComp::gaintable_t::num_L` [private]

5.71.4.2 unsigned int `DynComp::gaintable_t::num_F` [private]

5.71.4.3 unsigned int `DynComp::gaintable_t::num_channels` [private]

5.71.4.4 `std::vector<mha_real_t> DynComp::gaintable_t::vL` [private]

5.71.4.5 `std::vector<mha_real_t> DynComp::gaintable_t::vF` [private]

5.71.4.6 `std::vector<mha_real_t> DynComp::gaintable_t::vFlog` [private]

5.71.4.7 `std::vector<std::vector<std::vector<mha_real_t>>> DynComp::gaintable_t::data`
[private]

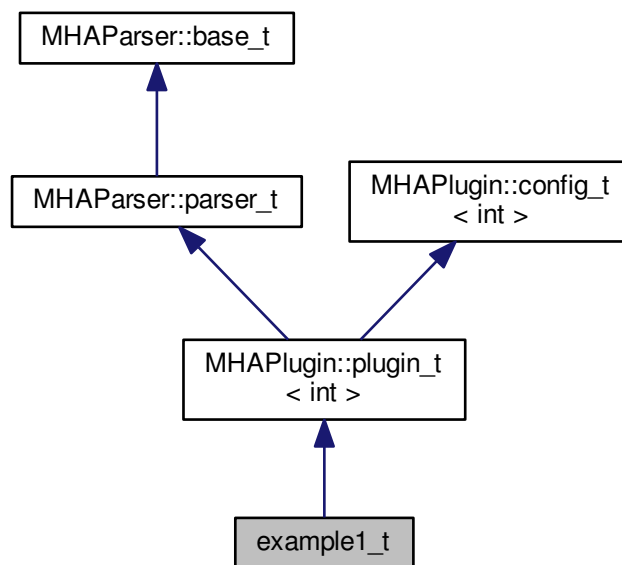
The documentation for this class was generated from the following files:

- **gaintable.h**
- **gaintable.cpp**

5.72 example1_t Class Reference

This C++ class implements the simplest example plugin for the step-by-step tutorial.

Inheritance diagram for example1_t:



Public Member Functions

- **example1_t** (**algo_comm_t** &**ac**, const std::string &**chain_name**, const std::string &**algo_name**)
Do-nothing constructor.
- void **release** (void)
Release may be empty.
- void **prepare** (**mhaconfig_t** &**signal_info**)
Plugin preparation.
- **mha_wave_t** * **process** (**mha_wave_t** ***signal**)
Signal processing performed by the plugin.

Additional Inherited Members

5.72.1 Detailed Description

This C++ class implements the simplest example plugin for the step-by-step tutorial.

It inherits from **MHAPLugin::plugin_t** (p. 659) for correct integration in the configuration language interface.

5.72.2 Constructor & Destructor Documentation

5.72.2.1 **example1_t::example1_t** (
 algo_comm_t & **ac**,
 const std::string & **chain_name**,
 const std::string & **algo_name**) [inline]

Do-nothing constructor.

The constructor has to take these three arguments, but it does not have to use them. However, the base class has to be initialized.

5.72.3 Member Function Documentation

5.72.3.1 void **example1_t::release** (
 void) [inline], [virtual]

Release may be empty.

Reimplemented from **MHAPLugin::plugin_t**< int > (p. 661).

5.72.3.2 void **example1_t::prepare** (
 mhaconfig_t & **signal_info**) [inline], [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains at least one channel

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

```
5.72.3.3 mha_wave_t* example1_t::process (
           mha_wave_t* signal ) [inline]
```

Signal processing performed by the plugin.

This plugin multiplies the signal in the first audio channel by a factor 0.1.

Parameters

| | |
|---------------|--|
| <i>signal</i> | 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)

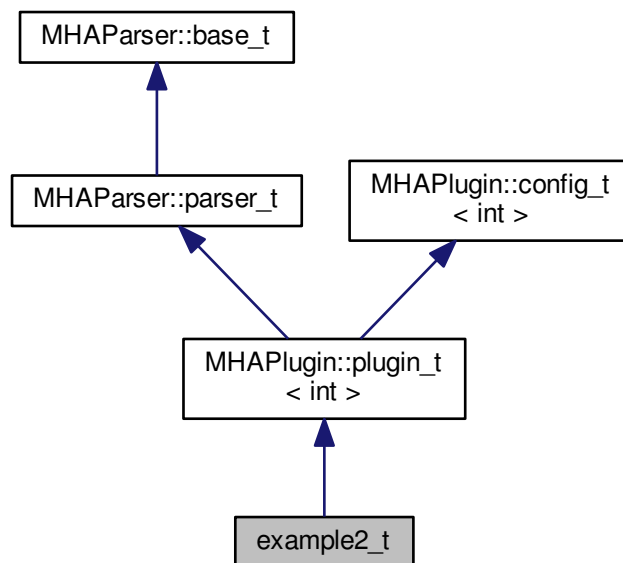
The documentation for this class was generated from the following file:

- **example1.cpp**

5.73 example2_t Class Reference

This C++ class implements the second example plugin for the step-by-step tutorial.

Inheritance diagram for example2_t:



Public Member Functions

- **example2_t** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
Plugin preparation.
- void **release** (void)
Undo restrictions posed in prepare.
- **mha_wave_t** * **process** (**mha_wave_t** *signal)
Signal processing performed by the plugin.

Private Attributes

- **MHAParser::int_t** **scale_ch**
Index of audio channel to scale.
- **MHAParser::float_t** **factor**
The scaling factor applied to the selected channel.

Additional Inherited Members

5.73.1 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.

5.73.2 Constructor & Destructor Documentation

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

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

5.73.3 Member Function Documentation

5.73.3.1 `void example2_t::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

5.73.3.2 `void example2_t::release (`
`void) [virtual]`

Undo restrictions posed in prepare.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. [661](#)).

5.73.3.3 `mha_wave_t * example2_t::process (`
`mha_wave_t * signal)`

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

| | |
|---------------|--|
| <i>signal</i> | 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)

5.73.4 Member Data Documentation

5.73.4.1 `MHAParser::int_t example2_t::scale_ch` [private]

Index of audio channel to scale.

5.73.4.2 `MHAParser::float_t example2_t::factor` [private]

The scaling factor applied to the selected channel.

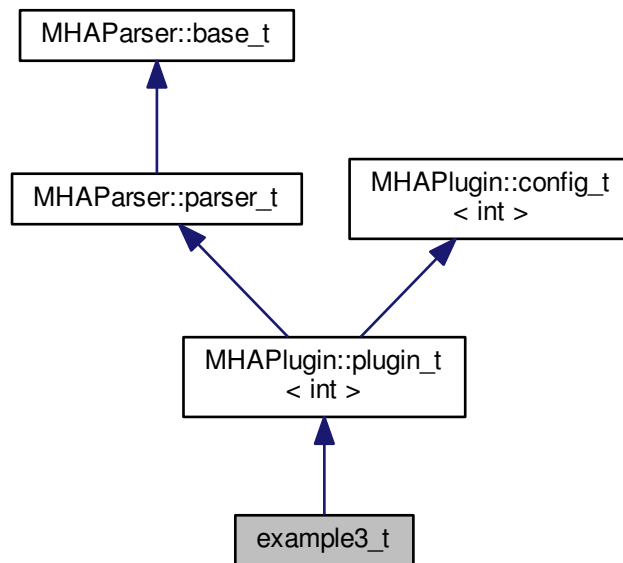
The documentation for this class was generated from the following file:

- **example2.cpp**

5.74 example3_t Class Reference

A Plugin class using the openMHA Event mechanism.

Inheritance diagram for example3_t:



Public Member Functions

- **example3_t** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
Plugin preparation.
- void **release** (void)
Bookkeeping only.
- **mha_wave_t** * **process** (**mha_wave_t** *signal)
Signal processing performed by the plugin.

Private Member Functions

- void **on_scale_ch_writeaccess** ()
- void **on_scale_ch_valuechanged** ()
- void **on_scale_ch_readaccess** ()
- void **on_prereadaccess** ()

Private Attributes

- **MHAParser::int_t scale_ch**
Index of audio channel to scale.
- **MHAParser::float_t factor**
The scaling factor applied to the selected channel.
- **MHAParser::int_mon_t prepared**
Keep Track of the prepare/release calls.
- **MHAEvents::patchbay_t < example3_t > patchbay**
The Event connector.

Additional Inherited Members

5.74.1 Detailed Description

A Plugin class using the openMHA Event mechanism.

This is the third example plugin for the step-by-step tutorial.

5.74.2 Constructor & Destructor Documentation

5.74.2.1 `example3_t::example3_t (`
 `algo_comm_t & ac,`
 `const std::string & chain_name,`
 `const std::string & algo_name)`

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

5.74.3 Member Function Documentation

5.74.3.1 `void example3_t::on_scale_ch_writeaccess () [private]`

5.74.3.2 `void example3_t::on_scale_ch_valuechanged () [private]`

5.74.3.3 `void example3_t::on_scale_ch_readaccess () [private]`

5.74.3.4 `void example3_t::on_prereadaccess () [private]`

5.74.3.5 `void example3_t::prepare (`
 `mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. 661).

5.74.3.6 void example3_t::release (
void) [virtual]

Bookkeeping only.

Reimplemented from **MHAPLugin::plugin_t< int >** (p. 661).

5.74.3.7 mha_wave_t * example3_t::process (
mha_wave_t * signal)

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

| | |
|---------------|--|
| <i>signal</i> | 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)

5.74.4 Member Data Documentation

5.74.4.1 MHAParser::int_t example3_t::scale_ch [private]

Index of audio channel to scale.

5.74.4.2 MHAParser::float_t example3_t::factor [private]

The scaling factor applied to the selected channel.

5.74.4.3 MHAParser::int_mon_t example3_t::prepared [private]

Keep Track of the prepare/release calls.

5.74.4.4 MHAEvents::patchbay_t<example3_t> example3_t::patchbay [private]

The Event connector.

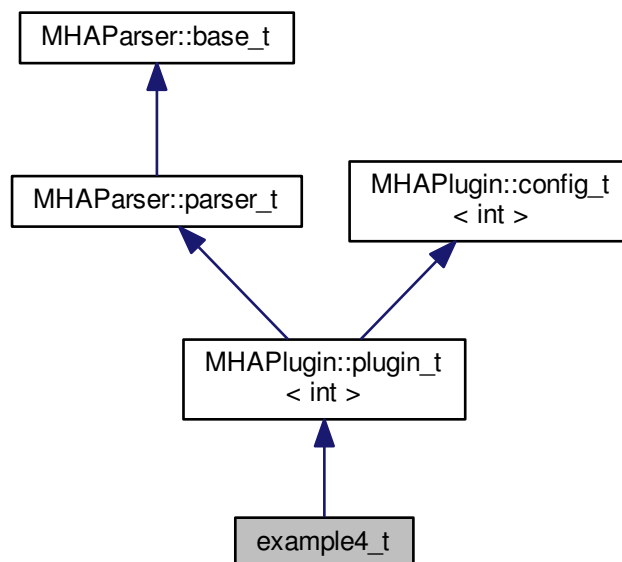
The documentation for this class was generated from the following file:

- **example3.cpp**

5.75 example4_t Class Reference

A Plugin class using the spectral signal.

Inheritance diagram for example4_t:



Public Member Functions

- **example4_t** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.
- void **prepare** (**mhaconfig_t** &signal_info)
Plugin preparation.
- void **release** (void)
Bookkeeping only.
- **mha_spec_t** * **process** (**mha_spec_t** *signal)
Signal processing performed by the plugin.

Private Member Functions

- void **on_scale_ch_writeaccess** ()
- void **on_scale_ch_valuechanged** ()
- void **on_scale_ch_readaccess** ()
- void **on_prereadaccess** ()

Private Attributes

- **MHAParser::int_t scale_ch**
Index of audio channel to scale.
- **MHAParser::float_t factor**
The scaling factor applied to the selected channel.
- **MHAParser::int_mon_t prepared**
Keep Track of the prepare/release calls.
- **MHAEvents::patchbay_t< example4_t > patchbay**
The Event connector.

Additional Inherited Members

5.75.1 Detailed Description

A Plugin class using the spectral signal.

This is the fourth example plugin for the step-by-step tutorial.

5.75.2 Constructor & Destructor Documentation

5.75.2.1 **example4_t::example4_t** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

This constructor initializes the configuration language variables and inserts them into the openMHA configuration tree.

It connects the openMHA Events triggered by these configuration variables to the respective callbacks.

5.75.3 Member Function Documentation

5.75.3.1 `void example4_t::on_scale_ch_writeaccess () [private]`

5.75.3.2 `void example4_t::on_scale_ch_valuechanged () [private]`

5.75.3.3 `void example4_t::on_scale_ch_readaccess () [private]`

5.75.3.4 `void example4_t::on_prereadaccess () [private]`

5.75.3.5 `void example4_t::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

| | |
|--------------------|--|
| <i>signal_info</i> | 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. 661).

5.75.3.6 `void example4_t::release (`
`void) [virtual]`

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 661).

5.75.3.7 `mha_spec_t * example4_t::process (`
`mha_spec_t * signal)`

Signal processing performed by the plugin.

This plugin multiplies the spectral signal in the selected audio channel by the configured factor.

Parameters

| | |
|---------------|--|
| <i>signal</i> | 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)

5.75.4 Member Data Documentation

5.75.4.1 `MHAParser::int_t example4_t::scale_ch` [private]

Index of audio channel to scale.

5.75.4.2 `MHAParser::float_t example4_t::factor` [private]

The scaling factor applied to the selected channel.

5.75.4.3 `MHAParser::int_mon_t example4_t::prepared` [private]

Keep Track of the prepare/release calls.

5.75.4.4 `MHAEvents::patchbay_t<example4_t> example4_t::patchbay` [private]

The Event connector.

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

- **example4.cpp**

5.76 `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`**

5.76.1 Constructor & Destructor Documentation

5.76.1.1 `example5_t::example5_t (`
 `unsigned int ichannel,`
 `unsigned int numchannels,`
 `mha_real_t iscale)`

5.76.2 Member Function Documentation

5.76.2.1 `mha_spec_t * example5_t::process (`
 `mha_spec_t * spec)`

5.76.3 Member Data Documentation

5.76.3.1 `unsigned int example5_t::channel` [private]

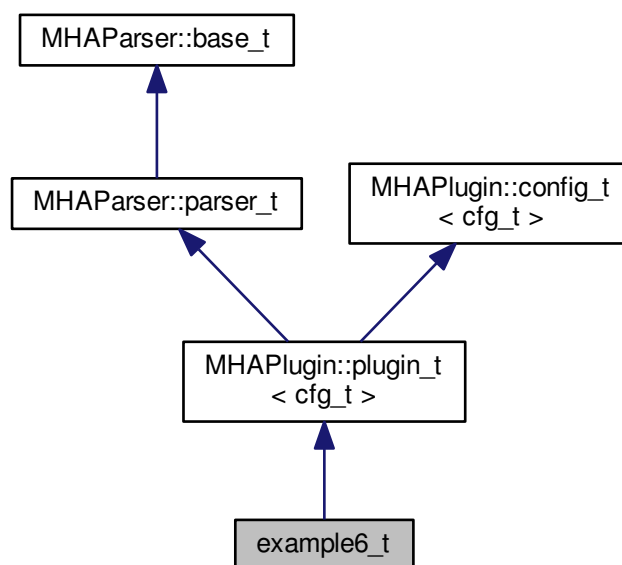
5.76.3.2 `mha_real_t example5_t::scale` [private]

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

- **example5.cpp**

5.77 example6_t Class Reference

Inheritance diagram for example6_t:



Public Member Functions

- **example6_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t * process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::int_t channel_no**
- float **rmsdb**
- **MHAEvents::patchbay_t< example6_t > patchbay**

Additional Inherited Members

5.77.1 Constructor & Destructor Documentation

5.77.1.1 **example6_t::example6_t** (
 const **algo_comm_t** & *i*ac,
 const std::string & ,
 const std::string &)

5.77.2 Member Function Documentation

5.77.2.1 **mha_wave_t * example6_t::process** (
 mha_wave_t * *wave*)

5.77.2.2 void **example6_t::prepare** (
 mhaconfig_t & *tfcfg*) [virtual]

Implements **MHAPLugin::plugin_t< cfg_t >** (p. 661).

5.77.2.3 void **example6_t::update_cfg** () [private]

5.77.3 Member Data Documentation

5.77.3.1 **MHAParser::int_t example6_t::channel_no** [private]

5.77.3.2 float **example6_t::rmsdb** [private]

5.77.3.3 **MHAEvents::patchbay_t<example6_t> example6_t::patchbay** [private]

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

- **example6.cpp**

5.78 expression_t Class Reference

Class for separating a string into a left hand value and a right hand value.

5.78.1 Detailed Description

Class for separating a string into a left hand value and a right hand value.

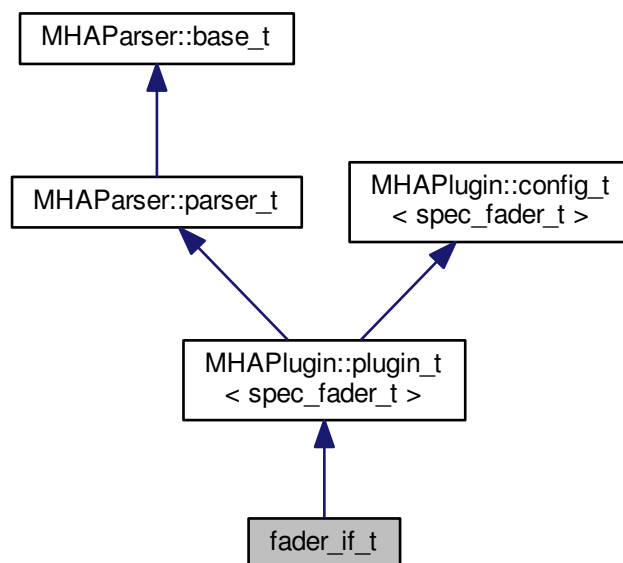
A list of valid operators can be provided. After construction, the class members lval, rval and op contain the appropriate contents.

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

- **mha_parser.cpp**

5.79 fader_if_t Class Reference

Inheritance diagram for fader_if_t:



Public Member Functions

- **fader_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t * process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t**< **fader_if_t** > **patchbay**
- **MHAParser::float_t** **tau**
- **MHAParser::vfloat_t** **newgains**
- **mha_real_t** * **actgains**

Additional Inherited Members

5.79.1 Constructor & Destructor Documentation

5.79.1.1 **fader_if_t::fader_if_t** (
 const **algo_comm_t** & *iac*,
 const std::string & ,
 const std::string &)

5.79.2 Member Function Documentation

5.79.2.1 **mha_spec_t** * **fader_if_t::process** (
 mha_spec_t * *s*)

5.79.2.2 void **fader_if_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Implements **MHAPLugin::plugin_t**< **spec_fader_t** > (p. 661).

5.79.2.3 void **fader_if_t::update_cfg** (
 void) [private]

5.79.3 Member Data Documentation

5.79.3.1 **MHAEvents::patchbay_t**<**fader_if_t**> **fader_if_t::patchbay** [private]

5.79.3.2 **MHAParser::float_t** **fader_if_t::tau** [private]

5.79.3.3 **MHAParser::vfloat_t** **fader_if_t::newgains** [private]

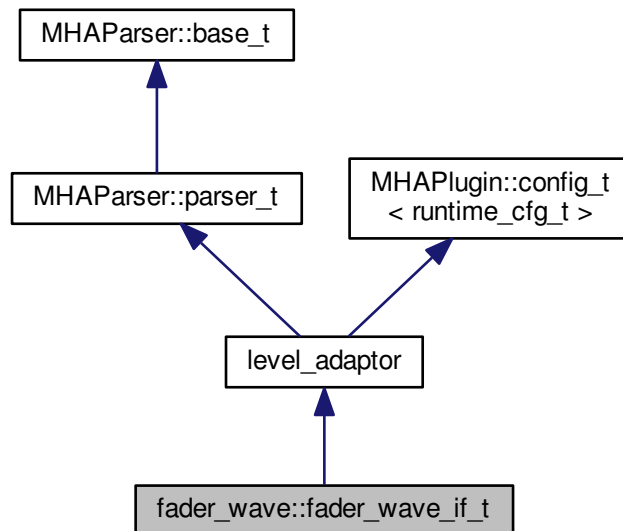
5.79.3.4 **mha_real_t*** **fader_if_t::actgains** [private]

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

- **fader_spec.cpp**

5.80 fader_wave::fader_wave_if_t Class Reference

Inheritance diagram for fader_wave::fader_wave_if_t:



Public Member Functions

- **fader_wave_if_t** (algo_comm_t, const char *, const char *)
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Member Functions

- void **set_level** ()

Private Attributes

- **MHAParser::vfloat_t** gain
- **MHAParser::float_t** ramplen
- **MHAEvents::patchbay_t** < fader_wave_if_t > patchbay
- bool **prepared**

Additional Inherited Members

5.80.1 Constructor & Destructor Documentation

5.80.1.1 `fader_wave::fader_wave_if_t::fader_wave_if_t (`
 `algo_comm_t iac,`
 `const char *,`
 `const char *)`

5.80.2 Member Function Documentation

5.80.2.1 `mha_wave_t * fader_wave::fader_wave_if_t::process (`
 `mha_wave_t * s)`

5.80.2.2 `void fader_wave::fader_wave_if_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPLugin::plugin_t< runtime_cfg_t >** (p. [661](#)).

5.80.2.3 `void fader_wave::fader_wave_if_t::release (`
 `void) [virtual]`

Reimplemented from **MHAPLugin::plugin_t< runtime_cfg_t >** (p. [661](#)).

5.80.2.4 `void fader_wave::fader_wave_if_t::set_level () [private]`

5.80.3 Member Data Documentation

5.80.3.1 **MHAParser::vfloat_t fader_wave::fader_wave_if_t::gain** [private]

5.80.3.2 **MHAParser::float_t fader_wave::fader_wave_if_t::ramplen** [private]

5.80.3.3 **MHAEvents::patchbay_t<fader_wave_if_t> fader_wave::fader_wave_if_t::patchbay**
 [private]

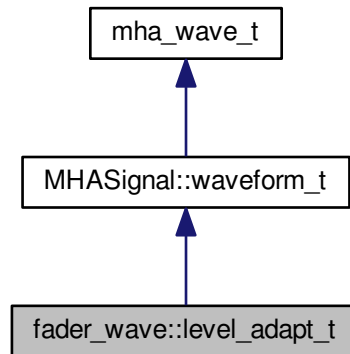
5.80.3.4 **bool fader_wave::fader_wave_if_t::prepared** [private]

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

- **fader_wave.cpp**

5.81 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

5.81.1 Constructor & Destructor Documentation

5.81.1.1 fader_wave::level_adapt_t::level_adapt_t (
 mhaconfig_t cf,
 mha_real_t adapt_len,
 std::vector< float > l_new_,
 std::vector< float > l_old_)

5.81.2 Member Function Documentation

5.81.2.1 void fader_wave::level_adapt_t::update_frame ()

5.81.2.2 std::vector<float> fader_wave::level_adapt_t::get_level () const [inline]

5.81.2.3 bool fader_wave::level_adapt_t::can_update () const [inline]

5.81.3 Member Data Documentation

5.81.3.1 unsigned int fader_wave::level_adapt_t::ilen [private]

5.81.3.2 unsigned int fader_wave::level_adapt_t::pos [private]

5.81.3.3 MHAWindow::fun_t fader_wave::level_adapt_t::wnd [private]

5.81.3.4 std::vector<float> fader_wave::level_adapt_t::l_new [private]

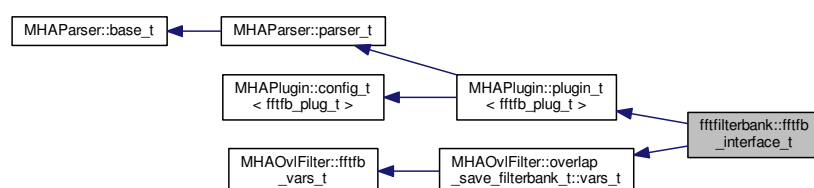
5.81.3.5 std::vector<float> fader_wave::level_adapt_t::l_old [private]

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

- fader_wave.cpp

5.82 fftfilterbank::fftfb_interface_t Class Reference

Inheritance diagram for fftfilterbank::fftfb_interface_t:



Public Member Functions

- **fftfb_interface_t** (const **algo_comm_t** &ac, const std::string &th, const std::string &al)
Default values are set and MHA configuration variables registered into the parser.
- void **prepare** (**mhaconfig_t** &)
Prepare all variables for processing.
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::bool_t** **return_imag**
- **MHAEvents::patchbay_t** < **fftfb_interface_t** > **patchbay**
- **MHA_AC::int_t** **nchannels**
- std::string **algo**
- bool **prepared**
- unsigned int **nbands**

Additional Inherited Members

5.82.1 Constructor & Destructor Documentation

5.82.1.1 **fftfilterbank::fftfb_interface_t::fftfb_interface_t** (
 const **algo_comm_t** & *ac*,
 const std::string & *th*,
 const std::string & *al*)

Default values are set and MHA configuration variables registered into the parser.

Parameters

| | |
|-----------|--------------------------------|
| <i>ac</i> | algorithm communication handle |
| <i>th</i> | chain name |
| <i>al</i> | algorithm name |

5.82.2 Member Function Documentation

5.82.2.1 void **fftfilterbank::fftfb_interface_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Prepare all variables for processing.

In this function, all variables are initialised and the filter shapes for each band are calculated. The filter shapes $W(f)$ are defined as

$$W(f) = W(T(S(f))) = W(x), \quad x = T(S(f)) = T(\hat{f}),$$

$W(x)$ being 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

| | |
|-----------|-----------------------|
| <i>tf</i> | Channel configuration |
|-----------|-----------------------|

Implements **MHAPLugin::plugin_t**< **fftfb_plug_t** > (p. 661).

5.82.2.2 void fftfilterbank::fftfb_interface_t::release (
void) [virtual]

Reimplemented from **MHAPLugin::plugin_t**< **fftfb_plug_t** > (p. 661).

5.82.2.3 mha_spec_t * fftfilterbank::fftfb_interface_t::process (
mha_spec_t * s)

5.82.2.4 mha_wave_t * fftfilterbank::fftfb_interface_t::process (
mha_wave_t * s)

5.82.2.5 void fftfilterbank::fftfb_interface_t::update_cfg (
void) [private]

5.82.3 Member Data Documentation

5.82.3.1 MHAParser::bool_t fftfilterbank::fftfb_interface_t::return_imag [private]

5.82.3.2 MHAEvents::patchbay_t<fftfb_interface_t> fftfilterbank::fftfb_interface_t::patchbay
[private]

5.82.3.3 MHA_AC::int_t fftfilterbank::fftfb_interface_t::nchannels [private]

5.82.3.4 std::string fftfilterbank::fftfb_interface_t::algo [private]

5.82.3.5 bool fftfilterbank::fftfb_interface_t::prepared [private]

5.82.3.6 unsigned int fftfilterbank::fftfb_interface_t::nbands [private]

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

- **fftfilterbank.cpp**

Additional Inherited Members

5.83.1 Constructor & Destructor Documentation

5.83.1.1 `fftfilterbank::fftfb_plug_t::fftfb_plug_t (`
 `MHAOviFilter::overlap_save_filterbank_t::vars_t & vars,`
 `mhaconfig_t chcfg,`
 `algo_comm_t ac,`
 `std::string alg,`
 `bool return_imag)`

5.83.2 Member Function Documentation

5.83.2.1 `mha_spec_t * fftfilterbank::fftfb_plug_t::process (`
 `mha_spec_t * s)`

5.83.2.2 `mha_wave_t * fftfilterbank::fftfb_plug_t::process (`
 `mha_wave_t * s)`

5.83.2.3 `void fftfilterbank::fftfb_plug_t::insert ()`

5.83.3 Member Data Documentation

5.83.3.1 `MHAOviFilter::fftfb_ac_info_t fftfilterbank::fftfb_plug_t::fb_acinfo` `[private]`

5.83.3.2 `MHASignal::spectrum_t fftfilterbank::fftfb_plug_t::s_out` `[private]`

5.83.3.3 `MHA_AC::waveform_t fftfilterbank::fftfb_plug_t::imag` `[private]`

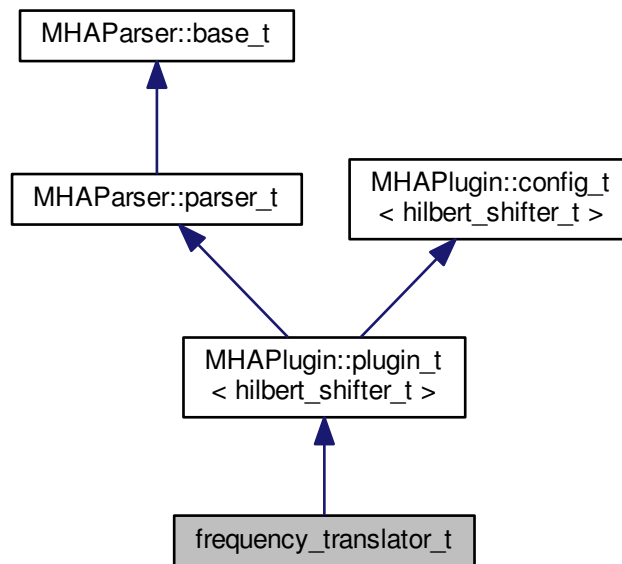
5.83.3.4 `bool fftfilterbank::fftfb_plug_t::return_imag_` `[private]`

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

- `fftfilterbank.cpp`

5.84 frequency_translator_t Class Reference

Inheritance diagram for frequency_translator_t:



Public Member Functions

- **frequency_translator_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t * process** (mha_spec_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t** < frequency_translator_t > patchbay
- **MHAParser::vfloat_t** df
- **MHAParser::float_t** fmin
- **MHAParser::float_t** fmax
- **MHAParser::int_t** irslen
- **MHAParser::kw_t** phasemode

Additional Inherited Members

5.84.1 Constructor & Destructor Documentation

5.84.1.1 frequency_translator_t::frequency_translator_t (
 const algo_comm_t & iac,
 const std::string & ith,
 const std::string & ial)

5.84.2 Member Function Documentation

5.84.2.1 mha_spec_t * frequency_translator_t::process (
 mha_spec_t * s)

5.84.2.2 void frequency_translator_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPLugin::plugin_t< hilbert_shifter_t >** (p. 661).

5.84.2.3 void frequency_translator_t::release (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t< hilbert_shifter_t >** (p. 661).

5.84.2.4 void frequency_translator_t::update () [private]

5.84.3 Member Data Documentation

5.84.3.1 MHAEvents::patchbay_t<frequency_translator_t> frequency_translator_t::patchbay
 [private]

5.84.3.2 MHAParser::vfloat_t frequency_translator_t::df [private]

5.84.3.3 MHAParser::float_t frequency_translator_t::fmin [private]

5.84.3.4 MHAParser::float_t frequency_translator_t::fmax [private]

5.84.3.5 MHAParser::int_t frequency_translator_t::irslen [private]

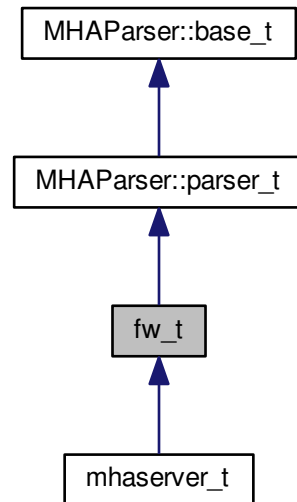
5.84.3.6 MHAParser::kw_t frequency_translator_t::phasemode [private]

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

- fshift_hilbert.cpp

5.85 fw_t Class Reference

Inheritance diagram for fw_t:



Public Member Functions

- **fw_t** ()
- **~fw_t** ()
- bool **exit_request** () const

Protected Attributes

- int **proc_error**
- int **io_error**

Private Types

Private Member Functions

- void **prepare** ()
preparation for processing
- void **start** ()
start of processing
- void **stop** ()

- stop/pause of processing*
- void **release** ()
 - release of IO device*
- void **quit** ()
 - controlled quit*
- void **stopped** (int, int)
- void **started** ()
- int **process** (mha_wave_t *, mha_wave_t **)
- void **exec_fw_command** ()
- void **load_proc_lib** ()
- void **load_io_lib** ()
- void **fw_sleep_cmd** ()
- void **fw_until_cmd** ()
- void **get_input_signal_dimension** ()
- void **async_read** ()
- void **async_poll_msg** ()
- void **get_parserstate** ()

Static Private Member Functions

- static void **stopped** (void *h, int proc_err, int io_err)
- static void **started** (void *h)
- static int **process** (void *h, mha_wave_t *sIn, mha_wave_t **sOut)

Private Attributes

- fw_vars_t prepare_vars
- MHAParser::int_mon_t nchannels_out
- MHAParser::string_t proc_name
- MHAParser::string_t io_name
- MHAParser::bool_t exit_on_stop
- MHAParser::int_t fw_sleep
- MHAParser::string_t fw_until
- MHAParser::kw_t fw_cmd
- MHAParser::string_mon_t parserstate
- MHAParser::string_t errorlog
- MHAParser::string_t fatallog
- MHAParser::vstring_t plugins
- MHAParser::vstring_t plugin_paths
- MHAParser::bool_t dump_mha
- MHAParser::string_t inst_name
 - A variable for naming MHA instances.*
- MHAKernel::algo_comm_class_t ac
- PluginLoader::mhapluginloader_t * proc_lib
- io_lib_t * io_lib

- **mhaconfig_t cfin**
- **mhaconfig_t cfout**
- **state_t state**
- **bool b_exit_request**
- **MHAParser::string_mon_t proc_error_string**
- **MHAEvents::patchbay_t< fw_t > patchbay**

Additional Inherited Members

5.85.1 Member Enumeration Documentation

5.85.1.1 enum fw_t::state_t [private]

Enumerator

fw_unprepared
fw_stopped
fw_starting
fw_running
fw_stopping
fw_exiting

5.85.2 Constructor & Destructor Documentation

5.85.2.1 fw_t::fw_t ()

5.85.2.2 fw_t::~~fw_t ()

5.85.3 Member Function Documentation

5.85.3.1 bool fw_t::exit_request () const [inline]

5.85.3.2 void fw_t::prepare (void) [private]

preparation for processing

5.85.3.3 void fw_t::start () [private]

start of processing

5.85.3.4 void fw_t::stop () [private]

stop/pause of processing

5.85.3.5 void fw_t::release () [private]

release of IO device

5.85.3.6 void fw_t::quit () [private]

controlled quit

5.85.3.7 static void fw_t::stopped (
void * *h*,
int *proc_err*,
int *io_err*) [inline], [static], [private]

5.85.3.8 static void fw_t::started (
void * *h*) [inline], [static], [private]

5.85.3.9 static int fw_t::process (
void * *h*,
mha_wave_t * *sIn*,
mha_wave_t ** *sOut*) [inline], [static], [private]

5.85.3.10 void fw_t::stopped (
int *proc_err*,
int *io_err*) [private]

5.85.3.11 void fw_t::started () [private]

5.85.3.12 int fw_t::process (
mha_wave_t * *sIn*,
mha_wave_t ** *sOut*) [private]

5.85.3.13 void fw_t::exec_fw_command () [private]

5.85.3.14 void fw_t::load_proc_lib () [private]

5.85.3.15 void fw_t::load_io_lib () [private]

5.85.3.16 void fw_t::fw_sleep_cmd () [private]

5.85.3.17 void fw_t::fw_until_cmd () [private]

5.85.3.18 void fw_t::get_input_signal_dimension () [private]

5.85.3.19 void fw_t::async_read () [inline], [private]

5.85.3.20 void fw_t::async_poll_msg () [private]

5.85.3.21 void fw_t::get_parserstate () [private]

5.85.4 Member Data Documentation

5.85.4.1 fw_vars_t fw_t::prepare_vars [private]

5.85.4.2 MHAParser::int_mon_t fw_t::nchannels_out [private]

5.85.4.3 MHAParser::string_t fw_t::proc_name [private]

5.85.4.4 MHAParser::string_t fw_t::io_name [private]

5.85.4.5 MHAParser::bool_t fw_t::exit_on_stop [private]

5.85.4.6 MHAParser::int_t fw_t::fw_sleep [private]

5.85.4.7 MHAParser::string_t fw_t::fw_until [private]

5.85.4.8 MHAParser::kw_t fw_t::fw_cmd [private]

5.85.4.9 MHAParser::string_mon_t fw_t::parserstate [private]

5.85.4.10 MHAParser::string_t fw_t::errorlog [private]

5.85.4.11 MHAParser::string_t fw_t::fatallog [private]

5.85.4.12 MHAParser::vstring_t fw_t::plugins [private]

5.85.4.13 MHAParser::vstring_t fw_t::plugin_paths [private]

5.85.4.14 MHAParser::bool_t fw_t::dump_mha [private]

5.85.4.15 MHAParser::string_t fw_t::inst_name [private]

A variable for naming MHA instances.

- 5.85.4.16 **MHAKernel::algo_comm_class_t fw_t::ac** [private]
- 5.85.4.17 **PluginLoader::mhapluginloader_t* fw_t::proc_lib** [private]
- 5.85.4.18 **io_lib_t* fw_t::io_lib** [private]
- 5.85.4.19 **mhaconfig_t fw_t::cfin** [private]
- 5.85.4.20 **mhaconfig_t fw_t::cfout** [private]
- 5.85.4.21 **state_t fw_t::state** [private]
- 5.85.4.22 **bool fw_t::b_exit_request** [private]
- 5.85.4.23 **int fw_t::proc_error** [protected]
- 5.85.4.24 **int fw_t::io_error** [protected]
- 5.85.4.25 **MHAParser::string_mon_t fw_t::proc_error_string** [private]
- 5.85.4.26 **MHAEvents::patchbay_t<fw_t> fw_t::patchbay** [private]

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

- **mhafw_lib.h**
- **mhafw_lib.cpp**

5.86 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**

5.86.1 Constructor & Destructor Documentation

5.86.1.1 `fw_vars_t::fw_vars_t (`
 `MHAParser::parser_t & p)`

5.86.2 Member Function Documentation

5.86.2.1 `void fw_vars_t::lock_srate_fragsize ()`

5.86.2.2 `void fw_vars_t::lock_channels ()`

5.86.2.3 `void fw_vars_t::unlock_srate_fragsize ()`

5.86.2.4 `void fw_vars_t::unlock_channels ()`

5.86.3 Member Data Documentation

5.86.3.1 `MHAParser::int_t fw_vars_t::pinchannels`

5.86.3.2 `MHAParser::int_t fw_vars_t::pfragmentsize`

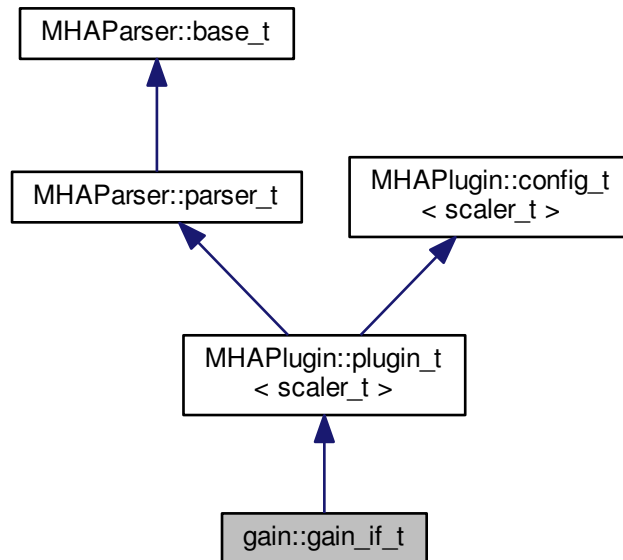
5.86.3.3 `MHAParser::float_t fw_vars_t::psrate`

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

- `mhafw_lib.h`
- `mhafw_lib.cpp`

5.87 gain::gain_if_t Class Reference

Inheritance diagram for gain::gain_if_t:



Public Member Functions

- **gain_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Member Functions

- void **update_gain** ()
- void **update_bbgain** ()
- void **update_minmax** ()

Private Attributes

- **MHAEvents::patchbay_t** < **gain_if_t** > **patchbay**
- **MHAParser::vfloat_t** **gains**
- **MHAParser::float_t** **bbgain**
- **MHAParser::float_t** **vmin**
- **MHAParser::float_t** **vmax**

Additional Inherited Members

5.87.1 Constructor & Destructor Documentation

5.87.1.1 `gain::gain_if_t::gain_if_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

5.87.2 Member Function Documentation

5.87.2.1 `mha_wave_t * gain::gain_if_t::process (`
 `mha_wave_t * s)`

5.87.2.2 `mha_spec_t * gain::gain_if_t::process (`
 `mha_spec_t * s)`

5.87.2.3 `void gain::gain_if_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPLugin::plugin_t< scaler_t >** (p. 661).

5.87.2.4 `void gain::gain_if_t::release (`
 `void) [virtual]`

Reimplemented from **MHAPLugin::plugin_t< scaler_t >** (p. 661).

5.87.2.5 `void gain::gain_if_t::update_gain () [private]`

5.87.2.6 `void gain::gain_if_t::update_bbgain () [private]`

5.87.2.7 `void gain::gain_if_t::update_minmax () [private]`

5.87.3 Member Data Documentation

5.87.3.1 **MHAEvents::patchbay_t<gain_if_t> gain::gain_if_t::patchbay** [private]

5.87.3.2 **MHAParser::vfloat_t gain::gain_if_t::gains** [private]

5.87.3.3 **MHAParser::float_t gain::gain_if_t::bbgain** [private]

5.87.3.4 **MHAParser::float_t gain::gain_if_t::vmin** [private]

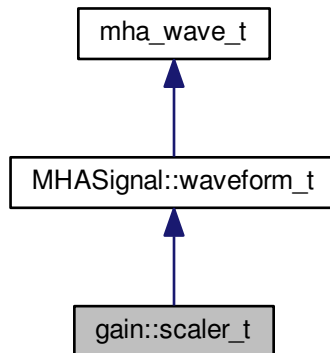
5.87.3.5 **MHAParser::float_t gain::gain_if_t::vmax** [private]

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

- **gain.cpp**

5.88 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

5.88.1 Constructor & Destructor Documentation

5.88.1.1 gain::scaler_t::scaler_t (
 const unsigned int & *channels*,
 const **MHAParser::vfloat_t** & *gains*)

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

- **gain.cpp**

5.89 hanning_ramps_t Class Reference

Public Member Functions

- **hanning_ramps_t** (unsigned int, unsigned int)
- **~hanning_ramps_t** ()
- void **operator()** (**MHASignal::waveform_t** &)

Private Attributes

- unsigned int **len_a**
- unsigned int **len_b**
- **mha_real_t * ramp_a**
- **mha_real_t * ramp_b**

5.89.1 Constructor & Destructor Documentation

5.89.1.1 `hanning_ramps_t::hanning_ramps_t (`
 unsigned int *la*,
 unsigned int *lb*)

5.89.1.2 `hanning_ramps_t::~~hanning_ramps_t (`
 void)

5.89.2 Member Function Documentation

5.89.2.1 `void hanning_ramps_t::operator() (`
 MHASignal::waveform_t & *b*)

5.89.3 Member Data Documentation

5.89.3.1 `unsigned int hanning_ramps_t::len_a` [private]

5.89.3.2 `unsigned int hanning_ramps_t::len_b` [private]

5.89.3.3 `mha_real_t* hanning_ramps_t::ramp_a` [private]

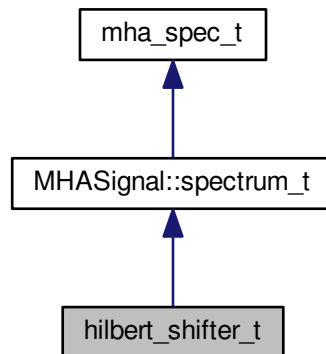
5.89.3.4 `mha_real_t* hanning_ramps_t::ramp_b` [private]

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

- **spec2wave.cpp**

5.90 hilbert_shifter_t Class Reference

Inheritance diagram for 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**
- **MHASignal::spectrum_t** **analytic**
- **MHASignal::waveform_t** **shifted**
- **MHASignal::spectrum_t** **mixw_shift**
- **MHASignal::spectrum_t** **mixw_ref**
- fftw_plan **plan_spec2analytic**
- **mha_fft_t** **mhafft**
- **MHASignal::waveform_t** **phi**
- **MHASignal::waveform_t** **dphi**
- unsigned int **kmin**
- unsigned int **kmax**
- unsigned int **frameshift**

Additional Inherited Members

5.90.1 Constructor & Destructor Documentation

5.90.1.1 `hilbert_shifter_t::hilbert_shifter_t (`
 unsigned int *fftl*en,
 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*)

5.90.1.2 `hilbert_shifter_t::~~hilbert_shifter_t ()`

5.90.2 Member Function Documentation

5.90.2.1 `void hilbert_shifter_t::process (`
 mha_spec_t * *s*)

5.90.3 Member Data Documentation

5.90.3.1 `MHASignal::spectrum_t hilbert_shifter_t::fullspec` [private]

5.90.3.2 `MHASignal::spectrum_t hilbert_shifter_t::analytic` [private]

5.90.3.3 `MHASignal::waveform_t hilbert_shifter_t::shifted` [private]

5.90.3.4 `MHASignal::spectrum_t hilbert_shifter_t::mixw_shift` [private]

5.90.3.5 `MHASignal::spectrum_t hilbert_shifter_t::mixw_ref` [private]

5.90.3.6 `fftw_plan hilbert_shifter_t::plan_spec2analytic` [private]

5.90.3.7 `mha_fft_t hilbert_shifter_t::mhafft` [private]

5.90.3.8 `MHASignal::waveform_t hilbert_shifter_t::phi` [private]

5.90.3.9 `MHASignal::waveform_t hilbert_shifter_t::dphi` [private]

5.90.3.10 `unsigned int hilbert_shifter_t::kmin` [private]

5.90.3.11 `unsigned int hilbert_shifter_t::kmax` [private]

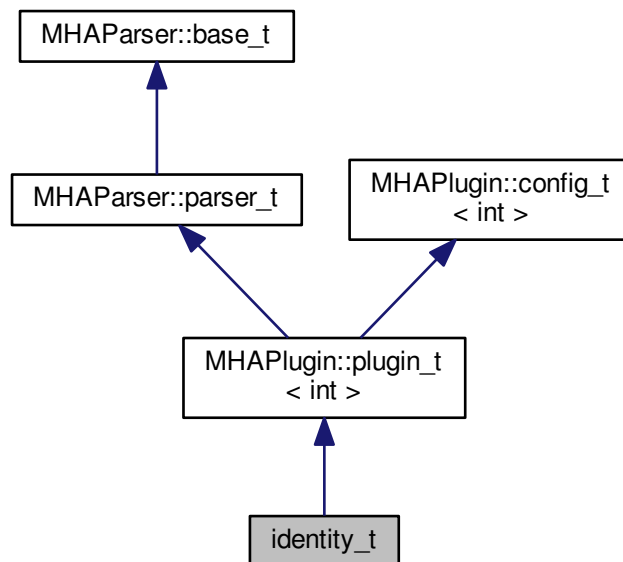
5.90.3.12 `unsigned int hilbert_shifter_t::frameshift` [private]

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

- `fshift_hilbert.cpp`

5.91 identity_t Class Reference

Inheritance diagram for identity_t:



Public Member Functions

- **identity_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** * **process** (mha_wave_t *)
- **mha_spec_t** * **process** (mha_spec_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Additional Inherited Members

5.91.1 Constructor & Destructor Documentation

5.91.1.1 identity_t::identity_t (
 const **algo_comm_t** & *i*ac,
 const std::string & ,
 const std::string &)

5.91.2 Member Function Documentation

5.91.2.1 `mha_wave_t * identity_t::process (`
`mha_wave_t * s)`

5.91.2.2 `mha_spec_t * identity_t::process (`
`mha_spec_t * s)`

5.91.2.3 `void identity_t::prepare (`
`mhaconfig_t & tf) [virtual]`

Implements **MHAPLugin::plugin_t< int >** (p. 661).

5.91.2.4 `void identity_t::release (`
`void) [virtual]`

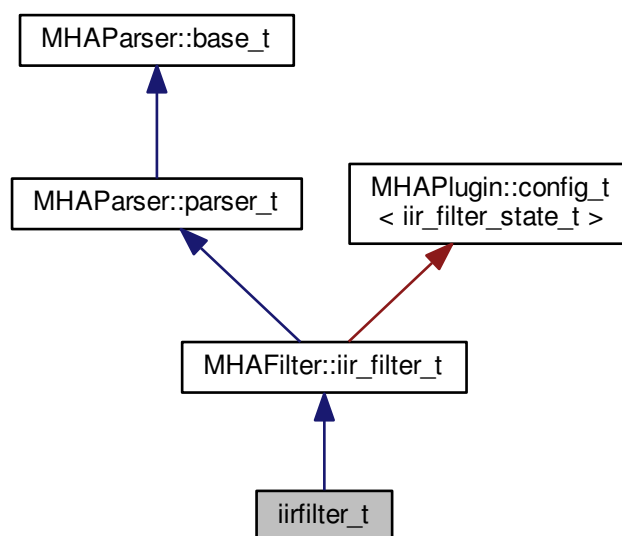
Reimplemented from **MHAPLugin::plugin_t< int >** (p. 661).

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

- **identity.cpp**

5.92 iirfilter_t Class Reference

Inheritance diagram for iirfilter_t:



Public Member Functions

- **iirfilter_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare_** (**mhaconfig_t** &)
- void **release_** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Additional Inherited Members

5.92.1 Constructor & Destructor Documentation

5.92.1.1 **iirfilter_t::iirfilter_t** (
 const **algo_comm_t** & *ac*,
 const std::string & *th*,
 const std::string & *al*)

5.92.2 Member Function Documentation

5.92.2.1 void **iirfilter_t::prepare_** (
 mhaconfig_t & *tf*)

5.92.2.2 void **iirfilter_t::release_** () [inline]

5.92.2.3 **mha_wave_t** * **iirfilter_t::process** (
 mha_wave_t * *s*)

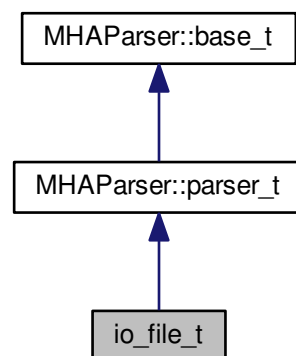
The documentation for this class was generated from the following file:

- **iirfilter.cpp**

5.93 io_file_t Class Reference

File IO.

Inheritance diagram for **io_file_t**:



Public Member Functions

- **io_file_t** (int **fragsize**, float **samplerate**, **IOProcessEvent_t** **proc_event**, void ***proc_**↵
handle, **IOStartedEvent_t** **start_event**, void ***start_handle**, **IOStoppedEvent_t** **stop**↵
_event, void ***stop_handle**)
- **~io_file_t** ()
- void **prepare** (int, int)
Allocate buffers, activate FILE client and install internal ports.
- void **start** ()
- void **stop** ()
- void **release** ()
Remove FILE client and deallocate internal ports and buffers.

Private Member Functions

- void **stopped** (int, int)
- void **setlock** (bool locked)
lock or unlock all parser variables.

Private Attributes

- int **fragsize**
- float **samplerate**
- int **nchannels_in**
- int **nchannels_file_in**
- int **nchannels_out**
- **IOProcessEvent_t** **proc_event**
- void * **proc_handle**
- **IOStartedEvent_t** **start_event**
- void * **start_handle**
- **IOStoppedEvent_t** **stop_event**
- void * **stop_handle**
- **MHAParser::string_t** **filename_input**
- **MHAParser::string_t** **filename_output**
- **MHAParser::kw_t** **output_sample_format**
- **MHAParser::int_t** **startsample**
- **MHAParser::int_t** **length**
- **MHAParser::bool_t** **strict_channel_match**
- **MHAParser::bool_t** **strict_srate_match**
- **MHASignal::waveform_t** * **s_in**
- **MHASignal::waveform_t** * **s_file_in**
- **mha_wave_t** * **s_out**
- bool **b_prepared**
- **SNDFILE** * **sf_in**
- **SNDFILE** * **sf_out**
- **SF_INFO** **sfinf_in**
- **SF_INFO** **sfinf_out**
- **sf_count_t** **total_read**

Additional Inherited Members

5.93.1 Detailed Description

File IO.

5.93.2 Constructor & Destructor Documentation

5.93.2.1 io_file_t::io_file_t (
 int *fragsize*,
 float *samplerate*,
 IOProcessEvent_t *proc_event*,
 void * *proc_handle*,
 IOStartedEvent_t *start_event*,
 void * *start_handle*,
 IOStoppedEvent_t *stop_event*,
 void * *stop_handle*)

5.93.2.2 io_file_t::~~io_file_t ()

5.93.3 Member Function Documentation

5.93.3.1 void io_file_t::prepare (
 int *nch_in*,
 int *nch_out*)

Allocate buffers, activate FILE client and install internal ports.

5.93.3.2 void io_file_t::start ()

5.93.3.3 void io_file_t::stop ()

5.93.3.4 void io_file_t::release (
 void)

Remove FILE client and deallocate internal ports and buffers.

5.93.3.5 void io_file_t::stopped (
 int *proc_err*,
 int *io_err*) [private]

5.93.3.6 void io_file_t::setlock (
 bool *locked*) [private]

lock or unlock all parser variables.

Used in prepare/release.

Parameters

| | |
|---------------|--|
| <i>locked</i> | When true, locks. When false, unlocks. |
|---------------|--|

5.93.4 Member Data Documentation

5.93.4.1 `int io_file_t::fragsize` [private]

5.93.4.2 `float io_file_t::samplerate` [private]

5.93.4.3 `int io_file_t::nchannels_in` [private]

5.93.4.4 `int io_file_t::nchannels_file_in` [private]

5.93.4.5 `int io_file_t::nchannels_out` [private]

5.93.4.6 `IOProcessEvent_t io_file_t::proc_event` [private]

5.93.4.7 `void* io_file_t::proc_handle` [private]

5.93.4.8 `IOStartedEvent_t io_file_t::start_event` [private]

5.93.4.9 `void* io_file_t::start_handle` [private]

5.93.4.10 `IOStoppedEvent_t io_file_t::stop_event` [private]

5.93.4.11 `void* io_file_t::stop_handle` [private]

5.93.4.12 `MHAParser::string_t io_file_t::filename_input` [private]

5.93.4.13 `MHAParser::string_t io_file_t::filename_output` [private]

5.93.4.14 `MHAParser::kw_t io_file_t::output_sample_format` [private]

5.93.4.15 `MHAParser::int_t io_file_t::startsample` [private]

5.93.4.16 `MHAParser::int_t io_file_t::length` [private]

5.93.4.17 `MHAParser::bool_t io_file_t::strict_channel_match` [private]

5.93.4.18 `MHAParser::bool_t io_file_t::strict_srate_match` [private]

5.93.4.19 MHASignal::waveform_t* io_file_t::s_in [private]

5.93.4.20 MHASignal::waveform_t* io_file_t::s_file_in [private]

5.93.4.21 mha_wave_t* io_file_t::s_out [private]

5.93.4.22 bool io_file_t::b_prepared [private]

5.93.4.23 SNDFILE* io_file_t::sf_in [private]

5.93.4.24 SNDFILE* io_file_t::sf_out [private]

5.93.4.25 SF_INFO io_file_t::sfinf_in [private]

5.93.4.26 SF_INFO io_file_t::sfinf_out [private]

5.93.4.27 sf_count_t io_file_t::total_read [private]

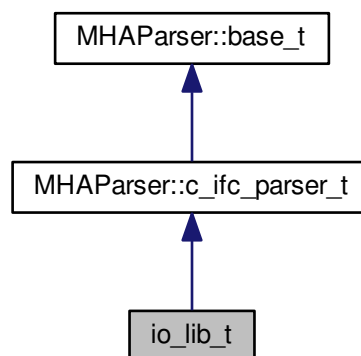
The documentation for this class was generated from the following file:

- MHAIOFile.cpp

5.94 io_lib_t Class Reference

Class for loading MHA sound IO module.

Inheritance diagram for io_lib_t:



Public Member Functions

- **io_lib_t** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, std::string libname)
load and initialize MHA sound io module.
- **~io_lib_t** ()
Deinitialize and unload this MHA sound io module.
- void **prepare** (unsigned int inch, unsigned int outch)
Prepare the sound io module.
- void **start** ()
Tell the sound io module to start sound processing.
- void **stop** ()
- void **release** ()
- std::string **lib_str_error** (int err)

Protected Member Functions

- void **test_error** ()

Protected Attributes

- int **lib_err**
- **dynamiclib_t** **lib_handle**
- void * **lib_data**
- **IOInit_t** **IOInit_cb**
- **IOPrepare_t** **IOPrepare_cb**
- **IOStart_t** **IOStart_cb**
- **IOStop_t** **IOStop_cb**
- **IORelease_t** **IORelease_cb**
- **IOSetVar_t** **IOSetVar_cb**
- **IOStrError_t** **IOStrError_cb**
- **IODestroy_t** **IODestroy_cb**

Additional Inherited Members

5.94.1 Detailed Description

Class for loading MHA sound IO module.

5.94.2 Constructor & Destructor Documentation

5.94.2.1 `io_lib_t::io_lib_t (`
 `int fragsize,`
 `float samplerate,`
 `IOProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle,`
 `std::string libname)`

load and initialize MHA sound io module.

5.94.2.2 `io_lib_t::~~io_lib_t ()`

Deinitialize and unload this MHA sound io module.

5.94.3 Member Function Documentation

5.94.3.1 `void io_lib_t::prepare (`
 `unsigned int inch,`
 `unsigned int outch)`

Prepare the sound io module.

After preparation, the sound io module may start the sound processing at any time (external trigger). When the sound processing is started, the sound io module will call the `start_event` callback.

Parameters

| | |
|--------------|---------------------------|
| <i>inch</i> | number of input channels |
| <i>outch</i> | number of output channels |

5.94.3.2 `void io_lib_t::start ()`

Tell the sound io module to start sound processing.

Some io modules need this, for others that wait for external events this method might do nothing.

5.94.3.3 void io_lib_t::stop ()

5.94.3.4 void io_lib_t::release ()

5.94.3.5 std::string io_lib_t::lib_str_error (
int err)

5.94.3.6 void io_lib_t::test_error () [protected]

5.94.4 Member Data Documentation

5.94.4.1 int io_lib_t::lib_err [protected]

5.94.4.2 dynamiclib_t io_lib_t::lib_handle [protected]

5.94.4.3 void* io_lib_t::lib_data [protected]

5.94.4.4 IOInit_t io_lib_t::IOInit_cb [protected]

5.94.4.5 IOPrepare_t io_lib_t::IOPrepare_cb [protected]

5.94.4.6 IOStart_t io_lib_t::IOStart_cb [protected]

5.94.4.7 IOSTop_t io_lib_t::IOStop_cb [protected]

5.94.4.8 IORelease_t io_lib_t::IORelease_cb [protected]

5.94.4.9 IOSetVar_t io_lib_t::IOSetVar_cb [protected]

5.94.4.10 IOStrError_t io_lib_t::IOStrError_cb [protected]

5.94.4.11 IODestroy_t io_lib_t::IODestroy_cb [protected]

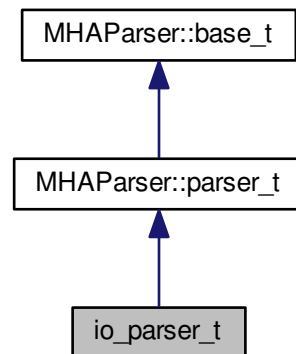
The documentation for this class was generated from the following files:

- mhafw_lib.h
- mhafw_lib.cpp

5.95 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

5.95.1 Detailed Description

Main class for Parser IO.

5.95.2 Constructor & Destructor Documentation

5.95.2.1 **io_parser_t::io_parser_t** (
 unsigned int *fragsize*,
 IOProcessEvent_t *proc_event*,
 void * *proc_handle*,
 IOStartedEvent_t *start_event*,
 void * *start_handle*,
 IOStoppedEvent_t *stop_event*,
 void * *stop_handle*)

5.95.2.2 **io_parser_t::~~io_parser_t** ()

5.95.3 Member Function Documentation

5.95.3.1 void **io_parser_t::prepare** (
 int *nch_in*,
 int *nch_out*)

Allocate buffers, activate JACK client and install internal ports.

5.95.3.2 void io_parser_t::start ()

5.95.3.3 void io_parser_t::stop ()

5.95.3.4 void io_parser_t::release (
 void)

Remove JACK client and deallocate internal ports and buffers.

5.95.3.5 void io_parser_t::stopped (
 int *proc_err*,
 int *io_err*) [private]

5.95.3.6 void io_parser_t::started () [private]

5.95.3.7 void io_parser_t::process_frame () [private]

5.95.4 Member Data Documentation

5.95.4.1 unsigned int io_parser_t::fragsize [private]

5.95.4.2 unsigned int io_parser_t::nchannels_in [private]

5.95.4.3 unsigned int io_parser_t::nchannels_out [private]

5.95.4.4 IOProcessEvent_t io_parser_t::proc_event [private]

5.95.4.5 void* io_parser_t::proc_handle [private]

5.95.4.6 IOStartedEvent_t io_parser_t::start_event [private]

5.95.4.7 void* io_parser_t::start_handle [private]

5.95.4.8 IOStoppedEvent_t io_parser_t::stop_event [private]

5.95.4.9 void* io_parser_t::stop_handle [private]

5.95.4.10 MHAParser::mfloat_t io_parser_t::input [private]

5.95.4.11 MHAParser::mfloat_mon_t io_parser_t::output [private]

5.95.4.12 MHASignal::waveform_t* io_parser_t::s_in [private]

5.95.4.13 mha_wave_t* io_parser_t::s_out [private]

5.95.4.14 bool io_parser_t::b_fw_started [private]

5.95.4.15 bool io_parser_t::b_stopped [private]

5.95.4.16 bool io_parser_t::b_prepared [private]

5.95.4.17 bool io_parser_t::b_starting [private]

5.95.4.18 MHAEvents::patchbay_t<io_parser_t> io_parser_t::patchbay [private]

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

- **MHAIOParser.cpp**

5.96 io_tcp_fwcb_t Class Reference

TCP sound-io library's interface to the framework callbacks.

Public Member Functions

- **io_tcp_fwcb_t** (IOProcessEvent_t proc_event, void *proc_handle, IOStartedEvent_t start_event, void *start_handle, IOStoppedEvent_t stop_event, void *stop_handle)
Constructor stores framework handles and initializes error numbers to 0.
- virtual ~**io_tcp_fwcb_t** ()
Do-nothing destructor.
- virtual void **start** ()
Call the framework's start callback.
- virtual int **process** (mha_wave_t *sIn, mha_wave_t *&sOut)
Call the frameworks processing callback.
- virtual void **set_errnos** (int proc_err, int io_err)
Save error numbers to use during.
- virtual void **stop** ()
Call the frameworks stop callback.

Private Attributes

- **IOProcessEvent_t proc_event**
Pointer to signal processing callback function.
- **IOStartedEvent_t start_event**
Pointer to start notification callback function.
- **IOStoppedEvent_t stop_event**
Pointer to stop notification callback function.
- void * **proc_handle**
Handles belonging to framework.
- void * **start_handle**
- void * **stop_handle**
- int **proc_err**
Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.
- int **io_err**

5.96.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

5.96.2 Constructor & Destructor Documentation

5.96.2.1 `io_tcp_fwcb_t::io_tcp_fwcb_t (`
 IOProcessEvent_t *proc_event*,
 void **proc_handle*,
 IOStartedEvent_t *start_event*,
 void **start_handle*,
 IOStoppedEvent_t *stop_event*,
 void **stop_handle*)

Constructor stores framework handles and initializes error numbers to 0.

5.96.2.2 `virtual io_tcp_fwcb_t::~io_tcp_fwcb_t ()` [inline],[virtual]

Do-nothing destructor.

5.96.3 Member Function Documentation

5.96.3.1 `void io_tcp_fwcb_t::start ()` [virtual]

Call the framework's start callback.

5.96.3.2 `int io_tcp_fwcb_t::process (`
 mha_wave_t **sIn*,
 mha_wave_t *&*sOut*) [virtual]

Call the frameworks processing callback.

Parameters

| | |
|-------------|---|
| <i>sIn</i> | The input sound data just received from TCP. |
| <i>sOut</i> | 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.

5.96.3.3 `void io_tcp_fwcb_t::set_errnos (`
 int *proc_err*,
 int *io_err*) [virtual]

Save error numbers to use during.

See also

stop (p. 322)

Parameters

| | |
|-----------------|---------------------------|
| <i>proc_err</i> | The error number from the |
|-----------------|---------------------------|

See also

process (p. 321) callback.

Parameters

| | |
|---------------|--|
| <i>io_err</i> | The error number from the io library itself. |
|---------------|--|

5.96.3.4 void io_tcp_fwcb_t::stop () [virtual]

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

set_errnos (p. 321).

5.96.4 Member Data Documentation

5.96.4.1 IOProcessEvent_t io_tcp_fwcb_t::proc_event [private]

Pointer to signal processing callback function.

5.96.4.2 IOStartedEvent_t io_tcp_fwcb_t::start_event [private]

Pointer to start notification callback function.

Called when a new TCP connection is established or the user issues the start command while there is a connection.

5.96.4.3 IOStoppedEvent_t io_tcp_fwcb_t::stop_event [private]

Pointer to stop notification callback function.

Called when the connection is closed.

5.96.4.4 void* io_tcp_fwcb_t::proc_handle [private]

Handles belonging to framework.

5.96.4.5 void * io_tcp_fwcb_t::start_handle [private]

5.96.4.6 void * io_tcp_fwcb_t::stop_handle [private]

5.96.4.7 int io_tcp_fwcb_t::proc_err [private]

Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.

MHAIOTCP is notified by the server when the connection has been taken down, and calls

See also

stop (p. 322) from that callback. Within **stop** (p. 322), these error numbers are read again and transmitted to the framework.

5.96.4.8 int io_tcp_fwcb_t::io_err [private]

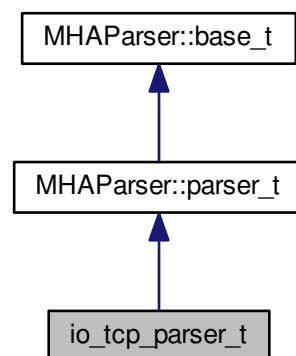
The documentation for this class was generated from the following file:

- **MHAIOTCP.cpp**

5.97 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 whether there currently exists an established sound data TCP connection or not.
- virtual void **set_connected** (bool connected)
Inform the parser about the existence 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 whether 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

5.97.1 Detailed Description

The parser interface of the IOTCP library.

5.97.2 Constructor & Destructor Documentation

5.97.2.1 io_tcp_parser_t::io_tcp_parser_t ()

Constructor initializes parser variables.

5.97.2.2 virtual io_tcp_parser_t::~~io_tcp_parser_t () [inline], [virtual]

Do-nothing destructor.

5.97.3 Member Function Documentation

5.97.3.1 virtual const std::string& io_tcp_parser_t::get_local_address () const [inline], [virtual]

Read parser variable `local_address`, this is the address of the network interface that should listen for incoming connections.

Returns

A string containing the address of the local interface as it was set by the user.

5.97.3.2 unsigned short io_tcp_parser_t::get_local_port () const [virtual]

Read parser variable `local_port`, this is the TCP port that should be used for incoming connections.

Returns

The local tcp port to listen on as it was chosen by the user. The port number is between `MIN_TCP_PORT` and `MAX_TCP_PORT`.

5.97.3.3 void io_tcp_parser_t::set_local_port (unsigned short *port*) [virtual]

Set parser variable `local_port`.

This is needed when it was set to 0 before: In this case, the OS chooses a free port for the TCP server socket, and the port that it chose has to be published to the user via the parser interface.

Parameters

| | |
|-------------|---|
| <i>port</i> | The TCP port number that is currently used. In the range [MIN_TCP_PORT, MAX_TCP_PORT], excluding 0. |
|-------------|---|

Precondition

get_local_port() (p. 325) currently returns 0.

5.97.3.4 `bool io_tcp_parser_t::get_server_port_open () const [virtual]`

Return the status of the server port as it is known to the parser.

Returns

false after initialization, or the value most recently set via

See also

set_server_port_open (p. 326).

5.97.3.5 `void io_tcp_parser_t::set_server_port_open (`
`bool open) [virtual]`

Inform the parser of the current status of the server socket.

Parameters

| | |
|-------------|---|
| <i>open</i> | Indicates whether the server socket has just been opened or closed. |
|-------------|---|

Precondition

open may only have the value true if **get_server_port_open()** (p. 326) currently returns false.

Postcondition**See also**

get_server_port_open (p. 326) returns the **value** (p. 52) of *open*.

5.97.3.6 bool io_tcp_parser_t::get_connected () const [virtual]

Return the parser's knowledge concerning whether 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. 327).

5.97.3.7 void io_tcp_parser_t::set_connected (bool *connected*) [virtual]

Inform the parser about the existence of a sound data connection.

Parameters

| | |
|------------------|---|
| <i>connected</i> | Indicates whether 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. 327).

Postcondition

See also

get_connected (p. 327) returns the **value** (p. 52) of open.

5.97.3.8 void io_tcp_parser_t::set_new_peer (unsigned short *port*, const std::string & *host*) [virtual]

Set parser monitor variables *peer_port* and *peer_address*, and calls **set_connected(true)**.

This method should be called when a new connection is established.

Parameters

| | |
|-------------|--|
| <i>port</i> | The TCP port number used by the peer. |
| <i>host</i> | The Internet host where the peer is located. |

Precondition**See also**

get_connected (p. 327) currently returns false.

Postcondition**See also**

get_connected (p. 327) returns true.

5.97.3.9 virtual void io_tcp_parser_t::debug (
const std::string & *message*) [inline],[virtual]

5.97.4 Member Data Documentation

5.97.4.1 **MHAParser::string_t io_tcp_parser_t::local_address** [private]

Lets the user set the local network interface to listen on.

5.97.4.2 **MHAParser::int_t io_tcp_parser_t::local_port** [private]

Lets the user choose the local tcp port to listen on.

5.97.4.3 **MHAParser::int_mon_t io_tcp_parser_t::server_port_open** [private]

Indicates wether the TCP server socket is currently open.

5.97.4.4 **MHAParser::int_mon_t io_tcp_parser_t::connected** [private]

Indicator if there currently is a sound data connection over TCP.

5.97.4.5 MHAParser::string_mon_t io_tcp_parser_t::peer_address [private]

Display the ip address of the currently connected sound data client.

5.97.4.6 MHAParser::int_mon_t io_tcp_parser_t::peer_port [private]

Display the tcp port used by the current sound data client.

5.97.4.7 MHAParser::string_t io_tcp_parser_t::debug_filename [private]

filename to write debugging info to (if non-empty)

5.97.4.8 FILE* io_tcp_parser_t::debug_file [private]

file handle to write debugging info to

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

- MHAIOTCP.cpp

5.98 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. 436) 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.

5.98.1 Detailed Description

Sound data handling of io tcp library.

5.98.2 Constructor & Destructor Documentation

5.98.2.1 **io_tcp_sound_t::io_tcp_sound_t** (
 int *fragsize*,
 float *samplerate*)

Initialize sound data handling.

Checks sound data type by calling

See also

check_sound_data_type (p. [331](#)).

Parameters

| | |
|-------------------|---|
| <i>fragsize</i> | Number of sound samples in each channel expected and returned from processing callback. |
| <i>samplerate</i> | Number of samples per second in each channel. |

5.98.2.2 `virtual io_tcp_sound_t::~io_tcp_sound_t() [inline], [virtual]`

Do-nothing destructor.

5.98.3 Member Function Documentation

5.98.3.1 `void io_tcp_sound_t::check_sound_data_type() [static], [private]`

Check if mha_real_t is a usable 32-bit floating point type.

Exceptions

| | |
|---|--|
| <i>MHA_Error</i> (p. 387) | if mha_real_t is not compatible to 32-bit float. |
|---|--|

5.98.3.2 `void io_tcp_sound_t::prepare (`
`int num_inchannels,`
`int num_outchannels) [virtual]`

Called during prepare, sets number of audio channels and allocates sound data storage.

Parameters

| | |
|------------------------|----------------------------------|
| <i>num_inchannels</i> | Number of input audio channels. |
| <i>num_outchannels</i> | Number of output audio channels. |

5.98.3.3 `void io_tcp_sound_t::release (`
`void) [virtual]`

Called during release.

Deletes sound data storage.

5.98.3.4 `int io_tcp_sound_t::chunkbytes_in() const [virtual]`

Number of bytes that constitute one input sound chunk.

Returns

Number of bytes to read from TCP connection before invoking signal processing.

5.98.3.5 `std::string io_tcp_sound_t::header() const [virtual]`

Create the tcp sound header lines.

5.98.3.6 `mha_wave_t * io_tcp_sound_t::ntoh (`
`const std::string & data) [virtual]`

Copy data received from tcp into **mha_wave_t** (p. 436) structure.

Doing network-to-host byte order swapping in the process.

Parameters

| | |
|-------------|-------------|
| <i>data</i> | One chunk (|
|-------------|-------------|

See also

chunkbytes_in (p. 331)) of sound data to process.

Returns

Pointer to the sound data storage.

5.98.3.7 `std::string io_tcp_sound_t::hton (`
`const mha_wave_t * s_out) [virtual]`

Copy sound data from the output sound structure to a string.

Doing host-to-network byte order swapping while at it.

Parameters

| | |
|--------------|---|
| <i>s_out</i> | Pointer to the storage of the sound to put out. |
|--------------|---|

Returns

The sound data in network byte order.

5.98.4 Member Data Documentation

5.98.4.1 `int io_tcp_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

5.98.4.2 `float io_tcp_sound_t::samplerate [private]`

Sampling rate.

Number of samples per second in each channel.

5.98.4.3 int io_tcp_sound_t::num_inchannels [private]

Number of input channels.

Number of channels expected from and returned by signal processing callback.

5.98.4.4 int io_tcp_sound_t::num_outchannels [private]

5.98.4.5 MHASignal::waveform_t* io_tcp_sound_t::s_in [private]

Storage for input signal.

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

- **MHAIOTCP.cpp**

5.99 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]

5.99.1 Detailed Description

This union helps in conversion of floats from host byte order to network byte order and back again.

5.99.2 Member Data Documentation

5.99.2.1 float io_tcp_sound_t::float_union::f

5.99.2.2 unsigned int io_tcp_sound_t::float_union::i

5.99.2.3 char io_tcp_sound_t::float_union::c[4]

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

- **MHAIOTCP.cpp**

5.100 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

5.100.1 Detailed Description

The tcp sound io library.

5.100.2 Constructor & Destructor Documentation

5.100.2.1 `io_tcp_t::io_tcp_t (`
 `int fragsize,`
 `float samplerate,`
 `IOProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle)`

5.100.2.2 `virtual io_tcp_t::~io_tcp_t () [inline],[virtual]`

5.100.3 Member Function Documentation

5.100.3.1 `void io_tcp_t::prepare (`
 `int num_inchannels,`
 `int num_outchannels)`

Allocate server socket and start thread waiting for sound data exchange.

prepare opens the tcp server socket and starts the io thread that listens for audio data on the tcp socket after doing some sanity checks

5.100.3.2 `void io_tcp_t::start ()`

Call frameworks start callback if there is a sound data connection at the moment.

5.100.3.3 `void io_tcp_t::stop ()`

Close the current connection if there is one.

stop IO thread

5.100.3.4 `void io_tcp_t::release (`
 `void)`

Close the current connection and close the server socket.

Stop IO thread and close server socket.

5.100.3.5 `void io_tcp_t::accept_loop () [virtual]`

IO thread executes this method.

5.100.3.6 `void io_tcp_t::connection_loop (`
 `MHA_TCP::Connection * c) [virtual]`

IO thread executes this method for each connection.

Parameters

| | |
|----------|---|
| c | pointer to connection. connection_loop deletes connection before exiting. |
|----------|---|

```
5.100.3.7 virtual void io_tcp_t::parse (
    const char * cmd,
    char * retval,
    unsigned int len ) [inline],[virtual]
```

Parser interface.

5.100.4 Member Data Documentation

5.100.4.1 `io_tcp_parser_t io_tcp_t::parser` [private]

5.100.4.2 `io_tcp_sound_t io_tcp_t::sound` [private]

5.100.4.3 `io_tcp_fwcb_t io_tcp_t::fwcb` [private]

5.100.4.4 `MHA_TCP::Server* io_tcp_t::server` [private]

5.100.4.5 `MHA_TCP::Thread* io_tcp_t::thread` [private]

5.100.4.6 `MHA_TCP::Async_Notify io_tcp_t::notify_start` [private]

5.100.4.7 `MHA_TCP::Async_Notify io_tcp_t::notify_stop` [private]

5.100.4.8 `MHA_TCP::Async_Notify io_tcp_t::notify_release` [private]

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

- **MHAIOTCP.cpp**

5.101 latex_doc_t Class Reference

Public Member Functions

- **latex_doc_t** (const std::string &**plugname**, const std::string &**plugin_macro**)
- std::string **get_latex_doc** ()
- std::string **get_main_category** ()
- std::vector< std::string > **get_categories** ()

Private Member Functions

- std::string **strdom** (mha_domain_t d)
- std::string **get_ac** (MHAKernel::algo_comm_class_t &ac, std::string txt)
- std::string **parsername** (std::string s)
- std::string **get_parser_var** (MHAParser::base_t *p, std::string name)
- std::string **get_parser_tab** (MHAParser::base_t *p, std::string prefix)

Private Attributes

- std::string **plugname**
- std::string **latex_plugname**
- MHAKernel::algo_comm_class_t **ac**
- PluginLoader::mhapluginloader_t **loader**
- std::string **plugin_macro**

5.101.1 Constructor & Destructor Documentation

5.101.1.1 latex_doc_t::latex_doc_t (
 const std::string & *plugname*,
 const std::string & *plugin_macro*)

5.101.2 Member Function Documentation

5.101.2.1 std::string latex_doc_t::get_latex_doc ()

5.101.2.2 std::string latex_doc_t::get_main_category ()

5.101.2.3 std::vector< std::string > latex_doc_t::get_categories ()

5.101.2.4 std::string latex_doc_t::strdom (
 mha_domain_t d) [private]

5.101.2.5 std::string latex_doc_t::get_ac (
 MHAKernel::algo_comm_class_t & ac,
 std::string txt) [private]

5.101.2.6 std::string latex_doc_t::parsername (
 std::string s) [private]

5.101.2.7 std::string latex_doc_t::get_parser_var (
 MHAParser::base_t * p,
 std::string name) [private]

5.101.2.8 `std::string latex_doc_t::get_parser_tab (`
 `MHAParser::base_t * p,`
 `std::string prefix) [private]`

5.101.3 Member Data Documentation

5.101.3.1 `std::string latex_doc_t::plugname [private]`

5.101.3.2 `std::string latex_doc_t::latex_plugname [private]`

5.101.3.3 `MHAKernel::algo_comm_class_t latex_doc_t::ac [private]`

5.101.3.4 `PluginLoader::mhapluginloader_t latex_doc_t::loader [private]`

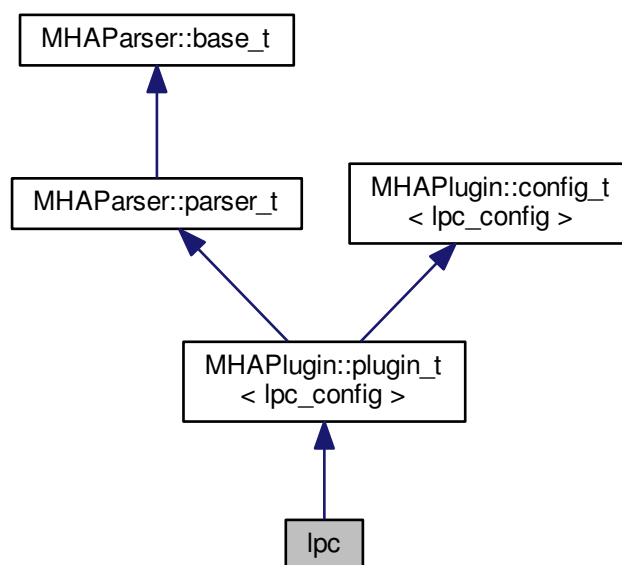
5.101.3.5 `std::string latex_doc_t::plugin_macro [private]`

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

- **generatemhaplugindoc.cpp**

5.102 lpc Class Reference

Inheritance diagram for lpc:



Public Member Functions

- **lpc** (**algo_comm_t** &**ac**, const std::string &**chain_name**, const std::string &**algo_name**)
Constructs our plugin.
- **~lpc** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- std::string **algo_name**
- MHAParser::int_t **lpc_order**
- MHAParser::int_t **lpc_buffer_size**
- MHAParser::bool_t **shift**
- MHAParser::int_t **comp_each_iter**
- MHAParser::bool_t **norm**
- MHAEvents::patchbay_t< lpc > **patchbay**

Additional Inherited Members

5.102.1 Constructor & Destructor Documentation

5.102.1.1 **lpc::lpc** (
 algo_comm_t &*ac*,
 const std::string &*chain_name*,
 const std::string &*algo_name*)

Constructs our plugin.

5.102.1.2 **lpc::~lpc** ()

5.102.2 Member Function Documentation

5.102.2.1 **mha_wave_t** * **lpc::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.102.2.2 void **lpc::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

5.102.2.3 `void lpc::release (`
`void) [inline],[virtual]`

Reimplemented from **MHAPLugin::plugin_t< lpc_config >** (p. [661](#)).

5.102.2.4 `void lpc::update_cfg () [private]`

5.102.3 Member Data Documentation

5.102.3.1 `std::string lpc::algo_name [private]`

5.102.3.2 `MHAParser::int_t lpc::lpc_order [private]`

5.102.3.3 `MHAParser::int_t lpc::lpc_buffer_size [private]`

5.102.3.4 `MHAParser::bool_t lpc::shift [private]`

5.102.3.5 `MHAParser::int_t lpc::comp_each_iter [private]`

5.102.3.6 `MHAParser::bool_t lpc::norm [private]`

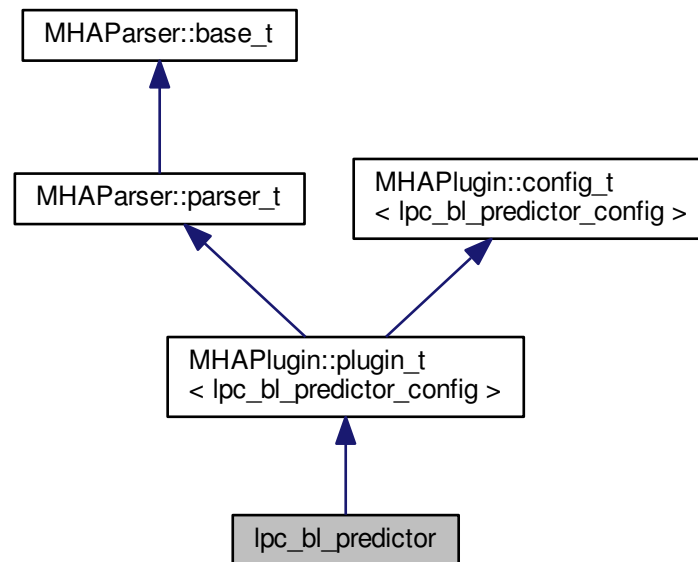
5.102.3.7 `MHAEvents::patchbay_t<lpc> lpc::patchbay [private]`

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

- **lpc.h**
- **lpc.cpp**

5.103 `lpc_bl_predictor` Class Reference

Inheritance diagram for `lpc_bl_predictor`:



Public Member Functions

- **`lpc_bl_predictor`** (**`algo_comm_t`** &**`ac`**, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **`~lpc_bl_predictor`** ()
- **`mha_wave_t * process`** (**`mha_wave_t *`**)
Checks for the most recent configuration and defers processing to it.
- void **`prepare`** (**`mhaconfig_t`** &)
Plugin preparation.
- void **`release`** (void)

Public Attributes

- **`MHAParser::int_t`** `lpc_order`
- **`MHAParser::string_t`** `name_kappa`
- **`MHAParser::string_t`** `name_lpc_f`
- **`MHAParser::string_t`** `name_lpc_b`
- **`MHAParser::string_t`** `name_f`
- **`MHAParser::string_t`** `name_b`

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t** < **lpc_bl_predictor** > **patchbay**

Additional Inherited Members

5.103.1 Constructor & Destructor Documentation

5.103.1.1 **lpc_bl_predictor::lpc_bl_predictor** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.103.1.2 **lpc_bl_predictor::~~lpc_bl_predictor** ()

5.103.2 Member Function Documentation

5.103.2.1 **mha_wave_t** * **lpc_bl_predictor::process** (
 mha_wave_t * *signal*)

Checks for the most recent configuration and defers processing to it.

5.103.2.2 void **lpc_bl_predictor::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. [661](#)).

5.103.2.3 `void lpc_bl_predictor::release (`
`void) [inline], [virtual]`

Reimplemented from **MHAPLugin::plugin_t< `lpc_bl_predictor_config` >** (p. [661](#)).

5.103.2.4 `void lpc_bl_predictor::update_cfg () [private]`

5.103.3 Member Data Documentation

5.103.3.1 **MHAParser::int_t** `lpc_bl_predictor::lpc_order`

5.103.3.2 **MHAParser::string_t** `lpc_bl_predictor::name_kappa`

5.103.3.3 **MHAParser::string_t** `lpc_bl_predictor::name_lpc_f`

5.103.3.4 **MHAParser::string_t** `lpc_bl_predictor::name_lpc_b`

5.103.3.5 **MHAParser::string_t** `lpc_bl_predictor::name_f`

5.103.3.6 **MHAParser::string_t** `lpc_bl_predictor::name_b`

5.103.3.7 **MHAEvents::patchbay_t<`lpc_bl_predictor`>** `lpc_bl_predictor::patchbay`
`[private]`

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

- `lpc_bl_predictor.h`
- `lpc_bl_predictor.cpp`

5.104 `lpc_bl_predictor_config` Class Reference

Public Member Functions

- **`lpc_bl_predictor_config`** (**`algo_comm_t`** &`iac`, const **`mhaconfig_t`** `in_cfg`, **`lpc_bl_↔`**
`predictor * _lpc`)
- **`~lpc_bl_predictor_config`** ()
- **`mha_wave_t * process`** (**`mha_wave_t *`**)

Private Attributes

- **algo_comm_t** **ac**
- **MHA_AC::waveform_t** **f_est**
- **MHA_AC::waveform_t** **b_est**
- **MHASignal::waveform_t** **forward**
- **MHASignal::waveform_t** **backward**
- **int** **lpc_order**
- **std::string** **name_km**
- **std::string** **name_f**
- **std::string** **name_b**
- **mha_wave_t** **km**
- **mha_wave_t** **s_f**
- **mha_wave_t** **s_b**

5.104.1 Constructor & Destructor Documentation

5.104.1.1 **lpc_bl_predictor_config::lpc_bl_predictor_config (**
 algo_comm_t & iac,
 const mhaconfig_t in_cfg,
 lpc_bl_predictor * _lpc)

5.104.1.2 **lpc_bl_predictor_config::~~lpc_bl_predictor_config ()**

5.104.2 Member Function Documentation

5.104.2.1 **mha_wave_t * lpc_bl_predictor_config::process (**
 mha_wave_t * wave)

5.104.3 Member Data Documentation

5.104.3.1 **algo_comm_t lpc_bl_predictor_config::ac** [private]

5.104.3.2 **MHA_AC::waveform_t lpc_bl_predictor_config::f_est** [private]

5.104.3.3 **MHA_AC::waveform_t lpc_bl_predictor_config::b_est** [private]

5.104.3.4 **MHASignal::waveform_t lpc_bl_predictor_config::forward** [private]

5.104.3.5 **MHASignal::waveform_t lpc_bl_predictor_config::backward** [private]

5.104.3.6 **int lpc_bl_predictor_config::lpc_order** [private]

5.104.3.7 **std::string lpc_bl_predictor_config::name_km** [private]

5.104.3.8 `std::string lpc_bl_predictor_config::name_f` [private]

5.104.3.9 `std::string lpc_bl_predictor_config::name_b` [private]

5.104.3.10 `mha_wave_t lpc_bl_predictor_config::km` [private]

5.104.3.11 `mha_wave_t lpc_bl_predictor_config::s_f` [private]

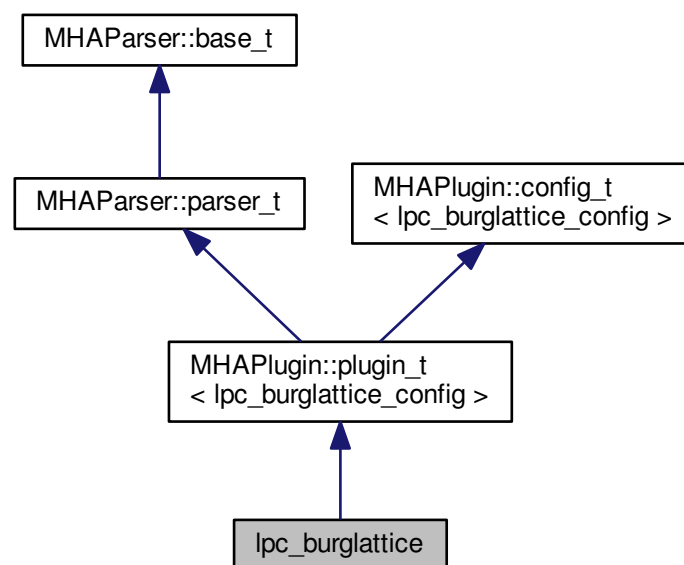
5.104.3.12 `mha_wave_t lpc_bl_predictor_config::s_b` [private]

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

- `lpc_bl_predictor.h`
- `lpc_bl_predictor.cpp`

5.105 lpc_burglattice Class Reference

Inheritance diagram for `lpc_burglattice`:



Public Member Functions

- **lpc_burglattice** (**algo_comm_t** &**ac**, const std::string &**chain_name**, const std::string &**algo_name**)
Constructs our plugin.
- **~lpc_burglattice** ()
- **mha_wave_t * process** (**mha_wave_t ***)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::int_t lpc_order**
- **MHAParser::string_t name_kappa**
- **MHAParser::string_t name_f**
- **MHAParser::string_t name_b**
- **MHAParser::float_t lambda**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< lpc_burglattice > patchbay**

Additional Inherited Members

5.105.1 Constructor & Destructor Documentation

5.105.1.1 **lpc_burglattice::lpc_burglattice** (
 algo_comm_t &*ac*,
 const std::string &*chain_name*,
 const std::string &*algo_name*)

Constructs our plugin.

5.105.1.2 `lpc_burglattice::~lpc_burglattice ()`

5.105.2 Member Function Documentation

5.105.2.1 `mha_wave_t * lpc_burglattice::process (mha_wave_t * signal)`

Checks for the most recent configuration and defers processing to it.

5.105.2.2 `void lpc_burglattice::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate). |
|--------------------|---|

Implements **MHAParser::plugin_t< lpc_burglattice_config >** (p. 661).

5.105.2.3 `void lpc_burglattice::release (void) [inline], [virtual]`

Reimplemented from **MHAParser::plugin_t< lpc_burglattice_config >** (p. 661).

5.105.2.4 `void lpc_burglattice::update_cfg () [private]`

5.105.3 Member Data Documentation

5.105.3.1 `MHAParser::int_t lpc_burglattice::lpc_order`5.105.3.2 `MHAParser::string_t lpc_burglattice::name_kappa`5.105.3.3 `MHAParser::string_t lpc_burglattice::name_f`5.105.3.4 `MHAParser::string_t lpc_burglattice::name_b`5.105.3.5 `MHAParser::float_t lpc_burglattice::lambda`5.105.3.6 `MHAEvents::patchbay_t<lpc_burglattice> lpc_burglattice::patchbay [private]`

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

- `lpc_burg-lattice.h`
- `lpc_burg-lattice.cpp`

5.106 `lpc_burglattice_config` Class Reference

Public Member Functions

- `lpc_burglattice_config` (`algo_comm_t` &*iac*, `const mhaconfig_t` *in_cfg*, `lpc_burglattice *`*lpc*)
- `~lpc_burglattice_config` ()
- `mha_wave_t * process` (`mha_wave_t *`)

Private Attributes

- `algo_comm_t` *ac*
- `MHASignal::waveform_t` *forward*
- `MHASignal::waveform_t` *backward*
- `MHASignal::waveform_t` *kappa*
- `MHA_AC::waveform_t` *kappa_block*
- `MHASignal::waveform_t` *dm*
- `MHASignal::waveform_t` *nm*
- `mha_real_t` *lambda*
- `int` *lpc_order*
- `std::string` *name_f*
- `std::string` *name_b*
- `mha_wave_t` *s_f*
- `mha_wave_t` *s_b*

5.106.1 Constructor & Destructor Documentation

5.106.1.1 `lpc_burglattice_config::lpc_burglattice_config` (
 `algo_comm_t` &*iac*,
 `const mhaconfig_t` *in_cfg*,
 `lpc_burglattice *`*lpc*)

5.106.1.2 `lpc_burglattice_config::~~lpc_burglattice_config` ()

5.106.2 Member Function Documentation

5.106.2.1 `mha_wave_t * lpc_burglattice_config::process` (
 `mha_wave_t * wave`)

5.106.3 Member Data Documentation

5.106.3.1 `algo_comm_t lpc_burglattice_config::ac` [private]

- 5.106.3.2 `MHASignal::waveform_t lpc_burglattice_config::forward` [private]
- 5.106.3.3 `MHASignal::waveform_t lpc_burglattice_config::backward` [private]
- 5.106.3.4 `MHASignal::waveform_t lpc_burglattice_config::kappa` [private]
- 5.106.3.5 `MHA_AC::waveform_t lpc_burglattice_config::kappa_block` [private]
- 5.106.3.6 `MHASignal::waveform_t lpc_burglattice_config::dm` [private]
- 5.106.3.7 `MHASignal::waveform_t lpc_burglattice_config::nm` [private]
- 5.106.3.8 `mha_real_t lpc_burglattice_config::lambda` [private]
- 5.106.3.9 `int lpc_burglattice_config::lpc_order` [private]
- 5.106.3.10 `std::string lpc_burglattice_config::name_f` [private]
- 5.106.3.11 `std::string lpc_burglattice_config::name_b` [private]
- 5.106.3.12 `mha_wave_t lpc_burglattice_config::s_f` [private]
- 5.106.3.13 `mha_wave_t lpc_burglattice_config::s_b` [private]

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

- `lpc_burg-lattice.h`
- `lpc_burg-lattice.cpp`

5.107 `lpc_config` Class Reference

Public Member Functions

- `lpc_config` (`algo_comm_t` &ac, const `mhaconfig_t` in_cfg, std::string &algo_name, unsigned int _order, unsigned int _lpc_buffer_size, bool _shift, unsigned int _comp_each←_iter, bool _norm)
- `~lpc_config` ()
- `mha_wave_t * process` (`mha_wave_t *`)
- void `insert` ()

Private Attributes

- bool **norm**
- bool **shift**
- unsigned int **comp_each_iter**
- unsigned int **order**
- unsigned int **lpc_buffer_size**
- unsigned int **N**
- unsigned int **comp_iter**
- **mha_wave_t** **sample**
- std::vector< **mha_real_t** > **R**
- std::vector< **mha_real_t** > **A**
- **MHASignal::ringbuffer_t** **inwave**
- **MHA_AC::waveform_t** **lpc_out**
- **MHA_AC::waveform_t** **corr_out**

5.107.1 Constructor & Destructor Documentation

5.107.1.1 `lpc_config::lpc_config (`
 `algo_comm_t & ac,`
 `const mhaconfig_t in_cfg,`
 `std::string & algo_name,`
 `unsigned int _order,`
 `unsigned int _lpc_buffer_size,`
 `bool _shift,`
 `unsigned int _comp_each_iter,`
 `bool _norm)`

5.107.1.2 `lpc_config::~~lpc_config ()`

5.107.2 Member Function Documentation

5.107.2.1 `mha_wave_t * lpc_config::process (`
 `mha_wave_t * wave)`

5.107.2.2 `void lpc_config::insert ()`

5.107.3 Member Data Documentation

5.107.3.1 `bool lpc_config::norm` [private]

5.107.3.2 `bool lpc_config::shift` [private]

5.107.3.3 `unsigned int lpc_config::comp_each_iter` [private]

- 5.107.3.4 unsigned int lpc_config::order [private]
- 5.107.3.5 unsigned int lpc_config::lpc_buffer_size [private]
- 5.107.3.6 unsigned int lpc_config::N [private]
- 5.107.3.7 unsigned int lpc_config::comp_iter [private]
- 5.107.3.8 mha_wave_t lpc_config::sample [private]
- 5.107.3.9 std::vector<mha_real_t> lpc_config::R [private]
- 5.107.3.10 std::vector<mha_real_t> lpc_config::A [private]
- 5.107.3.11 MHASignal::ringbuffer_t lpc_config::inwave [private]
- 5.107.3.12 MHA_AC::waveform_t lpc_config::lpc_out [private]
- 5.107.3.13 MHA_AC::waveform_t lpc_config::corr_out [private]

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

- lpc.h
- lpc.cpp

5.108 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**

5.108.1 Constructor & Destructor Documentation

5.108.1.1 `cfg_t::cfg_t (`
 `std::vector< std::vector< float > > imixer,`
 `unsigned int ci,`
 `unsigned int co,`
 `unsigned int fragsize,`
 `unsigned int nfft)`

5.108.2 Member Function Documentation

5.108.2.1 `mha_wave_t * cfg_t::process (`
 `mha_wave_t * s)`

5.108.2.2 `mha_spec_t * cfg_t::process (`
 `mha_spec_t * s)`

5.108.3 Member Data Documentation

5.108.3.1 `MHASignal::waveform_t matrixmixer::cfg_t::m` [private]

5.108.3.2 `MHASignal::waveform_t matrixmixer::cfg_t::wout` [private]

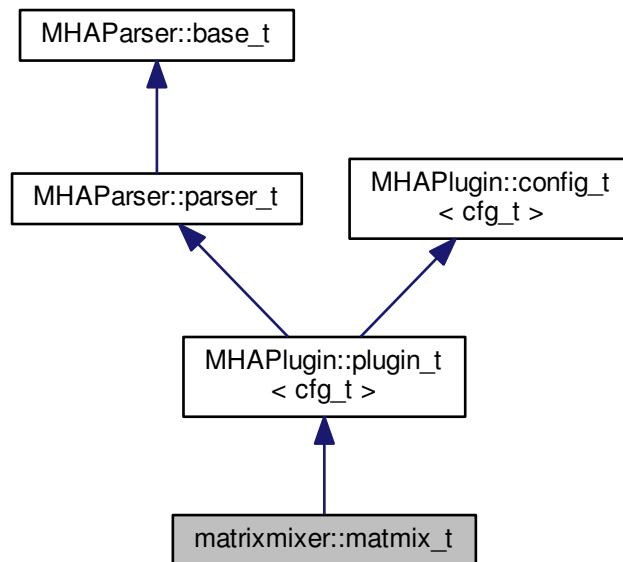
5.108.3.3 `MHASignal::spectrum_t matrixmixer::cfg_t::sout` [private]

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

- `matrixmixer.cpp`

5.109 matrixmixer::matmix_t Class Reference

Inheritance diagram for matrixmixer::matmix_t:



Public Member Functions

- **matmix_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update_m** ()

Private Attributes

- **MHAEvents::patchbay_t** < **matmix_t** > **patchbay**
- **MHAParser::mfloat_t** **mixer**
- unsigned int **ci**
- unsigned int **co**

Additional Inherited Members

5.109.1 Constructor & Destructor Documentation

5.109.1.1 `matrixmixer::matmix_t::matmix_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

5.109.2 Member Function Documentation

5.109.2.1 `void matrixmixer::matmix_t::prepare (`
 `mhaconfig_t & tf)` `[virtual]`

Implements **MHAPLugin::plugin_t**< **cfg_t** > (p. 661).

5.109.2.2 `mha_wave_t * matrixmixer::matmix_t::process (`
 `mha_wave_t * s)`

5.109.2.3 `mha_spec_t * matrixmixer::matmix_t::process (`
 `mha_spec_t * s)`

5.109.2.4 `void matrixmixer::matmix_t::update_m (`
 `void)` `[private]`

5.109.3 Member Data Documentation

5.109.3.1 `MHAEvents::patchbay_t<matmix_t> matrixmixer::matmix_t::patchbay` `[private]`

5.109.3.2 `MHAParser::mfloat_t matrixmixer::matmix_t::mixer` `[private]`

5.109.3.3 `unsigned int matrixmixer::matmix_t::ci` `[private]`

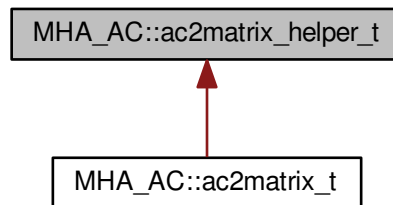
5.109.3.4 `unsigned int matrixmixer::matmix_t::co` `[private]`

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

- **matrixmixer.cpp**

5.110 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**

5.110.1 Constructor & Destructor Documentation

5.110.1.1 MHA_AC::ac2matrix_helper_t::ac2matrix_helper_t (
 algo_comm_t *iac*,
 const std::string & *iname*)

5.110.2 Member Function Documentation

5.110.2.1 void MHA_AC::ac2matrix_helper_t::getvar ()

5.110.3 Member Data Documentation

5.110.3.1 `algo_comm_t MHA_AC::ac2matrix_helper_t::ac`

5.110.3.2 `std::string MHA_AC::ac2matrix_helper_t::name`

5.110.3.3 `std::string MHA_AC::ac2matrix_helper_t::username`

5.110.3.4 `MHASignal::uint_vector_t MHA_AC::ac2matrix_helper_t::size`

5.110.3.5 `bool MHA_AC::ac2matrix_helper_t::is_complex`

5.110.3.6 `comm_var_t MHA_AC::ac2matrix_helper_t::acvar` `[protected]`

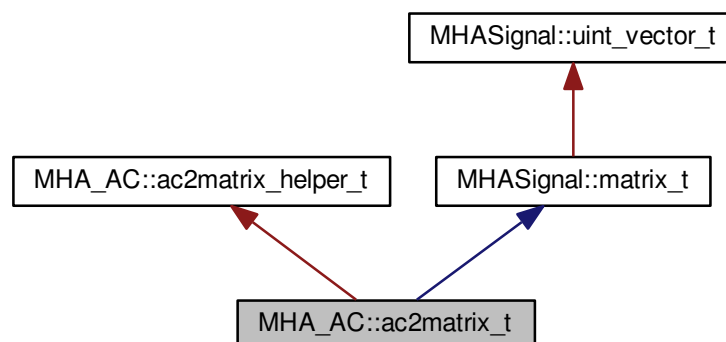
The documentation for this class was generated from the following files:

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

5.111 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

5.111.1 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. 357) function is real-time safe.

5.111.2 Constructor & Destructor Documentation

5.111.2.1 MHA_AC::ac2matrix_t::ac2matrix_t (**algo_comm_t** ac, const std::string & *name*)

Constructor.

Parameters

| | |
|-------------|----------------------------------|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of AC variable to be copied |

5.111.3 Member Function Documentation

5.111.3.1 void MHA_AC::ac2matrix_t::update ()

Update contents of the matrix from the AC space.

This function is real-time safe. The copy operation performance is of the order of the number of elements in the matrix.

5.111.3.2 `const std::string& MHA_AC::ac2matrix_t::getname () const` `[inline]`

Return name of AC variable/matrix.

5.111.3.3 `const std::string& MHA_AC::ac2matrix_t::getusername () const` `[inline]`

Return user specified name of AC variable/matrix.

5.111.3.4 `void MHA_AC::ac2matrix_t::insert (`
`algo_comm_t ac)`

Insert matrix into an AC space (other than source AC space)

Parameters

| | |
|-----------------|--------------------------------|
| <code>ac</code> | AC space handle to insert data |
|-----------------|--------------------------------|

Note

The AC variable data buffer points to the data of the matrix. Modifications of the AC variable directly modify the data of the matrix; after deletion of the matrix, the data buffer is invalid.

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

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

5.112 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**

5.112.1 Detailed Description

Copy all or a subset of all numeric AC variables into an array of matrixes.

5.112.2 Constructor & Destructor Documentation

5.112.2.1 **MHA_AC::acspace2matrix_t::acspace2matrix_t** (
 algo_comm_t ac,
 const std::vector< std::string > & *names*)

Constructor.

Scan all given AC variables and allocate corresponding matrixes.

Parameters

| | |
|--------------|--|
| <i>ac</i> | AC handle. |
| <i>names</i> | Names of AC variables, or empty for all. |

5.112.2.2 **MHA_AC::acspace2matrix_t::acspace2matrix_t** (
 const **MHA_AC::acspace2matrix_t** & *src*)

Constructor with initialization from an instance.

Parameters

| | |
|------------|------------------------|
| <i>src</i> | Instance to be copied. |
|------------|------------------------|

5.112.2.3 MHA_AC::acspace2matrix_t::~~acspace2matrix_t()**5.112.3 Member Function Documentation****5.112.3.1 MHA_AC::acspace2matrix_t & MHA_AC::acspace2matrix_t::operator=(
const MHA_AC::acspace2matrix_t & *src*)**

Copy all contents (deep copy).

Parameters

| | |
|------------|---------------------------------|
| <i>src</i> | Array of matrixes to be copied. |
|------------|---------------------------------|

**5.112.3.2 MHA_AC::ac2matrix_t & MHA_AC::acspace2matrix_t::operator[](
unsigned int *k*) [inline]**

Access operator.

Parameters

| | |
|----------|---|
| <i>k</i> | index into array; should not exceed size() (p. 361)-1. |
|----------|---|

Return values

| | |
|------------------|------------|
| <i>Reference</i> | to matrix. |
|------------------|------------|

**5.112.3.3 const MHA_AC::ac2matrix_t & MHA_AC::acspace2matrix_t::operator[](
unsigned int *k*) const [inline]**

Constant access operator.

Parameters

| | |
|----------|---|
| <i>k</i> | index into array; should not exceed size() (p. 361)-1. |
|----------|---|

Return values

| | |
|-----------------|----------------------|
| <i>Constant</i> | reference to matrix. |
|-----------------|----------------------|

5.112.3.4 void MHA_AC::acspace2matrix_t::update () [inline]

Update function.

This function updates all matrixes from their corresponding AC variables. It can be called from the MHA Framework prepare function or in the processing callback.

5.112.3.5 unsigned int MHA_AC::acspace2matrix_t::size () const [inline]

Number of matrixes in AC space.

5.112.3.6 unsigned int MHA_AC::acspace2matrix_t::frame () const [inline]

Actual frame number.

5.112.3.7 void MHA_AC::acspace2matrix_t::insert (algo_comm_t ac)

Insert AC space copy into an AC space (other than source AC space)

Parameters

| | |
|-----------|--------------------------------|
| <i>ac</i> | AC space handle to insert data |
|-----------|--------------------------------|

5.112.4 Member Data Documentation

5.112.4.1 unsigned int MHA_AC::acspace2matrix_t::len [private]

5.112.4.2 MHA_AC::ac2matrix_t** MHA_AC::acspace2matrix_t::data [private]

5.112.4.3 unsigned int MHA_AC::acspace2matrix_t::frameno [private]

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

- mha_algo_comm.h
- mha_algo_comm.cpp

5.113 MHA_AC::double_t Class Reference

Insert a double precision floating point variable into the AC space.

Public Member Functions

- **double_t** (**algo_comm_t**, std::string, double=0)
- **~double_t** ()

Public Attributes

- double **data**
Floating point value variable.

Private Attributes

- **algo_comm_t ac**

5.113.1 Detailed Description

Insert a double precision floating point variable into the AC space.

The variable is automatically removed on destruction.

5.113.2 Constructor & Destructor Documentation

5.113.2.1 **MHA_AC::double_t::double_t** (
 algo_comm_t iac,
 std::string *n*,
 double *v* = 0)

5.113.2.2 **MHA_AC::double_t::~~double_t** ()

5.113.3 Member Data Documentation

5.113.3.1 double **MHA_AC::double_t::data**

Floating point value variable.

5.113.3.2 **algo_comm_t MHA_AC::double_t::ac** [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

5.114 MHA_AC::float_t Class Reference

Insert a float point variable into the AC space.

Public Member Functions

- **float_t** (**algo_comm_t**, std::string, float=0)
Constructor.
- **~float_t** ()

Public Attributes

- float **data**
Floating point value variable.

Private Attributes

- **algo_comm_t ac**

5.114.1 Detailed Description

Insert a float point variable into the AC space.

The variable is automatically removed on destruction.

5.114.2 Constructor & Destructor Documentation

5.114.2.1 MHA_AC::float_t::float_t (
 algo_comm_t iac,
 std::string *n*,
 float *v* = 0)

Constructor.

5.114.2.2 MHA_AC::float_t::~~float_t ()

5.114.3 Member Data Documentation

5.114.3.1 float MHA_AC::float_t::data

Floating point value variable.

5.114.3.2 algo_comm_t MHA_AC::float_t::ac [private]

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

- mha_algo_comm.h
- mha_algo_comm.cpp

5.115 MHA_AC::int_t Class Reference

Insert a integer variable into the AC space.

Public Member Functions

- **int_t** (algo_comm_t, std::string, int=0)
- **~int_t** ()

Public Attributes

- int **data**
Integer value variable.

Private Attributes

- **algo_comm_t ac**

5.115.1 Detailed Description

Insert a integer variable into the AC space.

The variable is automatically removed on destruction.

5.115.2 Constructor & Destructor Documentation

5.115.2.1 MHA_AC::int_t::int_t (
 algo_comm_t iac,
 std::string n,
 int v = 0)

5.115.2.2 MHA_AC::int_t::~~int_t ()

5.115.3 Member Data Documentation

5.115.3.1 int MHA_AC::int_t::data

Integer value variable.

5.115.3.2 algo_comm_t MHA_AC::int_t::ac [private]

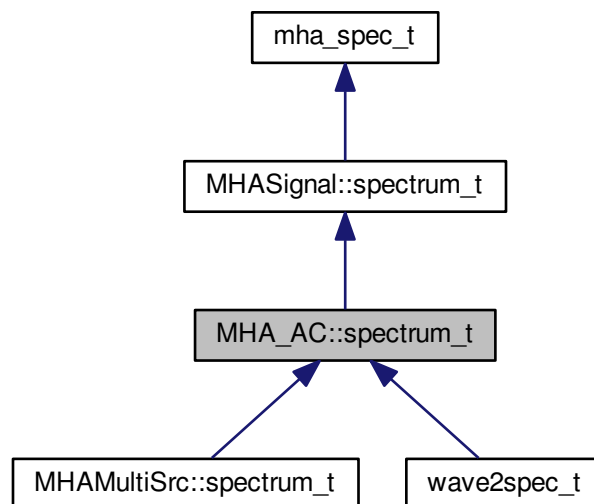
The documentation for this class was generated from the following files:

- mha_algo_comm.h
- mha_algo_comm.cpp

5.116 MHA_AC::spectrum_t Class Reference

Insert a **MHASignal::spectrum_t** (p. 731) class into the AC space.

Inheritance diagram for MHA_AC::spectrum_t:



Public Member Functions

- **spectrum_t** (**algo_comm_t** ac, std::string **name**, unsigned int bins, unsigned int **channels**, bool insert_now)
Create the AC variable.
- **~spectrum_t** ()
- void **insert** ()
Insert AC variable into AC space.

Protected Attributes

- **algo_comm_t** ac
- std::string **name**

Additional Inherited Members

5.116.1 Detailed Description

Insert a **MHASignal::spectrum_t** (p. 731) class into the AC space.

The variable is automatically removed on destruction.

5.116.2 Constructor & Destructor Documentation

5.116.2.1 **MHA_AC::spectrum_t::spectrum_t** (
 algo_comm_t *ac*,
 std::string *name*,
 unsigned int *bins*,
 unsigned int *channels*,
 bool *insert_now*)

Create the AC variable.

Parameters

| | |
|-------------------|--|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of variable in AC space |
| <i>bins</i> | Number of FFT bins in the waveform_t (p. 368) class |
| <i>channels</i> | Number of audio channels in the waveform_t (p. 368) class |
| <i>insert_now</i> | Insert implicitly in the constructor (true) or explicitly in the insert() (p. 366) function (false) |

5.116.2.2 **MHA_AC::spectrum_t::~~spectrum_t** (
 void) [*virtual*]

Reimplemented from **MHASignal::spectrum_t** (p. 733).

5.116.3 Member Function Documentation

5.116.3.1 **void MHA_AC::spectrum_t::insert** ()

Insert AC variable into AC space.

5.116.4 Member Data Documentation

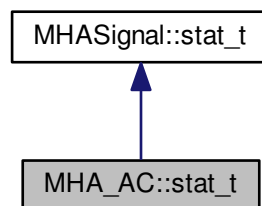
5.116.4.1 `algo_comm_t MHA_AC::spectrum_t::ac` [protected]5.116.4.2 `std::string MHA_AC::spectrum_t::name` [protected]

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

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

5.117 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

5.117.1 Constructor & Destructor Documentation

5.117.1.1 `MHA_AC::stat_t::stat_t (`
 `algo_comm_t ac,`
 `const std::string & name,`
 `const unsigned int & frames,`
 `const unsigned int & channels,`
 `bool insert_now)`

5.117.2 Member Function Documentation

5.117.2.1 `void MHA_AC::stat_t::update ()`

5.117.2.2 `void MHA_AC::stat_t::insert ()`

5.117.3 Member Data Documentation

5.117.3.1 `MHA_AC::waveform_t MHA_AC::stat_t::mean` [private]

5.117.3.2 `MHA_AC::waveform_t MHA_AC::stat_t::std` [private]

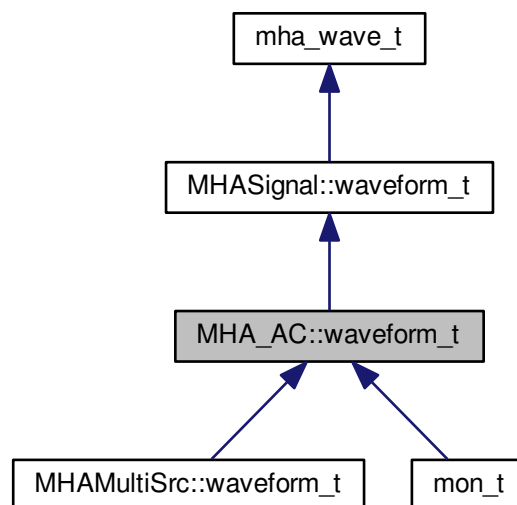
The documentation for this class was generated from the following files:

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

5.118 MHA_AC::waveform_t Class Reference

Insert a **MHASignal::waveform_t** (p. 743) class into the AC space.

Inheritance diagram for `MHA_AC::waveform_t`:



Public Member Functions

- **waveform_t** (**algo_comm_t** **ac**, std::string **name**, unsigned int **frames**, unsigned int **channels**, bool **insert_now**)
Create the AC variable.
- **~waveform_t** ()
- void **insert** ()
Insert AC variable into AC space.

Protected Attributes

- **algo_comm_t** **ac**
- std::string **name**

Additional Inherited Members

5.118.1 Detailed Description

Insert a **MHASignal::waveform_t** (p. 743) class into the AC space.

The variable is automatically removed on destruction.

5.118.2 Constructor & Destructor Documentation

5.118.2.1 MHA_AC::waveform_t::waveform_t (
 algo_comm_t **ac**,
 std::string **name**,
 unsigned int **frames**,
 unsigned int **channels**,
 bool **insert_now**)

Create the AC variable.

Parameters

| | |
|-------------------|--|
| <i>ac</i> | AC handle |
| <i>name</i> | Name of variable in AC space |
| <i>frames</i> | Number of frames in the waveform_t (p. 368) class |
| <i>channels</i> | Number of audio channels in the waveform_t (p. 368) class |
| <i>insert_now</i> | Insert implicitly in the constructor (true) or explicitly in the insert() (p. 370) function (false) |

5.118.2.2 `MHA_AC::waveform_t::~~waveform_t (`
`void) [virtual]`

Reimplemented from `MHASignal::waveform_t` (p. 746).

5.118.3 Member Function Documentation

5.118.3.1 `void MHA_AC::waveform_t::insert ()`

Insert AC variable into AC space.

5.118.4 Member Data Documentation

5.118.4.1 `algo_comm_t MHA_AC::waveform_t::ac [protected]`

5.118.4.2 `std::string MHA_AC::waveform_t::name [protected]`

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

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

5.119 mha_audio_descriptor_t Struct Reference

Description of an audio fragment (planned as a replacement of `mhaconfig_t` (p. 444)).

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.

5.119.1 Detailed Description

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 444)).

5.119.2 Member Data Documentation

5.119.2.1 unsigned int mha_audio_descriptor_t::n_samples

Number of samples.

5.119.2.2 unsigned int mha_audio_descriptor_t::n_channels

Number of audio channels.

5.119.2.3 unsigned int mha_audio_descriptor_t::n_freqs

Number of frequency bands.

5.119.2.4 unsigned int mha_audio_descriptor_t::is_complex

Flag about sample type.

5.119.2.5 mha_real_t mha_audio_descriptor_t::dt

Time distance between samples (only equidistant samples allowed)

5.119.2.6 mha_real_t* mha_audio_descriptor_t::cf

Center frequencies of frequency bands.

5.119.2.7 mha_real_t* mha_audio_descriptor_t::chdir

Hint on source direction of channel, values below zero is left, values above zero is right, zero means unknown.

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

- **mha.h**

5.120 mha_audio_t Struct Reference

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406)).

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. 371) is unset.*

- **mha_complex_t * cdata**

*Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 371) is set.*

5.120.1 Detailed Description

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406)).

The data alignment is $(t_0, c_0, f_0), (t_0, c_0, f_1), \dots, (t_0, c_0, f_{freqs}), (t_0, c_1, f_0), \dots$. This allows a direct cast of the current **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406) data pointers into corresponding **mha_audio_t** (p. 371) objects.

5.120.2 Member Data Documentation

5.120.2.1 mha_audio_descriptor_t mha_audio_t::descriptor

Dimension and description of the data.

5.120.2.2 mha_real_t* mha_audio_t::rdata

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 371) is unset.

5.120.2.3 mha_complex_t* mha_audio_t::cdata

Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 371) is set.

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

- **mha.h**

5.121 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.

5.121.1 Detailed Description

Channel information structure.

5.121.2 Member Data Documentation

5.121.2.1 int mha_channel_info_t::id

channel id

5.121.2.2 char mha_channel_info_t::idstr[32]

channel id

5.121.2.3 unsigned int mha_channel_info_t::side

side (left/right)

5.121.2.4 mha_direction_t mha_channel_info_t::dir

source direction

5.121.2.5 mha_real_t mha_channel_info_t::peaklevel

Peak level corresponds to this SPL (dB) level.

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

- **mha.h**

5.122 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.

5.122.1 Detailed Description

Type for complex floating point values.

5.122.2 Member Data Documentation

5.122.2.1 mha_real_t mha_complex_t::re

Real part.

5.122.2.2 mha_real_t mha_complex_t::im

Imaginary part.

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

- **mha.h**

5.123 mha_dblbuf_t< FIFO > Class Template Reference

The doublebuffer adapts block sizes 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.

5.123.1 Detailed Description

```
template<class FIFO>
class mha_dblbuf_t< FIFO >
```

The doublebuffer adapts block sizes 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.

5.123.2 Member Typedef Documentation

5.123.2.1 `template<class FIFO > typedef FIFO::value_type mha_dblbuf_t< FIFO >::value_type`

The datatype exchanged by the FIFO and this doublebuffer.

5.123.3 Constructor & Destructor Documentation

5.123.3.1 `template<class FIFO > mha_dblbuf_t< FIFO >::mha_dblbuf_t (`
`unsigned outer_size,`
`unsigned inner_size,`
`unsigned delay,`
`unsigned input_channels,`
`unsigned output_channels,`
`const value_type & delay_data)`

Constructor creates FIFOs with specified delay.

Warning

The doublebuffer may block or raise an exception if the delay is too small. To avoid this, the delay should be

$$delay \geq (inner_size - gcd(inner_size, outer_size))$$

.

Parameters

| | |
|------------------------|---|
| <i>outer_size</i> | The block size used by the outer process. |
| <i>inner_size</i> | The block size used by the inner process. |
| <i>delay</i> | The total delay |
| <i>input_channels</i> | Number of input channels |
| <i>output_channels</i> | Number of output channels |
| <i>delay_data</i> | The delay consists of copies of this value. |

5.123.3.2 `template<class FIFO > mha_dblbuf_t< FIFO >::~~mha_dblbuf_t()` [virtual]

5.123.4 Member Function Documentation

5.123.4.1 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_inner_size()` const [inline], [virtual]

5.123.4.2 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_outer_size()` const [inline], [virtual]

5.123.4.3 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_delay()` const [inline], [virtual]

5.123.4.4 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_fifo_size()` const [inline], [virtual]

5.123.4.5 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_channels()` const [inline], [virtual]

5.123.4.6 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_channels()` const [inline], [virtual]

5.123.4.7 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_fifo_fill_count()` const [inline], [virtual]

5.123.4.8 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_fifo_fill_count()` const [inline], [virtual]

5.123.4.9 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_input_fifo_space()` const [inline], [virtual]

5.123.4.10 `template<class FIFO > virtual unsigned mha_dblbuf_t< FIFO >::get_output_fifo_space()` const [inline], [virtual]

5.123.4.11 `template<class FIFO > virtual MHA_Error* mha_dblbuf_t< FIFO >::get_inner_error (`
`) const [inline], [virtual]`

5.123.4.12 `template<class FIFO > void mha_dblbuf_t< FIFO >::provoke_inner_error (`
`const MHA_Error & error) [virtual]`

5.123.4.13 `template<class FIFO > void mha_dblbuf_t< FIFO >::provoke_outer_error (`
`const MHA_Error & error) [virtual]`

5.123.4.14 `template<class FIFO > void mha_dblbuf_t< FIFO >::process (`
`const value_type * input_signal,`
`value_type * output_signal,`
`unsigned count) [virtual]`

The outer process has to call this method to propagate the input signal to the inner process, and receives back the output signal.

Parameters

| | |
|----------------------|--|
| <i>input_signal</i> | Pointer to the input signal array. |
| <i>output_signal</i> | Pointer to the output signal array. |
| <i>count</i> | The number of data instances provided and expected, lower or equal to inner_size given to constructor. |

Exceptions

| | |
|---------------------------|--|
| MHA_Error (p. 387) | When count is > outer_size as given to constructor or the underlying fifo implementation detects an error. |
|---------------------------|--|

5.123.4.15 `template<class FIFO > void mha_dblbuf_t< FIFO >::input (`
`value_type * input_signal) [virtual]`

The inner process has to call this method to receive its input signal.

Parameters

| | |
|---------------------|--|
| <i>input_signal</i> | Array where the doublebuffer can store the signal. |
|---------------------|--|

Exceptions

| | |
|---------------------------|---|
| MHA_Error (p. 387) | When the underlying fifo implementation detects an error. |
|---------------------------|---|

5.123.4.16 `template<class FIFO > void mha_dblbuf_t< FIFO >::output (const value_type * output_signal) [virtual]`

The outer process has to call this method to deliver its output signal.

Parameters

| | |
|----------------------|---|
| <i>output_signal</i> | Array from which doublebuffer reads outputsignal. |
|----------------------|---|

Exceptions

| | |
|----------------------------------|---|
| <i>MHA_Error</i> (p. 387) | When the underlying fifo implementation detects an error. |
|----------------------------------|---|

5.123.5 Member Data Documentation

5.123.5.1 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::outer_size [private]`

The block size used by the outer process.

5.123.5.2 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::inner_size [private]`

The block size used by the inner process.

5.123.5.3 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::delay [private]`

The delay introduced by bidirectional buffer size adaptation.

5.123.5.4 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::fifo_size [private]`

The size of each of the FIFOs.

5.123.5.5 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::input_channels [private]`

The number of input channels.

5.123.5.6 `template<class FIFO > unsigned mha_dblbuf_t< FIFO >::output_channels [private]`

The number of output channels.

5.123.5.7 `template<class FIFO > FIFO mha_dblbuf_t< FIFO >::input_fifo [private]`

The FIFO for transporting the input signal from the outer process to the inner process.

5.123.5.8 `template<class FIFO > FIFO mha_dblbuf_t< FIFO >::output_fifo [private]`

The FIFO for transporting the output signal from the inner process to the outer process.

5.123.5.9 `template<class FIFO > MHA_Error* mha_dblbuf_t< FIFO >::inner_error [private]`

Owned copy of exception to be thrown in inner thread.

5.123.5.10 `template<class FIFO > MHA_Error* mha_dblbuf_t< FIFO >::outer_error [private]`

Owned copy of exception to be thrown in outer thread.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

5.124 mha_direction_t Struct Reference

Channel source direction structure.

Public Attributes

- **mha_real_t azimuth**
azimuth in radians
- **mha_real_t elevation**
elevation in radians
- **mha_real_t distance**
distance in meters

5.124.1 Detailed Description

Channel source direction structure.

5.124.2 Member Data Documentation

5.124.2.1 mha_real_t mha_direction_t::azimuth

azimuth in radians

5.124.2.2 mha_real_t mha_direction_t::elevation

elevation in radiants

5.124.2.3 mha_real_t mha_direction_t::distance

distance in meters

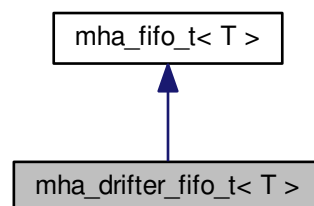
The documentation for this struct was generated from the following file:

- **mha.h**

5.125 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. 387) 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. 387) 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. 384) or `mha_drifter_fifo_t<T>::write` (p. 383) when their xrun in succession counter exceeds its limit.
- virtual void **starting** ()
Called by `mha_drifter_fifo_t<T>::read` (p. 384) or `mha_drifter_fifo_t<T>::write` (p. 383) when the respective flag (`mha_drifter_fifo_t<T>::reader_started` (p. 386) or `mha_drifter_fifo_t<T>::writer_started` (p. 385)) 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. 386) and **mha_drifter_fifo_t<T>::writer_started** (p. 385) are true, then first **mha_drifter_fifo_t<T>::desired_fill_count** (p. 385) instances of **mha_drifter_fifo_t<T>::null_data** (p. 387) are delivered to the reader.

Additional Inherited Members

5.125.1 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.

5.125.2 Constructor & Destructor Documentation

```
5.125.2.1  template<class T > mha_drifter_fifo_t< T >::mha_drifter_fifo_t (
            unsigned min_fill_count,
            unsigned desired_fill_count,
            unsigned max_fill_count )
```

Create drifter FIFO.

```
5.125.2.2  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.

5.125.3 Member Function Documentation

```
5.125.3.1  template<class T > void mha_drifter_fifo_t< T >::write (
            const T * data,
            unsigned count ) [virtual]
```

write data to fifo

Sets **writer_started** (p. 385) to true.

When processing has started, i.e. both **reader_started** (p. 386) and **writer_started** (p. 385) 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. 386) exceeds **maximum_writer_xruns_in_succession_before_stop** (p. 386).

Parameters

| | |
|--------------|-----------------------------|
| <i>data</i> | Pointer to source data. |
| <i>count</i> | Number of instances to copy |

Reimplemented from **mha_fifo_t< T >** (p. 396).

```
5.125.3.2  template<class T > void mha_drifter_fifo_t< T >::read (
            T * buf,
            unsigned count ) [virtual]
```

Read data from fifo.

Sets **reader_started** (p. 386) to true.

When processing has started, i.e. both **reader_started** (p. 386) and **writer_started** (p. 385) are true, then read specified ammount of data from the fifo. As long as **startup_zeros** (p. 387) is > 0, **null_data** (p. 387) is delivered to the reader and **startup_zeros** (p. 387) is diminished. Only when **startup_zeros** (p. 387) 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. 385), then only so much data are read that **minimum_fill_count** (p. 385) entries remain in the fifo, the missing data is replaced with **null_data** (p. 387), and the reader xrun counters are increased.

Processing is stopped when **reader_xruns_in_succession** (p. 386) exceeds **maximum_reader_xruns_in_succession_before_stop** (p. 386).

Parameters

| | |
|--------------|------------------------------|
| <i>buf</i> | Pointer to the target buffer |
| <i>count</i> | Number of instances to copy |

Reimplemented from **mha_fifo_t< T >** (p. 397).

```
5.125.3.3  template<class T > unsigned mha_drifter_fifo_t< T >::get_fill_count ( ) const
            [virtual]
```

Return fill_count, adding **mha_drifter_fifo_t<T>::startup_zeros** (p. 387) to the number of samples actually in the fifo's buffer.

Reimplemented from **mha_fifo_t< T >** (p. 397).

```
5.125.3.4  template<class T > unsigned mha_drifter_fifo_t< T >::get_available_space ( ) const
            [virtual]
```

Return available space, subtracting number of **mha_drifter_fifo_t<T>::startup_zeros** (p. 387) 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. 397).

5.125.3.5 `template<class T> virtual unsigned mha_drifter_fifo_t< T >::get_des_fill_count ()`
`const [inline], [virtual]`

The desired fill count of this fifo.

5.125.3.6 `template<class T> virtual unsigned mha_drifter_fifo_t< T >::get_min_fill_count ()`
`const [inline], [virtual]`

The minimum fill count of this fifo.

5.125.3.7 `template<class T> void mha_drifter_fifo_t< T >::stop () [virtual]`

Called by `mha_drifter_fifo_t<T>::read` (p. 384) or `mha_drifter_fifo_t<T>::write` (p. 383) when their xrun in succession counter exceeds its limit.

Called by `read` (p. 384) or `write` (p. 383) when their xrun in succession counter exceeds its limit.

May also be called explicitly.

5.125.3.8 `template<class T> void mha_drifter_fifo_t< T >::starting () [virtual]`

Called by `mha_drifter_fifo_t<T>::read` (p. 384) or `mha_drifter_fifo_t<T>::write` (p. 383) when the respective flag (`mha_drifter_fifo_t<T>::reader_started` (p. 386) or `mha_drifter_fifo_t<T>::writer_started` (p. 385)) is about to be toggled from false to true.

The fifo's buffer is emptied, this method resets `startup_zeros` (p. 387) to `desired_fill_count` (p. 385), and it also resets `reader_xruns_since_start` (p. 386) and `writer_xruns_since_start` (p. 386) to 0.

5.125.4 Member Data Documentation

5.125.4.1 `template<class T> const unsigned mha_drifter_fifo_t< T >::minimum_fill_count`
`[private]`

The minimum fill count of this fifo.

5.125.4.2 `template<class T> const unsigned mha_drifter_fifo_t< T >::desired_fill_count`
`[private]`

The desired fill count of the fifo.

The fifo is initialized with this ammount of data when data transmission starts.

5.125.4.3 `template<class T> bool mha_drifter_fifo_t< T >::writer_started [private]`

Flag set to true when write is called the first time.

5.125.4.4 `template<class T> bool mha_drifter_fifo_t< T>::reader_started` [private]

Flag set to true when read is called for the first time.

5.125.4.5 `template<class T> unsigned mha_drifter_fifo_t< T>::writer_xruns_total` [private]

The number of xruns seen by the writer since object instantiation.

5.125.4.6 `template<class T> unsigned mha_drifter_fifo_t< T>::reader_xruns_total` [private]

The number of xruns seen by the reader since object instantiation.

5.125.4.7 `template<class T> unsigned mha_drifter_fifo_t< T>::writer_xruns_since_start`
[private]

The number of xruns seen by the writer since the last start of processing.

5.125.4.8 `template<class T> unsigned mha_drifter_fifo_t< T>::reader_xruns_since_start`
[private]

The number of xruns seen by the reader since the last start of processing.

5.125.4.9 `template<class T> unsigned mha_drifter_fifo_t< T>::writer_xruns_in_succession`
[private]

The number of xruns seen by the writer in succession.

Reset to 0 every time a write succeeds without xrun.

5.125.4.10 `template<class T> unsigned mha_drifter_fifo_t< T>::reader_xruns_in_succession`
[private]

The number of xruns seen by the reader in succession.

Reset to 0 every time a read succeeds without xrun.

5.125.4.11 `template<class T> unsigned mha_drifter_fifo_t< T>::maximum_writer_xruns_in_↵
succession_before_stop` [private]

A limit to the number of xruns seen in succession during write before the data transmission through the FIFO is stopped.

5.125.4.12 `template<class T> unsigned mha_drifter_fifo_t< T>::maximum_reader_xruns_in_↵
succession_before_stop` [private]

A limit to the number of xruns seen in succession during read before the data transmission through the FIFO is stopped.

5.125.4.13 `template<class T> mha_fifo_t<T>::value_type mha_drifter_fifo_t<T>::null_data`
`[private]`

The value used in place of missing data.

5.125.4.14 `template<class T> unsigned mha_drifter_fifo_t<T>::startup_zeros` `[private]`

When processing starts, that is when both `mha_drifter_fifo_t<T>::reader_started` (p. 386) and `mha_drifter_fifo_t<T>::writer_started` (p. 385) are true, then first `mha_drifter_fifo_t<T>::desired_fill_count` (p. 385) instances of `mha_drifter_fifo_t<T>::null_data` (p. 387) are delivered to the reader.

These `null_data` (p. 387) instances are not transmitted through the fifo because filling the fifo with enough `null_data` (p. 387) might not be realtime safe and this filling has to be initiated by **starting** (p. 385) or **stop** (p. 385) (this implementation: **starting** (p. 385)) which are be called with realtime constraints.

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

- `mha_fifo.h`

5.126 MHA_Error Class Reference

Error reporting exception class.

Inherits exception.

Public Member Functions

- **MHA_Error** (const char *file, int line, const char *fmt,...)
*Create an instance of a **MHA_Error** (p. 387).*
- **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::exception::what()

Private Attributes

- char * **msg**
- char * **longmsg**

5.126.1 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. 387) class holds source file name, line number and an error message.

5.126.2 Constructor & Destructor Documentation

5.126.2.1 **MHA_Error::MHA_Error** (
 const char * *s_file*,
 int *l*,
 const char * *fmt*,
 ...)

Create an instance of a **MHA_Error** (p. 387).

Parameters

| | |
|---------------|--|
| <i>s_file</i> | source file name (FILE) |
| <i>l</i> | source line (LINE) |
| <i>fmt</i> | format string for error message (as in printf) |

5.126.2.2 **MHA_Error::MHA_Error** (
 const **MHA_Error** & *p*)

5.126.2.3 **MHA_Error::~~MHA_Error** () throw)

5.126.3 Member Function Documentation

5.126.3.1 **MHA_Error & MHA_Error::operator=** (
 const **MHA_Error** & *p*)

5.126.3.2 const char* **MHA_Error::get_msg** () const [inline]

Return the error message without source position.

5.126.3.3 const char* **MHA_Error::get_longmsg** () const [inline]

Return the error message with source position.

5.126.3.4 `const char* MHA_Error::what () const throw` `[inline]`

overwrite `std::exception::what()`

5.126.4 Member Data Documentation

5.126.4.1 `char* MHA_Error::msg` `[private]`

5.126.4.2 `char* MHA_Error::longmsg` `[private]`

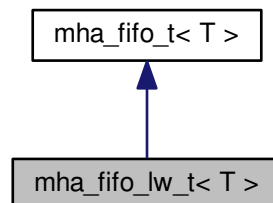
The documentation for this class was generated from the following files:

- `mha_error.hh`
- `mha_error.cpp`

5.127 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

5.127.1 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.

5.127.2 Constructor & Destructor Documentation

5.127.2.1 `template<class T > mha_fifo_lw_t< T >::mha_fifo_lw_t (unsigned max_fill_count) [explicit]`

Create FIFO with fixed buffer size.

5.127.2.2 `template<class T > mha_fifo_lw_t< T >::~~mha_fifo_lw_t () [virtual]`

release synchronization object

5.127.3 Member Function Documentation

5.127.3.1 `template<class T > void mha_fifo_lw_t< T >::write (const T * data, unsigned count) [virtual]`

write specified ammount of data to the fifo.

If there is not enough space, then wait for more space.

Parameters

| | |
|--------------|------------------------------|
| <i>data</i> | Pointer to source data. |
| <i>count</i> | Number of instances to copy. |

Exceptions

| | |
|----------------------------------|--------------------------------------|
| <i>MHA_Error</i> (p. 387) | when detecting a deadlock situation. |
|----------------------------------|--------------------------------------|

Reimplemented from **mha_fifo_t< T >** (p. 396).

```
5.127.3.2  template<class T> void mha_fifo_lw_t< T >::read (
            T * buf,
            unsigned count ) [virtual]
```

read data from fifo.

If there is not enough data, then wait for more data.

Parameters

| | |
|--------------|-------------------------------|
| <i>buf</i> | Pointer to the target buffer. |
| <i>count</i> | Number of instances to copy. |

Exceptions

| | |
|----------------------------------|--------------------------------------|
| <i>MHA_Error</i> (p. 387) | when detecting a deadlock situation. |
|----------------------------------|--------------------------------------|

Reimplemented from **mha_fifo_t< T >** (p. 397).

```
5.127.3.3  template<class T> void mha_fifo_lw_t< T >::set_error (
            unsigned index,
            MHA_Error * error ) [virtual]
```

Process waiting for more data or space should bail out, throwing this error.

Parameters

| | |
|--------------|---|
| <i>index</i> | Use 0 for terminating reader, 1 for terminating writer. |
| <i>error</i> | <i>MHA_Error</i> (p. 387) to be thrown |

5.127.4 Member Data Documentation

5.127.4.1 `template<class T> mha_fifo_thread_platform_t* mha_fifo_lw_t<T>::sync`
`[private]`

platform specific thread synchronization

5.127.4.2 `template<class T> MHA_Error* mha_fifo_lw_t<T>::error[2] [private]`

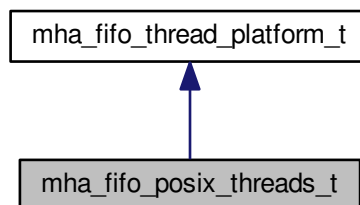
If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

5.128 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 **acquire_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**

5.128.1 Constructor & Destructor Documentation

5.128.1.1 mha_fifo_posix_threads_t::mha_fifo_posix_threads_t() [inline]

5.128.1.2 virtual mha_fifo_posix_threads_t::~~mha_fifo_posix_threads_t() [inline],
[virtual]

5.128.2 Member Function Documentation

5.128.2.1 virtual void mha_fifo_posix_threads_t::aquire_mutex() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 401).

5.128.2.2 virtual void mha_fifo_posix_threads_t::release_mutex() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 401).

5.128.2.3 virtual void mha_fifo_posix_threads_t::wait_for_decrease() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 401).

5.128.2.4 virtual void mha_fifo_posix_threads_t::wait_for_increase() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 401).

5.128.2.5 virtual void mha_fifo_posix_threads_t::increment() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 401).

5.128.2.6 virtual void mha_fifo_posix_threads_t::decrement() [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. 402).

5.128.3 Member Data Documentation

5.128.3.1 `pthread_mutex_t mha_fifo_posix_threads_t::mutex` `[private]`

5.128.3.2 `pthread_cond_t mha_fifo_posix_threads_t::decrease_condition` `[private]`

5.128.3.3 `pthread_cond_t mha_fifo_posix_threads_t::increase_condition` `[private]`

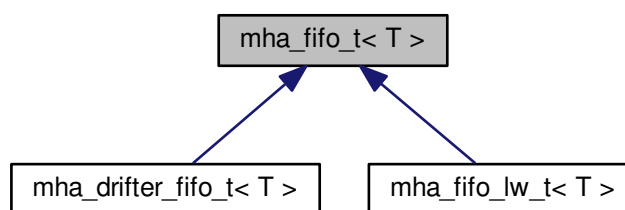
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.129 mha_fifo_t< T > Class Template Reference

A FIFO class for blocksize adaptation Synchronization: None.

Inheritance diagram for mha_fifo_t< T >:



Public Types

- `typedef T value_type`

The data type exchanged by this fifo.

Public Member Functions

- virtual void **write** (const T *data, unsigned count)
write specified ammount of data to the fifo.
- virtual void **read** (T *buf, unsigned count)
read data from fifo
- virtual unsigned **get_fill_count** () const
Read-only access to fill_count.
- virtual unsigned **get_available_space** () const
Read-only access to available_space.
- virtual unsigned **get_max_fill_count** () const
The capacity of this fifo.
- **mha_fifo_t** (unsigned **max_fill_count**)
Create FIFO with fixed buffer size.
- **mha_fifo_t** (unsigned **max_fill_count**, const T &t)
Create FIFO with fixed buffer size, where all (initially unused) copies of T are initialized as copies of t.
- **mha_fifo_t** (const **mha_fifo_t** &src)
Copy constructor.
- virtual ~**mha_fifo_t** ()
Destroy FIFO.
- **mha_fifo_t**< T > & **operator=** (const **mha_fifo_t**< T > &)
Assignment operator.

Protected Member Functions

- void **clear** ()
Empty the fifo at once.

Private Attributes

- const unsigned **max_fill_count**
The maximum fill count of this FIFO.
- T * **buf**
The memory allocated to store the data.
- T * **write_ptr**
points to location where to write next
- const T * **read_ptr**
points to location where to read next
- bool **buf_uses_placement_new**
wether buf was allocated using placement new or array new.

5.129.1 Detailed Description

```
template<class T>
class mha_fifo_t< T >
```

A FIFO class for blocksize adaptation Synchronization: None.

Use external synchronisation or synchronization in inheriting class.

5.129.2 Member Typedef Documentation

5.129.2.1 `template<class T> typedef T mha_fifo_t< T >::value_type`

The data type exchanged by this fifo.

5.129.3 Constructor & Destructor Documentation

5.129.3.1 `template<class T> mha_fifo_t< T >::mha_fifo_t (unsigned max_fill_count) [explicit]`

Create FIFO with fixed buffer size.

5.129.3.2 `template<class T> mha_fifo_t< T >::mha_fifo_t (unsigned max_fill_count, const T & t)`

Create FIFO with fixed buffer size, where all (initially unused) copies of T are initialized as copies of t.

5.129.3.3 `template<class T> mha_fifo_t< T >::mha_fifo_t (const mha_fifo_t< T > & src)`

Copy constructor.

5.129.3.4 `template<class T> mha_fifo_t< T >::~~mha_fifo_t () [virtual]`

Destroy FIFO.

5.129.4 Member Function Documentation

5.129.4.1 `template<class T> void mha_fifo_t< T >::write (const T * data, unsigned count) [virtual]`

write specified ammount of data to the fifo.

Parameters

| | |
|--------------|-----------------------------|
| <i>data</i> | Pointer to source data. |
| <i>count</i> | Number of instances to copy |

Exceptions

| | |
|----------------------------------|---|
| <i>MHA_Error</i> (p. 387) | when there is not enough space available. |
|----------------------------------|---|

Reimplemented in **mha_fifo_lw_t< T >** (p. 390), and **mha_drifter_fifo_t< T >** (p. 383).

5.129.4.2 `template<class T> void mha_fifo_t< T >::read (`
`T * buf,`
`unsigned count) [virtual]`

read data from fifo

Parameters

| | |
|--------------|------------------------------|
| <i>buf</i> | Pointer to the target buffer |
| <i>count</i> | Number of instances to copy |

Exceptions

| | |
|----------------------------------|--|
| <i>MHA_Error</i> (p. 387) | when there is not enough data available. |
|----------------------------------|--|

Reimplemented in **mha_fifo_lw_t< T >** (p. 391), and **mha_drifter_fifo_t< T >** (p. 384).

5.129.4.3 `template<class T> unsigned mha_fifo_t< T >::get_fill_count () const [virtual]`

Read-only access to fill_count.

Reimplemented in **mha_drifter_fifo_t< T >** (p. 384).

5.129.4.4 `template<class T> unsigned mha_fifo_t< T >::get_available_space () const`
`[virtual]`

Read-only access to available_space.

Reimplemented in **mha_drifter_fifo_t< T >** (p. 384).

5.129.4.5 `template<class T> virtual unsigned mha_fifo_t< T >::get_max_fill_count () const`
`[inline], [virtual]`

The capacity of this fifo.

5.129.4.6 `template<class T> mha_fifo_t< T > & mha_fifo_t< T >::operator= (const mha_fifo_t< T > & src)`

Assignment operator.

5.129.4.7 `template<class T> void mha_fifo_t< T >::clear () [inline],[protected]`

Empty the fifo at once.

Should be called by the reader, or when the reader is inactive.

5.129.5 Member Data Documentation

5.129.5.1 `template<class T> const unsigned mha_fifo_t< T >::max_fill_count [private]`

The maximum fill count of this FIFO.

5.129.5.2 `template<class T> T* mha_fifo_t< T >::buf [private]`

The memory allocated to store the data.

`max_fill_count + 1` locations are allocated: At least one location is always unused, because we have `max_fill_count + 1` possible fillcounts `[0:max_fill_count]` that we need to distinguish.

5.129.5.3 `template<class T> T* mha_fifo_t< T >::write_ptr [private]`

points to location where to write next

5.129.5.4 `template<class T> const T* mha_fifo_t< T >::read_ptr [private]`

points to location where to read next

5.129.5.5 `template<class T> bool mha_fifo_t< T >::buf_uses_placement_new [private]`

wether buf was allocated using placement new or array new.

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

- **mha_fifo.h**

5.130 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`

5.130.1 Detailed Description

Simple Mutex Guard Class.

5.130.2 Constructor & Destructor Documentation

5.130.2.1 `mha_fifo_thread_guard_t::mha_fifo_thread_guard_t(mha_fifo_thread_platform_t * sync)` `[inline]`

5.130.2.2 `mha_fifo_thread_guard_t::~~mha_fifo_thread_guard_t()` `[inline]`

5.130.3 Member Data Documentation

5.130.3.1 `mha_fifo_thread_platform_t * mha_fifo_thread_guard_t::sync` `[private]`

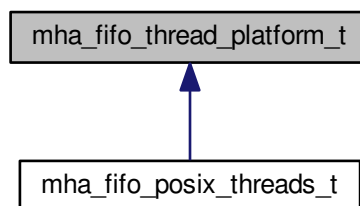
The documentation for this class was generated from the following file:

- `mha_fifo.h`

5.131 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 **acquire_mutex** ()=0
Calling thread waits until it acquires 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** &)

5.131.1 Detailed Description

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Works only with single producer and single consumer.

5.131.2 Constructor & Destructor Documentation

5.131.2.1 `virtual mha_fifo_thread_platform_t::~~mha_fifo_thread_platform_t() [inline],
[virtual]`

Make destructor virtual.

5.131.2.2 `mha_fifo_thread_platform_t::mha_fifo_thread_platform_t(
const mha_fifo_thread_platform_t &) [private]`

5.131.2.3 `mha_fifo_thread_platform_t::mha_fifo_thread_platform_t() [inline]`

Make default constructor accessible.

5.131.3 Member Function Documentation

5.131.3.1 virtual void mha_fifo_thread_platform_t::acquire_mutex () [pure virtual]

Calling thread waits until it acquires the lock.

Must not be called when the lock is already acquired.

Implemented in **mha_fifo_posix_threads_t** (p. 393).

5.131.3.2 virtual void mha_fifo_thread_platform_t::release_mutex () [pure virtual]

Calling thread releases the lock.

May only be called when lock is owned.

Implemented in **mha_fifo_posix_threads_t** (p. 393).

5.131.3.3 virtual void mha_fifo_thread_platform_t::wait_for_decrease () [pure virtual]

Calling producer thread must own the lock.

Method releases lock, and waits for consumer thread to call decrease(). Then reacquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. 393).

5.131.3.4 virtual void mha_fifo_thread_platform_t::wait_for_increase () [pure virtual]

Calling consumer thread must own the lock.

Method releases lock, and waits for producer thread to call increase(). Then reacquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. 393).

5.131.3.5 virtual void mha_fifo_thread_platform_t::increment () [pure virtual]

To be called by producer thread after producing.

Producer thread needs to own the lock to call this method.

Implemented in **mha_fifo_posix_threads_t** (p. 393).

5.131.3.6 `virtual void mha_fifo_thread_platform_t::decrement ()` [pure virtual]

To be called by consumer thread after consuming.

Consumer thread needs to own the lock to call this method.

Implemented in `mha_fifo_posix_threads_t` (p. 393).

5.131.3.7 `mha_fifo_thread_platform_t& mha_fifo_thread_platform_t::operator= (const mha_fifo_thread_platform_t &)` [private]

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

- `mha_fifo.h`

5.132 `mha_rt_fifo_element_t`< T > Class Template Reference

Object wrapper for `mha_rt_fifo_t` (p. 403).

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.

5.132.1 Detailed Description

```
template<class T>
class mha_rt_fifo_element_t< T >
```

Object wrapper for `mha_rt_fifo_t` (p. 403).

5.132.2 Constructor & Destructor Documentation

5.132.2.1 `template<class T> mha_rt_fifo_element_t< T >::mha_rt_fifo_element_t (T * data)` [inline]

Constructor.

This element assumes ownership of user data.

Parameters

| | |
|-------------|---|
| <i>data</i> | User data. Has to be allocated on the heap with standard operator new, because it will be deleted in this element's destructor. |
|-------------|---|

5.132.2.2 `template<class T > mha_rt_fifo_element_t< T >::~~mha_rt_fifo_element_t()`
`[inline]`

5.132.3 Member Data Documentation

5.132.3.1 `template<class T > mha_rt_fifo_element_t<T>* mha_rt_fifo_element_t< T >::next`

Pointer to next fifo element. NULL for the last (newest) fifo element.

5.132.3.2 `template<class T > bool mha_rt_fifo_element_t< T >::abandoned`

Indicates that this element will no longer be used and may be deleted.

5.132.3.3 `template<class T > T* mha_rt_fifo_element_t< T >::data`

Pointer to user data.

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

- **mha_fifo.h**

5.133 mha_rt_fifo_t< T > Class Template Reference

Template class for thread safe, half real time safe fifo without explicit 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 abandoned.
- **void push (T *data)**
Add element to the Fifo.

Private Member Functions

- void **remove_abandoned** ()
Deletes abandoned 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. 405) or **poll_1** (p. 405).*

5.133.1 Detailed Description

```
template<class T>
class mha_rt_fifo_t< T >
```

Template class for thread safe, half real time safe fifo without explicit 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. 405). The push operation is not real time safe, it allocates and deallocates memory. The latest element is retrieved by calling **poll** (p. 405). This operation will skip fifo elements if more than one **push** (p. 405) has been occurred since the last poll. To avoid skipping, call the **poll_1** (p. 405) operation instead.

5.133.2 Constructor & Destructor Documentation

5.133.2.1 `template<class T> mha_rt_fifo_t< T >::mha_rt_fifo_t() [inline]`

Construct empty fifo.

5.133.2.2 `template<class T> mha_rt_fifo_t< T >::~~mha_rt_fifo_t() [inline]`

Destructor will delete all data currently in the fifo.

5.133.3 Member Function Documentation

5.133.3.1 `template<class T> T* mha_rt_fifo_t< T >::poll () [inline]`

Retrieve the latest element in the Fifo.

Will skip fifo elements if more than one element has been added since last poll invocation. Will return the same element as on last call if no elements have been added in the mean time. Marks former elements as abandonned.

Returns

The latest element in this Fifo. Returns NULL if the Fifo is empty.

5.133.3.2 `template<class T> T* mha_rt_fifo_t< T >::poll_1 () [inline]`

Retrieve the next element in the Fifo, if there is one, and mark the previous element as abandonned.

Else, if there is no newer element, returns the same element as on last **poll()** (p. 405) or **poll_1()** (p. 405) invocation.

Returns

The next element in this Fifo, if there is one, or the same as before. Returns NULL if the Fifo is empty.

5.133.3.3 `template<class T> void mha_rt_fifo_t< T >::push (T* data) [inline]`

Add element to the Fifo.

Deletes abandonned elements in the fifo.

Parameters

| | |
|-------------|--|
| <i>data</i> | 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. |
|-------------|--|

5.133.3.4 `template<class T> void mha_rt_fifo_t< T >::remove_abandonned () [inline], [private]`

Deletes abandonned elements.

5.133.3.5 `template<class T> void mha_rt_fifo_t<T>::remove_all() [inline], [private]`

Deletes all elements.

5.133.4 Member Data Documentation

5.133.4.1 `template<class T> mha_rt_fifo_element_t<T>* mha_rt_fifo_t<T>::root [private]`

The first element in the fifo. Deleting elements starts here.

5.133.4.2 `template<class T> mha_rt_fifo_element_t<T>* mha_rt_fifo_t<T>::current [private]`

The element most recently returned by **poll** (p. 405) or **poll_1** (p. 405).

Searching for new elements starts here.

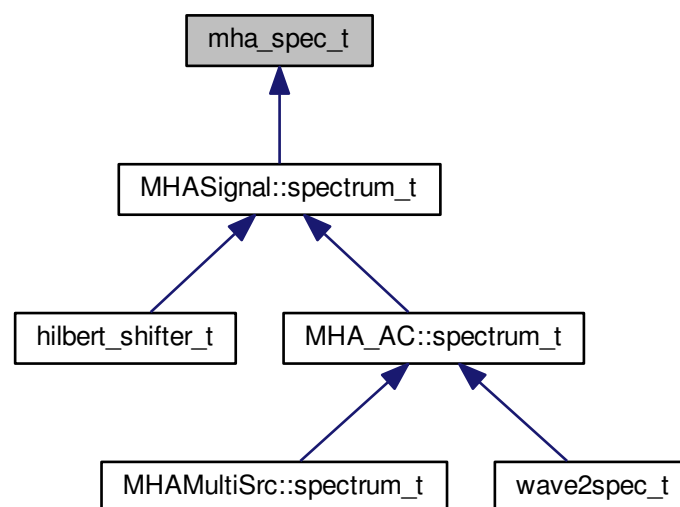
The documentation for this class was generated from the following file:

- **mha_fifo.h**

5.134 mha_spec_t Struct Reference

Spectrum signal structure.

Inheritance diagram for mha_spec_t:



Public Attributes

- **mha_complex_t * buf**
signal buffer
- unsigned int **num_channels**
number of channels
- unsigned int **num_frames**
number of frames in each channel
- **mha_channel_info_t * channel_info**
detailed channel description

5.134.1 Detailed Description

Spectrum signal structure.

This structure contains the short time fourier transform output of the windowed input signal. The member `num_frames` describes the number of frequency bins in each channel. For an even FFT length N , this is $N/2 + 1$. With odd FFT lengths, it is $(N + 1)/2$. The imaginary part of the first bin is zero. For even FFT lengths, also the imaginary part at the Nyquist frequency is zero.

| | | | | | | | | |
|-------------------|---------|---------|---------|---------|---------|-----|-------------|-----------|
| buff[k].re | $Re(0)$ | $Re(1)$ | $Re(2)$ | $Re(3)$ | $Re(4)$ | ... | $Re(n/2-1)$ | $Re(n/2)$ |
| buff[k].im | | $Im(1)$ | $Im(2)$ | $Im(3)$ | $Im(4)$ | ... | $Im(n/2-1)$ | |
| k | 0 | 1 | 2 | 3 | 4 | | $n/2-1$ | $n/2$ |

Figure 4 Data order of FFT spectrum.

5.134.2 Member Data Documentation

5.134.2.1 mha_complex_t* mha_spec_t::buf

signal buffer

5.134.2.2 unsigned int mha_spec_t::num_channels

number of channels

5.134.2.3 unsigned int mha_spec_t::num_frames

number of frames in each channel

5.134.2.4 mha_channel_info_t* mha_spec_t::channel_info

detailed channel description

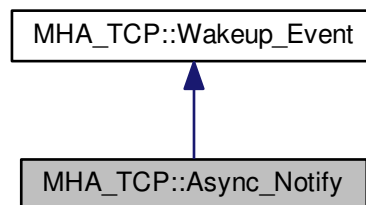
The documentation for this struct was generated from the following file:

- **mha.h**

5.135 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

5.135.1 Detailed Description

Portable Multiplexable cross-thread notification.

5.135.2 Constructor & Destructor Documentation

5.135.2.1 Async_Notify::Async_Notify ()

5.135.2.2 Async_Notify::~Async_Notify () [virtual]

5.135.3 Member Function Documentation

5.135.3.1 void Async_Notify::reset () [virtual]

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 435).

5.135.3.2 void Async_Notify::set () [virtual]

5.135.4 Member Data Documentation

5.135.4.1 int MHA_TCP::Async_Notify::pipe[2] [private]

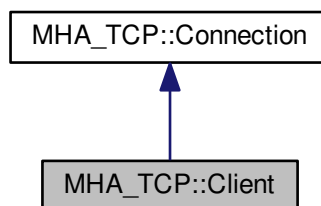
The documentation for this class was generated from the following files:

- mha_tcp.hh
- mha_tcp.cpp

5.136 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

5.136.1 Detailed Description

A portable class for a tcp client connections.

5.136.2 Constructor & Destructor Documentation

5.136.2.1 Client::Client (const std::string & host, unsigned short port)

Constructor connects to host, port via TCP.

Parameters

| | |
|-------------|---|
| <i>host</i> | The hostname of the TCP Server (p. 421). |
| <i>port</i> | The port or the TCP Server (p. 421). |

5.136.2.2 Client::Client (const std::string & host, unsigned short port, **Timeout_Watcher** & timeout_watcher)

Constructor connects to host, port via TCP, using a timeout.

Parameters

| | |
|------------------------|---|
| <i>host</i> | The hostname of the TCP Server (p. 421). |
| <i>port</i> | The port or the TCP Server (p. 421). |
| <i>timeout_watcher</i> | an Event watcher that implements a timeout. |

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

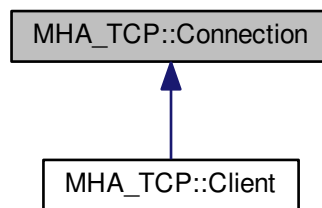
- **mha_tcp.hh**

- `mha_tcp.cpp`

5.137 MHA_TCP::Connection Class Reference

Connection (p. 411) handles Communication between client and server, is used on both sides.

Inheritance diagram for MHA_TCP::Connection:



Public Member Functions

- **Socketread_Event * get_read_event ()**
- **Socketwrite_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 whether at least 1 byte can be read without blocking.

- bool **can_syswrite** ()

Determine whether 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**

5.137.1 Detailed Description



Connection (p. 411) handles Communication between client and server, is used on both sides.

5.137.2 Constructor & Destructor Documentation

5.137.2.1 MHA_TCP::Connection::Connection (
 SOCKET_fd) [protected]

Create a connection instance from a socket filedescriptor.

Parameters

| | |
|---|--|
|  | The file descriptor of the TCP Socket. This file descriptor is closed again in the destructor. |
|  | |
| <i>fd</i> | |

Exceptions

| | |
|---------------------------|--------------------------------|
| MHA_Error (p. 387) | If the file descriptor is < 0. |
|---------------------------|--------------------------------|

5.137.2.2 Connection::~~Connection () [virtual]

Destructor closes the underlying file descriptor.

5.137.3 Member Function Documentation

5.137.3.1 void MHA_TCP::Connection::init_peer_data () [private]

determine peer address and port

5.137.3.2 bool Connection::can_sysread () [private]

Determine whether at least 1 byte can be read without blocking.

5.137.3.3 bool Connection::can_syswrite () [private]

Determine whether at least 1 byte can be written without blocking.

5.137.3.4 std::string Connection::sysread (
 unsigned bytes) [private]

Call the system's read function and try to read bytes.

This will block in a situation where can_sysread returns false.

Parameters

| | |
|--------------|-----------------------------------|
| <i>bytes</i> | 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.

5.137.3.5 `std::string Connection::syswrite (`
`const std::string & data) [private]`

Call the system's write function and try to write all characters in the string data.

May write fewer characters, but will at least write one character.

Parameters

| | |
|-------------|--|
| <i>data</i> | A string of characters to write to the socket. |
|-------------|--|

Returns

The rest of the characters that have not yet been written.

5.137.3.6 `Sockread_Event * Connection::get_read_event ()`

5.137.3.7 `Sockwrite_Event * Connection::get_write_event ()`

5.137.3.8 `std::string Connection::get_peer_address ()`

Get peer's IP Address.

5.137.3.9 `unsigned short Connection::get_peer_port ()`

Get peer's TCP port.

5.137.3.10 `SOCKET MHA_TCP::Connection::get_fd () const [inline]`

Return the (protected) file descriptor of the connection.

Will be required for SSL.

5.137.3.11 bool Connection::eof ()

Checks if the peer has closed the connection.

As a side effect, this method fills the internal "incoming" buffer if it was empty and the socket is readable and not eof.

**5.137.3.12 bool Connection::can_read_line (
char *delim* = ' \n ')**

Checks if a full line of text has arrived by now.

This method reads data from the socket into the internal "incoming" buffer if it can be done without blocking.

Parameters

| | |
|--------------|---------------------|
| <i>delim</i> | 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.

**5.137.3.13 bool Connection::can_read_bytes (
unsigned *howmany*)**

Checks if the specified ammount of data can be read.

This method reads data from the socket into an internal "incoming" buffer if it can be done without blocking.

Parameters

| | |
|----------------|--|
| <i>howmany</i> | 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

**5.137.3.14 std::string Connection::read_line (
char *delim* = ' \n ')**

Reads a single line of data from the socket.

Blocks if necessary.

Parameters

| | |
|--------------|---------------------|
| <i>delim</i> | 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.

5.137.3.15 `std::string Connection::read_bytes (`
`unsigned howmany)`

Reads the specified ammount of dat from the socket.

Blocks if necessary.

Parameters

| | |
|----------------|------------------------------|
| <i>howmany</i> | The number of bytes to read. |
|----------------|------------------------------|

Returns

The string of characters read. The string may be shorter if EOF is encountered.

5.137.3.16 `void Connection::try_write (`
`const std::string & data = " ")`

Adds data to the internal "outgoing" buffer, and then tries to write as much data from that buffer to the socket as possible without blocking.

Parameters

| | |
|-------------|-------------------------------|
| <i>data</i> | data to send over the socket. |
|-------------|-------------------------------|

5.137.3.17 `void Connection::write (`
`const std::string & data = " ")`

Adds data to the internal "outgoing" buffer, and then writes that that buffer to the socket, regardless of blocking.

Parameters

| | |
|-------------|-------------------------------|
| <i>data</i> | data to send over the socket. |
|-------------|-------------------------------|

5.137.3.18 `bool Connection::needs_write ()`

Checks if the internal "outgoing" buffer contains data.

5.137.3.19 `unsigned Connection::buffered_incoming_bytes () const`

Returns the number of bytes in the internal "incoming" buffer.

5.137.3.20 `unsigned Connection::buffered_outgoing_bytes () const`

Returns the number of bytes in the internal "outgoing" buffer.

5.137.4 Member Data Documentation**5.137.4.1** `std::string MHA_TCP::Connection::outbuf` [private]**5.137.4.2** `std::string MHA_TCP::Connection::inbuf` [private]**5.137.4.3** `Socketread_Event* MHA_TCP::Connection::read_event` [private]**5.137.4.4** `Socketwrite_Event* MHA_TCP::Connection::write_event` [private]**5.137.4.5** `bool MHA_TCP::Connection::closed` [private]**5.137.4.6** `struct sockaddr_in MHA_TCP::Connection::peer_addr` [private]**5.137.4.7** `SOCKET MHA_TCP::Connection::fd` [protected]

The file descriptor of the TCP Socket.

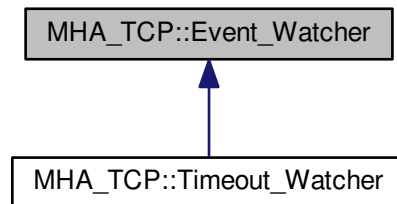
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.138 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.

5.138.1 Detailed Description

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

5.138.2 Member Typedef Documentation

5.138.2.1 `typedef std::set<Wakeup_Event*> MHA_TCP::Event_Watcher::Events`

5.138.2.2 `typedef std::set<Wakeup_Event*>::iterator MHA_TCP::Event_Watcher::iterator`

5.138.3 Constructor & Destructor Documentation

5.138.3.1 `Event_Watcher::~Event_Watcher () [virtual]`

5.138.4 Member Function Documentation

5.138.4.1 `void Event_Watcher::observe (Wakeup_Event * event)`

Add an event to this observer.

5.138.4.2 `void Event_Watcher::ignore (Wakeup_Event * event)`

Remove an event from this observer.

5.138.4.3 `std::set< Wakeup_Event * > Event_Watcher::wait ()`

\ Wait for some event to occur.

Return all events that are ready

5.138.5 Member Data Documentation

5.138.5.1 `std::set<Wakeup_Event*> MHA_TCP::Event_Watcher::events [private]`

The list of events to watch.

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

- `mha_tcp.hh`
- `mha_tcp.cpp`

5.139 MHA_TCP::OS_EVENT_TYPE Struct Reference

Public Types

Public Attributes

- enum MHA_TCP::OS_EVENT_TYPE:: { ... } **mode**
- union {
 - int **fd**
 - double **timeout**
- };

5.139.1 Member Enumeration Documentation

5.139.1.1 anonymous enum

Enumerator

R

W

X

T

5.139.2 Member Data Documentation

5.139.2.1 enum { ... } MHA_TCP::OS_EVENT_TYPE::mode

5.139.2.2 int MHA_TCP::OS_EVENT_TYPE::fd

5.139.2.3 double MHA_TCP::OS_EVENT_TYPE::timeout

5.139.2.4 union { ... }

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

- **mha_tcp.hh**

5.140 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. 418).*
- **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**

5.140.1 Constructor & Destructor Documentation

5.140.1.1 **Server::Server** (
 unsigned short **port** = 0,
 const std::string & **iface** = "0.0.0.0")

Create a TCP server socket.

Parameters

| | |
|--------------|-----------------------------------|
| <i>port</i> | The TCP port to listen to. |
| <i>iface</i> | The network interface to bind to. |

5.140.1.2 `Server::Server (`
 `const std::string & iface,`
 `unsigned short port = 0)`

Create a TCP server socket.

Parameters

| | |
|--------------|-----------------------------------|
| <i>port</i> | The TCP port to listen to. |
| <i>iface</i> | The network interface to bind to. |

5.140.1.3 `Server::~~Server ()`

Close the TCP server socket.

5.140.2 Member Function Documentation

5.140.2.1 `void Server::initialize (`
 `const std::string & iface,`
 `unsigned short port)` `[private]`

5.140.2.2 `std::string Server::get_interface () const`

Get the name given in the constructor for the network interface.

5.140.2.3 `unsigned short Server::get_port () const`

Get the port that the TCP server socket currently listens to.

5.140.2.4 `Sockaccept_Event * Server::get_accept_event ()`

Produces an event that can be observed by an **Event_Watcher** (p. 418).

This event signals incoming connections that can be accepted.

5.140.2.5 Connection * Server::accept ()

Accept an incoming connection.

blocks if necessary.

Returns

The new TCP connection. The connection has to be deleted by the caller.

5.140.2.6 Connection * Server::try_accept ()

Accept an incoming connection if it can be done without blocking.

Returns

The new TCP connection or 0 if there is no immediate connection. The connection has to be deleted by the caller.

5.140.3 Member Data Documentation

5.140.3.1 sockaddr_in MHA_TCP::Server::sock_addr [private]

5.140.3.2 SOCKET MHA_TCP::Server::serversocket [private]

5.140.3.3 std::string MHA_TCP::Server::iface [private]

5.140.3.4 unsigned short MHA_TCP::Server::port [private]

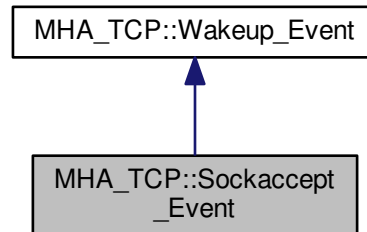
5.140.3.5 Sockaccept_Event* MHA_TCP::Server::accept_event [private]

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

- mha_tcp.hh
- mha_tcp.cpp

5.141 MHA_TCP::Sockaccept_Event Class Reference

Inheritance diagram for MHA_TCP::Sockaccept_Event:



Public Member Functions

- **Sockaccept_Event (SOCKET)**

Additional Inherited Members

5.141.1 Constructor & Destructor Documentation

5.141.1.1 MHA_TCP::Sockaccept_Event::Sockaccept_Event (SOCKET s)

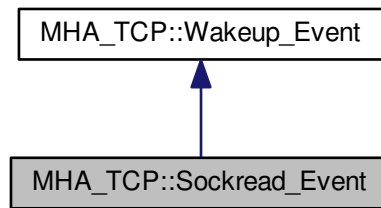
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.142 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

5.142.1 Detailed Description

Watch socket for incoming data.

5.142.2 Constructor & Destructor Documentation

5.142.2.1 MHA_TCP::Sockread_Event::Sockread_Event (SOCKET s)

Set socket to watch for.

Parameters

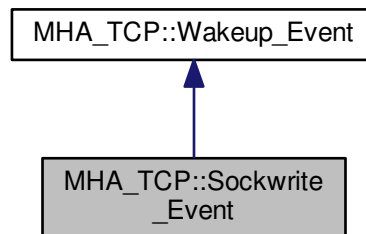
| | |
|---|---|
| s | The socket to observe incoming data on. |
|---|---|

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

- mha_tcp.hh
- mha_tcp.cpp

5.143 MHA_TCP::Sockwrite_Event Class Reference

Inheritance diagram for MHA_TCP::Sockwrite_Event:



Public Member Functions

- **Sockwrite_Event (SOCKET s)**

Additional Inherited Members

5.143.1 Constructor & Destructor Documentation

5.143.1.1 MHA_TCP::Sockwrite_Event::Sockwrite_Event (SOCKET s)

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.144 MHA_TCP::Thread Class Reference

A very simple class for portable threads.

Public Types

- **typedef void *(* **thr_f**) (void *)**
The thread function signature to use with this class.

Public Member Functions

- **Thread** (**thr_f** func, void ***arg**=0)
Constructor starts a new thread.
- virtual ~**Thread** ()
*The destructor should only be called when the **Thread** (p. 426) is finished.*
- virtual void **run** ()
*The internal method that delegated the new thread to the registered **Thread** (p. 426) 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. 387) 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.

5.144.1 Detailed Description

A very simple class for portable threads.

5.144.2 Member Typedef Documentation

5.144.2.1 `typedef void*(* MHA_TCP::Thread::thr_f)(void *)`

The thread function signature to use with this class.

Derive from this class and call protected standard constructor to start threads differently.

5.144.3 Member Enumeration Documentation

5.144.3.1 anonymous enum

The current state of the thread.

Enumerator

PREPARED

RUNNING

FINISHED

5.144.4 Constructor & Destructor Documentation

5.144.4.1 `MHA_TCP::Thread::Thread ()` [protected]

Default constructor may only be used by derived classes that want to start the thread themselves.

5.144.4.2 `Thread::Thread (` `Thread::thr_f func,` `void * arg = 0)`

Constructor starts a new thread.

Parameters

| | |
|-------------|--|
| <i>func</i> | The function to be executed by the thread. |
| <i>arg</i> | The argument given to pass to the thread function. |

5.144.4.3 Thread::~~Thread () [virtual]

The destructor should only be called when the **Thread** (p. 426) is finished.

There is preliminary support for forceful thread cancellation in the destructor, but probably not very robust or portable..

5.144.5 Member Function Documentation

5.144.5.1 void Thread::run () [virtual]

The internal method that delegated the new thread to the registered **Thread** (p. 426) function.

5.144.6 Member Data Documentation

5.144.6.1 pthread_t MHA_TCP::Thread::thread_handle [private]

The posix thread handle.

5.144.6.2 pthread_attr_t MHA_TCP::Thread::thread_attr [private]

The posix thread attribute structure.

Required for starting a thread in detached state. Detachment is required to eliminate the need for joining this thread.

5.144.6.3 void* MHA_TCP::Thread::arg [protected]

The argument for the client's thread function.

5.144.6.4 void* MHA_TCP::Thread::return_value [protected]

The return value from the client's thread function is stored here When that function returns.

5.144.6.5 Async_Notify MHA_TCP::Thread::thread_finish_event

Event will be triggered when the thread exits.

5.144.6.6 enum { ... } MHA_TCP::Thread::state

The current state of the thread.

5.144.6.7 thr_f MHA_TCP::Thread::thread_func

The thread function that the client has registered.

5.144.6.8 void* MHA_TCP::Thread::thread_arg

The argument that the client wants to be handed through to the thread function.

5.144.6.9 MHA_Error* MHA_TCP::Thread::error

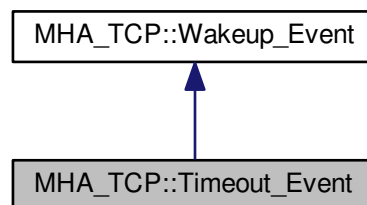
The **MHA_Error** (p. 387) that caused the thread to abort, if any.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.145 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

5.145.1 Constructor & Destructor Documentation

5.145.1.1 Timeout_Event::Timeout_Event (
double *interval*)

5.145.2 Member Function Documentation

5.145.2.1 OS_EVENT_TYPE Timeout_Event::get_os_event () [virtual]

Reimplemented from **MHA_TCP::Wakeup_Event** (p. [434](#)).

5.145.3 Member Data Documentation

5.145.3.1 double MHA_TCP::Timeout_Event::end_time [private]

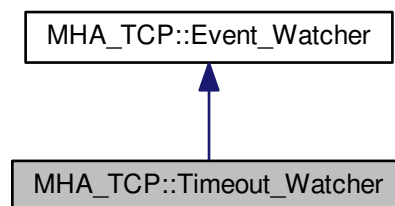
The documentation for this class was generated from the following files:

- mha_tcp.hh
- mha_tcp.cpp

5.146 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

5.146.1 Detailed Description

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

5.146.2 Constructor & Destructor Documentation

5.146.2.1 **Timeout_Watcher::Timeout_Watcher** (
 double *interval*) [explicit]

5.146.2.2 **Timeout_Watcher::~~Timeout_Watcher** () [virtual]

5.146.3 Member Data Documentation

5.146.3.1 **Timeout_Event** MHA_TCP::Timeout_Watcher::timeout [private]

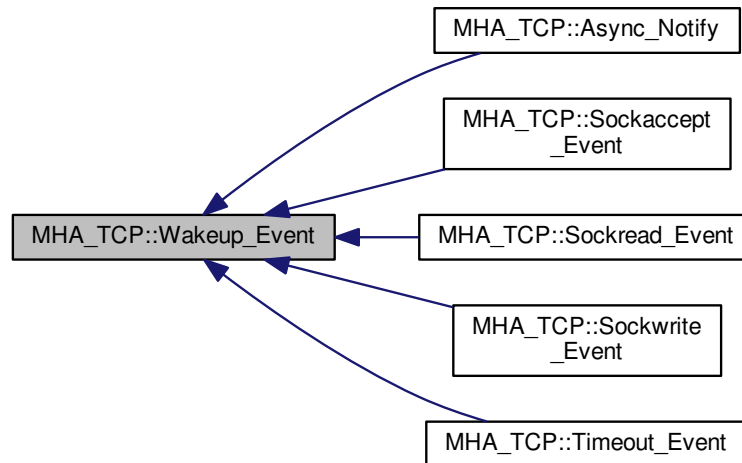
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.147 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. 418) when this event is added to its list of observed events.*
- virtual void **ignored_by** (**Event_Watcher** *observer)
*Called by the **Event_Watcher** (p. 418) 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. 418) instances that this **Wakeup_Event** (p. 433) is observed by (stored here for proper deregistering).*

5.147.1 Detailed Description

A base class for asynchronous wakeup events.

5.147.2 Constructor & Destructor Documentation

5.147.2.1 `Wakeup_Event::Wakeup_Event ()`

Event Constructor.

The new event has invalid state.

5.147.2.2 `Wakeup_Event::~~Wakeup_Event ()` [virtual]

Destructor deregisters from observers.

5.147.3 Member Function Documentation

5.147.3.1 `void Wakeup_Event::observed_by (` `Event_Watcher * observer)` [virtual]

Called by the **Event_Watcher** (p. 418) when this event is added to its list of observed events.

5.147.3.2 `void Wakeup_Event::ignored_by (` `Event_Watcher * observer)` [virtual]

Called by the **Event_Watcher** (p. 418) when this event is removed from its list of observed events.

5.147.3.3 `OS_EVENT_TYPE Wakeup_Event::get_os_event ()` [virtual]

Get necessary information for the Event Watcher.

Reimplemented in **MHA_TCP::Timeout_Event** (p. 431).

5.147.3.4 void Wakeup_Event::reset () [virtual]

For pure notification events, reset the "signalled" status.

Reimplemented in **MHA_TCP::Async_Notify** (p. 409).

5.147.3.5 bool Wakeup_Event::status () [virtual]

Query whether the event is in signalled state now.

5.147.4 Member Data Documentation

5.147.4.1 std::set<class Event_Watcher *> MHA_TCP::Wakeup_Event::observers [private]

A list of all **Event_Watcher** (p. 418) instances that this **Wakeup_Event** (p. 433) is observed by (stored here for proper deregistering).

5.147.4.2 OS_EVENT_TYPE MHA_TCP::Wakeup_Event::os_event [protected]

5.147.4.3 bool MHA_TCP::Wakeup_Event::os_event_valid [protected]

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

5.148 mha_tictoc_t Struct Reference

Public Attributes

- struct timeval **tv1**
- struct timeval **tv2**
- struct timezone **tz**
- float **t**

5.148.1 Member Data Documentation

5.148.1.1 struct timeval mha_tictoc_t::tv1

5.148.1.2 struct timeval mha_tictoc_t::tv2

5.148.1.3 struct timezone mha_tictoc_t::tz

5.148.1.4 float mha_tictoc_t::t

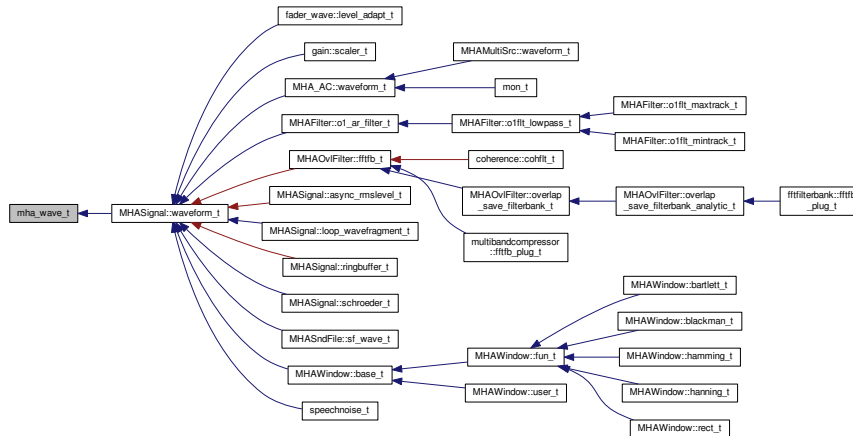
The documentation for this struct was generated from the following file:

- **mha_profiling.h**

5.149 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

5.149.1 Detailed Description

Waveform signal structure.

This structure contains one fragment of a waveform signal. The member num_frames describes the number of audio samples in each audio channel.

The field channel_info must be an array of num_channels entries or NULL.

5.149.2 Member Data Documentation

5.149.2.1 mha_real_t* mha_wave_t::buf

signal buffer

5.149.2.2 unsigned int mha_wave_t::num_channels

number of channels

5.149.2.3 unsigned int mha_wave_t::num_frames

number of frames in each channel

5.149.2.4 mha_channel_info_t* mha_wave_t::channel_info

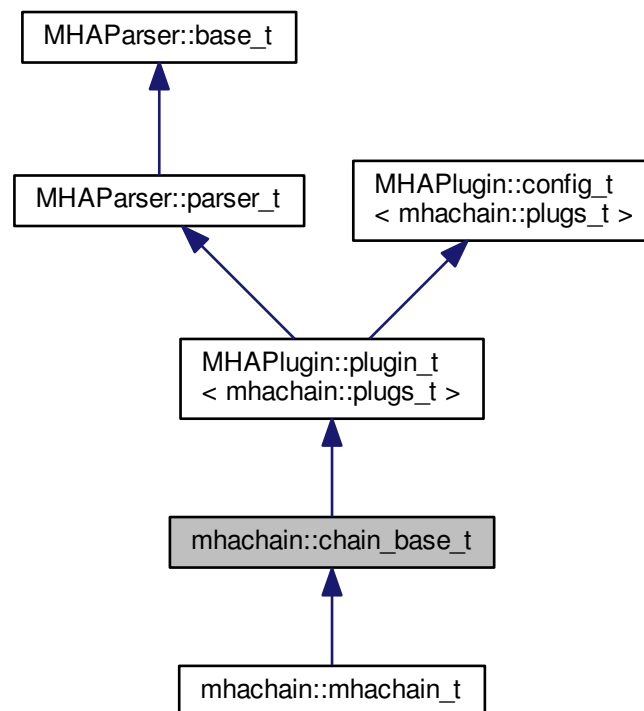
detailed channel description

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

- **mha.h**

5.150 mhachain::chain_base_t Class Reference

Inheritance diagram for mhachain::chain_base_t:



Public Member Functions

- **chain_base_t** (**algo_comm_t**, const std::string &, const std::string &)
- void **process** (**mha_wave_t** *, **mha_wave_t** **)
- void **process** (**mha_spec_t** *, **mha_wave_t** **)
- void **process** (**mha_wave_t** *, **mha_spec_t** **)
- void **process** (**mha_spec_t** *, **mha_spec_t** **)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Protected Attributes

- **MHAParser::bool_t** **bprofiling**
- **MHAParser::vstring_t** **algos**

Private Member Functions

- void **update** ()

Private Attributes

- std::vector< std::string > **old_algos**
- **MHAEvents::patchbay_t**< **mhachain::chain_base_t** > **patchbay**
- **mhaconfig_t** **cfin**
- **mhaconfig_t** **cfout**
- bool **b_prepared**
- std::string **chain**

Additional Inherited Members

5.150.1 Constructor & Destructor Documentation

5.150.1.1 **mhachain::chain_base_t::chain_base_t** (
 algo_comm_t *iac*,
 const std::string & *ichain*,
 const std::string & *ialgo*)

5.150.2 Member Function Documentation

5.150.2.1 void **mhachain::chain_base_t::process** (
 mha_wave_t * *sin*,
 mha_wave_t ** *sout*)

5.150.2.2 void mhachain::chain_base_t::process (
 mha_spec_t * sin,
 mha_wave_t ** sout)

5.150.2.3 void mhachain::chain_base_t::process (
 mha_wave_t * sin,
 mha_spec_t ** sout)

5.150.2.4 void mhachain::chain_base_t::process (
 mha_spec_t * sin,
 mha_spec_t ** sout)

5.150.2.5 void mhachain::chain_base_t::prepare (
 mhaconfig_t & cf) [virtual]

Implements **MHAPLugin::plugin_t**< **mhachain::plugs_t** > (p. 661).

5.150.2.6 void mhachain::chain_base_t::release (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t**< **mhachain::plugs_t** > (p. 661).

5.150.2.7 void mhachain::chain_base_t::update () [private]

5.150.3 Member Data Documentation

5.150.3.1 **MHAParser::bool_t** mhachain::chain_base_t::bprofiling [protected]

5.150.3.2 **MHAParser::vstring_t** mhachain::chain_base_t::algos [protected]

5.150.3.3 **std::vector<std::string>** mhachain::chain_base_t::old_algos [private]

5.150.3.4 **MHAEvents::patchbay_t**< **mhachain::chain_base_t** >
 mhachain::chain_base_t::patchbay [private]

5.150.3.5 **mhaconfig_t** mhachain::chain_base_t::cfin [private]

5.150.3.6 **mhaconfig_t** mhachain::chain_base_t::cfout [private]

5.150.3.7 **bool** mhachain::chain_base_t::b_prepared [private]

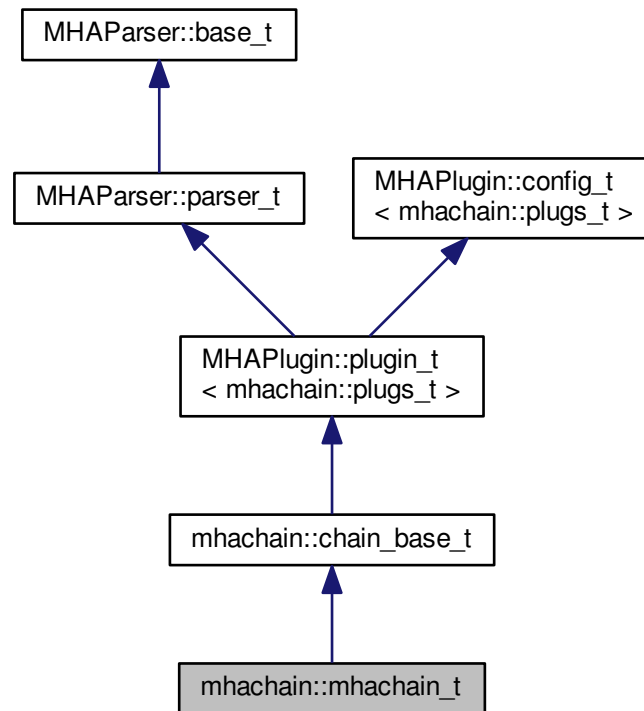
5.150.3.8 **std::string** mhachain::chain_base_t::chain [private]

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

- mha_generic_chain.h
- mha_generic_chain.cpp

5.151 mhachain::mhachain_t Class Reference

Inheritance diagram for mhachain::mhachain_t:



Public Member Functions

- **mhachain_t** (**algo_comm_t** iac, const std::string &ichain, const std::string &ialgo)

Additional Inherited Members

5.151.1 Constructor & Destructor Documentation

5.151.1.1 mhachain::mhachain_t::mhachain_t (
 algo_comm_t iac,
 const std::string & ichain,
 const std::string & ialgo)

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

- **mhachain.cpp**

5.152 mhachain::plugs_t Class Reference

Public Member Functions

- **plugs_t** (std::vector< std::string > **algos**, **mhaconfig_t** cfin, **mhaconfig_t** cfout, bool do_prepare, **MHAParser::parser_t** &p, **algo_comm_t** iac, std::string ichain, bool use_↵ profiling)
- **~plugs_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- void **process** (**mha_wave_t** *, **mha_spec_t** *, **mha_wave_t** **, **mha_spec_t** **)
- bool **prepared** () const

Private Member Functions

- void **alloc_plugs** (std::vector< std::string > **algos**)
- void **cleanup_plugs** ()
- void **update_proc_load** ()

Private Attributes

- bool **b_prepared**
- std::vector< **PluginLoader::mhapluginloader_t** * > **algos**
- **MHAParser::parser_t** & **parser**
- **algo_comm_t** **ac**
- std::string **chain**
- **MHAParser::parser_t** **profiling**
- **MHAParser::vstring_mon_t** **prof_algos**
- **MHAParser::vfloat_mon_t** **prof_init**
- **MHAParser::vfloat_mon_t** **prof_prepare**
- **MHAParser::vfloat_mon_t** **prof_release**
- **MHAParser::vfloat_mon_t** **prof_process**
- **MHAParser::float_mon_t** **prof_process_tt**
- **MHAParser::vfloat_mon_t** **prof_process_load**
- unsigned int **proc_cnt**
- **mhaconfig_t** **prof_cfg**
- **MHAEvents::connector_t**< **mhachain::plugs_t** > **prof_load_con**
- **MHAEvents::connector_t**< **mhachain::plugs_t** > **prof_tt_con**
- bool **b_use_profiling**
- **mha_platform_tictoc_t** **tictoc**

5.152.1 Constructor & Destructor Documentation

5.152.1.1 `mhachain::plugs_t::plugs_t (`
 `std::vector< std::string > algos,`
 `mhaconfig_t cfin,`
 `mhaconfig_t cfout,`
 `bool do_prepare,`
 `MHAParser::parser_t & p,`
 `algo_comm_t iac,`
 `std::string ichain,`
 `bool use_profiling)`

5.152.1.2 `mhachain::plugs_t::~~plugs_t ()`

5.152.2 Member Function Documentation

5.152.2.1 `void mhachain::plugs_t::prepare (`
 `mhaconfig_t & tf)`

5.152.2.2 `void mhachain::plugs_t::release (`
 `void)`

5.152.2.3 `void mhachain::plugs_t::process (`
 `mha_wave_t * win,`
 `mha_spec_t * sin,`
 `mha_wave_t ** wout,`
 `mha_spec_t ** sout)`

5.152.2.4 `bool mhachain::plugs_t::prepared () const` `[inline]`

5.152.2.5 `void mhachain::plugs_t::alloc_plugs (`
 `std::vector< std::string > algos)` `[private]`

5.152.2.6 `void mhachain::plugs_t::cleanup_plugs ()` `[private]`

5.152.2.7 `void mhachain::plugs_t::update_proc_load ()` `[private]`

5.152.3 Member Data Documentation

5.152.3.1 `bool mhachain::plugs_t::b_prepared` `[private]`

5.152.3.2 `std::vector< PluginLoader::mhapluginloader_t* > mhachain::plugs_t::algos`
`[private]`

5.152.3.3 `MHAParser::parser_t & mhachain::plugs_t::parser` `[private]`

- 5.152.3.4 **algo_comm_t** mhachain::plugs_t::ac [private]
- 5.152.3.5 **std::string** mhachain::plugs_t::chain [private]
- 5.152.3.6 **MHAParser::parser_t** mhachain::plugs_t::profiling [private]
- 5.152.3.7 **MHAParser::vstring_mon_t** mhachain::plugs_t::prof_algos [private]
- 5.152.3.8 **MHAParser::vfloat_mon_t** mhachain::plugs_t::prof_init [private]
- 5.152.3.9 **MHAParser::vfloat_mon_t** mhachain::plugs_t::prof_prepare [private]
- 5.152.3.10 **MHAParser::vfloat_mon_t** mhachain::plugs_t::prof_release [private]
- 5.152.3.11 **MHAParser::vfloat_mon_t** mhachain::plugs_t::prof_process [private]
- 5.152.3.12 **MHAParser::float_mon_t** mhachain::plugs_t::prof_process_tt [private]
- 5.152.3.13 **MHAParser::vfloat_mon_t** mhachain::plugs_t::prof_process_load [private]
- 5.152.3.14 **unsigned int** mhachain::plugs_t::proc_cnt [private]
- 5.152.3.15 **mhaconfig_t** mhachain::plugs_t::prof_cfg [private]
- 5.152.3.16 **MHAEvents::connector_t<mhachain::plugs_t>** mhachain::plugs_t::prof_load_con [private]
- 5.152.3.17 **MHAEvents::connector_t<mhachain::plugs_t>** mhachain::plugs_t::prof_tt_con [private]
- 5.152.3.18 **bool** mhachain::plugs_t::b_use_profiling [private]
- 5.152.3.19 **mha_platform_tictoc_t** mhachain::plugs_t::tictoc [private]

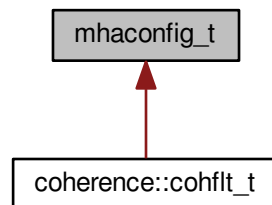
The documentation for this class was generated from the following files:

- **mha_generic_chain.h**
- **mha_generic_chain.cpp**

5.153 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 **ftlen**
FFT length of spectral data.
- **mha_real_t srate**
Sampling rate in Hz.

5.153.1 Detailed Description

MHA prepare configuration structure.

This structure contains information about channel number and domain for input and output signals of a openMHA Plugin. Each plugin can change any of these parameters, e.g. by resampling of the signal. The only limitation is that the callback frequency is fixed (except for the plugins `db` and `dbasync`).

Todo Add information on number of bands and on center frequencies, or replace by **mha_↔ audio_descriptor_t** (p. 370).

5.153.2 Member Data Documentation

5.153.2.1 unsigned int mhaconfig_t::channels

Number of audio channels.

5.153.2.2 unsigned int mhaconfig_t::domain

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.153.2.3 unsigned int mhaconfig_t::fragsize

Fragment size of waveform data.

5.153.2.4 unsigned int mhaconfig_t::wndlen

Window length of spectral data.

5.153.2.5 unsigned int mhaconfig_t::fftl

FFT length of spectral data.

5.153.2.6 mha_real_t mhaconfig_t::srate

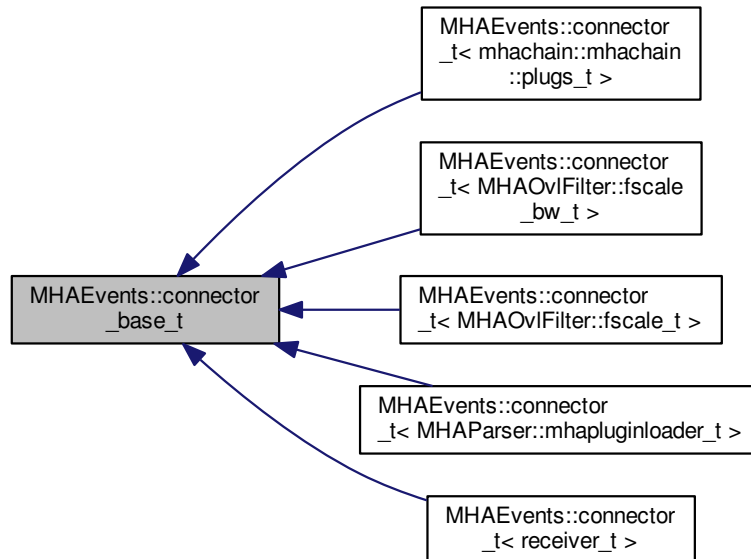
Sampling rate in Hz.

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

- **mha.h**

5.154 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**

5.154.1 Constructor & Destructor Documentation

5.154.1.1 MHAEvents::connector_base_t::connector_base_t ()

5.154.1.2 MHAEvents::connector_base_t::~~connector_base_t () [virtual]

5.154.2 Member Function Documentation

5.154.2.1 void MHAEvents::connector_base_t::emit_event () [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 449), **MHAEvents::connector_t< MHAOvIFilter::fscale_bw_t >** (p. 449), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 449), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 449), and **MHAEvents::connector_t< MHAOvIFilter::fscale_t >** (p. 449).

5.154.2.2 void MHAEvents::connector_base_t::emit_event (const std::string &) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 449), **MHAEvents::connector_t< MHAOvIFilter::fscale_bw_t >** (p. 449), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 449), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 449), and **MHAEvents::connector_t< MHAOvIFilter::fscale_t >** (p. 449).

5.154.2.3 void MHAEvents::connector_base_t::emit_event (const std::string & , unsigned int , unsigned int) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 449), **MHAEvents::connector_t< MHAOvIFilter::fscale_bw_t >** (p. 449), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 449), **MHAEvents::connector_t< mhachain::mhachain::plugs_t >** (p. 449), and **MHAEvents::connector_t< MHAOvIFilter::fscale_t >** (p. 449).

5.154.2.4 void MHAEvents::connector_base_t::emitter_die ()

5.154.3 Member Data Documentation

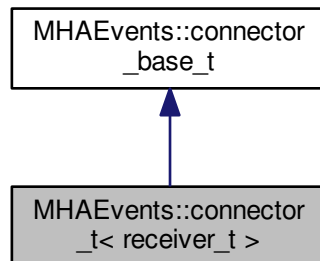
5.154.3.1 bool MHAEvents::connector_base_t::emitter_is_alive [protected]

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

- mha_event_emitter.h
- mha_events.cpp

5.155 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

5.155.1 Constructor & Destructor Documentation

5.155.1.1 `template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)() rfun)`

5.155.1.2 `template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &) rfun)`

5.155.1.3 `template<class receiver_t> MHAEvents::connector_t< receiver_t >::connector_t (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &, unsigned int, unsigned int) rfun)`

5.155.1.4 `template<class receiver_t> MHAEvents::connector_t< receiver_t >::~~connector_t ()`

5.155.2 Member Function Documentation

5.155.2.1 `template<class receiver_t> void MHAEvents::connector_t< receiver_t >::emit_event () [private],[virtual]`

Reimplemented from **MHAEvents::connector_base_t** (p. 447).

5.155.2.2 `template<class receiver_t> void MHAEvents::connector_t< receiver_t >::emit_event (const std::string & arg) [private],[virtual]`

Reimplemented from **MHAEvents::connector_base_t** (p. 447).

5.155.2.3 `template<class receiver_t> void MHAEvents::connector_t< receiver_t >::emit_event (const std::string & arg, unsigned int arg2, unsigned int arg3) [private],[virtual]`

Reimplemented from **MHAEvents::connector_base_t** (p. 447).

5.155.3 Member Data Documentation

5.155.3.1 `template<class receiver_t> emitter_t* MHAEvents::connector_t< receiver_t >::emitter [private]`

5.155.3.2 `template<class receiver_t> receiver_t* MHAEvents::connector_t< receiver_t >::receiver [private]`

5.155.3.3 `template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler) () [private]`

5.155.3.4 `template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_s) (const std::string &) [private]`

5.155.3.5 `template<class receiver_t> void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler_suu) (const std::string &, unsigned int, unsigned int) [private]`

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

- **mha_events.h**

5.156 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**

5.156.1 Detailed Description

Class for emitting openMHA events.

Use the template class **MHAEvents::patchbay_t** (p. 452) for connecting to an emitter.

5.156.2 Constructor & Destructor Documentation

5.156.2.1 MHAEvents::emitter_t::~~emitter_t ()

5.156.3 Member Function Documentation

5.156.3.1 void MHAEvents::emitter_t::operator() ()

Emit an event without parameter.

5.156.3.2 void MHAEvents::emitter_t::operator() (const std::string & arg)

Emit an event with string parameter.

5.156.3.3 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.

5.156.3.4 void MHAEvents::emitter_t::connect (connector_base_t * c)

5.156.3.5 void MHAEvents::emitter_t::disconnect (connector_base_t * c)

5.156.4 Member Data Documentation

5.156.4.1 std::list<connector_base_t*> MHAEvents::emitter_t::connections [private]

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

- mha_event_emitter.h
- mha_events.cpp

5.157 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**

5.157.1 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. 452) 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 class or structure; the event callback can be either a member function without arguments or with const std::string& argument.

5.157.2 Constructor & Destructor Documentation

5.157.2.1 `template<class receiver_t > MHAEvents::patchbay_t< receiver_t >::~~patchbay_t ()`

5.157.3 Member Function Documentation

5.157.3.1 `template<class receiver_t> void MHAEvents::patchbay_t< receiver_t >::connect (emitter_t * e, receiver_t * r, void(receiver_t::*)() rfun)`

Connect a receiver member function void (receiver_t::*)() with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

| | |
|-------------|--|
| <i>e</i> | Pointer to an event emitter |
| <i>r</i> | Pointer to the receiver |
| <i>rfun</i> | Pointer to a member function of the receiver class |

5.157.3.2 `template<class receiver_t> void MHAEvents::patchbay_t< receiver_t >::connect (emitter_t * e, receiver_t * r, void(receiver_t::*)(const std::string &) rfun)`

Connect a receiver member function void (receiver_t::*)(const std::string&) with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

| | |
|-------------|--|
| <i>e</i> | Pointer to an event emitter |
| <i>r</i> | Pointer to the receiver |
| <i>rfun</i> | Pointer to a member function of the receiver class |

5.157.3.3 `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)`

5.157.4 Member Data Documentation

5.157.4.1 `template<class receiver_t> std::list<connector_t<receiver_t>*> MHAEvents::patchbay_t< receiver_t >::cons [private]`

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

- **mha_events.h**

5.158 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**

5.158.1 Constructor & Destructor Documentation

5.158.1.1 **MHAFilter::adapt_filter_param_t::adapt_filter_param_t (**
 mha_real_t imu,
 bool ierr_in)

5.158.2 Member Data Documentation

5.158.2.1 **mha_real_t MHAFilter::adapt_filter_param_t::mu**

5.158.2.2 **bool MHAFilter::adapt_filter_param_t::err_in**

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.159 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**

5.159.1 Constructor & Destructor Documentation

5.159.1.1 MHAFilter::adapt_filter_state_t::adapt_filter_state_t (
 int *ntaps*,
 int *nchannels*)

5.159.2 Member Function Documentation

5.159.2.1 void MHAFilter::adapt_filter_state_t::filter (
 mha_wave_t *y*,
 mha_wave_t *e*,
 mha_wave_t *x*,
 mha_wave_t *d*,
 mha_real_t *mu*,
 bool *err_in*)

5.159.3 Member Data Documentation

5.159.3.1 int MHAFilter::adapt_filter_state_t::ntaps [private]

5.159.3.2 int MHAFilter::adapt_filter_state_t::nchannels [private]

5.159.3.3 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::W [private]

5.159.3.4 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::X [private]

5.159.3.5 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::od [private]

5.159.3.6 MHASignal::waveform_t MHAFilter::adapt_filter_state_t::oy [private]

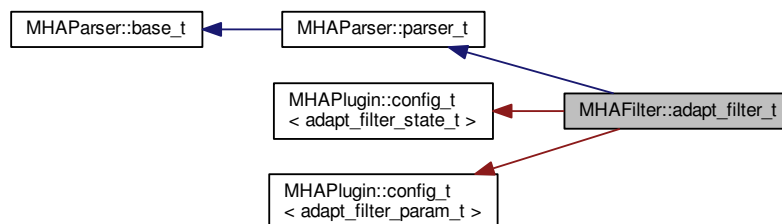
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.160 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

5.160.1 Detailed Description

Adaptive filter.

5.160.2 Constructor & Destructor Documentation

5.160.2.1 **MHAFilter::adapt_filter_t::adapt_filter_t** (
std::string *help*)

5.160.3 Member Function Documentation

5.160.3.1 void **MHAFilter::adapt_filter_t::filter** (
mha_wave_t y,
mha_wave_t e,
mha_wave_t x,
mha_wave_t d)

5.160.3.2 void **MHAFilter::adapt_filter_t::set_channelcnt** (
unsigned int *nch*)

5.160.3.3 void **MHAFilter::adapt_filter_t::update_mu** () [private]

5.160.3.4 void MHAFilter::adapt_filter_t::update_ntaps () [private]

5.160.4 Member Data Documentation

5.160.4.1 MHAParser::float_t MHAFilter::adapt_filter_t::mu [private]

5.160.4.2 MHAParser::int_t MHAFilter::adapt_filter_t::ntaps [private]

5.160.4.3 MHAParser::bool_t MHAFilter::adapt_filter_t::err_in [private]

5.160.4.4 MHAEvents::patchbay_t<adapt_filter_t> MHAFilter::adapt_filter_t::connector [private]

5.160.4.5 unsigned int MHAFilter::adapt_filter_t::nchannels [private]

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

- mha_filter.hh
- mha_filter.cpp

5.161 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)
Constructs 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**

5.161.1 Detailed Description

A class that does polyphase resampling and takes into account block processing.

5.161.2 Constructor & Destructor Documentation

5.161.2.1 `MHAFilter::blockprocessing_polyphase_resampling_t::blockprocessing_polyphase_resampling_t (`
`float source_srate,`
`unsigned source_fragsize,`
`float target_srate,`
`unsigned target_fragsize,`
`float nyquist_ratio,`
`float irslen,`
`unsigned nchannels,`
`bool add_delay)`

Constructs a polyphase resampling filter that can be used for blockprocessing with the given parameters.

Parameters

| | |
|------------------------|--|
| <i>source_srate</i> | Source sampling rate / Hz |
| <i>source_fragsize</i> | Fragment size of incoming audio blocks / frames at source_srate |
| <i>target_srate</i> | Target sampling rate / Hz |
| <i>target_fragsize</i> | Fragment size of produced audio blocks / frames at target_srate |
| <i>nyquist_ratio</i> | Low pass filter cutoff frequency relative to the nyquist frequency of the smaller of the two sampling rates. Example values: 0.8, 0.9 |
| <i>irslen</i> | Impulse response length used for low pass filtering / s |
| <i>nchannels</i> | Number of audio channels |
| <i>add_delay</i> | To avoid underruns, a delay is generally necessary for round trip block size adaptations. It is only necessary to add this delay to one of the two resampling chains. Set this parameter to true for the first resampling object of a round trip pair. It will add the necessary delay, and calculate the size of the ring buffer appropriately, When set to false, only the ringbuffer size will be set sufficiently. |

5.161.2.2 `virtual MHAFilter::blockprocessing_polyphase_resampling_t::~~blockprocessing_polyphase_resampling_t () [inline],[virtual]`

5.161.3 Member Function Documentation

5.161.3.1 `void MHAFilter::blockprocessing_polyphase_resampling_t::write (`
`mha_wave_t & signal)`

Write signal to the ringbuffer.

Parameters

| | |
|---------------|--|
| <i>signal</i> | input signal in original sampling rate |
|---------------|--|

Exceptions

| | |
|---------------------------|--|
| MHA_Error (p. 387) | Raises exception if there is not enough room, if the number of channels does not match, or if the number of frames is not equal to the number specified in the constructor |
|---------------------------|--|

5.161.3.2 void MHAFilter::blockprocessing_polyphase_resampling_t::read (mha_wave_t & *signal*)

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

| | |
|---------------|--|
| <i>signal</i> | buffer to write the resampled signal to. |
|---------------|--|

Exceptions

| | |
|---------------------------|---|
| MHA_Error (p. 387) | Raises exception if there is not enough input signal, if the number of channels of frames does not match. |
|---------------------------|---|

5.161.3.3 bool MHAFilter::blockprocessing_polyphase_resampling_t::can_read () const [inline]

Checks if the resampling ring buffer can produce another output signal block.

5.161.4 Member Data Documentation

5.161.4.1 polyphase_resampling_t* MHAFilter::blockprocessing_polyphase_resampling_t↔
::resampling [private]

5.161.4.2 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_in [private]

5.161.4.3 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_out [private]

5.161.4.4 unsigned MHAFilter::blockprocessing_polyphase_resampling_t::num_channels
[private]

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

- mha_filter.hh
- mha_filter.cpp

5.162 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**

5.162.1 Detailed Description

Complex bandpass filter.

5.162.2 Constructor & Destructor Documentation

5.162.2.1 MHAFilter::complex_bandpass_t::complex_bandpass_t (
std::vector< mha_complex_t > *A*,
std::vector< mha_complex_t > *B*)

Constructor with filter coefficients (one per channel)

Parameters

| | |
|----------|---|
| <i>A</i> | complex filter coefficients, one per band |
| <i>B</i> | complex weights |

5.162.3 Member Function Documentation

5.162.3.1 std::vector< mha_complex_t > MHAFilter::complex_bandpass_t::creator_A (
std::vector< mha_real_t > *cf*,
std::vector< mha_real_t > *bw*,
mha_real_t *srate*,
unsigned int *order*) [static]5.162.3.2 std::vector< mha_complex_t > MHAFilter::complex_bandpass_t::creator_B (
std::vector< mha_complex_t > *A*,
unsigned int *order*) [static]5.162.3.3 void MHAFilter::complex_bandpass_t::set_state (
mha_real_t *val*)5.162.3.4 void MHAFilter::complex_bandpass_t::set_state (
std::vector< mha_real_t > *val*)5.162.3.5 void MHAFilter::complex_bandpass_t::set_state (
mha_complex_t *val*)5.162.3.6 void MHAFilter::complex_bandpass_t::set_weights (
std::vector< mha_complex_t > *new_B*)

Allow to modify the input weights at a later stage.

5.162.3.7 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::get_weights () const
[inline]5.162.3.8 void MHAFilter::complex_bandpass_t::filter (
const mha_wave_t & *X*,
mha_spec_t & *Y*) [inline]

Filter method for real value input.

```
5.162.3.9 void MHAFilter::complex_bandpass_t::filter (
           const mha_wave_t & X,
           mha_wave_t & Yre,
           mha_wave_t & Yim ) [inline]
```

Filter method for real value input.

```
5.162.3.10 void MHAFilter::complex_bandpass_t::filter (
            const mha_spec_t & X,
            mha_spec_t & Y ) [inline]
```

Filter method for complex value input.

```
5.162.3.11 void MHAFilter::complex_bandpass_t::filter (
            const mha_wave_t & Xre,
            const mha_wave_t & Xim,
            mha_wave_t & Yre,
            mha_wave_t & Yim ) [inline]
```

Filter method for complex value input.

```
5.162.3.12 std::string MHAFilter::complex_bandpass_t::inspect ( ) const [inline]
```

5.162.4 Member Data Documentation

```
5.162.4.1 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::A_ [private]
```

```
5.162.4.2 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::B_ [private]
```

```
5.162.4.3 std::vector<mha_complex_t> MHAFilter::complex_bandpass_t::Yn [private]
```

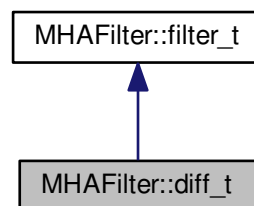
The documentation for this class was generated from the following files:

- **complex_filter.h**
- **complex_filter.cpp**

5.163 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

5.163.1 Detailed Description

Differentiator class (non-normalized)

5.163.2 Constructor & Destructor Documentation

5.163.2.1 MHAFilter::diff_t::diff_t (unsigned int *ch*)

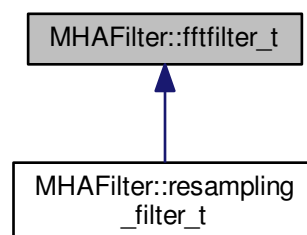
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.164 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 **fftl**)
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** *pw←IRS)
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** *ps←Weights)
Apply filter with changing coefficients to a waveform fragment.

Private Attributes

- unsigned int **fragsize**
- unsigned int **channels**
- unsigned int **fftl**
- **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

5.164.1 Detailed Description

FFT based FIR filter implementation.

The maximal number of coefficients can be FFT length - fragsize + 1.

5.164.2 Constructor & Destructor Documentation

- #### 5.164.2.1 MHAFilter::fftfilter_t::fftfilter_t (
- unsigned int *fragsize*,
unsigned int *channels*,
unsigned int *fftl*)

Constructor.

Parameters

| | |
|-----------------|---|
| <i>fragsize</i> | Number of frames expected in input signal (each cycle). |
| <i>channels</i> | Number of channels expected in input signal. |
| <i>fftlen</i> | FFT length of filter. |

5.164.2.2 MHAFilter::fftfilter_t::~~fftfilter_t ()

5.164.3 Member Function Documentation

5.164.3.1 void MHAFilter::fftfilter_t::update_coeffs (
const mha_wave_t * pwIRS)

Update the set of coefficients.

Parameters

| | |
|--------------|------------------------|
| <i>pwIRS</i> | 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).

5.164.3.2 void MHAFilter::fftfilter_t::filter (
const mha_wave_t * pwIn,
mha_wave_t ** ppwOut,
const mha_wave_t * pwIRS)

Apply filter with changing coefficients to a waveform fragment.

Parameters

| | |
|-------------|-----------------------|
| <i>pwIn</i> | Input signal pointer. |
|-------------|-----------------------|

Return values

| | |
|---------------|--|
| <i>ppwOut</i> | Pointer to output signal pointer, will be set to a valid signal. |
|---------------|--|

Parameters

| | |
|--------------|--|
| <i>pwIRS</i> | Pointer to FIR coefficients structure. |
|--------------|--|


```
5.164.3.3 void MHAFilter::fftfilter_t::filter (
            const mha_wave_t * pwIn,
            mha_wave_t ** ppwOut )
```

Apply filter to waveform fragment, without changing the coefficients.

Parameters

| | |
|-------------|-----------------------|
| <i>pwIn</i> | Input signal pointer. |
|-------------|-----------------------|

Return values

| | |
|---------------|---|
| <i>ppwOut</i> | Pointer to output signal pointer, will be set to a valid signal |
|---------------|---|

```
5.164.3.4 void MHAFilter::fftfilter_t::filter (
            const mha_wave_t * pwIn,
            mha_wave_t ** ppwOut,
            const mha_spec_t * psWeights )
```

Apply filter with changing coefficients to a waveform fragment.

Parameters

| | |
|-------------|-----------------------|
| <i>pwIn</i> | Input signal pointer. |
|-------------|-----------------------|

Return values

| | |
|---------------|--|
| <i>ppwOut</i> | Pointer to output signal pointer, will be set to a valid signal. |
|---------------|--|

Parameters

| | |
|------------------|--------------------------------------|
| <i>psWeights</i> | Pointer to filter weights structure. |
|------------------|--------------------------------------|

5.164.4 Member Data Documentation

```
5.164.4.1 unsigned int MHAFilter::fftfilter_t::fragsize [private]
```

```
5.164.4.2 unsigned int MHAFilter::fftfilter_t::channels [private]
```

```
5.164.4.3 unsigned int MHAFilter::fftfilter_t::ftflen [private]
```

- 5.164.4.4 MHASignal::waveform_t MHAFilter::fftfilter_t::wInput_fft [private]
- 5.164.4.5 mha_wave_t MHAFilter::fftfilter_t::wInput [private]
- 5.164.4.6 MHASignal::waveform_t MHAFilter::fftfilter_t::wOutput_fft [private]
- 5.164.4.7 mha_wave_t MHAFilter::fftfilter_t::wOutput [private]
- 5.164.4.8 MHASignal::spectrum_t MHAFilter::fftfilter_t::sInput [private]
- 5.164.4.9 MHASignal::spectrum_t MHAFilter::fftfilter_t::sWeights [private]
- 5.164.4.10 MHASignal::waveform_t MHAFilter::fftfilter_t::wIRS_fft [private]
- 5.164.4.11 mha_fft_t MHAFilter::fftfilter_t::fft [private]

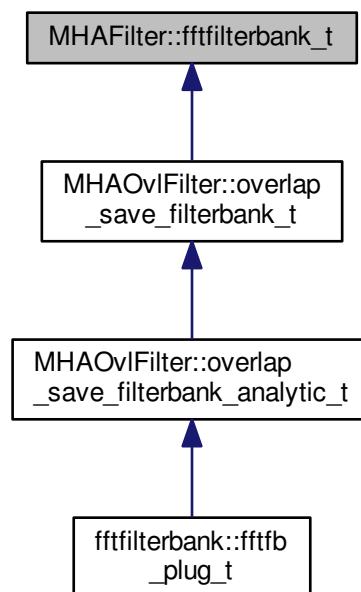
The documentation for this class was generated from the following files:

- mha_filter.hh
- mha_filter.cpp

5.165 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 **fftl**)
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 **fftl**
- **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

5.165.1 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.

5.165.2 Constructor & Destructor Documentation

- ##### 5.165.2.1 MHASignal::fftfilterbank_t::fftfilterbank_t (
- unsigned int *fragsize*,
 unsigned int *inputchannels*,
 unsigned int *firchannels*,
 unsigned int *fftl*)

Constructor.

Parameters

| | |
|----------------------|---|
| <i>fragsize</i> | Number of frames expected in input signal (each cycle). |
| <i>inputchannels</i> | Number of channels expected in input signal. |
| <i>firchannels</i> | Number of channels expected in FIR filter coefficients (= number of bands). |
| <i>fftlen</i> | FFT length of filter. |

The number of output channels is `inputchannels*firchannels`.

5.165.2.2 MHAFilter::fftfilterbank_t::~fftfilterbank_t ()

5.165.3 Member Function Documentation

5.165.3.1 void MHAFilter::fftfilterbank_t::update_coeffs (
const mha_wave_t * *h*)

Update the set of coefficients.

Parameters

| | |
|----------|------------------------|
| <i>h</i> | 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`.

5.165.3.2 void MHAFilter::fftfilterbank_t::filter (
const mha_wave_t * *s_in*,
mha_wave_t ** *s_out*,
const mha_wave_t * *h*)

Apply filter with changing coefficients to a waveform fragment.

Parameters

| | |
|-------------|-----------------------|
| <i>s_in</i> | Input signal pointer. |
|-------------|-----------------------|

Return values

| | |
|--------------|---|
| <i>s_out</i> | Pointer to output signal pointer, will be set to a valid signal |
|--------------|---|

Parameters

| | |
|----------|------------------|
| <i>h</i> | FIR coefficients |
|----------|------------------|

5.165.3.3 void MHAFilter::fftfilterbank_t::filter (
const mha_wave_t * *s_in*,
mha_wave_t ** *s_out*)

Apply filter to waveform fragment, without changing the coefficients.

Parameters

| | |
|--------------------------|-----------------------|
| <i>s</i> ↔ <i>_in</i> | Input signal pointer. |
|--------------------------|-----------------------|

Return values

| | |
|--------------|---|
| <i>s_out</i> | Pointer to output signal pointer, will be set to a valid signal |
|--------------|---|

5.165.3.4 const mha_wave_t* MHAFilter::fftfilterbank_t::get_irs () const [inline]

Return the current IRS.

5.165.4 Member Data Documentation

5.165.4.1 unsigned int MHAFilter::fftfilterbank_t::fragsize [private]

5.165.4.2 unsigned int MHAFilter::fftfilterbank_t::inputchannels [private]

5.165.4.3 unsigned int MHAFilter::fftfilterbank_t::firchannels [private]

5.165.4.4 unsigned int MHAFilter::fftfilterbank_t::outputchannels [private]

5.165.4.5 unsigned int MHAFilter::fftfilterbank_t::fftlen [private]

5.165.4.6 MHASignal::waveform_t MHAFilter::fftfilterbank_t::hw [private]

5.165.4.7 MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Hs [private]

5.165.4.8 MHASignal::waveform_t MHAFilter::fftfilterbank_t::xw [private]

5.165.4.9 MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Xs [private]

- 5.165.4.10 MHAFilter::fftfilterbank_t::yw [private]
- 5.165.4.11 MHAFilter::fftfilterbank_t::Ys [private]
- 5.165.4.12 MHAFilter::fftfilterbank_t::yw_temp [private]
- 5.165.4.13 MHAFilter::fftfilterbank_t::tail [private]
- 5.165.4.14 mha_fft_t MHAFilter::fftfilterbank_t::fft [private]

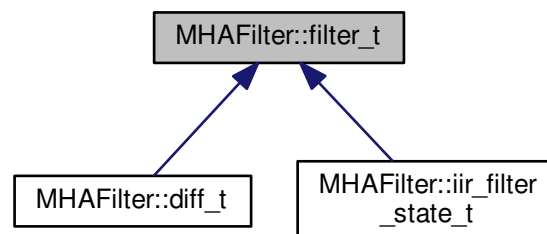
The documentation for this class was generated from the following files:

- mha_filter.hh
- mha_filter.cpp

5.166 MHAFilter::filter_t Class Reference

Generic IIR filter class.

Inheritance diagram for MHAFilter::filter_t:



Public Member Functions

- **filter_t** (unsigned int ch, unsigned int lena, unsigned int lenb)
Constructor.
- **filter_t** (unsigned int ch, const std::vector< **mha_real_t** > &vA, const std::vector< **mha_real_t** > &vB)
Constructor with initialization of coefficients.
- **filter_t** (const **MHAFilter::filter_t** &src)
- **~filter_t** ()
- void **filter** (**mha_wave_t** *out, const **mha_wave_t** *in)

Filter all channels in a waveform structure.

- void **filter** (**mha_real_t** *dest, const **mha_real_t** *src, unsigned int dframes, unsigned int frame_dist, unsigned int channel_dist, unsigned int channel_begin, unsigned int channel_end)

Filter parts of a waveform structure.

- **mha_real_t filter** (**mha_real_t** x, unsigned int ch)

Filter one sample.

- unsigned int **get_len_A** () const

Return length of recursive coefficients.

- unsigned int **get_len_B** () const

Return length of non-recursive coefficients.

Public Attributes

- double * **A**

Pointer to recursive coefficients.

- double * **B**

Pointer to non-recursive coefficients.

Private Attributes

- unsigned int **len_A**
- unsigned int **len_B**
- unsigned int **len**
- unsigned int **channels**
- double * **state**

5.166.1 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. 436) structs. The filter coefficients can be directly accessed.

Todo Implement a more robust filter form.

5.166.2 Constructor & Destructor Documentation

5.166.2.1 **MHAFilter::filter_t::filter_t** (
 unsigned int *ch*,
 unsigned int *lena*,
 unsigned int *lenb*)

Constructor.

Parameters

| | |
|-------------|--------------------------------------|
| <i>ch</i> | Number of channels |
| <i>lena</i> | Number of recursive coefficients |
| <i>lenb</i> | Number of non-recursive coefficients |

5.166.2.2 MHAFilter::filter_t::filter_t (
 unsigned int *ch*,
 const std::vector< mha_real_t > & *vA*,
 const std::vector< mha_real_t > & *vB*)

Constructor with initialization of coefficients.

Parameters

| | |
|-----------|-----------------------------|
| <i>ch</i> | Number of channels. |
| <i>vA</i> | Recursive coefficients. |
| <i>vB</i> | Non-recursive coefficients. |

5.166.2.3 MHAFilter::filter_t::filter_t (
 const MHAFilter::filter_t & *src*)

5.166.2.4 MHAFilter::filter_t::~~filter_t ()

5.166.3 Member Function Documentation

5.166.3.1 void MHAFilter::filter_t::filter (
 mha_wave_t * *out*,
 const mha_wave_t * *in*)

Filter all channels in a waveform structure.

Parameters

| | |
|------------|---------------|
| <i>out</i> | Output signal |
| <i>in</i> | Input signal |


```

5.166.3.2 void MHAFilter::filter_t::filter (
            mha_real_t * dest,
            const mha_real_t * src,
            unsigned int dframes,
            unsigned int frame_dist,
            unsigned int channel_dist,
            unsigned int channel_begin,
            unsigned int channel_end )

```

Filter parts of a waveform structure.

Parameters

| | |
|----------------------|--|
| <i>dest</i> | Output signal. |
| <i>src</i> | Input signal. |
| <i>dframes</i> | Number of frames to be filtered. |
| <i>frame_dist</i> | Index distance between frames of one channel |
| <i>channel_dist</i> | Index distance between audio channels |
| <i>channel_begin</i> | Number of first channel to be processed |
| <i>channel_end</i> | Number of last channel to be processed |

```

5.166.3.3 mha_real_t MHAFilter::filter_t::filter (
            mha_real_t x,
            unsigned int ch )

```

Filter one sample.

Parameters

| | |
|-----------|---------------------------------------|
| <i>x</i> | Input value |
| <i>ch</i> | Channel number to use in filter state |

```

5.166.3.4 unsigned int MHAFilter::filter_t::get_len_A ( ) const [inline]

```

Return length of recursive coefficients.

```

5.166.3.5 unsigned int MHAFilter::filter_t::get_len_B ( ) const [inline]

```

Return length of non-recursive coefficients.

5.166.4 Member Data Documentation

```

5.166.4.1 double* MHAFilter::filter_t::A

```

Pointer to recursive coefficients.

5.166.4.2 double* MHAFilter::filter_t::B

Pointer to non-recursive coefficients.

5.166.4.3 unsigned int MHAFilter::filter_t::len_A [private]

5.166.4.4 unsigned int MHAFilter::filter_t::len_B [private]

5.166.4.5 unsigned int MHAFilter::filter_t::len [private]

5.166.4.6 unsigned int MHAFilter::filter_t::channels [private]

5.166.4.7 double* MHAFilter::filter_t::state [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.167 MHAFilter::gammaflt_t Class Reference

Class for gammatone filter.

Public Member Functions

- **gammaflt_t** (std::vector< **mha_real_t** > cf, std::vector< **mha_real_t** > bw, **mha_real_t** state, unsigned int order)
Constructor.
- **~gammaflt_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_`

5.167.1 Detailed Description

Class for gammatone filter.

5.167.2 Constructor & Destructor Documentation

5.167.2.1 `MHAFilter::gammaflt_t::gammaflt_t (`
 `std::vector< mha_real_t > cf,`
 `std::vector< mha_real_t > bw,`
 `mha_real_t srate,`
 `unsigned int order)`

Constructor.

Parameters

| | |
|--------------|--|
| <i>cf</i> | Center frequency in Hz. |
| <i>bw</i> | Bandwidth in Hz (same number of entries as in cf). |
| <i>srate</i> | Sampling frequency in Hz. |
| <i>order</i> | Filter order. |

5.167.2.2 `MHAFilter::gammaflt_t::~~gammaflt_t ()`

5.167.3 Member Function Documentation

5.167.3.1 `void MHAFilter::gammaflt_t::operator() (`
 `mha_wave_t & X,`
 `mha_spec_t & Y) [inline]`

Filter method.

```
5.167.3.2 void MHAFilter::gammaflt_t::operator() (
    mha_wave_t & X,
    mha_wave_t & Yre,
    mha_wave_t & Yim ) [inline]
```

Filter method.

```
5.167.3.3 void MHAFilter::gammaflt_t::operator() (
    mha_wave_t & Yre,
    mha_wave_t & Yim,
    unsigned int stage ) [inline]
```

Filter method for specific stage.

```
5.167.3.4 void MHAFilter::gammaflt_t::phase_correction (
    unsigned int desired_delay,
    unsigned int inchannels )
```

```
5.167.3.5 void MHAFilter::gammaflt_t::set_weights (
    std::vector<mha_complex_t> new_B )
```

```
5.167.3.6 void MHAFilter::gammaflt_t::set_weights (
    unsigned int stage,
    std::vector<mha_complex_t> new_B )
```

```
5.167.3.7 std::vector<mha_complex_t> MHAFilter::gammaflt_t::get_weights ( ) const
[inline]
```

```
5.167.3.8 std::vector<mha_complex_t> MHAFilter::gammaflt_t::get_weights (
    unsigned int stage ) const [inline]
```

```
5.167.3.9 std::vector<mha_real_t> MHAFilter::gammaflt_t::get_resynthesis_gain ( ) const
[inline]
```

```
5.167.3.10 void MHAFilter::gammaflt_t::reset_state ( )
```

```
5.167.3.11 const std::vector<mha_complex_t>& MHAFilter::gammaflt_t::get_A ( ) [inline]
```

```
5.167.3.12 std::string MHAFilter::gammaflt_t::inspect ( ) const [inline]
```

5.167.4 Member Data Documentation

```
5.167.4.1 std::vector<mha_complex_t> MHAFilter::gammaflt_t::A [private]
```

```
5.167.4.2 std::vector<complex_bandpass_t> MHAFilter::gammaflt_t::GF [private]
```

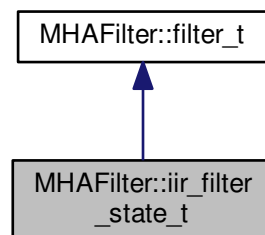
- 5.167.4.3 `MHASignal::delay_t* MHAFilter::gammaflt_t::delay` [private]
- 5.167.4.4 `std::vector<int> MHAFilter::gammaflt_t::envelope_delay` [private]
- 5.167.4.5 `std::vector<mha_real_t> MHAFilter::gammaflt_t::resynthesis_gain` [private]
- 5.167.4.6 `std::vector<mha_real_t> MHAFilter::gammaflt_t::cf_` [private]
- 5.167.4.7 `std::vector<mha_real_t> MHAFilter::gammaflt_t::bw_` [private]
- 5.167.4.8 `mha_real_t MHAFilter::gammaflt_t::srate_` [private]

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

- `complex_filter.h`
- `complex_filter.cpp`

5.168 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

5.168.1 Constructor & Destructor Documentation

5.168.1.1 MHAFilter::iir_filter_state_t::iir_filter_state_t (
 unsigned int *channels*,
 std::vector< float > *cf_A*,
 std::vector< float > *cf_B*)

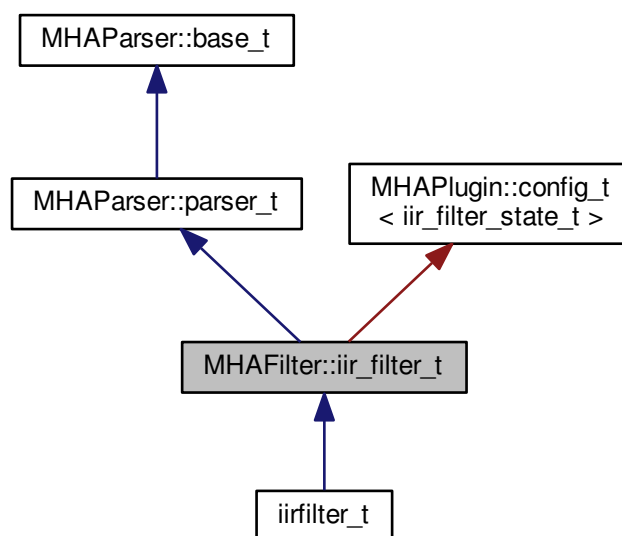
The documentation for this class was generated from the following files:

- mha_filter.hh
- mha_filter.cpp

5.169 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

5.169.1 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. 620), 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.

5.169.2 Constructor & Destructor Documentation

5.169.2.1 **MHAFilter::iir_filter_t::iir_filter_t** (
 std::string **help** = "IIR filter structure",
 std::string **def_A** = "[1]",
 std::string **def_B** = "[1]",
 unsigned int **channels** = 1)

Constructor of the IIR filter.

Initialises the sub-parser structure and the memory for holding the filter's state.

Parameters

| | |
|-----------------|--|
| <i>help</i> | 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. |
| <i>def_A</i> | The initial value of the vector of the recursive filter coefficients, represented as string. |
| <i>def_B</i> | The initial value of the vector of the non-recursive filter coefficients, represented as string. |
| <i>channels</i> | The number of independent audio channels to process with this filter. Needed to allocate a state vector for each audio channel. |

5.169.3 Member Function Documentation

5.169.3.1 void MHAFilter::iir_filter_t::filter (mha_wave_t * y, const mha_wave_t * x)

The filter processes the audio signal.

All channels in the audio signal are processed using the same filter coefficients. Independent state is stored between calls for each audio channel.

Parameters

| | |
|----------|--|
| <i>y</i> | 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. |
| <i>x</i> | Pointer to input signal holder. Number of channels has to be the same as given to the constructor, or to the resize (p. 482) method. |

5.169.3.2 mha_real_t MHAFilter::iir_filter_t::filter (mha_real_t x, unsigned int ch)

Filter a single audio sample.

Parameters

| | |
|-----------|--|
| <i>x</i> | The single audio sample |
| <i>ch</i> | 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. 482) method. |

Returns

the filtered result sample.

5.169.3.3 void MHAFilter::iir_filter_t::resize (
 unsigned int *channels*)

Change the number of channels after object creation.

Parameters

| | |
|-----------------|---|
| <i>channels</i> | The new number of channels. Old filter states are lost. |
|-----------------|---|

5.169.3.4 void MHAFilter::iir_filter_t::update_filter () [private]

5.169.4 Member Data Documentation

5.169.4.1 MHAParser::vfloat_t MHAFilter::iir_filter_t::A [private]

5.169.4.2 MHAParser::vfloat_t MHAFilter::iir_filter_t::B [private]

5.169.4.3 MHAEvents::patchbay_t<iir_filter_t> MHAFilter::iir_filter_t::connector [private]

5.169.4.4 unsigned int MHAFilter::iir_filter_t::nchannels [private]

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

- mha_filter.hh
- mha_filter.cpp

5.170 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**

5.170.1 Detailed Description

First order recursive filter.

5.170.2 Constructor & Destructor Documentation

5.170.2.1 MHAFilter::iir_ord1_real_t::iir_ord1_real_t (
std::vector< **mha_real_t** > A,
std::vector< **mha_real_t** > B)

Constructor with filter coefficients (one per channel)

5.170.2.2 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)

5.170.3 Member Function Documentation

5.170.3.1 `void MHAFilter::iir_ord1_real_t::set_state (mha_real_t val)`

5.170.3.2 `void MHAFilter::iir_ord1_real_t::set_state (std::vector< mha_real_t > val)`

5.170.3.3 `void MHAFilter::iir_ord1_real_t::set_state (mha_complex_t val)`

5.170.3.4 `mha_real_t MHAFilter::iir_ord1_real_t::operator() (unsigned int ch, mha_real_t x) [inline]`

Filter method for real value input, one element.

5.170.3.5 `mha_complex_t MHAFilter::iir_ord1_real_t::operator() (unsigned int ch, mha_complex_t x) [inline]`

Filter method for complex input, one element.

5.170.3.6 `void MHAFilter::iir_ord1_real_t::operator() (const mha_wave_t & X, mha_wave_t & Y) [inline]`

Filter method for real value input.

5.170.3.7 `void MHAFilter::iir_ord1_real_t::operator() (const mha_spec_t & X, mha_spec_t & Y) [inline]`

Filter method for complex value input.

5.170.3.8 `void MHAFilter::iir_ord1_real_t::operator() (const mha_wave_t & Xre, const mha_wave_t & Xim, mha_wave_t & Yre, mha_wave_t & Yim) [inline]`

Filter method for complex value input.

5.170.4 Member Data Documentation

5.170.4.1 `std::vector<mha_real_t> MHAFilter::iir_ord1_real_t::A_` [private]

5.170.4.2 `std::vector<mha_real_t> MHAFilter::iir_ord1_real_t::B_` [private]

5.170.4.3 `std::vector<mha_complex_t> MHAFilter::iir_ord1_real_t::Yn` [private]

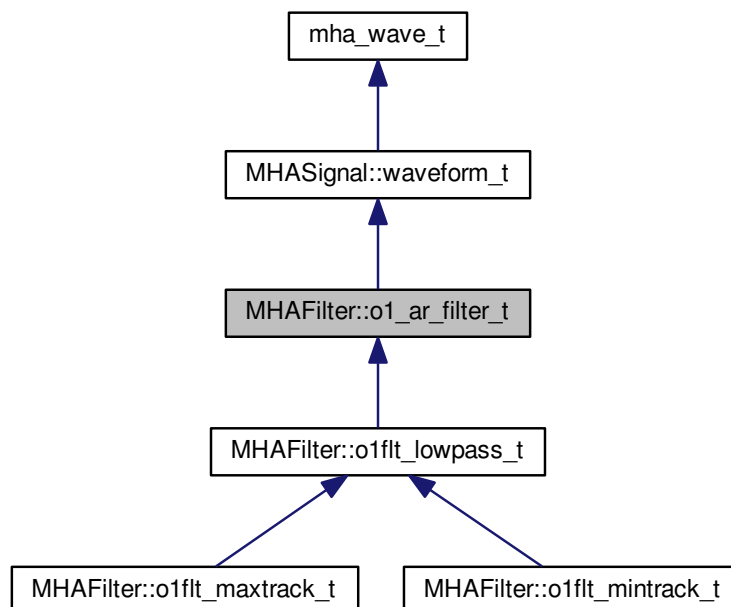
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

5.171 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. 436) 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

5.171.1 Detailed Description

First order attack-release lowpass filter.

This filter is the base of first order lowpass filter, maximum tracker and minimum tracker.

5.171.2 Constructor & Destructor Documentation

5.171.2.1 MHAFilter::o1_ar_filter_t::o1_ar_filter_t (
 unsigned int *channels*,
 mha_real_t *fs* = 1.0f,
 std::vector< **mha_real_t** > *tau_a* = std::vector<float>(1, 0.0f),
 std::vector< **mha_real_t** > *tau_r* = std::vector<float>(1, 0.0f))

Constructor, setting all taus to zero.

The filter state can be accessed through the member functions of **MHASignal::waveform_t** (p. 743).

Parameters

| | |
|-----------------|--|
| <i>channels</i> | Number of independent filters |
| <i>fs</i> | Sampling rate (optional, default = 1) |
| <i>tau_a</i> | Attack time constants (optional, default = 0) |
| <i>tau_r</i> | Release time constants (optional, default = 0) |

5.171.3 Member Function Documentation

5.171.3.1 void MHAFilter::o1_ar_filter_t::set_tau_attack (
 unsigned int *ch*,
 mha_real_t *tau*)

Set the attack time constant.

Parameters

| | |
|------------|----------------|
| <i>ch</i> | Channel number |
| <i>tau</i> | Time constant |

5.171.3.2 void MHAFilter::o1_ar_filter_t::set_tau_release (
 unsigned int *ch*,
 mha_real_t *tau*)

Set the release time constant.

Parameters

| | |
|------------|----------------|
| <i>ch</i> | Channel number |
| <i>tau</i> | Time constant |

5.171.3.3 mha_real_t MHAFilter::o1_ar_filter_t::operator() (
 unsigned int *ch*,
 mha_real_t *x*) [inline]

Apply filter to value *x*, using state channel *ch*.

Parameters

| | |
|-----------|----------------|
| <i>ch</i> | Channel number |
| <i>x</i> | Input value |

Returns

Output value

```
5.171.3.4 void MHAFilter::o1_ar_filter_t::operator() (
            const mha_wave_t & in,
            mha_wave_t & out ) [inline]
```

Apply filter to a **mha_wave_t** (p. 436) data.

Parameters

| | |
|------------|---------------|
| <i>in</i> | Input signal |
| <i>out</i> | Output signal |

The number of channels must match the number of filter bands.

5.171.4 Member Data Documentation

5.171.4.1 **MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c1_a** [protected]

5.171.4.2 **MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c2_a** [protected]

5.171.4.3 **MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c1_r** [protected]

5.171.4.4 **MHASignal::waveform_t MHAFilter::o1_ar_filter_t::c2_r** [protected]

5.171.4.5 **mha_real_t MHAFilter::o1_ar_filter_t::fs** [protected]

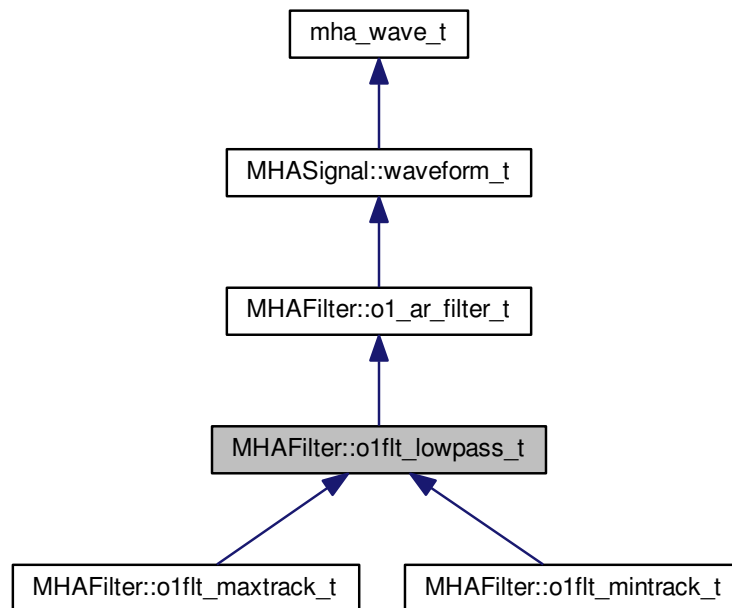
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.172 MHAFilter::o1flt_lowpass_t Class Reference

First order low pass filter.

Inheritance diagram for MHAFilter::o1flt_lowpass_t:



Public Member Functions

- **o1flt_lowpass_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
Constructor of low pass filter, sets sampling rate and time constants.
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
set time constant in all channels to tau
- **mha_real_t** **get_c1** (unsigned int ch) const
- **mha_real_t** **get_last_output** (unsigned int ch) const

Additional Inherited Members

5.172.1 Detailed Description

First order low pass filter.

5.172.2 Constructor & Destructor Documentation

5.172.2.1 `MHAFilter::o1flt_lowpass_t::o1flt_lowpass_t (`
`const std::vector< mha_real_t > & tau,`
`mha_real_t fs,`
`mha_real_t startval = 0)`

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

| | |
|-----------------|------------------------------|
| <i>tau</i> | Vector of time constants |
| <i>fs</i> | Sampling rate |
| <i>startval</i> | Initial internal state value |

5.172.3 Member Function Documentation

5.172.3.1 `void MHAFilter::o1flt_lowpass_t::set_tau (`
`unsigned int ch,`
`mha_real_t tau)`

change the time constant in one channel

5.172.3.2 `void MHAFilter::o1flt_lowpass_t::set_tau (`
`mha_real_t tau)`

set time constant in all channels to tau

5.172.3.3 `mha_real_t MHAFilter::o1flt_lowpass_t::get_c1 (`
`unsigned int ch) const` `[inline]`

5.172.3.4 `mha_real_t MHAFilter::o1flt_lowpass_t::get_last_output (`
`unsigned int ch) const` `[inline]`

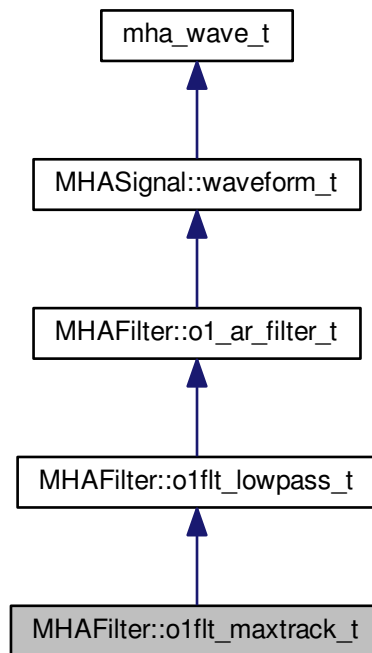
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

5.173 MHAFilter::o1flt_maxtrack_t Class Reference

First order maximum tracker.

Inheritance diagram for MHAFilter::o1flt_maxtrack_t:



Public Member Functions

- **o1flt_maxtrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
Constructor of low pass filter, sets sampling rate and time constants.
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
set time constant in all channels to tau

Additional Inherited Members

5.173.1 Detailed Description

First order maximum tracker.

5.173.2 Constructor & Destructor Documentation

5.173.2.1 `MHAFilter::o1flt_maxtrack_t::o1flt_maxtrack_t (`
`const std::vector< mha_real_t > & tau,`
`mha_real_t fs,`
`mha_real_t startval = 0)`

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

| | |
|-----------------|------------------------------|
| <i>tau</i> | Vector of time constants |
| <i>fs</i> | Sampling rate |
| <i>startval</i> | Initial internal state value |

5.173.3 Member Function Documentation

5.173.3.1 `void MHAFilter::o1flt_maxtrack_t::set_tau (`
`unsigned int ch,`
`mha_real_t tau)`

change the time constant in one channel

5.173.3.2 `void MHAFilter::o1flt_maxtrack_t::set_tau (`
`mha_real_t tau)`

set time constant in all channels to tau

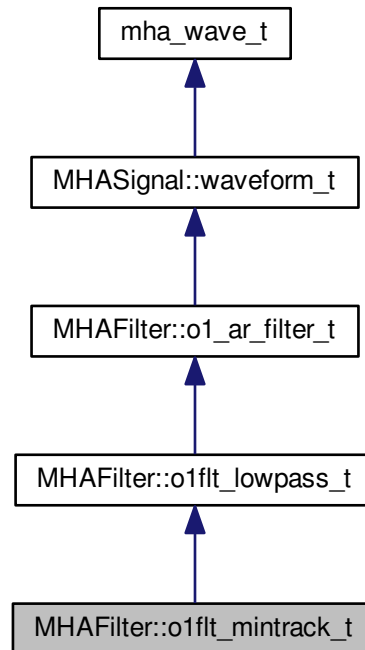
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.174 MHAFilter::o1flt_mintrack_t Class Reference

First order minimum tracker.

Inheritance diagram for MHAFilter::o1flt_mintrack_t:



Public Member Functions

- **o1flt_mintrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
set time constant in all channels to tau

Additional Inherited Members

5.174.1 Detailed Description

First order minimum tracker.

5.174.2 Constructor & Destructor Documentation

5.174.2.1 `MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (`
 `const std::vector< mha_real_t > & tau,`
 `mha_real_t fs,`
 `mha_real_t startval = 0)`

5.174.3 Member Function Documentation

5.174.3.1 `void MHAFilter::o1flt_mintrack_t::set_tau (`
 `unsigned int ch,`
 `mha_real_t tau)`

change the time constant in one channel

5.174.3.2 `void MHAFilter::o1flt_mintrack_t::set_tau (`
 `mha_real_t tau)`

set time constant in all channels to tau

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

- `mha_filter.hh`
- `mha_filter.cpp`

5.175 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.

5.175.1 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.

5.175.2 Constructor & Destructor Documentation

5.175.2.1 MHAFilter::partitioned_convolution_t::partitioned_convolution_t (
 unsigned int *fragsize*,
 unsigned int *nchannels_in*,
 unsigned int *nchannels_out*,
 const transfer_matrix_t & *transfer*)

Create a new partitioned convolver.

Parameters

| | |
|----------------------|---|
| <i>fragsize</i> | Audio fragment size, equal to partition size. |
| <i>nchannels_in</i> | Number of input audio channels. |
| <i>nchannels_out</i> | Number of output audio channels. |
| <i>transfer</i> | A sparse matrix of impulse responses. |

5.175.2.2 MHAFilter::partitioned_convolution_t::~~partitioned_convolution_t ()

Free fftw resource allocated in constructor.

5.175.3 Member Function Documentation**5.175.3.1 mha_wave_t * MHAFilter::partitioned_convolution_t::process (const mha_wave_t * s_in)**

processing

5.175.4 Member Data Documentation**5.175.4.1 unsigned int MHAFilter::partitioned_convolution_t::fragsize**

Audio fragment size, always equal to partition size.

5.175.4.2 unsigned int MHAFilter::partitioned_convolution_t::nchannels_in

Number of audio input channels.

5.175.4.3 unsigned int MHAFilter::partitioned_convolution_t::nchannels_out

Number of audio output channels.

5.175.4.4 unsigned int MHAFilter::partitioned_convolution_t::output_partitions

The maximum number of partitions in any of the impulse responses.

Determines the size of the delay line.

5.175.4.5 unsigned int MHAFilter::partitioned_convolution_t::filter_partitions

The total number of non-zero impulse response partitions.

5.175.4.6 MHASignal::waveform_t MHAFilter::partitioned_convolution_t::input_signal_wave

Buffer for input signal.

Has nchannels_in channels and fragsize*2 frames

5.175.4.7 unsigned int MHAFilter::partitioned_convolution_t::current_input_signal_buffer_half_index

A counter modulo 2.

Indicates the buffer half in input signal wave into which to copy the current input signal.

5.175.4.8 MHASignal::spectrum_t MHAFilter::partitioned_convolution_t::input_signal_spec

Buffer for FFT transformed input signal.

Has nchannels_in channels and fragsize+1 frames (fft bins).

5.175.4.9 MHASignal::spectrum_t MHAFilter::partitioned_convolution_t::frequency_response

Buffers for frequency response spectra of impulse response partitions.

Each "channel" contains another partition of some impulse response. The bookkeeping array is used to keep track what to do with these frequency responses. This container has filter_partitions channels and fragsize+1 frames (fft bins).

5.175.4.10 std::vector<index_t> MHAFilter::partitioned_convolution_t::bookkeeping

Keeps track of input channels, output channels, impulse response partition, and delay.

The index into this array is the same as the "channel" index into the frequency_response array. Array has filter_partitions entries.

5.175.4.11 std::vector<MHASignal::spectrum_t> MHAFilter::partitioned_convolution_t::output_signal_spec

Buffers for FFT transformed output signal.

For each array member, Number of channels is equal to nchannels_out, number of frames (fft bins) is equal to fragsize+1. Array size is equal to output_partitions.

5.175.4.12 unsigned int MHAFilter::partitioned_convolution_t::current_output_partition_index

A counter modulo output_partitions, indexing the "current" output partition.

5.175.4.13 MHASignal::waveform_t MHAFilter::partitioned_convolution_t::output_signal_wave

Buffer for the wave output signal.

Number of channels is equal to nchannels_out, number of frames is equal to fragsize

5.175.4.14 mha_fft_t MHAFilter::partitioned_convolution_t::fft

The FFT transformer.

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.176 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.

5.176.1 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 impulse response partitions.

5.176.2 Constructor & Destructor Documentation

5.176.2.1 MHAFilter::partitioned_convolution_t::index_t::index_t(
 unsigned int src,
 unsigned int tgt,
 unsigned int dly) [inline]

Data constructor.

Parameters

| | |
|------------|---|
| <i>src</i> | The input channel index to apply the current partition to. |
| <i>tgt</i> | The index of the output channel to which the filter result should go. |
| <i>dly</i> | The delay (in blocks) of this partition |

5.176.2.2 MHAFilter::partitioned_convolution_t::index_t::index_t() [inline]

Default constructor for STL compatibility.

5.176.3 Member Data Documentation

5.176.3.1 unsigned int MHAFilter::partitioned_convolution_t::index_t::source_channel_index

The input channel index to apply the current partition to.

5.176.3.2 unsigned int MHAFilter::partitioned_convolution_t::index_t::target_channel_index

The index of the output channel to which the filter result should go.

5.176.3.3 unsigned int MHAFilter::partitioned_convolution_t::index_t::delay

The delay (in blocks) of this partition.

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

- **mha_filter.hh**

5.177 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.

5.177.1 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 possibilities 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.

5.177.2 Constructor & Destructor Documentation

5.177.2.1 MHAFilter::polyphase_resampling_t::polyphase_resampling_t (

```
    unsigned n_up,
    unsigned n_down,
    mha_real_t nyquist_ratio,
    unsigned n_irs,
    unsigned n_ringbuffer,
    unsigned n_channels,
    unsigned n_prefill )
```

Construct a polyphase resampler instance.

Allocates a ringbuffer with the given capacity *n_ringbuffer*. Client that triggers the constructor must ensure that the capacity *n_ringbuffer* and the delay *n_prefill* are sufficient, i.e. enough old and new samples are always available to compute sufficient samples in using an impulse response of length *n_irs*. Audio block sizes at both sides of the resampler have to be taken into account. Class `MHASignal::blockprocessing_polyphase_resampling_t` takes care of this, and it is recommended to use this class for block-based processing.

Based on *n_up*, *n_down*, *n_irs* and *nyquist_ratio*, a suitable sinc impulse response is computed and windowed with a hanning window to limit its extent.

The actual source sampling rate, target sampling rate, and interpolation sampling rate are not parameters to this constructors, because only their ratios matter.

Parameters

| | |
|----------------------|--|
| <i>n_up</i> | upsampling factor, ratio between interpolation rate and source rate. |
| <i>n_down</i> | downsampling factor, ratio between interpolation rate and target rate. |
| <i>nyquist_ratio</i> | 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 |
| <i>n_irs</i> | length of impulse response (in samples at interpolation rate) |
| <i>n_ringbuffer</i> | length of ringbuffer, in samples at source sampling rate |
| <i>n_channels</i> | audio channels count |
| <i>n_prefill</i> | Prefill the ringbuffer with this many zero frames in samples at source sampling rate |

5.177.3 Member Function Documentation

5.177.3.1 void MHAFilter::polyphase_resampling_t::write (mha_wave_t & signal)

Write signal to the ringbuffer.

Signal contained in signal is appended to the audio frames already present in the ringbuffer.

Parameters

| | |
|---------------|--|
| <i>signal</i> | input signal in original sampling rate |
|---------------|--|

Exceptions

| | |
|---------------------------|---|
| MHA_Error (p. 387) | Raises exception if there is not enough room or if the number of channels does not match. |
|---------------------------|---|

5.177.3.2 void MHAFilter::polyphase_resampling_t::read (mha_wave_t & signal)

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

| | |
|---------------|--|
| <i>signal</i> | buffer to write the resampled signal to. |
|---------------|--|

Exceptions

| | |
|---------------------------|--|
| MHA_Error (p. 387) | Raises exception if there is not enough input signal or if the number of channels is too high. |
|---------------------------|--|

5.177.3.3 unsigned MHAFilter::polyphase_resampling_t::readable_frames () const [inline]

Number of frames at target sampling rate that can be produced.

This method only checks for enough future samples present, therefore, this number can be positive and a read operation can still fail if there are not enough past samples present to perform the filtering for the first output sample. This could only happen if the constructor parameters *n_ringbuffer* or *n_prefill* have been chosen too small, because otherwise the method **read** (p. 502) ensures that enough past samples are present to compute the next target sample.

5.177.4 Member Data Documentation

5.177.4.1 unsigned MHAFilter::polyphase_resampling_t::upsampling_factor [private]

Integer upsampling factor.

Interpolation rate divided by source rate.

5.177.4.2 unsigned MHAFilter::polyphase_resampling_t::downsampling_factor [private]

Integer downsampling factor.

Interpolation rate divided by target rate.

5.177.4.3 unsigned MHAFilter::polyphase_resampling_t::now_index [private]

Index of "now" in the interpolated sampling rate.

Todo Index into what? What is the meaning of now?

5.177.4.4 bool MHAFilter::polyphase_resampling_t::underflow [private]

Set to true when an underflow has occurred.

When this is true, then the object can no longer be used. Underflows have to be avoided by clients, e.g. by checking that enough **readable_frames** (p. 502) are present before calling **read** (p. 502)

5.177.4.5 MHAWindow::hanning_t MHAFilter::polyphase_resampling_t::impulse_response [private]

Contains the impulse response of the lowpass filter needed for anti-aliasing.

The impulse response is stored at the interpolation sampling rate. We use an instance of **MHAWindow::hanning_t** (p. 771) 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. 771) can be altered with ***=**, which our constructor does.

5.177.4.6 MHASignal::ringbuffer_t MHAFilter::polyphase_resampling_t::ringbuffer [private]

Storage of input signal.

Part of the polyphase resampling optimization is that apart from the FIR impulse response, nothing is stored at the interpolation rate, saving memory and computation cycles.

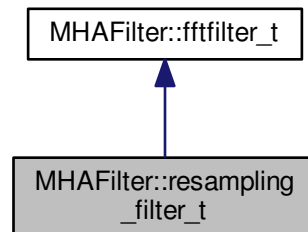
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

5.178 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 **fftl**en, unsigned int **irsl**en, unsigned int **chann**els, unsigned int **Nup**, unsigned int **Ndown**, double **fCutOff**)

Constructor.

Static Public Member Functions

- static unsigned int **fragsize_validator** (unsigned int **fftl**en, unsigned int **irsl**en)

Private Attributes

- unsigned int **fragsize**

5.178.1 Detailed Description

Hann shaped low pass filter for resampling.

This class uses FFT filter at upsampled rate.

5.178.2 Constructor & Destructor Documentation

5.178.2.1 MHAFilter::resampling_filter_t::resampling_filter_t (

unsigned int *fftl*en,

unsigned int *irsl*en,

unsigned int *chann*els,

unsigned int *Nup*,

unsigned int *Ndown*,

double *fCutOff*)

Constructor.

Parameters

| | |
|----------------|---|
| <i>fftl</i> | FFT length. |
| <i>irsl</i> | Length of filter. |
| <i>chann</i> | Number of channels to be filtered. |
| <i>Nup</i> | Upsampling ratio. |
| <i>Ndown</i> | Downsampling ratio. |
| <i>fCutOff</i> | Cut off frequency (relative to lower Nyquist Frequency) |

5.178.3 Member Function Documentation

5.178.3.1 unsigned int MHAFilter::resampling_filter_t::fragsize_validator (unsigned int *fftl*, unsigned int *irsl*) [static]

5.178.4 Member Data Documentation

5.178.4.1 unsigned int MHAFilter::resampling_filter_t::fragsize [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.179 MHAFilter::smoothspec_t Class Reference

Smooth spectral gains, create a windowed impulse response.

Public Member Functions

- **smoothspec_t** (unsigned int *fftl*, 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 **fftl**
- 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**

5.179.1 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 symmetrical around zero, i.e. to the first and last samples of the inverse Fourier transform of the input spectrum. The **spec2fir()** (p. 507) function creates a causal impulse response by circular shifting the impulse response by half of the window length.

The signal dimensions of the arguments of **smoothspec()** (p. 507) must correspond to the FFT length and number of channels provided in the constructor. The function **spec2fir()** (p. 507) can fill signal structures with more than window length frames.

5.179.2 Constructor & Destructor Documentation

5.179.2.1 **MHAFilter::smoothspec_t::smoothspec_t** (
 unsigned int *fftl*,
 unsigned int *nchannels*,
 const **MHAWindow::base_t** & *window*,
 bool *minphase*,
 bool *linphase_asym* = false)

Constructor.

Parameters

| | |
|----------------------|---|
| <i>fftlen</i> | FFT length of input spectrum (fftlen/2+1 bins) |
| <i>nchannels</i> | Number of channels in input spectrum |
| <i>window</i> | Window used for smoothing |
| <i>minphase</i> | Use minimal phase (true) or original phase (false) |
| <i>linphase_asym</i> | Keep phase, but apply full window at beginning of IRS |

5.179.2.2 MHAFilter::smoothspec_t::~smoothspec_t ()

5.179.3 Member Function Documentation

5.179.3.1 void MHAFilter::smoothspec_t::smoothspec (

const mha_spec_t & s_in,

mha_spec_t & s_out)

Create a smoothed spectrum.

Parameters

| | |
|-------------|----------------|
| <i>s_in</i> | Input spectrum |
|-------------|----------------|

Return values

| | |
|--------------|-----------------|
| <i>s_out</i> | Output spectrum |
|--------------|-----------------|

5.179.3.2 void MHAFilter::smoothspec_t::smoothspec (

mha_spec_t & spec) [inline]

Create a smoothed spectrum (in place)

Parameters

| | |
|-------------|--------------------------|
| <i>spec</i> | Spectrum to be smoothed. |
|-------------|--------------------------|

5.179.3.3 void MHAFilter::smoothspec_t::spec2fir (

const mha_spec_t & spec,

mha_wave_t & fir)

Return FIR coefficients.

Parameters

| | |
|-------------|----------------|
| <i>spec</i> | Input spectrum |
|-------------|----------------|

Return values

| | |
|------------|---|
| <i>fir</i> | FIR coefficients, minimum length is window length |
|------------|---|

5.179.3.4 void MHAFilter::smoothspec_t::internal_fir (
const mha_spec_t & s_in) [private]

5.179.4 Member Data Documentation

5.179.4.1 unsigned int MHAFilter::smoothspec_t::fftlens [private]

5.179.4.2 unsigned int MHAFilter::smoothspec_t::nchannels [private]

5.179.4.3 MHAWindow::base_t MHAFilter::smoothspec_t::window [private]

5.179.4.4 MHASignal::waveform_t MHAFilter::smoothspec_t::tmp_wave [private]

5.179.4.5 MHASignal::spectrum_t MHAFilter::smoothspec_t::tmp_spec [private]

5.179.4.6 MHASignal::minphase_t* MHAFilter::smoothspec_t::minphase [private]

5.179.4.7 bool MHAFilter::smoothspec_t::_linphase_asym [private]

5.179.4.8 mha_fft_t MHAFilter::smoothspec_t::fft [private]

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

- mha_filter.hh
- mha_filter.cpp

5.180 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::gammaflt_t **fb**
- MHASignal::waveform_t **out_chunk**
- MHASignal::waveform_t **out_chunk_im**

5.180.1 Constructor & Destructor Documentation

5.180.1.1 MHAFilter::thirddoctave_analyzer_t::thirddoctave_analyzer_t (mhaconfig_t *cfg*)

5.180.2 Member Function Documentation

5.180.2.1 mha_wave_t * MHAFilter::thirddoctave_analyzer_t::process (mha_wave_t * *sln*)

5.180.2.2 unsigned int MHAFilter::thirddoctave_analyzer_t::nbands ()

5.180.2.3 unsigned int MHAFilter::thirddoctave_analyzer_t::nchannels ()

5.180.2.4 std::vector< mha_real_t > MHAFilter::thirddoctave_analyzer_t::get_cf_hz ()

5.180.2.5 std::vector< mha_real_t > MHAFilter::thirddoctave_analyzer_t::cf_generator (mhaconfig_t *cfg*) [static]

5.180.2.6 std::vector< mha_real_t > MHAFilter::thirddoctave_analyzer_t::bw_generator (mhaconfig_t *cfg*) [static]

5.180.2.7 std::vector< mha_real_t > MHAFilter::thirddoctave_analyzer_t::dup (std::vector< mha_real_t > *vec*, mhaconfig_t *cfg*) [static]

5.180.3 Member Data Documentation

5.180.3.1 mhaconfig_t MHAFilter::thirddoctave_analyzer_t::cfg_ [private]

- 5.180.3.2 `std::vector<mha_real_t> MHAFilter::thirdoctave_analyzer_t::cf` [private]
- 5.180.3.3 `MHAFilter::gammaflt_t MHAFilter::thirdoctave_analyzer_t::fb` [private]
- 5.180.3.4 `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk` [private]
- 5.180.3.5 `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk_im`
[private]

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

- `complex_filter.h`
- `complex_filter.cpp`

5.181 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.

5.181.1 Detailed Description

a structure containing a source channel number, a target channel number, and an impulse response.

5.181.2 Constructor & Destructor Documentation

5.181.2.1 MHAFilter::transfer_function_t::transfer_function_t () `[inline]`

Default constructor for STL conformity.

Not used.

5.181.2.2 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

| | |
|-----------------------------|--|
| <i>source_channel_index</i> | Source audio channel index for this transfer function |
| <i>target_channel_index</i> | Target audio channel index for this transfer function |
| <i>impulse_response</i> | Impulse response of transfer from source to target channel |

5.181.3 Member Function Documentation

5.181.3.1 unsigned int MHAFilter::transfer_function_t::partitions (unsigned int *fragsize*) const `[inline]`

for the given partition size, return the number of partitions of the impulse response.

Parameters

| | |
|-----------------|----------------|
| <i>fragsize</i> | partition size |
|-----------------|----------------|

Returns

number of partitions occupied by the impulse response

5.181.3.2 `unsigned int MHAFilter::transfer_function_t::non_empty_partitions (`
`unsigned int fragsize) const` `[inline]`

for the given partition size, return the number of non-empty partitions of the impulse response.

Parameters

| | |
|-----------------|----------------|
| <i>fragsize</i> | partition size |
|-----------------|----------------|

Returns

the number of non-empty partitions of the impulse response, i.e. partitions containing only zeros are not counted.

5.181.3.3 `bool MHAFilter::transfer_function_t::isempty (`
`unsigned int fragsize,`
`unsigned int index) const` `[inline]`

checks if the partition contains only zeros

Parameters

| | |
|-----------------|-----------------|
| <i>fragsize</i> | partition size |
| <i>index</i> | partition index |

Returns

true when this partition of the impulse response contains only zeros.

5.181.4 Member Data Documentation

5.181.4.1 `unsigned int MHAFilter::transfer_function_t::source_channel_index`

Source audio channel index for this transfer function.

5.181.4.2 `unsigned int MHAFilter::transfer_function_t::target_channel_index`

Target audio channel index for this transfer function.

5.181.4.3 `std::vector<float> MHAFilter::transfer_function_t::impulse_response`

Impulse response of transfer from source to target channel.

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

- **mha_filter.hh**
- **mha_filter.cpp**

5.182 MHAFilter::transfer_matrix_t Struct Reference

A sparse matrix of transfer function partitionss.

Inherits vector< transfer_function_t >.

Public Member Functions

- std::valarray< unsigned int > **partitions** (unsigned fragsize) const
*Returns an array of the results of calling the **partitions()** (p. 513) 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. 513) method on every matrix member.*

5.182.1 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.

5.182.2 Member Function Documentation

5.182.2.1 std::valarray<unsigned int> MHAFilter::transfer_matrix_t::partitions (
unsigned fragsize) const [inline]

Returns an array of the results of calling the **partitions()** (p. 513) method on every matrix member.

5.182.2.2 std::valarray<unsigned int> MHAFilter::transfer_matrix_t::non_empty_partitions (
unsigned int fragsize) const [inline]

Returns an array of the results of calling the **non_empty_partitions()** (p. 513) method on every matrix member.

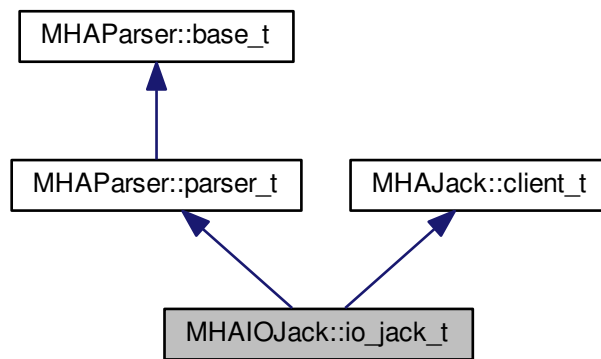
The documentation for this struct was generated from the following file:

- mha_filter.hh

5.183 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

5.183.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

5.183.2 Constructor & Destructor Documentation

```
5.183.2.1 io_jack_t::io_jack_t (
    unsigned int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

5.183.3 Member Function Documentation

```
5.183.3.1 void io_jack_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

5.183.3.2 void io_jack_t::release (void)

5.183.3.3 void io_jack_t::reconnect_inports () [private]

Connect the input ports when connection variable is accessed.

5.183.3.4 void io_jack_t::reconnect_outports () [private]

Connect the output ports when connection variable is accessed.

5.183.3.5 void io_jack_t::get_physical_input_ports () [private]

5.183.3.6 void io_jack_t::get_physical_output_ports () [private]

5.183.3.7 void io_jack_t::get_all_input_ports () [private]

5.183.3.8 void io_jack_t::get_all_output_ports () [private]

5.183.3.9 void io_jack_t::get_delays_in () [private]

5.183.3.10 void io_jack_t::get_delays_out () [private]

5.183.3.11 void io_jack_t::read_get_cpu_load () [private]

5.183.3.12 void io_jack_t::read_get_xruns () [private]

5.183.3.13 void io_jack_t::read_get_scheduler () [private]

5.183.4 Member Data Documentation

5.183.4.1 unsigned int MHAIOJack::io_jack_t::fw_fragsize [private]

5.183.4.2 float MHAIOJack::io_jack_t::fw_samplerate [private]

5.183.4.3 MHAParser::string_t MHAIOJack::io_jack_t::servername [private]

5.183.4.4 MHAParser::string_t MHAIOJack::io_jack_t::clientname [private]

5.183.4.5 MHAParser::vstring_t MHAIOJack::io_jack_t::connections_in [private]

5.183.4.6 MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_in [private]

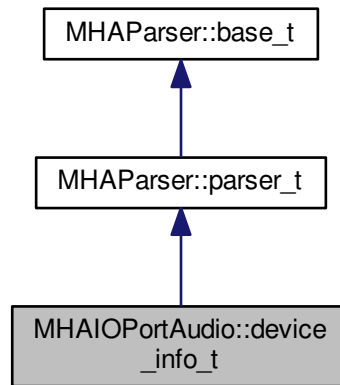
- 5.183.4.7 MHAParser::vstring_t MHAIOJack::io_jack_t::connections_out [private]
- 5.183.4.8 MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_out [private]
- 5.183.4.9 MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_in [private]
- 5.183.4.10 MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_out [private]
- 5.183.4.11 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_physical [private]
- 5.183.4.12 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_physical [private]
- 5.183.4.13 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_all [private]
- 5.183.4.14 MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_all [private]
- 5.183.4.15 MHAParser::parser_t MHAIOJack::io_jack_t::ports_parser [private]
- 5.183.4.16 MHAParser::float_mon_t MHAIOJack::io_jack_t::state_cpuload [private]
- 5.183.4.17 MHAParser::int_mon_t MHAIOJack::io_jack_t::state_xruns [private]
- 5.183.4.18 MHAParser::int_mon_t MHAIOJack::io_jack_t::state_priority [private]
- 5.183.4.19 MHAParser::string_mon_t MHAIOJack::io_jack_t::state_scheduler [private]
- 5.183.4.20 MHAParser::parser_t MHAIOJack::io_jack_t::state_parser [private]
- 5.183.4.21 MHAEvents::patchbay_t<io_jack_t> MHAIOJack::io_jack_t::patchbay [private]

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

- MHAIOJack.cpp

5.184 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

5.184.1 Constructor & Destructor Documentation

5.184.1.1 MHAIOPortAudio::device_info_t::device_info_t() [inline]

5.184.2 Member Function Documentation

5.184.2.1 void MHAIOPortAudio::device_info_t::fill_info() [inline]

5.184.3 Member Data Documentation

5.184.3.1 MHAParser::int_mon_t MHAIOPortAudio::device_info_t::numDevices

5.184.3.2 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::structVersion

5.184.3.3 MHAParser::vstring_mon_t MHAIOPortAudio::device_info_t::name

5.184.3.4 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::hostApi

5.184.3.5 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxInputChannels

5.184.3.6 MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxOutputChannels

5.184.3.7 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowInputLatency

5.184.3.8 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowOutputLatency

5.184.3.9 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighInputLatency

5.184.3.10 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighOutputLatency

5.184.3.11 MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultSampleRate

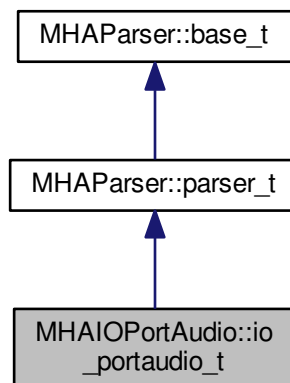
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.185 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_updated** ()
- void **device_index_updated** ()
- ~**io_portaudio_t** ()
- void **cmd_prepare** (int, int)
- void **cmd_start** ()
- void **cmd_stop** ()
- void **cmd_release** ()
- int **portaudio_callback** (const void *input, void *output, unsigned long frame_count, const PaStreamCallbackTimeInfo *time_info, PaStreamCallbackFlags status_flags)

Private Attributes

- **device_info_t** **device_info**
- **MHASignal::waveform_t** * **s_in**
- **mha_wave_t** * **s_out**
- float **samplerate**
- unsigned int **nchannels_out**
- unsigned int **nchannels_in**

- unsigned int **fragsize**
- **IOProcessEvent_t** **proc_event**
- void * **proc_handle**
- **IOStartedEvent_t** **start_event**
- void * **start_handle**
- **IOStoppedEvent_t** **stop_event**
- void * **stop_handle**
- PaStream * **portaudio_stream**
- **MHAParser::string_t** **device_name**
- **MHAParser::int_t** **device_index**
- **MHAEvents::patchbay_t** < **io_portaudio_t** > **patchbay**

Additional Inherited Members

5.185.1 Detailed Description

Main class for Portaudio sound IO.

5.185.2 Constructor & Destructor Documentation

5.185.2.1 MHAIOPortAudio::io_portaudio_t::io_portaudio_t (
 unsigned int *fragsize*,
 float *samplerate*,
 IOProcessEvent_t *proc_event*,
 void * *proc_handle*,
 IOStartedEvent_t *start_event*,
 void * *start_handle*,
 IOStoppedEvent_t *stop_event*,
 void * *stop_handle*) [inline]

5.185.2.2 MHAIOPortAudio::io_portaudio_t::~io_portaudio_t () [inline]

5.185.3 Member Function Documentation

5.185.3.1 void MHAIOPortAudio::io_portaudio_t::device_name_updated () [inline]

5.185.3.2 void MHAIOPortAudio::io_portaudio_t::device_index_updated () [inline]

5.185.3.3 void MHAIOPortAudio::io_portaudio_t::cmd_prepare (
 int *nchannels_in*,
 int *nchannels_out*)

5.185.3.4 void MHAIOPortAudio::io_portaudio_t::cmd_start ()

- 5.185.3.5 void MHAIOPortAudio::io_portaudio_t::cmd_stop ()
- 5.185.3.6 void MHAIOPortAudio::io_portaudio_t::cmd_release ()
- 5.185.3.7 int MHAIOPortAudio::io_portaudio_t::portaudio_callback (
 - const void * *input*,
 - void * *output*,
 - unsigned long *frame_count*,
 - const PaStreamCallbackTimeInfo * *time_info*,
 - PaStreamCallbackFlags *status_flags*)
- 5.185.4 Member Data Documentation
- 5.185.4.1 device_info_t MHAIOPortAudio::io_portaudio_t::device_info [private]
- 5.185.4.2 MHASignal::waveform_t* MHAIOPortAudio::io_portaudio_t::s_in [private]
- 5.185.4.3 mha_wave_t* MHAIOPortAudio::io_portaudio_t::s_out [private]
- 5.185.4.4 float MHAIOPortAudio::io_portaudio_t::samplerate [private]
- 5.185.4.5 unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_out [private]
- 5.185.4.6 unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_in [private]
- 5.185.4.7 unsigned int MHAIOPortAudio::io_portaudio_t::fragsize [private]
- 5.185.4.8 IOProcessEvent_t MHAIOPortAudio::io_portaudio_t::proc_event [private]
- 5.185.4.9 void* MHAIOPortAudio::io_portaudio_t::proc_handle [private]
- 5.185.4.10 IOStartedEvent_t MHAIOPortAudio::io_portaudio_t::start_event [private]
- 5.185.4.11 void* MHAIOPortAudio::io_portaudio_t::start_handle [private]
- 5.185.4.12 IOStoppedEvent_t MHAIOPortAudio::io_portaudio_t::stop_event [private]
- 5.185.4.13 void* MHAIOPortAudio::io_portaudio_t::stop_handle [private]
- 5.185.4.14 PaStream* MHAIOPortAudio::io_portaudio_t::portaudio_stream [private]
- 5.185.4.15 MHAParser::string_t MHAIOPortAudio::io_portaudio_t::device_name [private]
- 5.185.4.16 MHAParser::int_t MHAIOPortAudio::io_portaudio_t::device_index [private]
- 5.185.4.17 MHAEvents::patchbay_t<io_portaudio_t> MHAIOPortAudio::io_portaudio_t::patchbay [private]

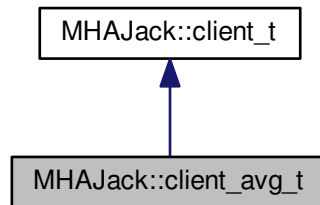
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

5.186 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

5.186.1 Detailed Description

Generic JACK client for averaging a system response across time.

5.186.2 Constructor & Destructor Documentation

5.186.2.1 MHAJack::client_avg_t::client_avg_t (
 const std::string & *name_*,
 const unsigned int & *nrep_*)

Constructor for averaging client.

Parameters

| | |
|---------------------|-----------------------|
| <i>name_</i> ↔ — | Name of JACK client |
| <i>nrep_</i> ↔ — | Number of repetitions |

5.186.3 Member Function Documentation

5.186.3.1 void MHAJack::client_avg_t::io (
 mha_wave_t * *is_out*,
 mha_wave_t * *is_in*,
 const std::vector< std::string > & *p_out*,
 const std::vector< std::string > & *p_in*,
 float * *srate* = NULL,
 unsigned int * *fragsize* = NULL)

Recording function.

long-description

Parameters

| | |
|-----------------|---|
| <i>is_out</i> | Input (test) signal, which will be repeated |
| <i>is_in</i> | System response (averaged, same length as input required) |
| <i>p_out</i> | Ports to play back the test signal |
| <i>p_in</i> | Ports to record from the system response |
| <i>srate</i> | Pointer to sampling rate variable, will be filled with server sampling rate |
| <i>fragsize</i> | Pointer to fragment size variable, will be filled with server fragment size |

5.186.3.2 int MHAJack::client_avg_t::proc (
 void * *handle*,
 mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [static], [private]

5.186.3.3 void MHAJack::client_avg_t::IOStoppedEvent (
 void * *handle*,
 int *proc_err*,
 int *io_err*) [static], [private]

5.186.3.4 void MHAJack::client_avg_t::proc (
 mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [private]

5.186.3.5 void MHAJack::client_avg_t::IOStoppedEvent () [private]

5.186.4 Member Data Documentation

5.186.4.1 bool MHAJack::client_avg_t::b_stopped [private]

5.186.4.2 unsigned int MHAJack::client_avg_t::pos [private]

5.186.4.3 mha_wave_t* MHAJack::client_avg_t::sn_in [private]

5.186.4.4 mha_wave_t* MHAJack::client_avg_t::sn_out [private]

5.186.4.5 std::string MHAJack::client_avg_t::name [private]

5.186.4.6 MHASignal::waveform_t* MHAJack::client_avg_t::frag_out [private]

5.186.4.7 const unsigned int MHAJack::client_avg_t::nrep [private]

5.186.4.8 unsigned int MHAJack::client_avg_t::n [private]

5.186.4.9 bool MHAJack::client_avg_t::b_ready [private]

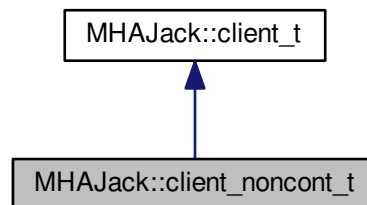
The documentation for this class was generated from the following files:

- mhajack.h
- mhajack.cpp

5.187 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

5.187.1 Detailed Description

Generic client for synchronous playback and recording of waveform fragments.

5.187.2 Constructor & Destructor Documentation

5.187.2.1 MHAJack::client_noncont_t::client_noncont_t (
 const std::string & *name*,
 bool *use_jack_transport* = false)

5.187.3 Member Function Documentation

5.187.3.1 void MHAJack::client_noncont_t::io (
 mha_wave_t * *s_out*,
 mha_wave_t * *s_in*,
 const std::vector< std::string > & *p_out*,
 const std::vector< std::string > & *p_in*,
 float * *srate* = NULL,
 unsigned int * *fragsize* = NULL)

5.187.3.2 int MHAJack::client_noncont_t::proc (
 void * *handle*,
 mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [static], [private]

5.187.3.3 void MHAJack::client_noncont_t::IOStoppedEvent (
 void * *handle*,
 int *proc_err*,
 int *io_err*) [static], [private]

5.187.3.4 void MHAJack::client_noncont_t::proc (
 mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [private]

5.187.3.5 void MHAJack::client_noncont_t::IOStoppedEvent () [private]

5.187.4 Member Data Documentation

5.187.4.1 bool MHAJack::client_noncont_t::b_stopped [private]

5.187.4.2 unsigned int MHAJack::client_noncont_t::pos [private]

5.187.4.3 `mha_wave_t* MHAJack::client_noncont_t::sn_in` [private]

5.187.4.4 `mha_wave_t* MHAJack::client_noncont_t::sn_out` [private]

5.187.4.5 `std::string MHAJack::client_noncont_t::name` [private]

5.187.4.6 `MHASignal::waveform_t* MHAJack::client_noncont_t::frag_out` [private]

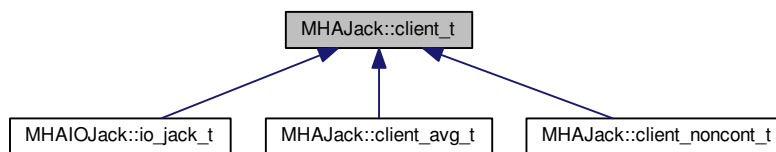
The documentation for this class was generated from the following files:

- `mhajack.h`
- `mhajack.cpp`

5.188 MHAJack::client_t Class Reference

Generic asynchronous JACK client.

Inheritance diagram for MHAJack::client_t:



Public Member Functions

- **client_t** (`IOProcessEvent_t proc_event`, `void *proc_handle=NULL`, `IOStartedEvent_t start_event=NULL`, `void *start_handle=NULL`, `IOStoppedEvent_t stop_event=NULL`, `void *stop_handle=NULL`, `bool use_jack_transport=false`)
- **void prepare** (`const std::string &client_name`, `const unsigned int &nchannels_in`, `const unsigned int &nchannels_out`)
Allocate buffers, activate JACK client and install internal ports.
- **void prepare** (`const std::string &server_name`, `const std::string &client_name`, `const unsigned int &nchannels_in`, `const unsigned int &nchannels_out`)
Allocate buffers, ports, and activates JACK client.
- **void release** ()
Remove JACK client and deallocate internal ports and buffers.
- **void start** (`bool fail_on_async_jack_error=true`)
- **void stop** ()
- **void connect_input** (`const std::vector< std::string > &`)

Connect the input ports when connection variable is accessed.

- void **connect_output** (const std::vector< std::string > &)

Connect the output ports when connection variable is accessed.

- unsigned int **get_fragsize** () const
- float **get_srate** () const
- unsigned long **get_xruns** ()
- unsigned long **get_xruns_reset** ()
- std::string **str_error** (int err)
- void **get_ports** (std::vector< std::string > &, unsigned long jack_flags)

Get a list of Jack ports.

- std::vector< std::string > **get_my_input_ports** ()
- std::vector< std::string > **get_my_output_ports** ()
- void **set_input_portnames** (const std::vector< std::string > &)
- void **set_output_portnames** (const std::vector< std::string > &)
- float **get_cpu_load** ()
- void **set_use_jack_transport** (bool ut)

Protected Attributes

- jack_client_t * **jc**

Private Member Functions

- void **prepare_impl** (const char *server_name, const char *client_name, const unsigned int &nchannels_in, const unsigned int &nchannels_out)

Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.

- void **internal_start** ()
- void **internal_stop** ()
- void **stopped** (int, int)
- int **jack_proc_cb** (jack_nframes_t)

This is the main processing callback.

- int **jack_xrun_cb** ()

Static Private Member Functions

- static int **jack_proc_cb** (jack_nframes_t, void *)
- static int **jack_xrun_cb** (void *)

Private Attributes

- unsigned long **num_xruns**
- unsigned int **fragsize**
- float **samplerate**
- unsigned int **nchannels_in**
- unsigned int **nchannels_out**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- **MHASignal::waveform_t * s_in**
- **mha_wave_t * s_out**
- **MHAJack::port_t ** inch**
- **MHAJack::port_t ** outch**
- unsigned int **flags**
- bool **b_prepared**
- bool **use_jack_transport**
- **jack_transport_state_t jstate_prev**
- **std::vector< std::string > input_portnames**
- **std::vector< std::string > output_portnames**
- bool **fail_on_async_jackerror**

5.188.1 Detailed Description

Generic asynchronous JACK client.

5.188.2 Constructor & Destructor Documentation

```
5.188.2.1 MHAJack::client_t::client_t(
    IOProcessEvent_t proc_event,
    void * proc_handle = NULL,
    IOStartedEvent_t start_event = NULL,
    void * start_handle = NULL,
    IOStoppedEvent_t stop_event = NULL,
    void * stop_handle = NULL,
    bool use_jack_transport = false )
```

5.188.3 Member Function Documentation

```
5.188.3.1 void MHAJack::client_t::prepare (
    const std::string & client_name,
    const unsigned int & nch_in,
    const unsigned int & nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

Registers the jack client with the default jack server and activates it.

Parameters

| | |
|--------------------|--------------------------|
| <i>client_name</i> | Name of this jack client |
| <i>nch_in</i> | Input ports to register |
| <i>nch_out</i> | Output ports to register |

5.188.3.2 void MHAJack::client_t::prepare (
 const std::string & *server_name*,
 const std::string & *client_name*,
 const unsigned int & *nch_in*,
 const unsigned int & *nch_out*)

Allocate buffers, ports, and activates JACK client.

Registers the jack client with specified jack server and activates it.

Parameters

| | |
|--------------------|--|
| <i>server_name</i> | Name of the jack server to register with |
| <i>client_name</i> | Name of this jack client |
| <i>nch_in</i> | Input ports to register |
| <i>nch_out</i> | Output ports to register |

5.188.3.3 void MHAJack::client_t::release (
 void)

Remove JACK client and deallocate internal ports and buffers.

5.188.3.4 void MHAJack::client_t::start (
 bool *fail_on_async_jack_error* = true)

5.188.3.5 void MHAJack::client_t::stop ()

5.188.3.6 void MHAJack::client_t::connect_input (
 const std::vector< std::string > & *con*)

Connect the input ports when connection variable is accessed.

5.188.3.7 void MHAJack::client_t::connect_output (
 const std::vector< std::string > & *con*)

Connect the output ports when connection variable is accessed.

5.188.3.8 unsigned int MHAJack::client_t::get_fragsize () const [inline]

5.188.3.9 float MHAJack::client_t::get_srate () const [inline]

5.188.3.10 unsigned long MHAJack::client_t::get_xruns () [inline]

5.188.3.11 unsigned long MHAJack::client_t::get_xruns_reset ()

5.188.3.12 std::string MHAJack::client_t::str_error (
 int *err*)

5.188.3.13 void MHAJack::client_t::get_ports (
 std::vector< std::string > & *res*,
 unsigned long *jack_flags*)

Get a list of Jack ports.

Parameters

| | |
|-------------------|--------------------------------------|
| <i>res</i> | Result string vector |
| <i>jack_flags</i> | Jack port flags (JackPortInput etc.) |

5.188.3.14 std::vector< std::string > MHAJack::client_t::get_my_input_ports ()

5.188.3.15 std::vector< std::string > MHAJack::client_t::get_my_output_ports ()

5.188.3.16 void MHAJack::client_t::set_input_portnames (
 const std::vector< std::string > & *names*)

5.188.3.17 void MHAJack::client_t::set_output_portnames (
 const std::vector< std::string > & *names*)

5.188.3.18 float MHAJack::client_t::get_cpu_load ()

5.188.3.19 void MHAJack::client_t::set_use_jack_transport (
 bool *ut*) [inline]

5.188.3.20 void MHAJack::client_t::prepare_impl (
 const char * *server_name*,
 const char * *client_name*,
 const unsigned int & *nch_in*,
 const unsigned int & *nch_out*) [private]

Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.

Parameters

| | |
|--------------------|--|
| <i>server_name</i> | Name of the jack server to register with |
| <i>client_name</i> | Name of this jack client |
| <i>nch_in</i> | Input ports to register |
| <i>nch_out</i> | Output ports to register |

5.188.3.21 void MHAJack::client_t::internal_start () [private]

5.188.3.22 void MHAJack::client_t::internal_stop () [private]

5.188.3.23 void MHAJack::client_t::stopped (
 int *proc_err*,
 int *io_err*) [private]

5.188.3.24 int MHAJack::client_t::jack_proc_cb (
 jack_nframes_t *n*,
 void * *h*) [static], [private]

5.188.3.25 int MHAJack::client_t::jack_proc_cb (
 jack_nframes_t *n*) [private]

This is the main processing callback.

Here happens double buffering and downsampling.

5.188.3.26 int MHAJack::client_t::jack_xrun_cb (
 void * *h*) [static], [private]

5.188.3.27 int MHAJack::client_t::jack_xrun_cb () [inline], [private]

5.188.4 Member Data Documentation

5.188.4.1 unsigned long MHAJack::client_t::num_xruns [private]

5.188.4.2 unsigned int MHAJack::client_t::fragsize [private]

5.188.4.3 float MHAJack::client_t::samplerate [private]

5.188.4.4 unsigned int MHAJack::client_t::nchannels_in [private]

5.188.4.5 unsigned int MHAJack::client_t::nchannels_out [private]

5.188.4.6 IOProcessEvent_t MHAJack::client_t::proc_event [private]

- 5.188.4.7 `void* MHAJack::client_t::proc_handle` [private]
- 5.188.4.8 `IOStartedEvent_t MHAJack::client_t::start_event` [private]
- 5.188.4.9 `void* MHAJack::client_t::start_handle` [private]
- 5.188.4.10 `IOStoppedEvent_t MHAJack::client_t::stop_event` [private]
- 5.188.4.11 `void* MHAJack::client_t::stop_handle` [private]
- 5.188.4.12 `MHASignal::waveform_t* MHAJack::client_t::s_in` [private]
- 5.188.4.13 `mha_wave_t* MHAJack::client_t::s_out` [private]
- 5.188.4.14 `MHAJack::port_t** MHAJack::client_t::inch` [private]
- 5.188.4.15 `MHAJack::port_t** MHAJack::client_t::outch` [private]
- 5.188.4.16 `jack_client_t* MHAJack::client_t::jc` [protected]
- 5.188.4.17 `unsigned int MHAJack::client_t::flags` [private]
- 5.188.4.18 `bool MHAJack::client_t::b_prepared` [private]
- 5.188.4.19 `bool MHAJack::client_t::use_jack_transport` [private]
- 5.188.4.20 `jack_transport_state_t MHAJack::client_t::jstate_prev` [private]
- 5.188.4.21 `std::vector<std::string> MHAJack::client_t::input_portnames` [private]
- 5.188.4.22 `std::vector<std::string> MHAJack::client_t::output_portnames` [private]
- 5.188.4.23 `bool MHAJack::client_t::fail_on_async_jackerror` [private]

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

- `mhajack.h`
- `mhajack.cpp`

5.189 MHAJack::port_t Class Reference

Class for one channel/port.

Public Types

Public Member Functions

- **port_t** (jack_client_t ***jc**, **dir_t** dir, int id)
- **port_t** (jack_client_t ***jc**, **dir_t** dir, const std::string &id)
Constructor to create port with specific name.
- ~**port_t** ()
- void **read** (mha_wave_t *s, unsigned int ch)
- void **write** (mha_wave_t *s, unsigned int ch)
- void **mute** (unsigned int n)
- void **connect_to** (const char *pn)
- const char * **get_short_name** ()
Return the port name.

Private Attributes

- **dir_t** **dir_type**
- jack_port_t * **port**
- jack_default_audio_sample_t * **iob**
- jack_client_t * **jc**

5.189.1 Detailed Description

Class for one channel/port.

This class represents one JACK port. Double buffering for asynchronous process callbacks is managed by this class.

5.189.2 Member Enumeration Documentation

5.189.2.1 enum MHAJack::port_t::dir_t

Enumerator

input
output

5.189.3 Constructor & Destructor Documentation

5.189.3.1 MHAJack::port_t::port_t (jack_client_t * *jc*, **dir_t** *dir*, int *id*)

Parameters

| | |
|------------|--|
| <i>jc</i> | JACK client. |
| <i>dir</i> | Direction (input/output). |
| <i>id</i> | Number in port name (starting with 1). |

5.189.3.2 `MHAJack::port_t::port_t (`
 `jack_client_t * jc,`
 `dir_t dir,`
 `const std::string & id)`

Constructor to create port with specific name.

Parameters

| | |
|------------|---------------------------|
| <i>jc</i> | JACK client. |
| <i>dir</i> | Direction (input/output). |
| <i>id</i> | Port name. |

5.189.3.3 `MHAJack::port_t::~~port_t ()`

5.189.4 Member Function Documentation

5.189.4.1 `void MHAJack::port_t::read (`
 `mha_wave_t * s,`
 `unsigned int ch)`

Parameters

| | |
|-----------|--|
| <i>s</i> | Signal structure to store the audio data. |
| <i>ch</i> | Channel number in audio data structure to be used. |

5.189.4.2 `void MHAJack::port_t::write (`
 `mha_wave_t * s,`
 `unsigned int ch)`

Parameters

| | |
|-----------|---|
| <i>s</i> | Signal structure from which the audio data is read. |
| <i>ch</i> | Channel number in audio data structure to be used. |

5.189.4.3 `void MHAJack::port_t::mute (`
 `unsigned int n)`

Parameters

| | |
|----------|---|
| <i>n</i> | Number of samples to be muted (must be the same as reported by Jack processing callback). |
|----------|---|

5.189.4.4 void MHAJack::port_t::connect_to (
const char * *pn*)

Parameters

| | |
|-----------|-------------------------|
| <i>pn</i> | Port name to connect to |
|-----------|-------------------------|

5.189.4.5 const char * MHAJack::port_t::get_short_name ()

Return the port name.

5.189.5 Member Data Documentation

5.189.5.1 dir_t MHAJack::port_t::dir_type [private]

5.189.5.2 jack_port_t* MHAJack::port_t::port [private]

5.189.5.3 jack_default_audio_sample_t* MHAJack::port_t::iob [private]

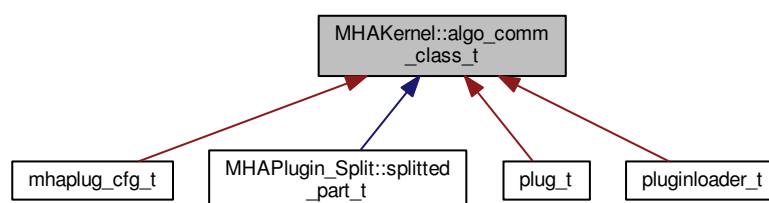
5.189.5.4 jack_client_t* MHAJack::port_t::jc [private]

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

- mhajack.h
- mhajack.cpp

5.190 MHAKernel::algo_comm_class_t Class Reference

Inheritance diagram for MHAKernel::algo_comm_class_t:



Public Member Functions

- **algo_comm_class_t** ()
- virtual **~algo_comm_class_t** ()
- **algo_comm_t** **get_c_handle** ()
- virtual void **local_insert_var** (const char *, **comm_var_t**)
- virtual void **local_remove_var** (const char *)
- virtual void **local_remove_ref** (void *)
- virtual bool **local_is_var** (const char *)
- virtual void **local_get_var** (const char *, **comm_var_t** *)
- virtual std::string **local_get_entries** ()
- virtual comm_var_map_t::size_type **size** () const

Static Public Member Functions

- static int **insert_var** (void *, const char *, **comm_var_t**)
- static int **insert_var_int** (void *, const char *, int *)
- static int **insert_var_float** (void *, const char *, float *)
- static int **remove_var** (void *, const char *)
- static int **remove_ref** (void *, void *)
- static int **is_var** (void *, const char *)
- static int **get_var** (void *, const char *, **comm_var_t** *)
- static int **get_var_int** (void *, const char *, int *)
- static int **get_var_float** (void *, const char *, float *)
- static int **get_entries** (void *, char *, unsigned int)
- static const char * **get_error** (int)

Public Attributes

- char * **algo_comm_id_string**

Private Attributes

- **algo_comm_t** **ac**
- int **algo_comm_id_string_len**
- **comm_var_map_t** **vars**

5.190.1 Constructor & Destructor Documentation

5.190.1.1 MHAKernel::algo_comm_class_t::algo_comm_class_t ()

5.190.1.2 MHAKernel::algo_comm_class_t::~~algo_comm_class_t () [virtual]

5.190.2 Member Function Documentation

5.190.2.1 algo_comm_t MHAKernel::algo_comm_class_t::get_c_handle ()

5.190.2.2 int MHAKernel::algo_comm_class_t::insert_var (
void * *handle*,
const char * *name*,
comm_var_t *var*) [static]

5.190.2.3 int MHAKernel::algo_comm_class_t::insert_var_int (
void * *handle*,
const char * *name*,
int * *ivar*) [static]

5.190.2.4 int MHAKernel::algo_comm_class_t::insert_var_float (
void * *handle*,
const char * *name*,
float * *ivar*) [static]

5.190.2.5 int MHAKernel::algo_comm_class_t::remove_var (
void * *handle*,
const char * *name*) [static]

5.190.2.6 int MHAKernel::algo_comm_class_t::remove_ref (
void * *handle*,
void * *ref*) [static]

5.190.2.7 int MHAKernel::algo_comm_class_t::is_var (
void * *handle*,
const char * *name*) [static]

5.190.2.8 int MHAKernel::algo_comm_class_t::get_var (
void * *handle*,
const char * *name*,
comm_var_t * *var*) [static]

5.190.2.9 int MHAKernel::algo_comm_class_t::get_var_int (
void * *handle*,
const char * *name*,
int * *ivar*) [static]

- 5.190.2.10 `int MHAKernel::algo_comm_class_t::get_var_float (`
`void * handle,`
`const char * name,`
`float * ivar) [static]`
- 5.190.2.11 `int MHAKernel::algo_comm_class_t::get_entries (`
`void * handle,`
`char * ret,`
`unsigned int len) [static]`
- 5.190.2.12 `const char * MHAKernel::algo_comm_class_t::get_error (`
`int e) [static]`
- 5.190.2.13 `void MHAKernel::algo_comm_class_t::local_insert_var (`
`const char * name,`
`comm_var_t var) [virtual]`
- 5.190.2.14 `void MHAKernel::algo_comm_class_t::local_remove_var (`
`const char * name) [virtual]`
- 5.190.2.15 `void MHAKernel::algo_comm_class_t::local_remove_ref (`
`void * addr) [virtual]`
- 5.190.2.16 `bool MHAKernel::algo_comm_class_t::local_is_var (`
`const char * name) [virtual]`
- 5.190.2.17 `void MHAKernel::algo_comm_class_t::local_get_var (`
`const char * name,`
`comm_var_t * var) [virtual]`
- 5.190.2.18 `std::string MHAKernel::algo_comm_class_t::local_get_entries () [virtual]`
- 5.190.2.19 `MHAKernel::comm_var_map_t::size_type MHAKernel::algo_comm_class_t::size () const`
`[virtual]`

5.190.3 Member Data Documentation

- 5.190.3.1 `char* MHAKernel::algo_comm_class_t::algo_comm_id_string`
- 5.190.3.2 `algo_comm_t MHAKernel::algo_comm_class_t::ac [private]`
- 5.190.3.3 `int MHAKernel::algo_comm_class_t::algo_comm_id_string_len [private]`
- 5.190.3.4 `comm_var_map_t MHAKernel::algo_comm_class_t::vars [private]`

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

- `mha_algo_comm.hh`
- `mha_algo_comm.cpp`

5.191 MHAKernel::comm_var_map_t Class Reference

Inherits map< std::string, comm_var_t >.

Public Member Functions

- bool **has_key** (const std::string &name)

5.191.1 Member Function Documentation

5.191.1.1 bool MHAKernel::comm_var_map_t::has_key (const std::string &name) [inline]

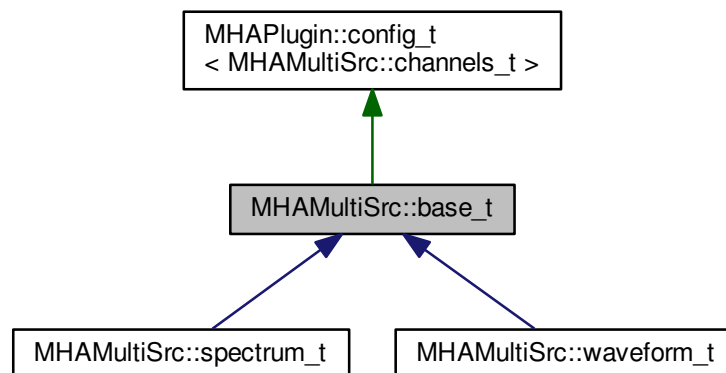
The documentation for this class was generated from the following file:

- mha_algo_comm.hh

5.192 MHAMultiSrc::base_t Class Reference

Base class for source selection.

Inheritance diagram for MHAMultiSrc::base_t:



Public Member Functions

- **base_t** (algo_comm_t iac)
- void **select_source** (const std::vector< std::string > &src, int in_channels)
Change the selection of input sources.

5.193 MHAMultiSrc::channel_t Class Reference

Public Attributes

- std::string **name**
- int **channel**

5.193.1 Member Data Documentation

5.193.1.1 std::string MHAMultiSrc::channel_t::name

5.193.1.2 int MHAMultiSrc::channel_t::channel

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

- **mha_multisrc.h**

5.194 MHAMultiSrc::channels_t Class Reference

Inherits vector< MHAMultiSrc::channel_t >.

Public Member Functions

- **channels_t** (const std::vector< std::string > &src, int in_channels)
Separate a list of input sources into a parsable channel list.

5.194.1 Constructor & Destructor Documentation

5.194.1.1 MHAMultiSrc::channels_t::channels_t (
const std::vector< std::string > &route,
int in_channels)

Separate a list of input sources into a parsable channel list.

The number of input channels if verified, a list of **MHAMultiSrc::channel_t** (p. 543) is filled.

Parameters

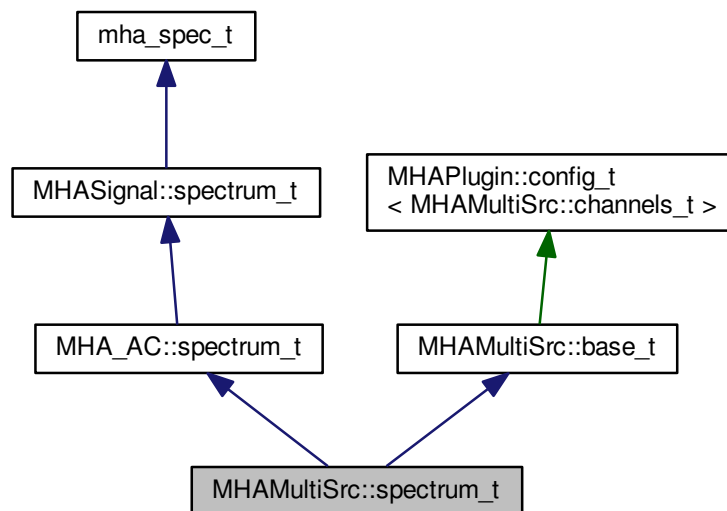
| | |
|--------------------|--|
| <i>route</i> | vector of source channel ids |
| <i>in_channels</i> | number of channels in the processed input signal |

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.195 MHAMultiSrc::spectrum_t Class Reference

Inheritance diagram for MHAMultiSrc::spectrum_t:



Public Member Functions

- **spectrum_t** (**algo_comm_t** iac, std::string **name**, unsigned int frames, unsigned int channels)
- **mha_spec_t * update** (**mha_spec_t** *s)
Update data of spectrum to hold actual input data.

Additional Inherited Members

5.195.1 Constructor & Destructor Documentation

5.195.1.1 MHAMultiSrc::spectrum_t::spectrum_t (
 algo_comm_t iac,
 std::string **name**,
 unsigned int **frames**,
 unsigned int **channels**)

5.195.2 Member Function Documentation

5.195.2.1 mha_spec_t * MHAMultiSrc::spectrum_t::update (
mha_spec_t * s)

Update data of spectrum to hold actual input data.

Parameters

| | |
|---|--------------------|
| s | Input signal chunk |
|---|--------------------|

Returns

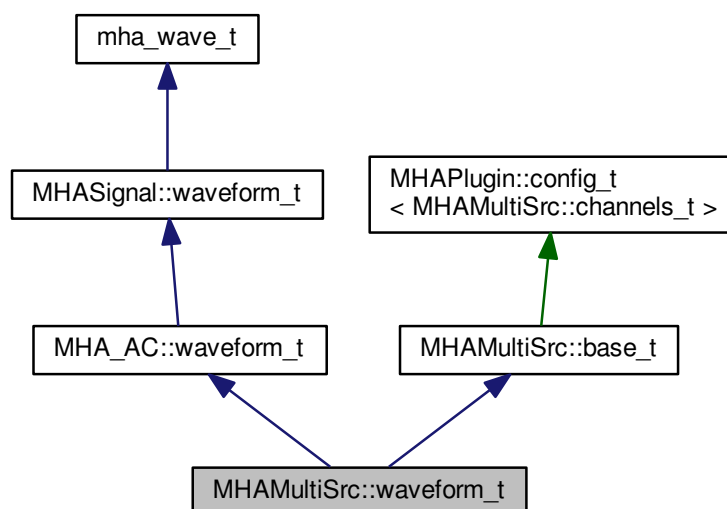
Return pointer to spectrum structure

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

- mha_multisrc.h
- mha_multisrc.cpp

5.196 MHAMultiSrc::waveform_t Class Reference

Inheritance diagram for MHAMultiSrc::waveform_t:



Public Member Functions

- **waveform_t** (**algo_comm_t** iac, std::string **name**, unsigned int frames, unsigned int channels)
- **mha_wave_t * update** (**mha_wave_t** *s)
Update data of waveform to hold actual input data.

Additional Inherited Members

5.196.1 Constructor & Destructor Documentation

5.196.1.1 MHAMultiSrc::waveform_t::waveform_t (
 algo_comm_t iac,
 std::string *name*,
 unsigned int *frames*,
 unsigned int *channels*)

5.196.2 Member Function Documentation

5.196.2.1 **mha_wave_t * MHAMultiSrc::waveform_t::update** (
 mha_wave_t * **s**)

Update data of waveform to hold actual input data.

Parameters

| | |
|----------|--------------------|
| s | Input signal chunk |
|----------|--------------------|

Returns

Return pointer to waveform structure

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

5.197 MHAOviFilter::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`

5.197.1 Member Data Documentation

5.197.1.1 `mha_real_t` MHAOvIFilter::band_descriptor_t::cf_l

5.197.1.2 `mha_real_t` MHAOvIFilter::band_descriptor_t::ef_l

5.197.1.3 `mha_real_t` MHAOvIFilter::band_descriptor_t::cf

5.197.1.4 `mha_real_t` MHAOvIFilter::band_descriptor_t::ef_h

5.197.1.5 `mha_real_t` MHAOvIFilter::band_descriptor_t::cf_h

5.197.1.6 `bool` MHAOvIFilter::band_descriptor_t::low_side_flat

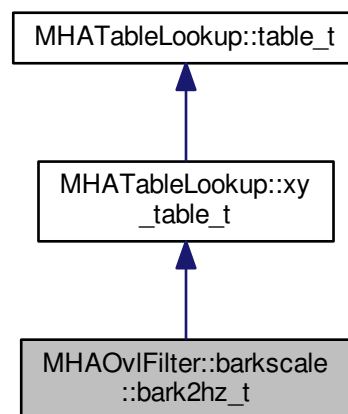
5.197.1.7 `bool` MHAOvIFilter::band_descriptor_t::high_side_flat

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

- `mha_fftfb.hh`

5.198 MHAOvIFilter::barkscale::bark2hz_t Class Reference

Inheritance diagram for MHAOvIFilter::barkscale::bark2hz_t:



Public Member Functions

- **bark2hz_t ()**
- **~bark2hz_t ()**

Additional Inherited Members

5.198.1 Constructor & Destructor Documentation

5.198.1.1 MHAOvIFilter::barkscale::bark2hz_t::bark2hz_t ()

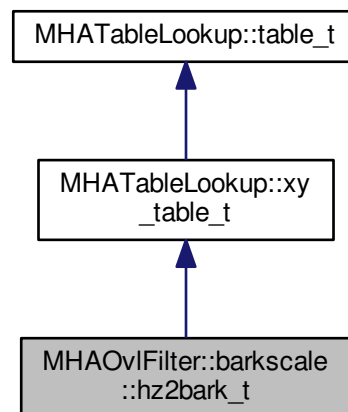
5.198.1.2 MHAOvIFilter::barkscale::bark2hz_t::~~bark2hz_t ()

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

- **mha_fftfb.cpp**

5.199 MHAOvIFilter::barkscale::hz2bark_t Class Reference

Inheritance diagram for MHAOvIFilter::barkscale::hz2bark_t:



Public Member Functions

- **hz2bark_t ()**
- **~hz2bark_t ()**

Additional Inherited Members

5.199.1 Constructor & Destructor Documentation

5.199.1.1 MHAOvFilter::barkscale::hz2bark_t::hz2bark_t ()

5.199.1.2 MHAOvFilter::barkscale::hz2bark_t::~~hz2bark_t ()

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

- **mha_fftfb.cpp**

5.200 MHAOvFilter::fftfb_ac_info_t Class Reference

Public Member Functions

- **fftfb_ac_info_t** (const **MHAOvFilter::fftfb_t** &fb, **algo_comm_t** ac, const std::string &prefix)
- void **insert** ()

Private Attributes

- **MHA_AC::waveform_t** cfv
vector of nominal center frequencies / Hz
- **MHA_AC::waveform_t** efv
vector of edge frequencies / Hz
- **MHA_AC::waveform_t** bwv
vector of band-weights (sum of squared fft-bin-weights)/num_frames
- **MHA_AC::waveform_t** cLTASS
vector of LTASS correction

5.200.1 Constructor & Destructor Documentation

5.200.1.1 MHAOvFilter::fftfb_ac_info_t::fftfb_ac_info_t (
const MHAOvFilter::fftfb_t &fb,
algo_comm_t ac,
const std::string &prefix)

5.200.2 Member Function Documentation

5.200.2.1 void MHAOvFilter::fftfb_ac_info_t::insert ()

5.200.3 Member Data Documentation

5.200.3.1 **MHA_AC::waveform_t** MHAOvFilter::fftfb_ac_info_t::cfv [private]

vector of nominal center frequencies / Hz

5.200.3.2 MHA_AC::waveform_t MHAOvfFilter::fftfb_ac_info_t::efv [private]

vector of edge frequencies / Hz

5.200.3.3 MHA_AC::waveform_t MHAOvfFilter::fftfb_ac_info_t::bwv [private]

vector of band-weights (sum of squared fft-bin-weights)/num_frames

5.200.3.4 MHA_AC::waveform_t MHAOvfFilter::fftfb_ac_info_t::cLTASS [private]

vector of LTASS correction

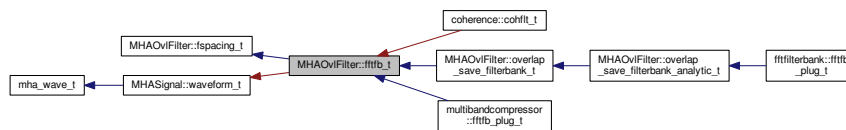
The documentation for this class was generated from the following files:

- mha_fftfb.hh
- mha_fftfb.cpp

5.201 MHAOvfFilter::fftfb_t Class Reference

FFT based overlapping filter bank.

Inheritance diagram for MHAOvfFilter::fftfb_t:



Public Member Functions

- **fftfb_t** (MHAOvfFilter::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_fbpow** (mha_wave_t *fbpow, const mha_spec_t *s_in)
- void **get_fbpow_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 **fftl**
- **mha_real_t** **samplingrate**

Additional Inherited Members

5.201.1 Detailed Description

FFT based overlapping filter bank.

5.201.2 Constructor & Destructor Documentation

5.201.2.1 MHAOvIFilter::fftfb_t::fftfb_t (
MHAOvIFilter::fftfb_vars_t & *par*,
unsigned int *nfft*,
mha_real_t *fs*)

Constructor for a FFT-based overlapping filter bank.

Parameters

| | |
|-------------|--|
| <i>par</i> | Parameters for the FFT filterbank that can not be deduced from the signal dimensions are taken from this set of configuration variables. |
| <i>nfft</i> | FFT length |
| <i>fs</i> | Sampling rate / Hz |

5.201.2.2 MHAOvIFilter::fftfb_t::~~fftfb_t ()

5.201.3 Member Function Documentation

5.201.3.1 void MHAOvIFilter::fftfb_t::apply_gains (
mha_spec_t * *s_out*,
const mha_spec_t * *s_in*,
const mha_wave_t * *gains*)

5.201.3.2 void MHAOvIFilter::fftfb_t::get_fbpower (
mha_wave_t * *fbpow*,
const mha_spec_t * *s_in*)

5.201.3.3 void MHAOvFilter::fftfb_t::get_fbpower_db (
 mha_wave_t * *fbpow*,
 const mha_spec_t * *s_in*)

5.201.3.4 std::vector< float > MHAOvFilter::fftfb_t::get_ltass_gain_db () const

5.201.3.5 unsigned int MHAOvFilter::fftfb_t::bin1 (
 unsigned int *band*) const [inline]

Return index of first non-zero filter shape window.

5.201.3.6 unsigned int MHAOvFilter::fftfb_t::bin2 (
 unsigned int *band*) const [inline]

Return index of first zero filter shape window above center frequency.

5.201.3.7 unsigned int MHAOvFilter::fftfb_t::get_fftlens () const [inline]

Return fft length.

5.201.3.8 mha_real_t MHAOvFilter::fftfb_t::w (
 unsigned int *k*,
 unsigned int *b*) const [inline]

Return filter shape window at index *k* in band *b*.

Parameters

| | |
|----------|-----------------|
| <i>k</i> | Frequency index |
| <i>b</i> | Band index |

5.201.4 Member Data Documentation

5.201.4.1 unsigned int* MHAOvFilter::fftfb_t::vbin1 [private]

5.201.4.2 unsigned int* MHAOvFilter::fftfb_t::vbin2 [private]

5.201.4.3 mha_real_t(* MHAOvFilter::fftfb_t::shape) (mha_real_t) [private]

5.201.4.4 unsigned int MHAOvFilter::fftfb_t::fftlens [private]

5.201.4.5 mha_real_t MHAOvFilter::fftfb_t::samplingrate [private]

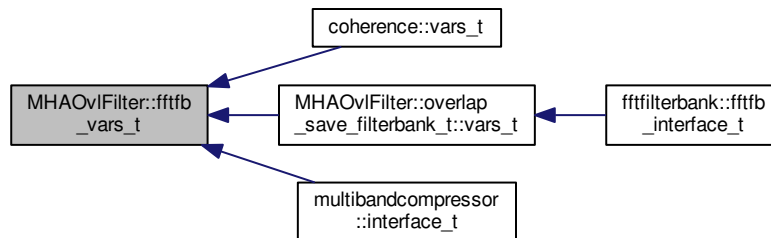
The documentation for this class was generated from the following files:

- mha_fftfb.hh
- mha_fftfb.cpp

5.202 MHAOvIFilter::fftfb_vars_t Class Reference

Set of configuration variables for FFT-based overlapping filters.

Inheritance diagram for MHAOvIFilter::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 whether edge or center frequencies are used.
- **fscale_t f**
Frequency.
- **MHAParser::bool_t normalize**
Normalize sum of channels.
- **MHAParser::bool_t fail_on_nonmonotonic**
Fail if frequency entries are non-monotonic (otherwise sort)
- **MHAParser::bool_t fail_on_unique_bins**
Fail if center frequencies share the same FFT bin.
- **MHAParser::vfloat_mon_t cf**
Final center frequencies in Hz.
- **MHAParser::vfloat_mon_t ef**
Final edge frequencies in Hz.
- **MHAParser::vfloat_mon_t cLTASS**
Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)
- **MHAParser::mfloat_mon_t shapes**

5.202.1 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. 553) 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.

5.202.2 Constructor & Destructor Documentation

5.202.2.1 MHAOvFilter::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

| | |
|----------|---|
| <i>p</i> | The node of the configuration tree where the variables created by this instance are inserted. |
|----------|---|

5.202.3 Member Data Documentation

5.202.3.1 scale_var_t MHAOvFilter::fftfb_vars_t::fscale

Frequency scale type (lin/bark/log/erb).

5.202.3.2 scale_var_t MHAOvFilter::fftfb_vars_t::ovltype

Filter shape (rect/lin/hann).

5.202.3.3 MHAParser::float_t MHAOvFilter::fftfb_vars_t::plateau

relative plateau width.

5.202.3.4 MHAParser::kw_t MHAOvFilter::fftfb_vars_t::ftype

Flag to decide whether edge or center frequencies are used.

5.202.3.5 fscale_t MHAOvFilter::fftfb_vars_t::f

Frequency.

5.202.3.6 MHAParser::bool_t MHAOvFilter::fftfb_vars_t::normalize

Normalize sum of channels.

5.202.3.7 MHAParser::bool_t MHAOvFilter::fftfb_vars_t::fail_on_nonmonotonic

Fail if frequency entries are non-monotonic (otherwise sort)

5.202.3.8 MHAParser::bool_t MHAOvFilter::fftfb_vars_t::fail_on_unique_bins

Fail if center frequencies share the same FFT bin.

5.202.3.9 MHAParser::vfloat_mon_t MHAOvFilter::fftfb_vars_t::cf

Final center frequencies in Hz.

5.202.3.10 MHAParser::vfloat_mon_t MHAOvFilter::fftfb_vars_t::ef

Final edge frequencies in Hz.

5.202.3.11 MHAParser::vfloat_mon_t MHAOvFilter::fftfb_vars_t::cLTASS

Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)

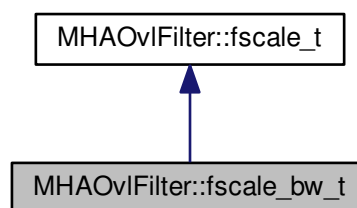
5.202.3.12 MHAParser::mfloat_mon_t MHAOvFilter::fftfb_vars_t::shapes

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.203 MHAOvFilter::fscale_bw_t Class Reference

Inheritance diagram for MHAOvFilter::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

5.203.1 Constructor & Destructor Documentation

5.203.1.1 MHAOvFilter::fscale_bw_t::fscale_bw_t (MHAParser::parser_t & *parent*)

5.203.2 Member Function Documentation

5.203.2.1 std::vector< mha_real_t > MHAOvFilter::fscale_bw_t::get_bw_hz () const

5.203.2.2 void MHAOvFilter::fscale_bw_t::update_hz () [private]

5.203.3 Member Data Documentation

5.203.3.1 MHAParser::vfloat_t MHAOvFilter::fscale_bw_t::bw [protected]

5.203.3.2 MHAParser::vfloat_mon_t MHAOvFilter::fscale_bw_t::bw_hz [protected]

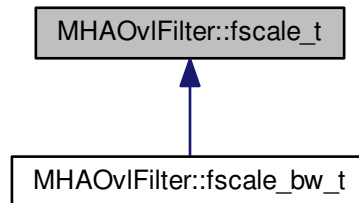
5.203.3.3 MHAEvents::connector_t<fscale_bw_t> MHAOvFilter::fscale_bw_t::updater [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.204 MHAOvFilter::fscale_t Class Reference

Inheritance diagram for MHAOvFilter::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**

5.204.1 Constructor & Destructor Documentation

5.204.1.1 MHAOvFilter::fscale_t::fscale_t (MHAParser::parser_t & parent)

5.204.2 Member Function Documentation

5.204.2.1 std::vector< mha_real_t > MHAOvFilter::fscale_t::get_f_hz () const

5.204.2.2 void MHAOvFilter::fscale_t::update_hz () [private]

5.204.3 Member Data Documentation

5.204.3.1 scale_var_t MHAOvFilter::fscale_t::unit

5.204.3.2 MHAParser::vfloat_t MHAOvFilter::fscale_t::f

5.204.3.3 MHAParser::vfloat_mon_t MHAOvFilter::fscale_t::f_hz

5.204.3.4 MHAEvents::connector_t<fscale_t> MHAOvFilter::fscale_t::updater [private]

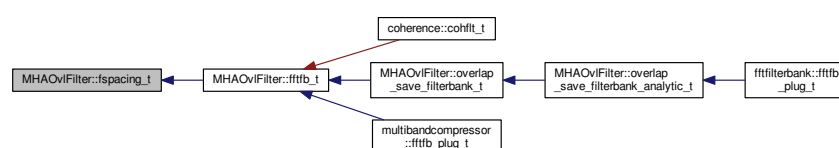
The documentation for this class was generated from the following files:

- mha_fftfb.hh
- mha_fftfb.cpp

5.205 MHAOvFilter::fspacing_t Class Reference

Class for frequency spacing, used by filterbank shape generator class.

Inheritance diagram for MHAOvFilter::fspacing_t:



Public Member Functions

- **fspacing_t** (const **MHAOvFilter::fftfb_vars_t** &par, unsigned int nfft, **mha_real_t** fs)
- std::vector< unsigned int > **get_cf_fftbins** () 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< **MHAOvFilter::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_**

5.205.1 Detailed Description

Class for frequency spacing, used by filterbank shape generator class.

5.205.2 Constructor & Destructor Documentation

5.205.2.1 `MHAOvFilter::fspacing_t::fspacing_t (`
 `const MHAOvFilter::fftfb_vars_t & par,`
 `unsigned int nfft,`
 `mha_real_t fs)`

5.205.3 Member Function Documentation

5.205.3.1 `std::vector< unsigned int > MHAOvFilter::fspacing_t::get_cf_fftbins () const`

5.205.3.2 `std::vector< mha_real_t > MHAOvFilter::fspacing_t::get_cf_hz () const`

5.205.3.3 `std::vector< mha_real_t > MHAOvFilter::fspacing_t::get_ef_hz () const`

5.205.3.4 `unsigned int MHAOvFilter::fspacing_t::nbands () const` `[inline]`

Return number of bands in filter bank.

5.205.3.5 `void MHAOvFilter::fspacing_t::fail_on_nonmonotonic_cf ()` `[protected]`

5.205.3.6 `void MHAOvFilter::fspacing_t::fail_on_unique_fftbins ()` `[protected]`

5.205.3.7 `void MHAOvFilter::fspacing_t::ef2bands (`
 `std::vector< mha_real_t > vef)` `[private]`

5.205.3.8 `void MHAOvFilter::fspacing_t::cf2bands (`
 `std::vector< mha_real_t > vcf)` `[private]`

5.205.3.9 `void MHAOvFilter::fspacing_t::equidist2bands (`
 `std::vector< mha_real_t > vcf)` `[private]`

5.205.4 Member Data Documentation

5.205.4.1 `std::vector<MHAOvFilter::band_descriptor_t> MHAOvFilter::fspacing_t::bands`
`[protected]`

5.205.4.2 `mha_real_t(* MHAOvFilter::fspacing_t::symmetry_scale)(mha_real_t)` `[protected]`

5.205.4.3 `unsigned int MHAOvFilter::fspacing_t::nfft_` `[private]`

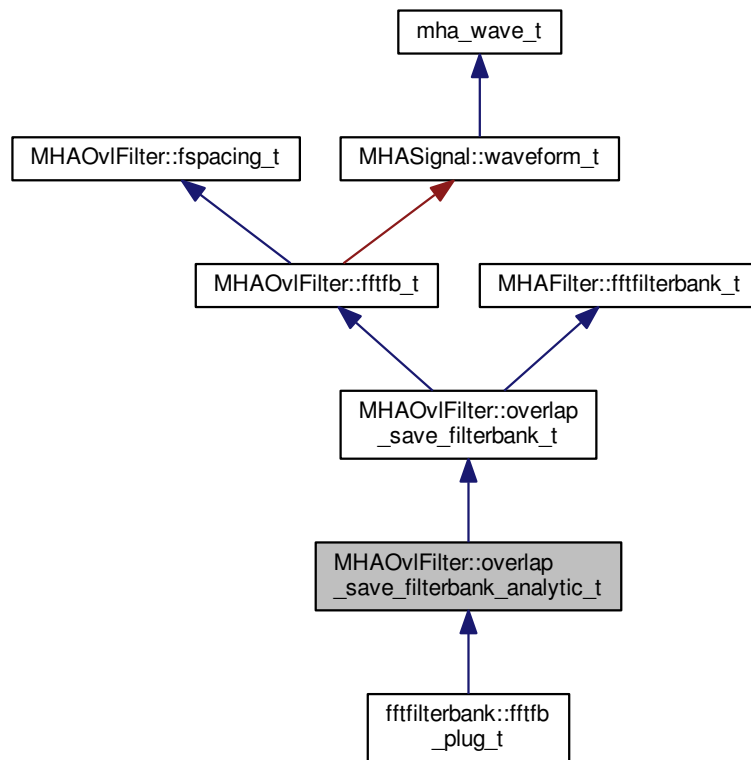
5.205.4.4 `mha_real_t MHAOvFilter::fspacing_t::fs_` `[private]`

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

5.206 MHAOvFilter::overlap_save_filterbank_analytic_t Class Reference

Inheritance diagram for MHAOvFilter::overlap_save_filterbank_analytic_t:



Public Member Functions

- **overlap_save_filterbank_analytic_t** (MHAOvFilter::overlap_save_filterbank_t↵
::vars_t &fbpar, mhaconfig_t channelconfig_in)
- void **filter_analytic** (const mha_wave_t *sIn, mha_wave_t **fltRe, mha_wave_t **flt↵
Im)

Private Attributes

- MHAFilter::fftfbfilterbank_t imagfb

Additional Inherited Members

5.206.1 Constructor & Destructor Documentation

5.206.1.1 **MHAOvFilter::overlap_save_filterbank_analytic_t::overlap_save_filterbank_analytic_t (**
MHAOvFilter::overlap_save_filterbank_t::vars_t & *fbpar*,
mhaconfig_t *channelconfig_in*)

5.206.2 Member Function Documentation

5.206.2.1 **void MHAOvFilter::overlap_save_filterbank_analytic_t::filter_analytic (**
const mha_wave_t * *sIn*,
mha_wave_t ** *fltRe*,
mha_wave_t ** *fltIm*)

5.206.3 Member Data Documentation

5.206.3.1 **MHAFilter::fftfilterbank_t MHAOvFilter::overlap_save_filterbank_analytic_t::imagfb**
 [private]

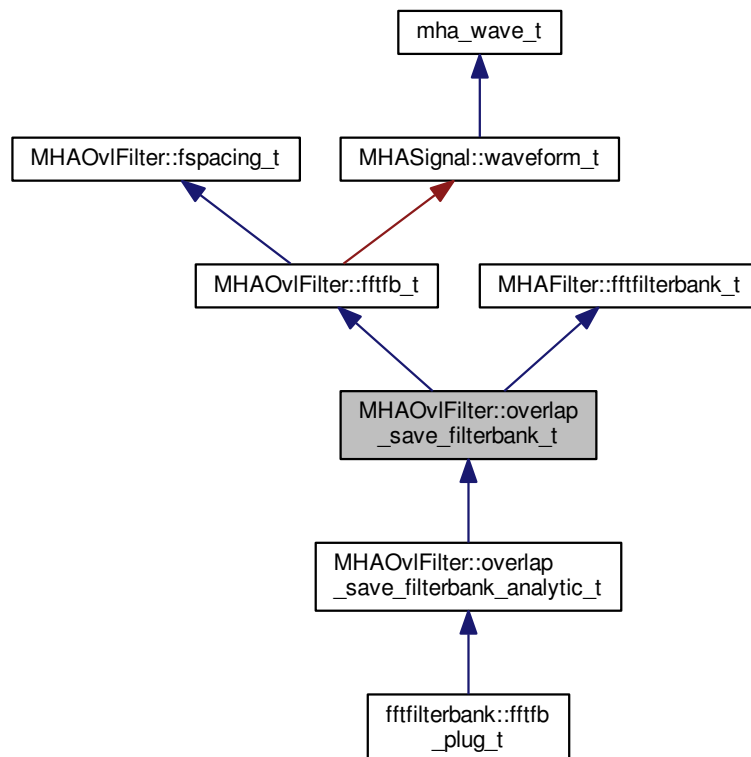
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.207 MHAOvFilter::overlap_save_filterbank_t Class Reference

A time-domain minimal phase filter bank with frequency shapes from **MHAOvFilter::fftfb_t** (p. [550](#)).

Inheritance diagram for MHAOvfFilter::overlap_save_filterbank_t:



Classes

- class **vars_t**

Public Member Functions

- **overlap_save_filterbank_t** (MHAOvfFilter::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

5.207.1 Detailed Description

A time-domain minimal phase filter bank with frequency shapes from **MHAovIFilter::fftfb_t** (p. 550).

5.207.2 Constructor & Destructor Documentation

5.207.2.1 `MHAOvIFilter::overlap_save_filterbank_t::overlap_save_filterbank_t (MHAOvIFilter::overlap_save_filterbank_t::vars_t & fbpar, mhaconfig_t channelconfig_in)`

5.207.3 Member Function Documentation

5.207.3.1 `mhaconfig_t MHAOvIFilter::overlap_save_filterbank_t::get_channelconfig () const`
[inline]

5.207.4 Member Data Documentation

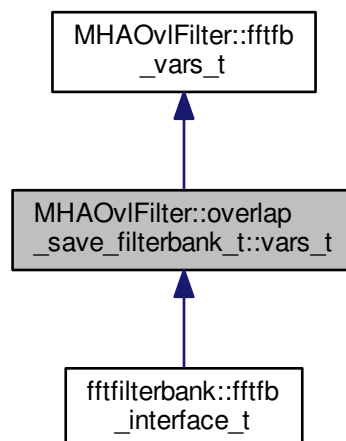
5.207.4.1 `mhaconfig_t MHAOvIFilter::overlap_save_filterbank_t::channelconfig_out_` [private]

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

5.208 MHAOvIFilter::overlap_save_filterbank_t::vars_t Class Reference

Inheritance diagram for `MHAOvIFilter::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**

5.208.1 Constructor & Destructor Documentation

5.208.1.1 MHAOvIFilter::overlap_save_filterbank_t::vars_t::vars_t (MHAParser::parser_t & p)

5.208.2 Member Data Documentation

5.208.2.1 MHAParser::int_t MHAOvIFilter::overlap_save_filterbank_t::vars_t::fftlen

5.208.2.2 MHAParser::kw_t MHAOvIFilter::overlap_save_filterbank_t::vars_t::phasemodel

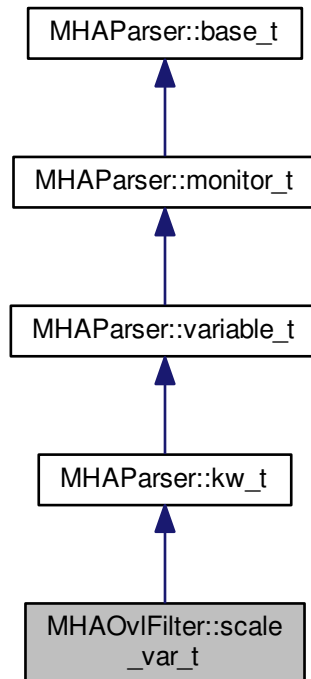
5.208.2.3 MHAParser::window_t MHAOvIFilter::overlap_save_filterbank_t::vars_t::irswnd

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

5.209 MHAOvFilter::scale_var_t Class Reference

Inheritance diagram for MHAOvFilter::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** * > **funcs**

Additional Inherited Members

5.209.1 Constructor & Destructor Documentation

5.209.1.1 MHAOvIFilter::scale_var_t::scale_var_t (
const std::string & *help*)

5.209.2 Member Function Documentation

5.209.2.1 void MHAOvIFilter::scale_var_t::add_fun (
const std::string & *name*,
scale_fun_t * *fun*)

5.209.2.2 std::string MHAOvIFilter::scale_var_t::get_name () const [inline]

5.209.2.3 scale_fun_t* MHAOvIFilter::scale_var_t::get_fun () const [inline]

5.209.2.4 mha_real_t MHAOvIFilter::scale_var_t::hz2unit (
mha_real_t *x*) const

5.209.2.5 mha_real_t MHAOvIFilter::scale_var_t::unit2hz (
mha_real_t *x*) const

5.209.3 Member Data Documentation

5.209.3.1 std::vector<std::string> MHAOvIFilter::scale_var_t::names [private]

5.209.3.2 std::vector<scale_fun_t*> MHAOvIFilter::scale_var_t::funs [private]

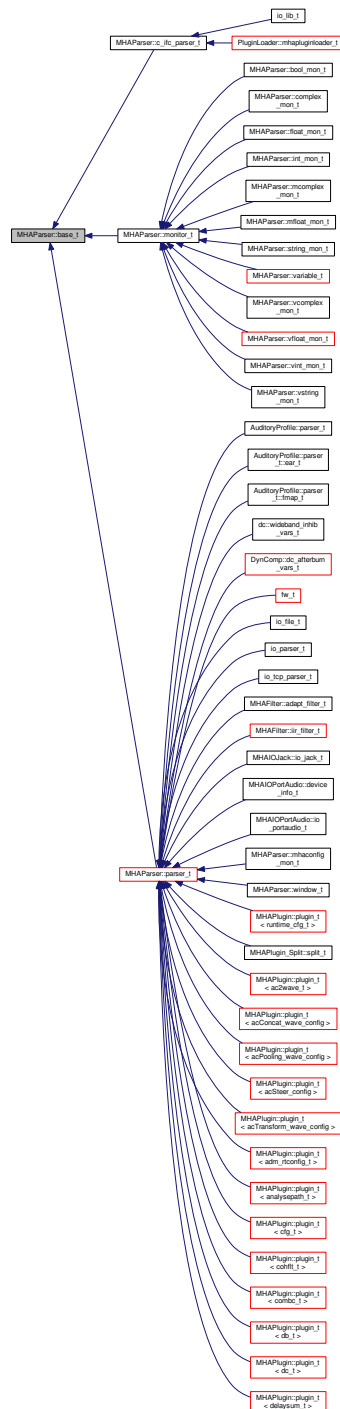
The documentation for this class was generated from the following files:

- mha_fftfb.hh
- mha_fftfb.cpp

5.210 MHParse::base_t Class Reference

Base class for all parser items.

Inheritance diagram for MHParse::base_t:



Classes

- class `replace_t`

Public Member Functions

- **base_t** (const std::string &)
Constructor for base class of all parser nodes.
- **base_t** (const **base_t** &)
- virtual ~**base_t** ()
- virtual std::string **parse** (const std::string &)
Causes this node to process a command in the openMHA configuration language.
- virtual void **parse** (const char *, char *, unsigned int)
This function parses a command and writes the parsing result into a C character array.
- virtual void **parse** (const std::vector< std::string > &, std::vector< std::string > &)
- virtual std::string **op_subparse** (**expression_t** &)
- virtual std::string **op_setval** (**expression_t** &)
- virtual std::string **op_query** (**expression_t** &)
- virtual std::string **query_dump** (const std::string &)
- virtual std::string **query_entries** (const std::string &)
- virtual std::string **query_perm** (const std::string &)
- virtual std::string **query_range** (const std::string &)
- virtual std::string **query_type** (const std::string &)
- virtual std::string **query_val** (const std::string &)
- virtual std::string **query_readfile** (const std::string &)
- virtual std::string **query_savefile** (const std::string &)
- virtual std::string **query_savefile_compact** (const std::string &)
- virtual std::string **query_savemons** (const std::string &)
- virtual std::string **query_listids** (const std::string &)
- std::string **query_version** (const std::string &)
- std::string **query_id** (const std::string &)
- std::string **query_subst** (const std::string &)
- std::string **query_addsubst** (const std::string &)
- std::string **query_help** (const std::string &)
- std::string **query_cmds** (const std::string &)
- void **set_node_id** (const std::string &)
Set the identification string of this parser node.
- void **set_help** (const std::string &)
Set the help comment of a variable or parser.
- void **add_parent_on_insert** (**parser_t** *, std::string)
- void **rm_parent_on_remove** (**parser_t** *)
- const std::string & **fullname** () const
Return the full dot-separated path name of this parser node in the openMHA configuration tree.

Public Attributes

- **MHAEvents::emitter_t** writeaccess
Event emitted on write access.
- **MHAEvents::emitter_t** valuechanged
Event emitted if the value has changed.

- **MHAEvents::emitter_t readaccess**

Event emitted on read access.

- **MHAEvents::emitter_t prereadaccess**

Event emitted on read access, before the data field is accessed.

Protected Member Functions

- void **activate_query** (const std::string &, **query_t**)
- void **notify** ()

Protected Attributes

- **query_map_t** queries
- bool **data_is_initialized**

Private Types

- typedef std::vector< **replace_t** > **repl_list_t**

Private Member Functions

- void **add_replace_pair** (const std::string &, const std::string &)
- std::string **oplist** ()

Private Attributes

- std::string **help**
- std::string **id_str**
- **opact_map_t** operators
- **repl_list_t** repl_list
- bool **nested_lock**
- **parser_t** * parent
- std::string **thefullname**

5.210.1 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. [571](#)) method. Parser proxy derivatives which overwrite any of the other **parse()** (p. [571](#)) methods to be the key method must make sure that the original **parse()** (p. [571](#)) method utilizes the new key method.

5.210.2 Member Typedef Documentation

5.210.2.1 `typedef std::vector<replace_t> MHAParser::base_t::repl_list_t` `[private]`

5.210.3 Constructor & Destructor Documentation

5.210.3.1 `MHAParser::base_t::base_t (`
`const std::string & h)`

Constructor for base class of all parser nodes.

Parameters

| | |
|----------|--|
| <i>h</i> | Help text describing this parser node. This help text is accessible to the configuration language through the "?help" query command. |
|----------|--|

5.210.3.2 `MHAParser::base_t::base_t (`
`const base_t & src)`

5.210.3.3 `MHAParser::base_t::~~base_t ()` `[virtual]`

5.210.4 Member Function Documentation

5.210.4.1 `std::string MHAParser::base_t::parse (`
`const std::string & cs)` `[virtual]`

Causes this node to process a command in the openMHA configuration language.

Parameters

| | |
|-----------|----------------------|
| <i>cs</i> | The command to parse |
|-----------|----------------------|

Returns

The response to the command, if successful

Exceptions

| | |
|---|---|
| <i>MHA_Error</i> (p. 387) | If the command cannot be executed successfully. The reason for failure is given in the message string of the exception. |
|---|---|

Reimplemented in **PluginLoader::mhapluginloader_t** (p. [808](#)), and **altplugins_t** (p. [178](#)).

```
5.210.4.2 void MHAParser::base_t::parse (
            const char * cmd,
            char * retv,
            unsigned int len ) [virtual]
```

This function parses a command and writes the parsing result into a C character array.

This base class implementation delegates to **parse(const std::string &)** (p. 571).

Parameters

| | |
|-------------|-----------------------|
| <i>cmd</i> | Command to be parsed |
| <i>retv</i> | Buffer for the result |
| <i>len</i> | Length of buffer |

Reimplemented in **altplugs_t** (p. 178).

```
5.210.4.3 void MHAParser::base_t::parse (
            const std::vector< std::string > & cs,
            std::vector< std::string > & retv ) [virtual]
```

```
5.210.4.4 std::string MHAParser::base_t::op_subparse (
            expression_t & ) [virtual]
```

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 583), and **MHAParser::parser_t** (p. 623).

```
5.210.4.5 std::string MHAParser::base_t::op_setval (
            expression_t & ) [virtual]
```

Reimplemented in **MHAParser::mcomplex_t** (p. 609), **MHAParser::mfloat_t** (p. 613), **MHAParser::vcomplex_t** (p. 637), **MHAParser::vfloat_t** (p. 641), **MHAParser::vint_t** (p. 645), **MHAParser::complex_t** (p. 589), **MHAParser::float_t** (p. 595), **MHAParser::int_t** (p. 600), **MHAParser::bool_t** (p. 581), **MHAParser::vstring_t** (p. 649), **MHAParser::string_t** (p. 631), **MHAParser::kw_t** (p. 605), **MHAParser::variable_t** (p. 633), **MHAParser::c_ifc_parser_t** (p. 583), and **MHAParser::parser_t** (p. 623).

```
5.210.4.6 std::string MHAParser::base_t::op_query (
            expression_t & ) [virtual]
```

Reimplemented in **MHAParser::monitor_t** (p. 619), **MHAParser::c_ifc_parser_t** (p. 583), and **MHAParser::parser_t** (p. 623).

```
5.210.4.7 std::string MHAParser::base_t::query_dump (
            const std::string & s ) [virtual]
```

Reimplemented in **MHAParser::monitor_t** (p. 620), and **MHAParser::parser_t** (p. 623).

5.210.4.8 `std::string MHAParser::base_t::query_entries (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.9 `std::string MHAParser::base_t::query_perm (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::variable_t** (p. 633), and **MHAParser::monitor_t** (p. 620).

5.210.4.10 `std::string MHAParser::base_t::query_range (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::kw_t** (p. 605), and **MHAParser::range_var_t** (p. 626).

5.210.4.11 `std::string MHAParser::base_t::query_type (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 607), **MHAParser::vcomplex_mon_t** (p. 635), **MHAParser::complex_mon_t** (p. 587), **MHAParser::float_mon_t** (p. 593), **MHAParser::mfloat_mon_t** (p. 611), **MHAParser::vfloat_mon_t** (p. 639), **MHAParser::vint_mon_t** (p. 643), **MHAParser::vstring_mon_t** (p. 647), **MHAParser::string_mon_t** (p. 629), **MHAParser::bool_mon_t** (p. 579), **MHAParser::int_mon_t** (p. 597), **MHAParser::mcomplex_t** (p. 609), **MHAParser::mfloat_t** (p. 613), **MHAParser::vcomplex_t** (p. 637), **MHAParser::vfloat_t** (p. 641), **MHAParser::vint_t** (p. 645), **MHAParser::complex_t** (p. 589), **MHAParser::float_t** (p. 595), **MHAParser::int_t** (p. 600), **MHAParser::bool_t** (p. 581), **MHAParser::vstring_t** (p. 649), **MHAParser::string_t** (p. 631), **MHAParser::kw_t** (p. 605), and **MHAParser::parser_t** (p. 623).

5.210.4.12 `std::string MHAParser::base_t::query_val (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 607), **MHAParser::vcomplex_mon_t** (p. 635), **MHAParser::complex_mon_t** (p. 587), **MHAParser::float_mon_t** (p. 593), **MHAParser::mfloat_mon_t** (p. 611), **MHAParser::vfloat_mon_t** (p. 639), **MHAParser::vint_mon_t** (p. 643), **MHAParser::vstring_mon_t** (p. 647), **MHAParser::string_mon_t** (p. 629), **MHAParser::bool_mon_t** (p. 579), **MHAParser::int_mon_t** (p. 597), **MHAParser::mcomplex_t** (p. 609), **MHAParser::mfloat_t** (p. 613), **MHAParser::vcomplex_t** (p. 637), **MHAParser::vfloat_t** (p. 641), **MHAParser::vint_t** (p. 645), **MHAParser::complex_t** (p. 589), **MHAParser::float_t** (p. 595), **MHAParser::int_t** (p. 600), **MHAParser::bool_t** (p. 581), **MHAParser::vstring_t** (p. 649), **MHAParser::string_t** (p. 631), **MHAParser::kw_t** (p. 605), and **MHAParser::parser_t** (p. 624).

5.210.4.13 `std::string MHAParser::base_t::query_readfile (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.14 `std::string MHAParser::base_t::query_savefile (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.15 `std::string MHAParser::base_t::query_savefile_compact (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.16 `std::string MHAParser::base_t::query_savemons (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.17 `std::string MHAParser::base_t::query_listids (`
`const std::string & s) [virtual]`

Reimplemented in **MHAParser::parser_t** (p. 624).

5.210.4.18 `std::string MHAParser::base_t::query_version (`
`const std::string &)`

5.210.4.19 `std::string MHAParser::base_t::query_id (`
`const std::string &)`

5.210.4.20 `std::string MHAParser::base_t::query_subst (`
`const std::string & s)`

5.210.4.21 `std::string MHAParser::base_t::query_addsubst (`
`const std::string & s)`

5.210.4.22 `std::string MHAParser::base_t::query_help (`
`const std::string & s)`

5.210.4.23 `std::string MHAParser::base_t::query_cmds (`
`const std::string & s)`

5.210.4.24 `void MHAParser::base_t::set_node_id (`
`const std::string & s)`

Set the identification string of this parser node.

The id can be queried from the configuration language using the ?id query command. Nodes can be found by id using the ?listid query command on a containing parser node.

Parameters

| | |
|----------|--------------------------------|
| s | The new identification string. |
|----------|--------------------------------|

5.210.4.25 void MHAParser::base_t::set_help (
const std::string & s)

Set the help comment of a variable or parser.

Parameters

| | |
|----------|-------------------|
| s | New help comment. |
|----------|-------------------|

5.210.4.26 void MHAParser::base_t::add_parent_on_insert (
parser_t * p,
std::string n)

5.210.4.27 void MHAParser::base_t::rm_parent_on_remove (
parser_t *)

5.210.4.28 const std::string & MHAParser::base_t::fullname () const

Return the full dot-separated path name of this parser node in the openMHA configuration tree.

5.210.4.29 void MHAParser::base_t::activate_query (
const std::string & n,
query_t a) [protected]

5.210.4.30 void MHAParser::base_t::notify () [protected]

5.210.4.31 void MHAParser::base_t::add_replace_pair (
const std::string & a,
const std::string & b) [private]

5.210.4.32 std::string MHAParser::base_t::oplist () [private]

5.210.5 Member Data Documentation

5.210.5.1 MHAEvents::emitter_t MHAParser::base_t::writeaccess

Event emitted on write access.

To connect a callback that is invoked on write access to this parser variable, use MHAEvents::patchbay_t<receiver_t> method connect(&writeaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

5.210.5.2 MHAEvents::emitter_t MHParse::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.

5.210.5.3 MHAEvents::emitter_t MHParse::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.

5.210.5.4 MHAEvents::emitter_t MHParse::base_t::preadaccess

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(&preadaccess,&receiver_t::callback)` where `callback` is a method that expects no parameters and returns void.

5.210.5.5 query_map_t MHParse::base_t::queries [protected]**5.210.5.6 bool MHParse::base_t::data_is_initialized** [protected]**5.210.5.7 std::string MHParse::base_t::help** [private]**5.210.5.8 std::string MHParse::base_t::id_str** [private]**5.210.5.9 opact_map_t MHParse::base_t::operators** [private]**5.210.5.10 repl_list_t MHParse::base_t::repl_list** [private]**5.210.5.11 bool MHParse::base_t::nested_lock** [private]**5.210.5.12 parser_t* MHParse::base_t::parent** [private]**5.210.5.13 std::string MHParse::base_t::thefullname** [private]

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.211 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**

5.211.1 Constructor & Destructor Documentation

5.211.1.1 MHAParser::base_t::replace_t (
const std::string & *ia*,
const std::string & *ib*)

5.211.2 Member Function Documentation

5.211.2.1 void MHAParser::base_t::replace_t::replace (
std::string & *s*)

5.211.2.2 const std::string& MHAParser::base_t::replace_t::get_a () const [inline]

5.211.2.3 const std::string& MHAParser::base_t::replace_t::get_b () const [inline]

5.211.3 Member Data Documentation

5.211.3.1 std::string MHAParser::base_t::replace_t::a [private]

5.211.3.2 std::string MHAParser::base_t::replace_t::b [private]

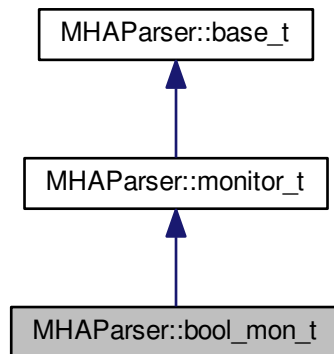
The documentation for this class was generated from the following files:

- mha_parser.hh
- mha_parser.cpp

5.212 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

5.212.1 Detailed Description

Monitor with string value.

5.212.2 Constructor & Destructor Documentation

5.212.2.1 MHAParser::bool_mon_t::bool_mon_t (const std::string & hlp)

Create a monitor variable for string values.

Parameters

| | |
|-------------|---|
| <i>help</i> | A help text describing this monitor variable. |
|-------------|---|

5.212.3 Member Function Documentation

5.212.3.1 `std::string MHAParser::bool_mon_t::query_val (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.212.3.2 `std::string MHAParser::bool_mon_t::query_type (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.212.4 Member Data Documentation

5.212.4.1 `bool MHAParser::bool_mon_t::data`

Data field.

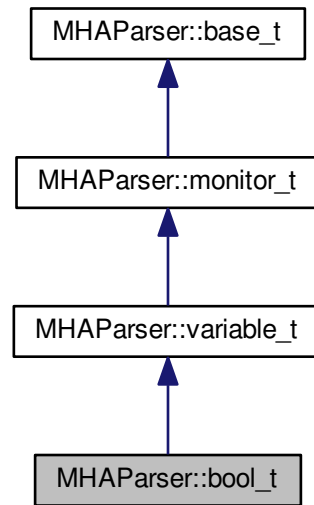
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.213 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

5.213.1 Detailed Description

Variable with a boolean value ("yes"/"no")

5.213.2 Constructor & Destructor Documentation

5.213.2.1 MHAParser::bool_t::bool_t (const std::string & *help_text*, const std::string & *initial_value*)

Constructor for a configuration language variable for boolean values.

Parameters

| | |
|----------------------|--|
| <i>help_text</i> | A human-readable text describing the purpose of this configuration variable. |
| <i>initial_value</i> | 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". |

5.213.3 Member Function Documentation

5.213.3.1 std::string MHAParser::bool_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.213.3.2 std::string MHAParser::bool_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.213.3.3 std::string MHAParser::bool_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.213.4 Member Data Documentation

5.213.4.1 bool MHAParser::bool_t::data

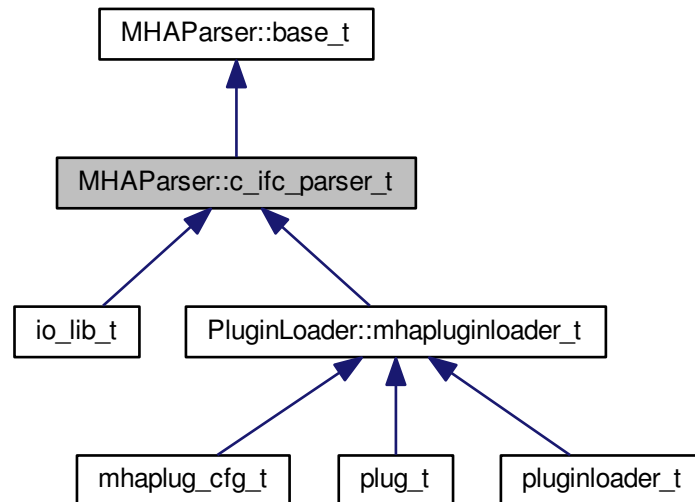
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.214 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 &module_name_)
- **~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

5.214.1 Constructor & Destructor Documentation

5.214.1.1 MHAParser::c_ifc_parser_t::c_ifc_parser_t (const std::string & *modulename*)

5.214.1.2 MHAParser::c ifc_parser t::~c ifc_parser t ()

5.214.2 Member Function Documentation

```
5.214.2.1 void MHAParser::c_ifc_parser_t::set_parse_cb (
                MHAParser::c_parse_cmd_t cb,
                MHAParser::c_parse_err_t strerr,
                void * d )
```

```
5.214.2.2  std::string MHAParser::c_ifc_parser_t::op_subparse (
            MHAParser::expression t & x ) [protected],[virtual]
```

Reimplemented from **MHAParser::base_t** (p. 572).

```
5.214.2.3  std::string MHAParser::c_ifc_parser_t::op_setval (
            MHAParser::expression t & x ) [protected],[virtual]
```

Reimplemented from **MHAParser::base_t** (p. 572).

```
5.214.2.4 std::string MHAParser::c_ifc_parser_t::op_query (
    MHAParser::expression t & x ) [protected],[virtual]
```

Reimplemented from **MHAParser::base_t** (p. 572).

5.214.2.5 void MHAParser::c_ifc_parser_t::test_error () [private]

5.214.3 Member Data Documentation

5.214.3.1 std::string MHAParser::c_ifc_parser_t::modulename [private]

5.214.3.2 c_parse_cmd_t MHAParser::c_ifc_parser_t::c_parse_cmd [private]

5.214.3.3 c_parse_err_t MHAParser::c_ifc_parser_t::c_parse_err [private]

5.214.3.4 int MHAParser::c_ifc_parser_t::liberr [private]

5.214.3.5 void* MHAParser::c_ifc_parser_t::libdata [private]

5.214.3.6 unsigned int MHAParser::c_ifc_parser_t::ret_size [private]

5.214.3.7 char* MHAParser::c_ifc_parser_t::retv [private]

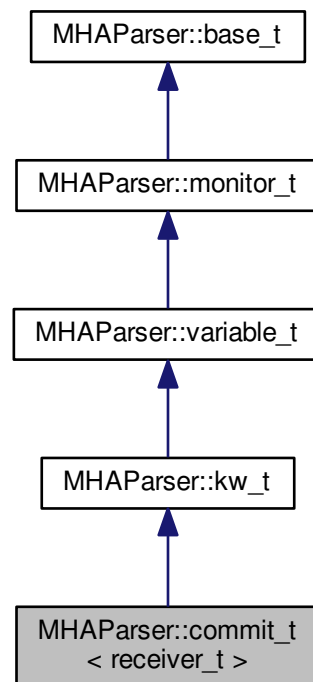
The documentation for this class was generated from the following files:

- mha_parser.hh
- mha_parser.cpp

5.215 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

5.215.1 Detailed Description

```

template<class receiver_t>
class MHAParser::commit_t< receiver_t >

```

Parser variable with event-emission functionality.

The **commit_t** (p. 584) variable can register an event receiver in its constructor, which is called whenever the variable is set to "commit".

5.215.2 Constructor & Destructor Documentation

5.215.2.1 `template<class receiver_t> MHAParser::commit_t< receiver_t >::commit_t (receiver_t * r, void(receiver_t::*)() rfun, const std::string & help = "Variable changes action")`

5.215.3 Member Data Documentation

5.215.3.1 `template<class receiver_t> MHAEvents::connector_t<receiver_t> MHAParser::commit_t< receiver_t >::extern_connector [private]`

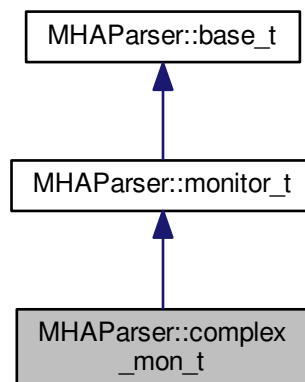
The documentation for this class was generated from the following file:

- `mha_parser.hh`

5.216 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

5.216.1 Detailed Description

Monitor with complex value.

5.216.2 Constructor & Destructor Documentation

5.216.2.1 MHAParser::complex_mon_t::complex_mon_t (const std::string & *hlp*)

Create a complex monitor variable.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.216.3 Member Function Documentation

5.216.3.1 std::string MHAParser::complex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.216.3.2 std::string MHAParser::complex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.216.4 Member Data Documentation

5.216.4.1 mha_complex_t MHAParser::complex_mon_t::data

Data field.

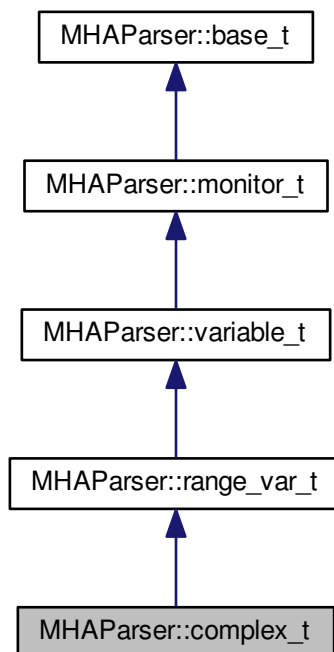
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.217 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

5.217.1 Detailed Description

Variable with complex value.

5.217.2 Constructor & Destructor Documentation

5.217.2.1 `MHAParser::complex_t::complex_t (`
 `const std::string & h,`
 `const std::string & v,`
 `const std::string & rg = " ")`

5.217.3 Member Function Documentation

5.217.3.1 `std::string MHAParser::complex_t::op_setval (`
 `expression_t & x)` `[protected]`, `[virtual]`

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.217.3.2 `std::string MHAParser::complex_t::query_type (`
 `const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.217.3.3 `std::string MHAParser::complex_t::query_val (`
 `const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.217.4 Member Data Documentation

5.217.4.1 `mha_complex_t` `MHAParser::complex_t::data`

Data field.

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

- `mha_parser.hh`
- `mha_parser.cpp`

5.218 `MHAParser::entry_t` Class Reference

Public Member Functions

- `entry_t` (`const std::string &`, `base_t *`)

Public Attributes

- `std::string name`
- `base_t * entry`

5.218.1 Constructor & Destructor Documentation

5.218.1.1 `MHAParser::entry_t::entry_t` (`const std::string & n`, `base_t * e`)

5.218.2 Member Data Documentation

5.218.2.1 `std::string` `MHAParser::entry_t::name`

5.218.2.2 `base_t*` `MHAParser::entry_t::entry`

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

- `mha_parser.hh`
- `mha_parser.cpp`

5.219 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**

5.219.1 Constructor & Destructor Documentation

5.219.1.1 **expression_t::expression_t** (
 const std::string & *s*,
 const std::string & *o*)

Constructor.

Parameters

| | |
|----------|---|
| <i>s</i> | String to be splitted |
| <i>o</i> | List of valid operators (single character only) |

5.219.1.2 **expression_t::expression_t** ()

5.219.2 Member Data Documentation

5.219.2.1 **std::string MHAParser::expression_t::lval**

5.219.2.2 **std::string MHAParser::expression_t::rval**

5.219.2.3 **std::string MHAParser::expression_t::op**

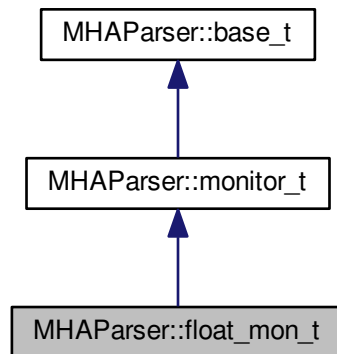
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.220 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

5.220.1 Detailed Description

Monitor with float value.

5.220.2 Constructor & Destructor Documentation

5.220.2.1 MHAParser::float_mon_t::float_mon_t (const std::string & hlp)

Initialize a floating point (32 bits) monitor variable.

Parameters

| | |
|-------------|---|
| <i>help</i> | A help text describing this monitor variable. |
|-------------|---|

5.220.3 Member Function Documentation

5.220.3.1 `std::string MHAParser::float_mon_t::query_val (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.220.3.2 `std::string MHAParser::float_mon_t::query_type (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.220.4 Member Data Documentation

5.220.4.1 `float MHAParser::float_mon_t::data`

Data field.

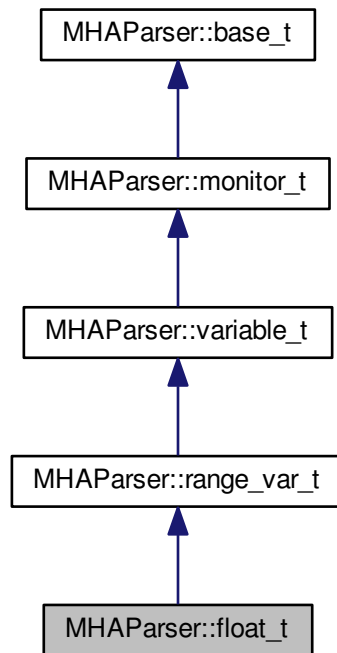
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.221 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

5.221.1 Detailed Description

Variable with float value.

5.221.2 Constructor & Destructor Documentation

5.221.2.1 MHAParser::float_t::float_t (
 const std::string & *help_text*,
 const std::string & *initial_value*,
 const std::string & *range* = " ")

Constructor for a configuration language variable for 32bit ieee floating-point values.

Parameters

| | |
|----------------------|---|
| <i>help_text</i> | A human-readable text describing the purpose of this configuration variable. |
| <i>initial_value</i> | 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. |
| <i>range</i> | 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. $a \leq b$. 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. |

5.221.3 Member Function Documentation

5.221.3.1 std::string MHAParser::float_t::op_setval (
 expression_t & x) [protected], [virtual]

Reimplemented from MHAParser::variable_t (p. 633).

5.221.3.2 std::string MHAParser::float_t::query_type (
 const std::string & s) [protected], [virtual]

Reimplemented from MHAParser::base_t (p. 573).

5.221.3.3 std::string MHAParser::float_t::query_val (
 const std::string & s) [protected], [virtual]

Reimplemented from MHAParser::base_t (p. 573).

5.221.4 Member Data Documentation

5.221.4.1 float MHAParser::float_t::data

Data field.

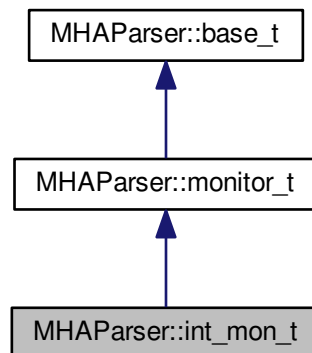
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.222 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

5.222.1 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.

5.222.2 Constructor & Destructor Documentation

5.222.2.1 `MHAParser::int_mon_t::int_mon_t (`
`const std::string & hlp)`

Create a monitor variable for integral values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.222.3 Member Function Documentation

5.222.3.1 `std::string MHAParser::int_mon_t::query_val (`
`const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from `MHAParser::base_t` (p. 573).

5.222.3.2 `std::string MHAParser::int_mon_t::query_type (`
`const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from `MHAParser::base_t` (p. 573).

5.222.4 Member Data Documentation

5.222.4.1 int MHAParser::int_mon_t::data

Data field.

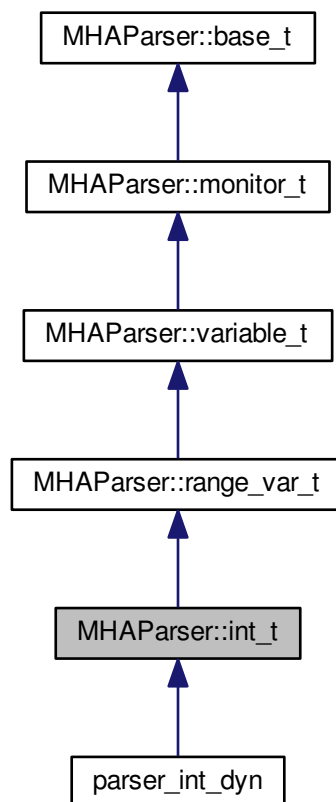
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.223 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

5.223.1 Detailed Description

Variable with integer value.

5.223.2 Constructor & Destructor Documentation

5.223.2.1 MHAParser::int_t::int_t (
 const std::string & *help_text*,
 const std::string & *initial_value*,
 const std::string & *range* = " ")

Constructor for a configuration language variable for integral values.

Parameters

| | |
|----------------------|---|
| <i>help_text</i> | A human-readable text describing the purpose of this configuration variable. |
| <i>initial_value</i> | 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. |
| <i>range</i> | 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. $a \leq b$. 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]). |

5.223.3 Member Function Documentation

5.223.3.1 `std::string MHAParser::int_t::op_setval (`
`expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.223.3.2 `std::string MHAParser::int_t::query_type (`
`const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.223.3.3 `std::string MHAParser::int_t::query_val (`
`const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.223.4 Member Data Documentation

5.223.4.1 `int MHAParser::int_t::data`

Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.224 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**

5.224.1 Detailed Description

Keyword list class.

The structure **keyword_list_t** (p. 600) defines a keyword list (vector of strings) with an index into the list. Used as **MHAParser::kw_t** (p. 603), it can be used to access a set of valid keywords through the parser (i.e. one of "pear apple banana").

5.224.2 Member Typedef Documentation

5.224.2.1 `typedef std::vector<std::string>::size_type MHAParser::keyword_list_t::size_t`

5.224.3 Constructor & Destructor Documentation

5.224.3.1 `MHAParser::keyword_list_t::keyword_list_t ()`

Constructor.

5.224.4 Member Function Documentation

5.224.4.1 void MHAParser::keyword_list_t::set_value (const std::string & s)

Select a value from keyword list.

This function selects a value from the keyword list. The index is set to the last matching entry.

Parameters

| | |
|---|-----------------------|
| s | Value to be selected. |
|---|-----------------------|

5.224.4.2 void MHAParser::keyword_list_t::set_entries (const std::string & s)

Set keyword list entries.

With this function, the keyword list can be set from a space separated string list.

Parameters

| | |
|---|-----------------------------|
| s | Space separated entry list. |
|---|-----------------------------|

5.224.4.3 const std::string & MHAParser::keyword_list_t::get_value () const

Return selected value.

5.224.4.4 const std::vector< std::string > & MHAParser::keyword_list_t::get_entries () const

Return keyword list.

5.224.4.5 const MHAParser::keyword_list_t::size_t & MHAParser::keyword_list_t::get_index () const

Return index of selected value.

5.224.4.6 void MHAParser::keyword_list_t::set_index (unsigned int idx)

5.224.4.7 void MHAParser::keyword_list_t::validate () const

Check if index of selected value is valid.

5.224.4.8 void MHAParser::keyword_list_t::add_entry (
const std::string & en) [inline]

5.224.5 Member Data Documentation

5.224.5.1 size_t MHAParser::keyword_list_t::index [private]

Index into list.

5.224.5.2 std::vector<std::string> MHAParser::keyword_list_t::entries [private]

List of valid entries.

5.224.5.3 std::string MHAParser::keyword_list_t::empty_string [private]

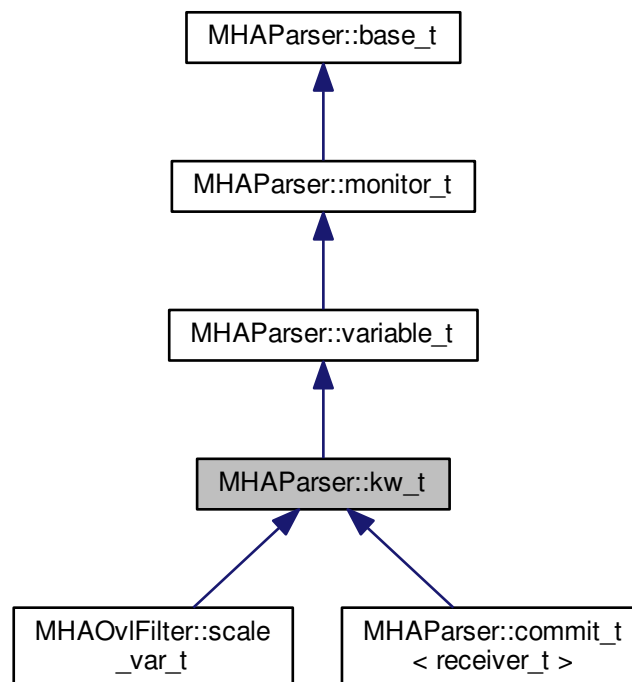
The documentation for this class was generated from the following files:

- mha_parser.hh
- mha_parser.cpp

5.225 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

5.225.1 Detailed Description

Variable with keyword list value.

5.225.2 Constructor & Destructor Documentation

5.225.2.1 **MHAParser::kw_t::kw_t** (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg*)

Constructor of a keyword list openMHA configuration variable.

Parameters

| | |
|-----------|---|
| <i>h</i> | A help string describing the purpose of this variable. |
| <i>v</i> | The initial value, has to be a value from the list of possible values given in the last parameter. |
| <i>rg</i> | 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 "[", ""]". |

5.225.2.2 MHAParser::kw_t::kw_t (
const kw_t & src)

Copy constructor.

5.225.3 Member Function Documentation

5.225.3.1 void MHAParser::kw_t::set_range (
const std::string & r)

Set/change the list of valid entries.

Parameters

| | |
|----------|--|
| <i>r</i> | 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 "[", "]". |
|----------|--|

5.225.3.2 bool MHAParser::kw_t::isval (
const std::string & testval) const

Test if the given value is selected.

5.225.3.3 void MHAParser::kw_t::validate (
const keyword_list_t & s) [protected]

5.225.3.4 std::string MHAParser::kw_t::op_setval (
expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 633).

5.225.3.5 std::string MHAParser::kw_t::query_range (
const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.225.3.6 std::string MHAParser::kw_t::query_val (
const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.225.3.7 std::string MHAParser::kw_t::query_type (
const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.225.4 Member Data Documentation

5.225.4.1 keyword_list_t MHAParser::kw_t::data

Variable data in its native type.

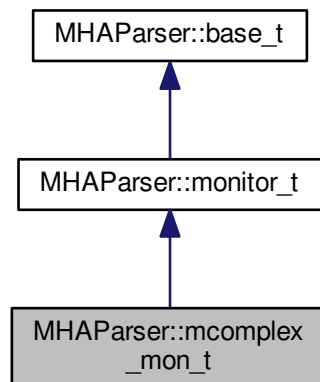
The documentation for this class was generated from the following files:

- mha_parser.hh
- mha_parser.cpp

5.226 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

5.226.1 Detailed Description

Matrix of complex numbers monitor.

5.226.2 Constructor & Destructor Documentation

5.226.2.1 MHAParser::mcomplex_mon_t::mcomplex_mon_t (`const std::string & hlp`)

Create a matrix of complex floating point monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.226.3 Member Function Documentation

5.226.3.1 `std::string MHAParser::mcomplex_mon_t::query_val (const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.226.3.2 `std::string MHAParser::mcomplex_mon_t::query_type (const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.226.4 Member Data Documentation

5.226.4.1 `std::vector< std::vector<mha_complex_t> > MHAParser::mcomplex_mon_t::data`

Data field.

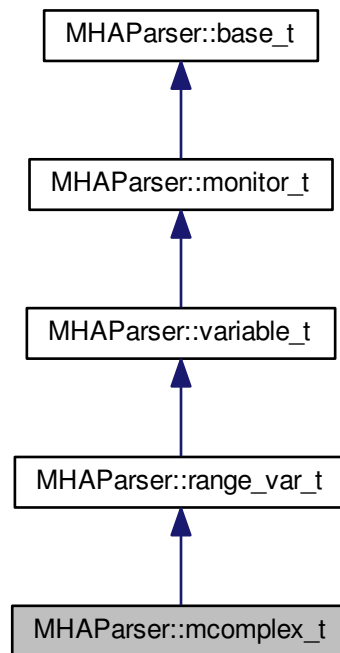
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.227 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

5.227.1 Detailed Description

Matrix variable with complex value.

5.227.2 Constructor & Destructor Documentation

5.227.2.1 MHAParser::mcomplex_t::mcomplex_t (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg* = " ")

5.227.3 Member Function Documentation

5.227.3.1 std::string MHAParser::mcomplex_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.227.3.2 std::string MHAParser::mcomplex_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.227.3.3 std::string MHAParser::mcomplex_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.227.4 Member Data Documentation

5.227.4.1 std::vector<std::vector<mha_complex_t> > MHAParser::mcomplex_t::data

Data field.

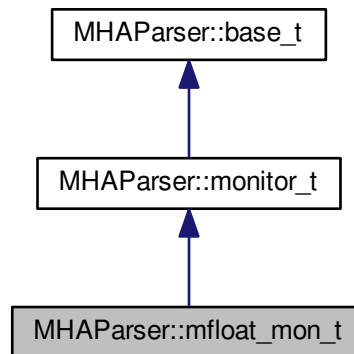
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.228 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

5.228.1 Detailed Description

Matrix of floats monitor.

5.228.2 Constructor & Destructor Documentation

5.228.2.1 MHAParser::mfloat_mon_t::mfloat_mon_t (const std::string & hlp)

Create a matrix of floating point monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.228.3 Member Function Documentation

5.228.3.1 `std::string MHAParser::mfloat_mon_t::query_val (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.228.3.2 `std::string MHAParser::mfloat_mon_t::query_type (`
`const std::string & s)` [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 573).

5.228.4 Member Data Documentation

5.228.4.1 `std::vector< std::vector<float> > MHAParser::mfloat_mon_t::data`

Data field.

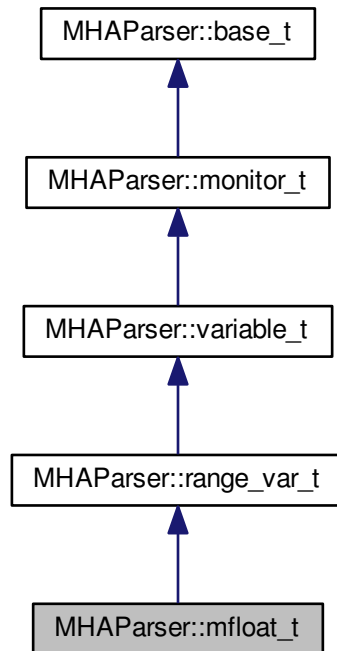
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.229 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

5.229.1 Detailed Description

Matrix variable with float value.

5.229.2 Constructor & Destructor Documentation

5.229.2.1 MHAParser::mfloat_t::mfloat_t (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg* = " ")

Create a float matrix parser variable.

Parameters

| | |
|-----------|---|
| <i>h</i> | A human-readable text describing the purpose of this configuration variable. |
| <i>v</i> | 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. |
| <i>rg</i> | The numeric range to enforce on all members of the matrix. |

5.229.3 Member Function Documentation

5.229.3.1 std::string MHAParser::mfloat_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.229.3.2 std::string MHAParser::mfloat_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.229.3.3 std::string MHAParser::mfloat_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.229.4 Member Data Documentation

5.229.4.1 `std::vector<std::vector<float>> MHAParser::mfloat_t::data`

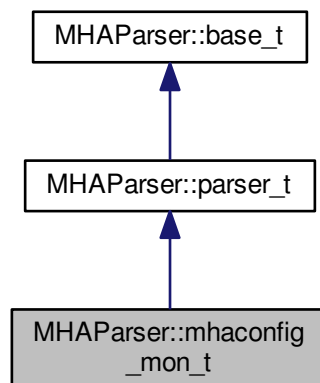
Data field.

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

- `mha_parser.hh`
- `mha_parser.cpp`

5.230 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

5.230.1 Constructor & Destructor Documentation

5.230.1.1 **MHAParser::mhaconfig_mon_t::mhaconfig_mon_t (**
 const std::string & help = " ")

5.230.2 Member Function Documentation

5.230.2.1 **void MHAParser::mhaconfig_mon_t::update (**
 const mhaconfig_t & cf)

5.230.3 Member Data Documentation

5.230.3.1 **MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::channels** [private]

Number of audio channels.

5.230.3.2 **MHAParser::string_mon_t MHAParser::mhaconfig_mon_t::domain** [private]

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

5.230.3.3 **MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::fragsize** [private]

Fragment size of waveform data.

5.230.3.4 **MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::wndlen** [private]

Window length of spectral data.

5.230.3.5 **MHAParser::int_mon_t** MHAParser::mhaconfig_mon_t::ffflen [private]

FFT length of spectral data.

5.230.3.6 **MHAParser::float_mon_t** MHAParser::mhaconfig_mon_t::srate [private]

Sampling rate in Hz.

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.231 **MHAParser::mhapluginloader_t** Class Reference

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

Public Member Functions

- **mhapluginloader_t** (**MHAParser::parser_t** &parent, **algo_comm_t** ac, const std::string &plugin_name="plugin_name", const std::string &prefix="")
- **~mhapluginloader_t** ()
- void **prepare** (**mhaconfig_t** &cf)
- void **release** ()
- void **process** (**mha_wave_t** *sIn, **mha_wave_t** **sOut)
- void **process** (**mha_spec_t** *sIn, **mha_spec_t** **sOut)
- void **process** (**mha_wave_t** *sIn, **mha_spec_t** **sOut)
- void **process** (**mha_spec_t** *sIn, **mha_wave_t** **sOut)
- **mhaconfig_t** **get_cfin** () const
- **mhaconfig_t** **get_cfout** () const
- const std::string & **get_last_name** () const

Protected Attributes

- **PluginLoader::mhapluginloader_t** * **plug**

Private Member Functions

- void **load_plug** ()

Private Attributes

- **MHAParser::parser_t & parent_**
- **MHAParser::string_t plugname**
- **std::string prefix_**
- **MHAEvents::connector_t< mhapluginloader_t > connector**
- **algo_comm_t ac_**
- **std::string last_name**
- **std::string plugname_name_**
- **mhaconfig_t cf_in_**
- **mhaconfig_t cf_out_**

Static Private Attributes

- static double **bookkeeping**

5.231.1 Detailed Description

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

5.231.2 Constructor & Destructor Documentation

5.231.2.1 **MHAParser::mhapluginloader_t::mhapluginloader_t (**
MHAParser::parser_t & parent,
algo_comm_t ac,
const std::string & plugname_name = "plugin_name",
const std::string & prefix = " ")

5.231.2.2 **MHAParser::mhapluginloader_t::~~mhapluginloader_t ()**

5.231.3 Member Function Documentation

5.231.3.1 **void MHAParser::mhapluginloader_t::prepare (**
mhaconfig_t & cf)

5.231.3.2 **void MHAParser::mhapluginloader_t::release ()**

5.231.3.3 **void MHAParser::mhapluginloader_t::process (**
mha_wave_t * sIn,
mha_wave_t ** sOut) [inline]

5.231.3.4 **void MHAParser::mhapluginloader_t::process (**
mha_spec_t * sIn,
mha_spec_t ** sOut) [inline]

- 5.231.3.5 `void MHAParser::mhapluginloader_t::process (`
`mha_wave_t * sIn,`
`mha_spec_t ** sOut) [inline]`
- 5.231.3.6 `void MHAParser::mhapluginloader_t::process (`
`mha_spec_t * sIn,`
`mha_wave_t ** sOut) [inline]`
- 5.231.3.7 `mhaconfig_t MHAParser::mhapluginloader_t::get_cfin () const [inline]`
- 5.231.3.8 `mhaconfig_t MHAParser::mhapluginloader_t::get_cfout () const [inline]`
- 5.231.3.9 `const std::string& MHAParser::mhapluginloader_t::get_last_name () const [inline]`
- 5.231.3.10 `void MHAParser::mhapluginloader_t::load_plug () [private]`

5.231.4 Member Data Documentation

- 5.231.4.1 `PluginLoader::mhapluginloader_t* MHAParser::mhapluginloader_t::plug`
`[protected]`
- 5.231.4.2 `MHAParser::parser_t& MHAParser::mhapluginloader_t::parent_ [private]`
- 5.231.4.3 `MHAParser::string_t MHAParser::mhapluginloader_t::plugname [private]`
- 5.231.4.4 `std::string MHAParser::mhapluginloader_t::prefix_ [private]`
- 5.231.4.5 `MHAEvents::connector_t<mhapluginloader_t> MHAParser::mhapluginloader_t↔`
`::connector [private]`
- 5.231.4.6 `algo_comm_t MHAParser::mhapluginloader_t::ac_ [private]`
- 5.231.4.7 `std::string MHAParser::mhapluginloader_t::last_name [private]`
- 5.231.4.8 `std::string MHAParser::mhapluginloader_t::plugname_name_ [private]`
- 5.231.4.9 `mhaconfig_t MHAParser::mhapluginloader_t::cf_in_ [private]`
- 5.231.4.10 `mhaconfig_t MHAParser::mhapluginloader_t::cf_out_ [private]`
- 5.231.4.11 `double MHAParser::mhapluginloader_t::bookkeeping [static], [private]`

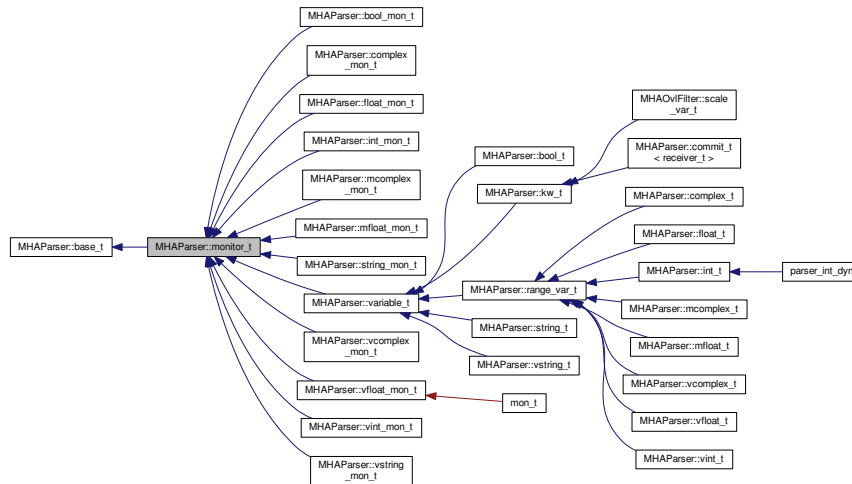
The documentation for this class was generated from the following files:

- `mhapluginloader.h`
- `mhapluginloader.cpp`

5.232 MHAParser::monitor_t Class Reference

Base class for monitors and variable nodes.

Inheritance diagram for MHAParser::monitor_t:



Public Member Functions

- **monitor_t** (const std::string &)
- **monitor_t** (const **monitor_t** &)
- std::string **op_query** (**expression_t** &)
- std::string **query_dump** (const std::string &)
- std::string **query_perm** (const std::string &)

Additional Inherited Members

5.232.1 Detailed Description

Base class for monitors and variable nodes.

5.232.2 Constructor & Destructor Documentation

5.232.2.1 MHAParser::monitor_t::monitor_t (const std::string & h)

5.232.2.2 MHAParser::monitor_t::monitor_t (const **monitor_t** & src)

5.232.3 Member Function Documentation

5.232.3.1 std::string MHAParser::monitor_t::op_query (**expression_t** & x) [virtual]

Reimplemented from **MHAParser::base_t** (p.572).

5.232.3.2 `std::string MHAParser::monitor_t::query_dump (`
`const std::string & s) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 572).

5.232.3.3 `std::string MHAParser::monitor_t::query_perm (`
`const std::string & s) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 573).

Reimplemented in **MHAParser::variable_t** (p. 633).

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

- **mha_parser.hh**
- **mha_parser.cpp**

5.233 MHAParser::parser_t Class Reference

Parser node class.

Inherits **MHAParser::base_t**.

Inherited by **AuditoryProfile::parser_t**, **AuditoryProfile::parser_t::ear_t**, **AuditoryProfile::parser_t::fmap_t**, **dc::wideband_inhib_vars_t**, **DynComp::dc_afterburn_vars_t**, **fw_t**, **io_file_t**, **io_parser_t**, **io_tcp_parser_t**, **MHAFilter::adapt_filter_t**, **MHAFilter::iir_filter_t**, **MHAIOJack::io_jack_t**, **MHAIOPortAudio::device_info_t**, **MHAIOPortAudio::io_portaudio_t**, **MHAParser::mhaconfig_mon_t**, **MHAParser::window_t**, **MHAPlugin::plugin_t< runtime_cfg_t >**, **MHAPlugin_Split::split_t**, **MHAPlugin::plugin_t< ac2wave_t >**, **MHAPlugin::plugin_t< acConcat_wave_config >**, **MHAPlugin::plugin_t< acPooling_wave_config >**, **MHAPlugin::plugin_t< acSteer_config >**, **MHAPlugin::plugin_t< acTransform_wave_config >**, **MHAPlugin::plugin_t< adm_rtconfig_t >**, **MHAPlugin::plugin_t< analysepath_t >**, **MHAPlugin::plugin_t< cfg_t >**, **MHAPlugin::plugin_t< cohflt_t >**, **MHAPlugin::plugin_t< combc_t >**, **MHAPlugin::plugin_t< db_t >**, **MHAPlugin::plugin_t< dc_t >**, **MHAPlugin::plugin_t< delaysum_t >**, **MHAPlugin::plugin_t< doasvm_classification_config >**, **MHAPlugin::plugin_t< doasvm_feature_extraction_config >**, **MHAPlugin::plugin_t< example5_t >**, **MHAPlugin::plugin_t< fftfb_plug_t >**, **MHAPlugin::plugin_t< float >**, **MHAPlugin::plugin_t< hilbert_shifter_t >**, **MHAPlugin::plugin_t< int >**, **MHAPlugin::plugin_t< lpc_bl_predictor_config >**, **MHAPlugin::plugin_t< lpc_burglattice_config >**, **MHAPlugin::plugin_t< lpc_config >**, **MHAPlugin::plugin_t< MHA_AC::spectrum_t >**, **MHAPlugin::plugin_t< MHA_AC::waveform_t >**, **MHAPlugin::plugin_t< mhachain::plugs_t >**, **MHAPlugin::plugin_t< MHASignal::delay_t >**, **MHAPlugin::plugin_t< MHASignal::waveform_t >**, **MHAPlugin::plugin_t< MHAWindow::fun_t >**, **MHAPlugin::plugin_t< noisePowProposed >**, **MHAPlugin::plugin_t< overlapadd_t >**, **MHAPlugin::plugin_t< prediction_error_config >**, **MHAPlugin::plugin_t< resampling_t >**, **MHAPlugin::plugin_t< rmslevel_t >**, **MHAPlugin::plugin_t< route::process_t >**, **MHAPlugin::plugin_t< rt_nlms_t >**, **MHAPlugin::plugin_t< scaler_t >**, **MHAPlugin::plugin_t< sine_cfg_t >**, **MHAPlugin::plugin_t< smoothspec_wrap_t >**, **MHAPlugin::plugin_t< spec2wave_t >**, **MHAPlugin::plugin_t< spec_fader_t >**, **MHAPlugin::plugin_t< steerbf_config >**, **MHAPlugin::plugin_t< timoConfig >**, **MHAPlugin::plugin_t< wave2spec_t >**, **MHAPlugin::plugin_t< wavwriter_t >**, and **softclipper_variables_t**.

Public Member Functions

- **parser_t** (const std::string &help_text="")
Construct detached node to be used in the configuration tree.
- **~parser_t** ()
- void **insert_item** (const std::string &, **base_t** *)
Register a parser item into this sub-parser.
- void **remove_item** (const std::string &)
Remove an item by name.
- void **force_remove_item** (const std::string &)
Remove an item by name.
- void **remove_item** (const **base_t** *)
Remove an item by address.

Protected Member Functions

- std::string **op_subparse** (**expression_t** &)
- std::string **op_setval** (**expression_t** &)
- std::string **op_query** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_dump** (const std::string &)
- std::string **query_entries** (const std::string &)
- std::string **query_readfile** (const std::string &)
- std::string **query_savefile** (const std::string &)
- std::string **query_savefile_compact** (const std::string &)
- std::string **query_savemons** (const std::string &)
- std::string **query_val** (const std::string &)
- std::string **query_listids** (const std::string &)
- void **set_id_string** (const std::string &)

Private Attributes

- **entry_map_t** **entries**
- std::string **id_string**
identification string
- std::string **srcfile**
- unsigned int **srcline**
- std::string **last_errormsg**

Additional Inherited Members

5.233.1 Detailed Description

Parser node class.

A **parser_t** (p. 620) instance is a node in the configuration tree. A parser node can contain any number of other **parser_t** (p. 620) instances or configuration language variables. These items are inserted into a parser node using the **parser_t::insert_item** (p. 622) method.

5.233.2 Constructor & Destructor Documentation

5.233.2.1 MHAParser::parser_t::parser_t (const std::string & *help_text* = " ")

Construct detached node to be used in the configuration tree.

Parameters

| | |
|------------------|--|
| <i>help_text</i> | 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. |
|------------------|--|

5.233.2.2 MHAParser::parser_t::~~parser_t ()

5.233.3 Member Function Documentation

5.233.3.1 void MHAParser::parser_t::insert_item (const std::string & *n*, MHAParser::base_t * *e*)

Register a parser item into this sub-parser.

This function registers an item under a given name into this sub-parser and makes it accessible to the parser interface.

Parameters

| | |
|----------|--|
| <i>n</i> | Name of the item in the configuration tree |
| <i>e</i> | C++ pointer to the item instance. <i>e</i> can either point to a variable, to a monitor, or to another sub-parser. |

5.233.3.2 void MHAParser::parser_t::remove_item (const std::string & *n*)

Remove an item by name.

If the item does not exist, an error is being reported.

Parameters

| | |
|----------|--|
| <i>n</i> | Name of parser item to be removed from list. |
|----------|--|

5.233.3.3 void MHAParser::parser_t::force_remove_item (
const std::string & *n*)

Remove an item by name.

Non-existing items are ignored.

Parameters

| | |
|----------|--|
| <i>n</i> | Name of parser item to be removed from list. |
|----------|--|

5.233.3.4 void MHAParser::parser_t::remove_item (
const base_t * *addr*)

Remove an item by address.

The item belonging to an address is being removed from the list of items.

Parameters

| | |
|-------------|---------------------------------------|
| <i>addr</i> | Address of parser item to be removed. |
|-------------|---------------------------------------|

5.233.3.5 std::string MHAParser::parser_t::op_subparse (
expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [572](#)).

5.233.3.6 std::string MHAParser::parser_t::op_setval (
expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [572](#)).

5.233.3.7 std::string MHAParser::parser_t::op_query (
expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [572](#)).

5.233.3.8 std::string MHAParser::parser_t::query_type (
const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.233.3.9 std::string MHAParser::parser_t::query_dump (
const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [572](#)).

5.233.3.10 `std::string MHAParser::parser_t::query_entries (`
 `const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 573).

5.233.3.11 `std::string MHAParser::parser_t::query_readfile (`
 `const std::string & fname) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 573).

5.233.3.12 `std::string MHAParser::parser_t::query_savefile (`
 `const std::string & fname) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 574).

5.233.3.13 `std::string MHAParser::parser_t::query_savefile_compact (`
 `const std::string & fname) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 574).

5.233.3.14 `std::string MHAParser::parser_t::query_savemons (`
 `const std::string & fname) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 574).

5.233.3.15 `std::string MHAParser::parser_t::query_val (`
 `const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 573).

5.233.3.16 `std::string MHAParser::parser_t::query_listids (`
 `const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 574).

5.233.3.17 `void MHAParser::parser_t::set_id_string (`
 `const std::string & s) [protected]`

5.233.4 Member Data Documentation

5.233.4.1 `entry_map_t MHAParser::parser_t::entries [private]`

5.233.4.2 `std::string MHAParser::parser_t::id_string [private]`

identification string

5.233.4.3 `std::string MHAParser::parser_t::srcfile` [private]

5.233.4.4 `unsigned int MHAParser::parser_t::srcline` [private]

5.233.4.5 `std::string MHAParser::parser_t::last_errormsg` [private]

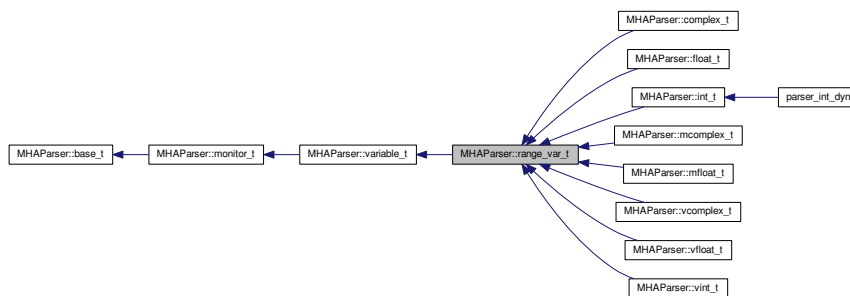
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.234 MHAParser::range_var_t Class Reference

Base class for all variables with a numeric value range.

Inheritance diagram for MHAParser::range_var_t:



Public Member Functions

- **range_var_t** (const std::string &, const std::string &="")
- **range_var_t** (const **range_var_t** &)
- std::string **query_range** (const std::string &)
- void **set_range** (const std::string &r)
 Change the valid range of a variable.
- void **validate** (const int &)
- void **validate** (const float &)
- void **validate** (const **mha_complex_t** &)
- void **validate** (const std::vector< int > &)
- void **validate** (const std::vector< float > &)
- void **validate** (const std::vector< **mha_complex_t** > &)
- void **validate** (const std::vector< std::vector< float > > &)
- void **validate** (const std::vector< std::vector< **mha_complex_t** > > &)

Protected Attributes

- float **low_limit**
Lower limit of range.
- float **up_limit**
Upper limit of range.
- bool **low_incl**
Lower limit is included (or excluded) in range.
- bool **up_incl**
Upper limit is included (or excluded) in range.
- bool **check_low**
Check lower limit.
- bool **check_up**
Check upper limit.
- bool **check_range**
Range checking is active.

Additional Inherited Members

5.234.1 Detailed Description

Base class for all variables with a numeric value range.

5.234.2 Constructor & Destructor Documentation

5.234.2.1 `MHAParser::range_var_t::range_var_t (`
 `const std::string & h,`
 `const std::string & r = " ")`

5.234.2.2 `MHAParser::range_var_t::range_var_t (`
 `const range_var_t & src)`

5.234.3 Member Function Documentation

5.234.3.1 `std::string MHAParser::range_var_t::query_range (`
 `const std::string & s)` `[virtual]`

Reimplemented from **MHAParser::base_t** (p. 573).

5.234.3.2 `void MHAParser::range_var_t::set_range (`
 `const std::string & r)`

Change the valid range of a variable.

Parameters

| | |
|----------|---|
| <i>r</i> | New range of the variable (string representation) |
|----------|---|

5.234.3.3 void MHAParser::range_var_t::validate (
const int & *v*)

5.234.3.4 void MHAParser::range_var_t::validate (
const float & *v*)

5.234.3.5 void MHAParser::range_var_t::validate (
const mha_complex_t & *v*)

5.234.3.6 void MHAParser::range_var_t::validate (
const std::vector< int > & *v*)

5.234.3.7 void MHAParser::range_var_t::validate (
const std::vector< float > & *v*)

5.234.3.8 void MHAParser::range_var_t::validate (
const std::vector< mha_complex_t > & *v*)

5.234.3.9 void MHAParser::range_var_t::validate (
const std::vector< std::vector< float > > & *v*)

5.234.3.10 void MHAParser::range_var_t::validate (
const std::vector< std::vector< mha_complex_t > > & *v*)

5.234.4 Member Data Documentation

5.234.4.1 float MHAParser::range_var_t::low_limit [protected]

Lower limit of range.

5.234.4.2 float MHAParser::range_var_t::up_limit [protected]

Upper limit of range.

5.234.4.3 bool MHAParser::range_var_t::low_incl [protected]

Lower limit is included (or excluded) in range.

5.234.4.4 bool MHAParser::range_var_t::up_incl [protected]

Upper limit is included (or excluded) in range.

5.234.4.5 `bool MHAParser::range_var_t::check_low` `[protected]`

Check lower limit.

5.234.4.6 `bool MHAParser::range_var_t::check_up` `[protected]`

Check upper limit.

5.234.4.7 `bool MHAParser::range_var_t::check_range` `[protected]`

Range checking is active.

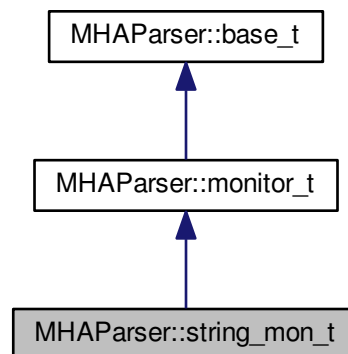
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.235 `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

5.235.1 Detailed Description

Monitor with string value.

5.235.2 Constructor & Destructor Documentation

5.235.2.1 MHAParser::string_mon_t::string_mon_t (const std::string & *hlp*)

Create a monitor variable for string values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.235.3 Member Function Documentation

5.235.3.1 std::string MHAParser::string_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.235.3.2 std::string MHAParser::string_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.235.4 Member Data Documentation

5.235.4.1 std::string MHAParser::string_mon_t::data

Data field.

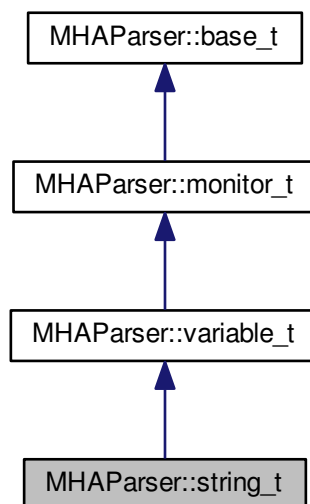
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.236 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

5.236.1 Detailed Description

Variable with a string value.

5.236.2 Constructor & Destructor Documentation

5.236.2.1 `MHAParser::string_t::string_t (`
 `const std::string & h,`
 `const std::string & v)`

Constructor of a openMHA configuration variable for string values.

Parameters

| | |
|----------|--|
| <i>h</i> | A help string describing the purpose of this variable. |
| <i>v</i> | The initial string value |

5.236.3 Member Function Documentation

5.236.3.1 `std::string MHAParser::string_t::op_setval (`
 `expression_t & x)` `[protected]`, `[virtual]`

Reimplemented from `MHAParser::variable_t` (p. [633](#)).

5.236.3.2 `std::string MHAParser::string_t::query_type (`
 `const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from `MHAParser::base_t` (p. [573](#)).

5.236.3.3 `std::string MHAParser::string_t::query_val (`
 `const std::string & s)` `[protected]`, `[virtual]`

Reimplemented from `MHAParser::base_t` (p. [573](#)).

Additional Inherited Members

5.237.1 Detailed Description

Base class for variable nodes.

5.237.2 Constructor & Destructor Documentation

5.237.2.1 MHAParser::variable_t::variable_t (
const std::string & h)

5.237.3 Member Function Documentation

5.237.3.1 std::string MHAParser::variable_t::op_setval (
expression_t & x) [virtual]

Reimplemented from MHAParser::base_t (p. 572).

Reimplemented in MHAParser::mcomplex_t (p. 609), MHAParser::mfloat_t (p. 613), MHAParser::vcomplex_t (p. 637), MHAParser::vfloat_t (p. 641), MHAParser::vint_t (p. 645), MHAParser::complex_t (p. 589), MHAParser::float_t (p. 595), MHAParser::int_t (p. 600), MHAParser::bool_t (p. 581), MHAParser::vstring_t (p. 649), MHAParser::string_t (p. 631), and MHAParser::kw_t (p. 605).

5.237.3.2 std::string MHAParser::variable_t::query_perm (
const std::string & s) [virtual]

Reimplemented from MHAParser::monitor_t (p. 620).

5.237.3.3 void MHAParser::variable_t::setlock (
const bool & b)

Lock a variable against write access.

Parameters

| | |
|----------|------------|
| <i>b</i> | Lock state |
|----------|------------|

5.237.4 Member Data Documentation

5.237.4.1 bool MHAParser::variable_t::locked [private]

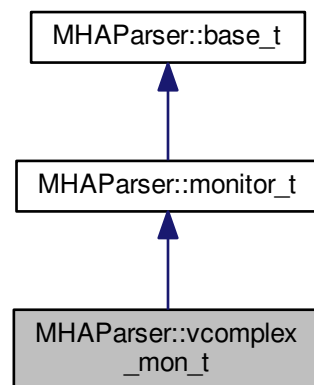
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.238 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

5.238.1 Detailed Description

Monitor with vector of complex values.

5.238.2 Constructor & Destructor Documentation

5.238.2.1 MHAParser::vcomplex_mon_t::vcomplex_mon_t (const std::string & *hlp*)

Create a vector of complex monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.238.3 Member Function Documentation

5.238.3.1 std::string MHAParser::vcomplex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.238.3.2 std::string MHAParser::vcomplex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.238.4 Member Data Documentation

5.238.4.1 std::vector<mha_complex_t> MHAParser::vcomplex_mon_t::data

Data field.

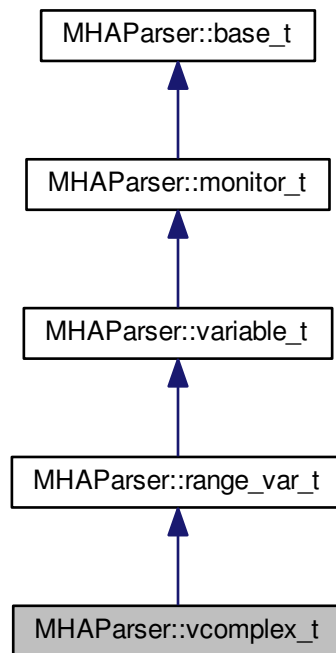
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.239 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

5.239.1 Detailed Description

Vector variable with complex value.

5.239.2 Constructor & Destructor Documentation

5.239.2.1 MHAParser::vcomplex_t::vcomplex_t (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg* = " ")

5.239.3 Member Function Documentation

5.239.3.1 std::string MHAParser::vcomplex_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.239.3.2 std::string MHAParser::vcomplex_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.239.3.3 std::string MHAParser::vcomplex_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.239.4 Member Data Documentation

5.239.4.1 std::vector<mha_complex_t> MHAParser::vcomplex_t::data

Data field.

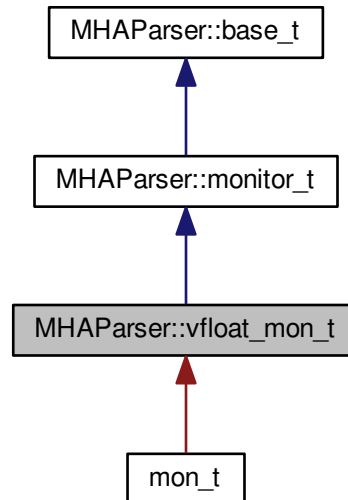
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.240 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

5.240.1 Detailed Description

Vector of floats monitor.

5.240.2 Constructor & Destructor Documentation

5.240.2.1 MHAParser::vfloat_mon_t::vfloat_mon_t (const std::string & *hlp*)

Create a vector of floating point monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.240.3 Member Function Documentation

5.240.3.1 std::string MHAParser::vfloat_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.240.3.2 std::string MHAParser::vfloat_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.240.4 Member Data Documentation

5.240.4.1 std::vector<float> MHAParser::vfloat_mon_t::data

Data field.

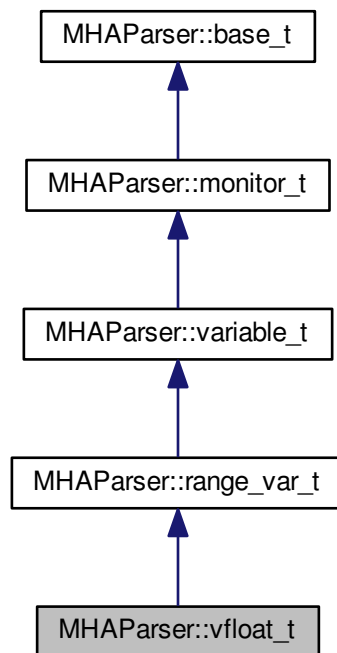
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.241 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

5.241.1 Detailed Description

Vector variable with float value.

5.241.2 Constructor & Destructor Documentation

5.241.2.1 MHAParser::vfloat_t::vfloat_t (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg* = " ")

Create a float vector parser variable.

Parameters

| | |
|-----------|--|
| <i>h</i> | A human-readable text describing the purpose of this configuration variable. |
| <i>v</i> | 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. |
| <i>rg</i> | The numeric range to enforce on all members of the vector. |

•

5.241.3 Member Function Documentation

5.241.3.1 std::string MHAParser::vfloat_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.241.3.2 std::string MHAParser::vfloat_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.241.3.3 std::string MHAParser::vfloat_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.241.4 Member Data Documentation

5.241.4.1 `std::vector<float> MHAParser::vfloat_t::data`

Data field.

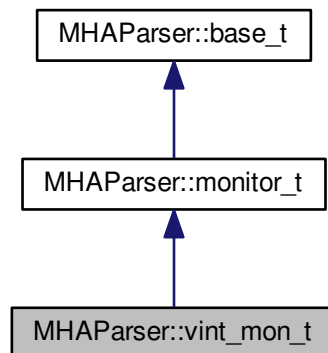
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.242 `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

5.242.1 Detailed Description

Vector of ints monitor.

5.242.2 Constructor & Destructor Documentation

5.242.2.1 MHAParser::vint_mon_t::vint_mon_t (const std::string & *hlp*)

Create a vector of integer monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.242.3 Member Function Documentation

5.242.3.1 std::string MHAParser::vint_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.242.3.2 std::string MHAParser::vint_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.242.4 Member Data Documentation

5.242.4.1 std::vector<int> MHAParser::vint_mon_t::data

Data field.

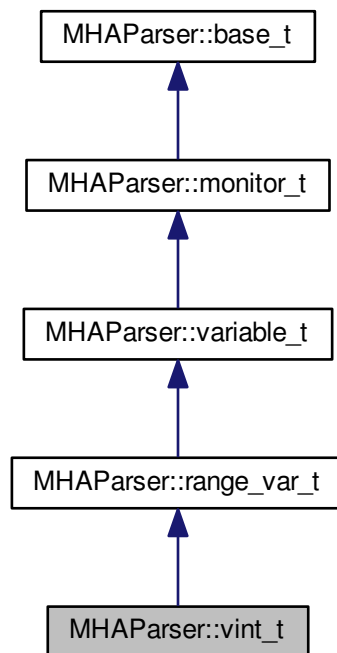
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.243 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

5.243.1 Detailed Description

Variable with vector<int> value.

5.243.2 Constructor & Destructor Documentation

5.243.2.1 MHAParser::vint_t::vint_t (
 const std::string & *h*,
 const std::string & *v*,
 const std::string & *rg* = " ")

Constructor.

Parameters

| | |
|-----------|---|
| <i>h</i> | help string |
| <i>v</i> | initial value |
| <i>rg</i> | optional: range constraint for all elements |

5.243.3 Member Function Documentation

5.243.3.1 std::string MHAParser::vint_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.243.3.2 std::string MHAParser::vint_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.243.3.3 std::string MHAParser::vint_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.243.4 Member Data Documentation

5.243.4.1 `std::vector<int> MHAParser::vint_t::data`

Data field.

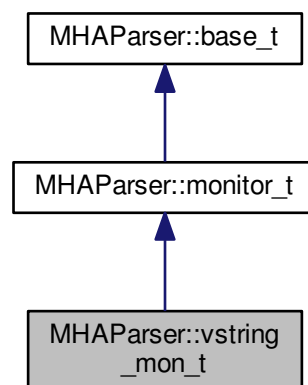
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

5.244 `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

5.244.1 Detailed Description

Vector of monitors with string value.

5.244.2 Constructor & Destructor Documentation

5.244.2.1 MHAParser::vstring_mon_t::vstring_mon_t (const std::string & *hlp*)

Create a vector of string monitor values.

Parameters

| | |
|------------|---|
| <i>hlp</i> | A help text describing this monitor variable. |
|------------|---|

5.244.3 Member Function Documentation

5.244.3.1 std::string MHAParser::vstring_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.244.3.2 std::string MHAParser::vstring_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.244.4 Member Data Documentation

5.244.4.1 std::vector<std::string> MHAParser::vstring_mon_t::data

Data field.

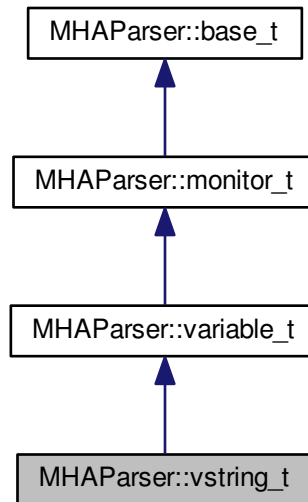
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

5.245 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

5.245.1 Detailed Description

Vector variable with string values.

5.245.2 Constructor & Destructor Documentation

5.245.2.1 MHAParser::vstring_t::vstring_t (
 const std::string & *h*,
 const std::string & *v*)

5.245.3 Member Function Documentation

5.245.3.1 std::string MHAParser::vstring_t::op_setval (
 expression_t & *x*) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. [633](#)).

5.245.3.2 std::string MHAParser::vstring_t::query_type (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.245.3.3 std::string MHAParser::vstring_t::query_val (
 const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. [573](#)).

5.245.4 Member Data Documentation

5.245.4.1 std::vector<std::string> MHAParser::vstring_t::data

Data field.

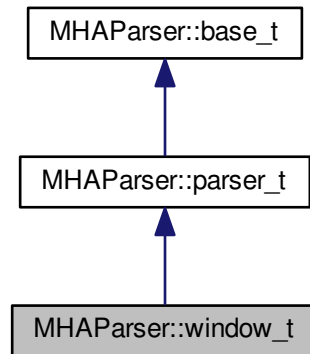
The documentation for this class was generated from the following files:

- mha_parser.hh
- mha_parser.cpp

5.246 MHAParser::window_t Class Reference

MHA configuration interface for a window function generator.

Inheritance diagram for MHAParser::window_t:



Public Types

Public Member Functions

- **window_t** (const std::string &help="Window type configuration.")
Constructor to create parser class.
- **MHAWindow::base_t get_window** (unsigned int len) const
Create a window instance, use default parameters.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool minin-cluded) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool minin-cluded, bool maxincluded) const
Create a window instance.
- **MHAParser::window_t::wtype_t get_type** () const
Return currently selected window type.

Private Attributes

- **MHAParser::kw_t wtype**
- **MHAParser::vfloat_t user**

Additional Inherited Members

5.246.1 Detailed Description

MHA configuration interface for a window function generator.

This class implements a configuration interface (sub-parser) for window type selection and user-defined window type. It provides member functions to generate an instance of **MHA↔Window::base_t** (p. 766) based on the values provided by the configuration interface.

The configuration interface is derived from **MHAParser::parser_t** (p. 620) and can thus be inserted into the configuration tree using the **insert_item()** (p. 622) method of the parent parser.

If one of the pre-defined window types is used, then the window is generated using the **MHA↔Window::fun_t** (p. 769) class constructor; for the user-defined type the values from the "user" variable are copied.

5.246.2 Member Enumeration Documentation

5.246.2.1 enum MHAParser::window_t::wtype_t

Enumerator

wnd_rect
wnd_hann
wnd_hamming
wnd_blackman
wnd_bartlett
wnd_user

5.246.3 Constructor & Destructor Documentation

5.246.3.1 MHAParser::window_t::window_t (const std::string & *help* = "Window type configuration.")

Constructor to create parser class.

5.246.4 Member Function Documentation

5.246.4.1 MHAWindow::base_t MHAParser::window_t::get_window (unsigned int *len*) const

Create a window instance, use default parameters.

5.246.4.2 MHAWindow::base_t MHAParser::window_t::get_window (
 unsigned int *len*,
 float *xmin*) const

Create a window instance.

5.246.4.3 MHAWindow::base_t MHAParser::window_t::get_window (
 unsigned int *len*,
 float *xmin*,
 float *xmax*) const

Create a window instance.

5.246.4.4 MHAWindow::base_t MHAParser::window_t::get_window (
 unsigned int *len*,
 float *xmin*,
 float *xmax*,
 bool *minincluded*) const

Create a window instance.

5.246.4.5 MHAWindow::base_t MHAParser::window_t::get_window (
 unsigned int *len*,
 float *xmin*,
 float *xmax*,
 bool *minincluded*,
 bool *maxincluded*) const

Create a window instance.

5.246.4.6 MHAParser::window_t::wtype_t MHAParser::window_t::get_type () const

Return currently selected window type.

5.246.5 Member Data Documentation

5.246.5.1 MHAParser::kw_t MHAParser::window_t::wtype [private]

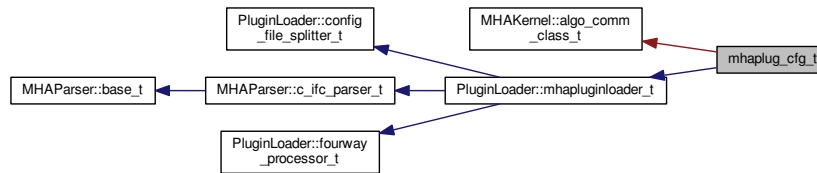
5.246.5.2 MHAParser::vfloat_t MHAParser::window_t::user [private]

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.247 mhaplug_cfg_t Class Reference

Inheritance diagram for mhaplug_cfg_t:



Public Member Functions

- **mhaplug_cfg_t** (**algo_comm_t** iac, const std::string &**libname**, bool use_own_ac)
- **~mhaplug_cfg_t** () throw ()

Additional Inherited Members

5.247.1 Constructor & Destructor Documentation

5.247.1.1 mhaplug_cfg_t::mhaplug_cfg_t (
 algo_comm_t iac,
 const std::string & *libname*,
 bool *use_own_ac*)

5.247.1.2 mhaplug_cfg_t::~mhaplug_cfg_t () throw () [inline]

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

- **altplugs.cpp**

5.248 MHAPlugin::cfg_chain_t< runtime_cfg_t > Class Template Reference

Public Member Functions

- **cfg_chain_t** (runtime_cfg_t *id)
- **~cfg_chain_t** ()

Public Attributes

- **cfg_chain_t**< runtime_cfg_t > * **next**
- bool **not_in_use**
- runtime_cfg_t * **data**

5.248.1 Constructor & Destructor Documentation

5.248.1.1 `template<class runtime_cfg_t> MHAPLugin::cfg_chain_t< runtime_cfg_t >::cfg_chain_t (runtime_cfg_t * id)`

5.248.1.2 `template<class runtime_cfg_t > MHAPLugin::cfg_chain_t< runtime_cfg_t >::~~cfg_chain_t ()`

5.248.2 Member Data Documentation

5.248.2.1 `template<class runtime_cfg_t> cfg_chain_t< runtime_cfg_t > * MHAPLugin::cfg_chain_t< runtime_cfg_t >::next`

5.248.2.2 `template<class runtime_cfg_t> bool MHAPLugin::cfg_chain_t< runtime_cfg_t >::not_in_use`

5.248.2.3 `template<class runtime_cfg_t> runtime_cfg_t * MHAPLugin::cfg_chain_t< runtime_cfg_t >::data`

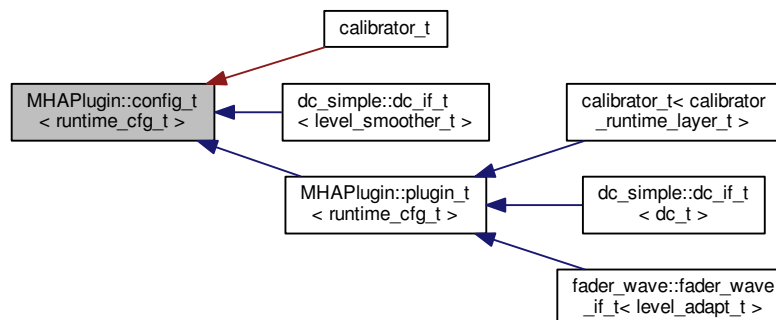
The documentation for this class was generated from the following file:

- **mha_plugin.hh**

5.249 MHAPLugin::config_t< runtime_cfg_t > Class Template Reference

Template class for thread safe configuration.

Inheritance diagram for MHAPLugin::config_t< runtime_cfg_t >:



Public Member Functions

- **config_t** ()
- **~config_t** ()

Protected Member Functions

- runtime_cfg_t * **poll_config** ()
Receive the latest run time configuration.
- runtime_cfg_t * **last_config** ()
Receive the latest run time configuration.
- void **push_config** (runtime_cfg_t *ncfg)
Push a new run time configuration into the configuration fifo.
- void **cleanup_unused_cfg** ()

Protected Attributes

- runtime_cfg_t * **cfg**

Private Member Functions

- void **remove_all_cfg** ()

Private Attributes

- **MHAPLugin::cfg_chain_t**< runtime_cfg_t > * **cfg_chain**
- **MHAPLugin::cfg_chain_t**< runtime_cfg_t > * **cfg_chain_current**

5.249.1 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. 103) 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. 658). In the processing thread, the actual configuration can be received by a call of **poll_config()** (p. 656).

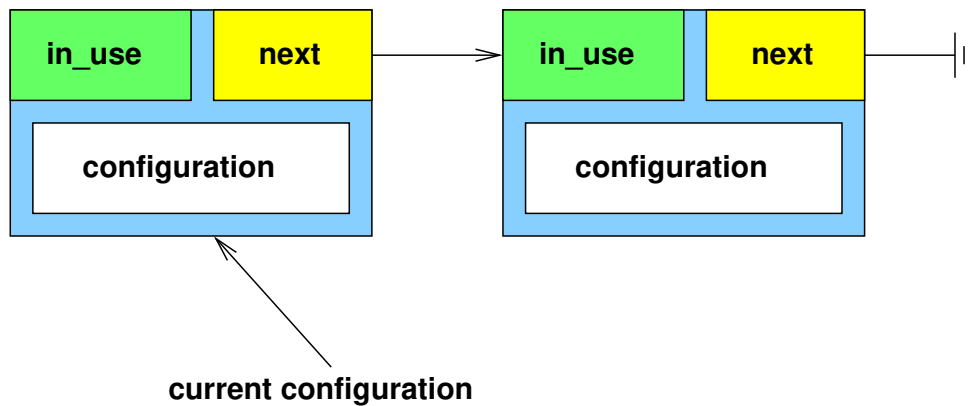


Figure 5 Schematic drawing of runtime configuration update: configuration updated, but not used yet.

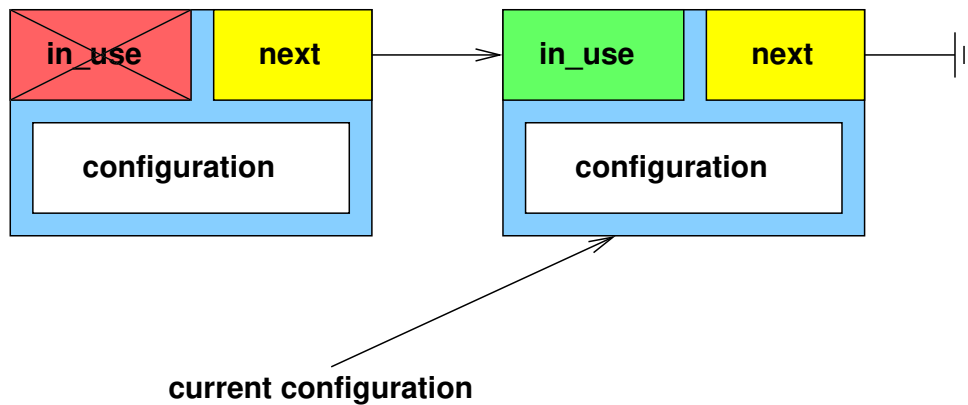


Figure 6 Schematic drawing of runtime configuration update: configuration in use.

5.249.2 Constructor & Destructor Documentation

5.249.2.1 `template<class runtime_cfg_t> MHAPLugin::config_t< runtime_cfg_t >::config_t ()`

5.249.2.2 `template<class runtime_cfg_t> MHAPLugin::config_t< runtime_cfg_t >::~~config_t ()`

5.249.3 Member Function Documentation

5.249.3.1 `template<class runtime_cfg_t> runtime_cfg_t * MHAPLugin::config_t< runtime_cfg_t >::poll_config () [protected]`

Receive the latest run time configuration.

This function stores the latest run time configuration into the protected class member variable 'cfg'. If no configuration exists, then an exception will be thrown. If no changes occurred, then

the value of 'cfg' will be untouched. This function should be called before any access to the 'cfg' variable, typically once in each signal processing call.

This function should be only called from the *processing* thread.

Exceptions

| | |
|----------------------------------|---|
| <i>MHA_Error</i> (p. 387) | if the resulting runtime configuration is NULL. This usually means that no <code>push_config</code> has occurred. |
|----------------------------------|---|

5.249.3.2 `template<class runtime_cfg_t> runtime_cfg_t * MHAPLugin::config_t< runtime_cfg_t >::last_config () [protected]`

Receive the latest run time configuration.

This function stores the latest run time configuration into the protected class member variable 'cfg'. If no configuration exists, then an exception will be thrown. If no changes occurred, then the value of 'cfg' will be untouched. This function may be called instead of `poll_config`.

The difference between `poll_config` and `last_config` is that `poll_config` marks previous configurations as ready for deletion, while this function does not. Therefore, memory usage of all runtime configurations will accumulate if only this function is called, but it enables safe access to previous runtime configurations.

Also, `last_config` does not raise an Exception when the latest run time configuration is NULL.

5.249.3.3 `template<class runtime_cfg_t> void MHAPLugin::config_t< runtime_cfg_t >::push_config (runtime_cfg_t * ncfg) [protected]`

Push a new run time configuration into the configuration fifo.

This function adds a new run time configuration. The next time **`poll_config`** (p. 656) is called, this configuration will be available. Configurations which are not in use or are outdated will be removed.

This function should be only called from the *configuration* thread.

Parameters

| | |
|-------------|--------------------------------|
| <i>ncfg</i> | pointer on a new configuration |
|-------------|--------------------------------|

Warning

The runtime configuration passed to this function will be removed by the internal garbage collector. Do not free manually.

5.249.3.4 `template<class runtime_cfg_t> void MHAPLugin::config_t< runtime_cfg_t >::cleanup_unused_cfg () [protected]`

5.249.3.5 `template<class runtime_cfg_t > void MHAPlugin::config_t< runtime_cfg_t >::remove_all_cfg () [private]`

5.249.4 Member Data Documentation

5.249.4.1 `template<class runtime_cfg_t> runtime_cfg_t* MHAPlugin::config_t< runtime_cfg_t >::cfg [protected]`

5.249.4.2 `template<class runtime_cfg_t> MHAPlugin::cfg_chain_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_chain [private]`

5.249.4.3 `template<class runtime_cfg_t> MHAPlugin::cfg_chain_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_chain_current [private]`

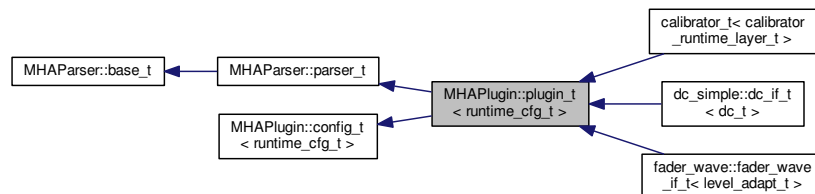
The documentation for this class was generated from the following file:

- `mha_plugin.hh`

5.250 MHAPlugin::plugin_t< runtime_cfg_t > Class Template Reference

The template class for C++ openMHA plugins.

Inheritance diagram for MHAPlugin::plugin_t< runtime_cfg_t >:



Public Member Functions

- **plugin_t** (const std::string &, const algo_comm_t &)
Constructor of plugin template.
- virtual ~**plugin_t** ()
- virtual void **prepare** (mhaconfig_t &)=0
- virtual void **release** ()
- void **prepare_** (mhaconfig_t &)
- void **release_** ()
- bool **is_prepared** () const
Flag, if the prepare method is successfully called (or currently evaluated)
- **mhaconfig_t input_cfg** () const
Current input channel configuration.
- **mhaconfig_t output_cfg** () const
Current output channel configuration.

Protected Attributes

- **mhaconfig_t tftype**
Member for storage of plugin interface configuration.
- **algo_comm_t ac**
AC handle of the chain.

Private Attributes

- bool **is_prepared_**
- **mhaconfig_t input_cfg_**
- **mhaconfig_t output_cfg_**
- **MHAParser::mhaconfig_mon_t mhaconfig_in**
- **MHAParser::mhaconfig_mon_t mhaconfig_out**

Additional Inherited Members

5.250.1 Detailed Description

```
template<class runtime_cfg_t>
class MHAPLugin::plugin_t< runtime_cfg_t >
```

The template class for C++ openMHA plugins.

Template Parameters

| | |
|----------------------------|-------------------------|
| <i>runtime_↔ cfg_t</i> | run-time configuration. |
|----------------------------|-------------------------|

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

This template class provides thread safe configuration handling and standard methods to be compatible to the C++ openMHA plugin wrapper macro **MHAPLUGIN_CALLBACKS** (p. 9).

The template parameter `runtime_cfg_t` should be the runtime configuration of the plugin.

See **MHAPLugin::config_t** (p. 654) for details on the thread safe communication update mechanism.

5.250.2 Constructor & Destructor Documentation

5.250.2.1 `template<class runtime_cfg_t > MHAPlugin::plugin_t< runtime_cfg_t >::plugin_t (const std::string & help, const algo_comm_t & iac)`

Constructor of plugin template.

Parameters

| | |
|-------------|--|
| <i>help</i> | Help comment to provide some general information about the plugin. |
| <i>iac</i> | AC space handle (will be stored into the member variable ac). |

5.250.2.2 `template<class runtime_cfg_t > MHAPlugin::plugin_t< runtime_cfg_t >::~plugin_t () [virtual]`

5.250.3 Member Function Documentation

5.250.3.1 `template<class runtime_cfg_t> virtual void MHAPlugin::plugin_t< runtime_cfg_t >::prepare (mhaconfig_t &) [pure virtual]`

Implemented in [bbcalib_interface_t](#) (p. 192), [calibrator_t](#) (p. 195), [analysispath_if_t](#) (p. 183), [adm_if_t](#) (p. 166), [overlapadd::overlapadd_if_t](#) (p. 791), [frequency_translator_t](#) (p. 293), [noisePowProposedScale::interface_t](#) (p. 787), [dc_simple::dc_if_t](#) (p. 227), [dc::dc_if_t](#) (p. 217), [multibandcompressor::interface_t](#) (p. 779), [combc_if_t](#) (p. 207), [coherence::cohflt_if_t](#) (p. 201), [plugin_interface_t](#) (p. 798), [example6_t](#) (p. 280), [smoothgains_bridge::overlapadd_if_t](#) (p. 838), [MHAPlugin_Resampling::resampling_if_t](#) (p. 665), [shadowfilter_end::shadowfilter_end_t](#) (p. 834), [ac2wave_if_t](#) (p. 126), [noise_t](#) (p. 785), [nlms_t](#) (p. 782), [prediction_error](#) (p. 813), [spec2wave_if_t](#) (p. 846), [mhachain::chain_base_t](#) (p. 439), [acsave::acsave_t](#) (p. 144), [fader_wave::fader_wave_if_t](#) (p. 284), [doasvm_feature_extraction](#) (p. 247), [rmslevel_if_t](#) (p. 818), [shadowfilter_begin::shadowfilter_begin_t](#) (p. 831), [example3_t](#) (p. 273), [example4_t](#) (p. 277), [lpc_bl_predictor](#) (p. 342), [lpc_burglattice](#) (p. 347), [steerbf](#) (p. 852), [delaysum::delaysum_if_t](#) (p. 239), [acPooling_wave](#) (p. 139), [lpc](#) (p. 339), [fader_if_t](#) (p. 282), [acConcat_wave](#) (p. 130), [db_if_t](#) (p. 213), [acSteer](#) (p. 151), [acTransform_wave](#) (p. 155), [wave2spec_if_t](#) (p. 868), [gain::gain_if_t](#) (p. 302), [droptect_t](#) (p. 251), [example1_t](#) (p. 267), [sine_t](#) (p. 836), [doasvm_classification](#) (p. 243), [example2_t](#) (p. 270), [wavrec_t](#) (p. 872), [fftfilterbank::fftfb_interface_t](#) (p. 287), [route::interface_t](#) (p. 821), [matrixmixer::matmix_t](#) (p. 354), [altplugs_t](#) (p. 177), [softclip_t](#) (p. 842), [save_spec_t](#) (p. 827), [save_wave_t](#) (p. 828), [acmon::acmon_t](#) (p. 136), [timoSmooth](#) (p. 864), [identity_t](#) (p. 308), [delay::interface_t](#) (p. 237), [cpuload_t](#) (p. 211), [ds_t](#) (p. 254), and [us_t](#) (p. 866).

5.250.3.2 `template<class runtime_cfg_t > void MHAPlugin::plugin_t< runtime_cfg_t >::release () [virtual]`

Reimplemented in [bbcalib_interface_t](#) (p. 192), [calibrator_t](#) (p. 195), [analysispath_if_t](#) (p. 183), [adm_if_t](#) (p. 166), [overlapadd::overlapadd_if_t](#) (p. 791), [frequency_translator_t](#) (p. 293), [dc_simple::dc_if_t](#) (p. 227), [multibandcompressor::interface_t](#) (p. 779),

coherence::cohflt_if_t (p. 201), **smoothgains_bridge::overlapadd_if_t** (p. 838), **MHAPlugin_Resampling::resampling_if_t** (p. 665), **ac2wave_if_t** (p. 126), **nlms_t** (p. 783), **prediction_error** (p. 813), **mhachain::chain_base_t** (p. 439), **acsave::acsave_t** (p. 144), **fader_wave::fader_wave_if_t** (p. 284), **doasvm_feature_extraction** (p. 248), **example3_t** (p. 274), **example4_t** (p. 277), **lpc_bl_predictor** (p. 343), **lpc_burglattice** (p. 347), **steerbf** (p. 853), **delaysum::delaysum_if_t** (p. 239), **acPooling_wave** (p. 140), **lpc** (p. 340), **acConcat_wave** (p. 131), **db_if_t** (p. 213), **acSteer** (p. 152), **acTransform_wave** (p. 156), **droptect_t** (p. 251), **gain::gain_if_t** (p. 302), **example2_t** (p. 270), **doasvm_classification** (p. 244), **wavrec_t** (p. 872), **fftfilterbank::fftfb_interface_t** (p. 289), **route::interface_t** (p. 821), **example1_t** (p. 267), **altplugs_t** (p. 178), **acmon::acmon_t** (p. 137), **timoSmooth** (p. 864), **identity_t** (p. 308), **ds_t** (p. 254), and **us_t** (p. 866).

5.250.3.3 `template<class runtime_cfg_t> void MHAPlugin::plugin_t< runtime_cfg_t >::prepare_ (mhaconfig_t & cf)`

5.250.3.4 `template<class runtime_cfg_t> void MHAPlugin::plugin_t< runtime_cfg_t >::release_ ()`

5.250.3.5 `template<class runtime_cfg_t> bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared () const [inline]`

Flag, if the prepare method is successfully called (or currently evaluated)

5.250.3.6 `template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg () const [inline]`

Current input channel configuration.

5.250.3.7 `template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg () const [inline]`

Current output channel configuration.

5.250.4 Member Data Documentation

5.250.4.1 `template<class runtime_cfg_t> mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::tftype [protected]`

Member for storage of plugin interface configuration.

This member is defined for convenience of the developer. Typically, the actual contents of **mhaconfig_t** (p. 444) are stored in this member in the **prepare()** (p. 661) method.

Note

This member is likely to be removed in later versions, use **input_cfg()** (p. 662) and **output_cfg()** (p. 662) instead.

5.250.4.2 `template<class runtime_cfg_t> algo_comm_t MHAPLugin::plugin_t< runtime_cfg_t
>::ac [protected]`

AC handle of the chain.

This variable is initialized in the constructor and can be used by derived plugins to access the AC space. Its contents should not be modified.

5.250.4.3 `template<class runtime_cfg_t> bool MHAPLugin::plugin_t< runtime_cfg_t
>::is_prepared_ [private]`

5.250.4.4 `template<class runtime_cfg_t> mhaconfig_t MHAPLugin::plugin_t< runtime_cfg_t
>::input_cfg_ [private]`

5.250.4.5 `template<class runtime_cfg_t> mhaconfig_t MHAPLugin::plugin_t< runtime_cfg_t
>::output_cfg_ [private]`

5.250.4.6 `template<class runtime_cfg_t> MHAParser::mhaconfig_mon_t
MHAPLugin::plugin_t< runtime_cfg_t >::mhaconfig_in [private]`

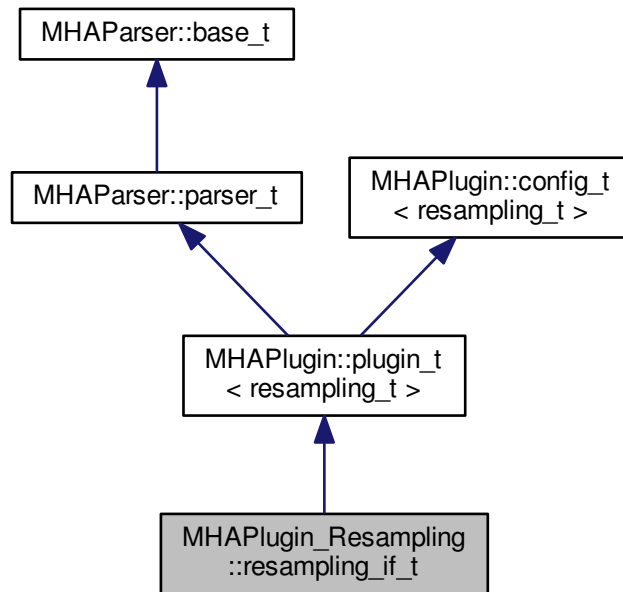
5.250.4.7 `template<class runtime_cfg_t> MHAParser::mhaconfig_mon_t
MHAPLugin::plugin_t< runtime_cfg_t >::mhaconfig_out [private]`

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

- **mha_plugin.hh**

5.251 MHAPLugin_Resampling::resampling_if_t Class Reference

Inheritance diagram for MHAPLugin_Resampling::resampling_if_t:



Public Member Functions

- **resampling_if_t** (algo_comm_t, std::string, std::string)
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)
- void **release** ()

Private Attributes

- **MHAParser::float_t srate**
- **MHAParser::int_t fragsize**
- **MHAParser::float_t nyquist_ratio**
- **MHAParser::float_t irslen_outer2inner**
- **MHAParser::float_t irslen_inner2outer**
- **MHAParser::mhapluginloader_t plugloader**
- std::string **chain**
- std::string **algo**

Additional Inherited Members

5.251.1 Constructor & Destructor Documentation

5.251.1.1 MHAPlugin_Resampling::resampling_if_t (
 algo_comm_t *iac*,
 std::string *th*,
 std::string *al*)

5.251.2 Member Function Documentation

5.251.2.1 **mha_wave_t*** MHAPlugin_Resampling::resampling_if_t::process (
 mha_wave_t* *s*)

5.251.2.2 **void** MHAPlugin_Resampling::resampling_if_t::prepare (
 mhaconfig_t & *conf*) [virtual]

Implements **MHAPlugin::plugin_t**< **resampling_t** > (p. 661).

5.251.2.3 **void** MHAPlugin_Resampling::resampling_if_t::release (
 void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **resampling_t** > (p. 661).

5.251.3 Member Data Documentation

5.251.3.1 **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::srate [private]

5.251.3.2 **MHAParser::int_t** MHAPlugin_Resampling::resampling_if_t::fragsize [private]

5.251.3.3 **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::nyquist_ratio
[private]

5.251.3.4 **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::irslen_outer2inner
[private]

5.251.3.5 **MHAParser::float_t** MHAPlugin_Resampling::resampling_if_t::irslen_inner2outer
[private]

5.251.3.6 **MHAParser::mhapluginloader_t** MHAPlugin_Resampling::resampling_if_t::plugloader
[private]

5.251.3.7 **std::string** MHAPlugin_Resampling::resampling_if_t::chain [private]

5.251.3.8 **std::string** MHAPlugin_Resampling::resampling_if_t::algo [private]

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

- **resampling.cpp**

5.252 MHAPugin_Resampling::resampling_t Class Reference

Public Member Functions

- **resampling_t** (unsigned int **outer_fragsize**, float **outer_srate**, unsigned int **inner_↵
fragsize**, float **inner_srate**, 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**

5.252.1 Constructor & Destructor Documentation

5.252.1.1 MHAPugin_Resampling::resampling_t::resampling_t (
 unsigned int *outer_fragsize*,
 float *outer_srate*,
 unsigned int *inner_fragsize*,
 float *inner_srate*,
 unsigned int *nch_in*,
 float *filter_length_in*,
 unsigned int *nch_out*,
 float *filter_length_out*,
 float *nyquist_ratio*,
 MHAParser::mhapluginloader_t & *plug*)

5.252.2 Member Function Documentation

5.252.2.1 **mha_wave_t * MHAPugin_Resampling::resampling_t::process (**
 mha_wave_t * s)

5.252.3 Member Data Documentation

- 5.252.3.1 unsigned MHAPlugin_Resampling::resampling_t::outer_fragsize [private]
- 5.252.3.2 unsigned MHAPlugin_Resampling::resampling_t::inner_fragsize [private]
- 5.252.3.3 float MHAPlugin_Resampling::resampling_t::outer_srate [private]
- 5.252.3.4 float MHAPlugin_Resampling::resampling_t::inner_srate [private]
- 5.252.3.5 unsigned MHAPlugin_Resampling::resampling_t::nchannels_in [private]
- 5.252.3.6 unsigned MHAPlugin_Resampling::resampling_t::nchannels_out [private]
- 5.252.3.7 MHAFilter::blockprocessing_polyphase_resampling_t
MHAPlugin_Resampling::resampling_t::outer2inner_resampling [private]
- 5.252.3.8 MHAFilter::blockprocessing_polyphase_resampling_t
MHAPlugin_Resampling::resampling_t::inner2outer_resampling [private]
- 5.252.3.9 MHAParser::mhapluginloader_t& MHAPlugin_Resampling::resampling_t::plugloader
[private]
- 5.252.3.10 MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::inner_signal
[private]
- 5.252.3.11 MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::output_signal
[private]

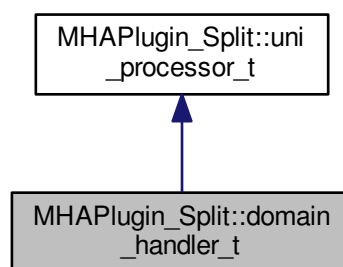
The documentation for this class was generated from the following file:

- **resampling.cpp**

5.253 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.

5.253.1 Detailed Description

Handles domain-specific partial input and output signal.

5.253.2 Constructor & Destructor Documentation

5.253.2.1 MHAPlugin_Split::domain_handler_t::domain_handler_t (
 const domain_handler_t &) [private]

Disallow copy constructor.

5.253.2.2 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.

5.253.2.3 virtual MHAPlugin_Split::domain_handler_t::~~domain_handler_t () [inline],
 [virtual]

Deallocation of signal holders.

5.253.3 Member Function Documentation

5.253.3.1 domain_handler_t& MHAPlugin_Split::domain_handler_t::operator= (
 const domain_handler_t &) [private]

Disallow assignment operator.

5.253.3.2 void MHAPlugin_Split::domain_handler_t::set_input_domain (
 const mhaconfig_t & settings_in) [inline]

Set parameters of input signal.

Parameters

| | |
|--------------------|---|
| <i>settings_in</i> | domain and dimensions of partial input signal |
|--------------------|---|

5.253.3.3 void MHAPlugin_Split::domain_handler_t::set_output_domain (
 const mhaconfig_t & settings_out) [inline]

Set output signal parameters.

Parameters

| | |
|---------------------|--|
| <i>settings_out</i> | domain and dimensions of partial output signal |
|---------------------|--|

5.253.3.4 void MHAPlugin_Split::domain_handler_t::deallocate_domains () [inline]

Deallocate domain indicators and signal holders.

5.253.3.5 unsigned MHAPlugin_Split::domain_handler_t::put_signal (
 mha_wave_t * s_in,
 unsigned start_channel) [inline]

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **wave_in** (p. 672).

Parameters

| | |
|----------------------|--|
| <i>s_in</i> | The combined waveform input signal. |
| <i>start_channel</i> | 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

5.253.3.6 unsigned MHAPlugin_Split::domain_handler_t::put_signal (
 mha_spec_t * s_in,
 unsigned start_channel) [inline]

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **spec_in** (p. 672).

Parameters

| | |
|----------------------|--|
| <i>s_in</i> | The combined spectrum input signal. |
| <i>start_channel</i> | 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

5.253.3.7 `unsigned MHAPLugin_Split::domain_handler_t::get_signal (`
`MHASignal::waveform_t * s_out,`
`unsigned start_channel) [inline]`

Store all partial signal output channels in the combined waveform signal with the given channel offset.

All channels present in **wave_out** (p. 672) will be copied. Caller may use (*wave_out)->num_channels to check the number of channels in advance.

Parameters

| | |
|----------------------|--|
| <i>s_out</i> | The combined waveform output signal. |
| <i>start_channel</i> | The channel offset (0-based) in s_out. |

Returns

The number of channels that were copied to the output signal

5.253.3.8 `unsigned MHAPLugin_Split::domain_handler_t::get_signal (`
`MHASignal::spectrum_t * s_out,`
`unsigned start_channel) [inline]`

Store all partial signal output channels in the combined spectrum signal with the given channel offset.

All channels present in **spec_out** (p. 672) will be copied. Caller may use (*spec_out)->num_channels to check the number of channels in advance.

Parameters

| | |
|----------------------|--|
| <i>s_out</i> | The combined spectrum output signal. |
| <i>start_channel</i> | The channel offset (0-based) in s_out. |

Returns

The number of channels that were copied to the output signal

5.253.3.9 `void MHAPLugin_Split::domain_handler_t::process () [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. 670) before this method may be called.

Implements **MHAPLugin_Split::uni_processor_t** (p. 691).

5.253.4 Member Data Documentation

5.253.4.1 MHASignal::waveform_t* MHAPlugin_Split::domain_handler_t::wave_in

Partial wave input signal.

5.253.4.2 mha_wave_t** MHAPlugin_Split::domain_handler_t::wave_out

Partial wave output signal.

5.253.4.3 MHASignal::spectrum_t* MHAPlugin_Split::domain_handler_t::spec_in

Partial spec input signal.

5.253.4.4 mha_spec_t** MHAPlugin_Split::domain_handler_t::spec_out

Partial spec input signal.

5.253.4.5 PluginLoader::fourway_processor_t* MHAPlugin_Split::domain_handler_t::processor

The domain-specific signal processing methods are implemented here.

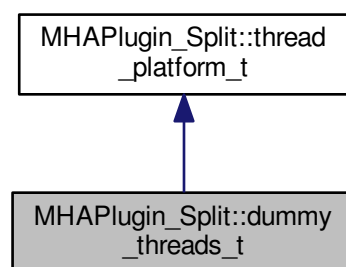
The documentation for this class was generated from the following file:

- **split.cpp**

5.254 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

5.254.1 Detailed Description

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

5.254.2 Constructor & Destructor Documentation

5.254.2.1 MHAPLugin_Split::dummy_threads_t::dummy_threads_t (
 uni_processor_t * *proc*,
 const std::string & *thread_scheduler*,
 int *thread_priority*) [inline]

Constructor.

Parameters

| | |
|-------------------------|---|
| <i>proc</i> | Pointer to the associated plugin loader |
| <i>thread_scheduler</i> | Unused in dummy thread platform |
| <i>thread_priority</i> | Unused in dummy thread platform |

5.254.3 Member Function Documentation

5.254.3.1 void MHAPLugin_Split::dummy_threads_t::kick_thread () [inline], [virtual]

perform signal processing immediately (no multiple threads in this dummy class)

Implements **MHAPLugin_Split::thread_platform_t** (p. 689).

5.254.3.2 void MHAPugin_Split::dummy_threads_t::catch_thread () [inline],[virtual]

No implementation needed: Processing has been completed during ummy_threads_t::kick_↵ thread.

Implements **MHAPugin_Split::thread_platform_t** (p. 689).

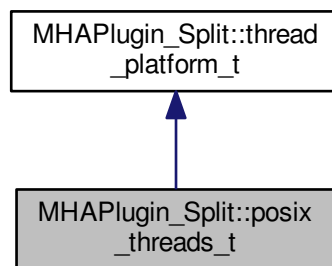
The documentation for this class was generated from the following file:

- **split.cpp**

5.255 MHAPugin_Split::posix_threads_t Class Reference

Posix threads specification of thread platform.

Inheritance diagram for MHAPugin_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

5.255.1 Detailed Description

Posix threads specification of thread platform.

5.255.2 Constructor & Destructor Documentation

5.255.2.1 MHAPLugin_Split::posix_threads_t::posix_threads_t (
 uni_processor_t * proc,
 const std::string & thread_scheduler,
 int thread_priority) [inline]

Constructor.

Parameters

| | |
|-------------------------|---|
| <i>proc</i> | Pointer to the associated signal processor instance |
| <i>thread_scheduler</i> | A string describing the posix thread scheduler. Possible values: "SCHED_OTHER", "SCHED_RR", "SCHED_FIFO". |
| <i>thread_priority</i> | The scheduling priority of the new thread. |

5.255.2.2 MHAPugin_Split::posix_threads_t::~~posix_threads_t () [inline]

Terminate thread.

5.255.3 Member Function Documentation**5.255.3.1 void MHAPugin_Split::posix_threads_t::kick_thread () [inline],[virtual]**

Start signal processing in separate thread.

Implements **MHAPugin_Split::thread_platform_t** (p. 689).

5.255.3.2 void MHAPugin_Split::posix_threads_t::catch_thread () [inline],[virtual]

Wait for signal processing to finish.

Implements **MHAPugin_Split::thread_platform_t** (p. 689).

5.255.3.3 static void* MHAPugin_Split::posix_threads_t::thread_start (void * thr) [inline],[static]

Thread start function.

5.255.3.4 void MHAPugin_Split::posix_threads_t::main () [inline]

Thread main loop. Wait for process/termination trigger, then act.

5.255.3.5 static std::string MHAPugin_Split::posix_threads_t::current_thread_scheduler () [inline],[static]**5.255.3.6 static int MHAPugin_Split::posix_threads_t::current_thread_priority () [inline],[static]****5.255.4 Member Data Documentation****5.255.4.1 pthread_mutex_t MHAPugin_Split::posix_threads_t::mutex [private]**

The mutex.

5.255.4.2 pthread_cond_t MHAPugin_Split::posix_threads_t::kick_condition [private]

The condition for signalling the kicking and termination.

5.255.4.3 pthread_cond_t MHAPugin_Split::posix_threads_t::catch_condition [private]

The condition for signalling the processing is finished.

5.255.4.4 pthread_attr_t MHAPugin_Split::posix_threads_t::attr [private]

Thread attributes.

5.255.4.5 struct sched_param MHAPugin_Split::posix_threads_t::priority [private]

Thread scheduling priority.

5.255.4.6 int MHAPugin_Split::posix_threads_t::scheduler [private]**5.255.4.7 pthread_t MHAPugin_Split::posix_threads_t::thread** [private]

The thread object.

5.255.4.8 bool MHAPugin_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.

5.255.4.9 bool MHAPugin_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.

5.255.4.10 bool MHAPugin_Split::posix_threads_t::termination_request [private]

Set to true by the destructor.

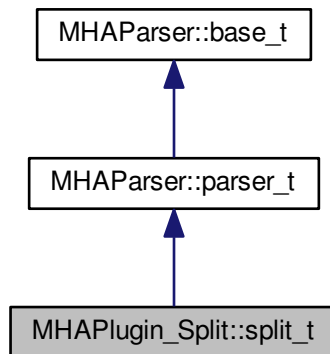
The documentation for this class was generated from the following file:

- **split.cpp**

5.256 MHAPLugin_Split::split_t Class Reference

Implements split plugin.

Inheritance diagram for MHAPLugin_Split::split_t:



Public Member Functions

- **split_t** (**algo_comm_t** iac, const std::string &chain_name, const std::string &algo_name)
Plugin constructor.
- **~split_t** ()
Plugin destructor. Unloads nested plugins.
- void **prepare_** (**mhaconfig_t** &)
Check signal parameters, prepare chains, and allocate output signal holders.
- void **release_** ()
Delete output signal holder and release chains.
- template<class SigTypeIn , class SigTypeOut >
void **process** (SigTypeIn *, SigTypeOut **)
Let the parallel plugins process channel groups of the input signal.

Private Member Functions

- void **update** ()
Load plugins in response to a value change in the algos variable.
- void **clear_chains** ()
Unload the plugins.
- **mha_wave_t** * **copy_output_wave** ()
- **mha_spec_t** * **copy_output_spec** ()

- `template<class SigType >`
`void trigger_processing (SigType *s_in)`
Split the argument input signal to groups of channels for the plugins and initiate signal processing.
- `template<class SigType >`
`void collect_result (SigType *s_out)`
Combine the output signal from the plugins.
- `MHASignal::waveform_t * signal_out (mha_wave_t **)`
Waveform domain output signal structure accessor.
- `MHASignal::spectrum_t * signal_out (mha_spec_t **)`
Spectrum domain output signal structure. Parameter is ignored.

Private Attributes

- `MHAEvents::patchbay_t< split_t > patchbay`
Reload plugins when the algos variable changes.
- `MHAParser::vstring_t algos`
Vector of plugins to load in parallel.
- `MHAParser::vint_t channels`
Number of channels to route through each plugin.
- `MHAParser::kw_t thread_platform`
Thread platform chooser.
- `MHAParser::kw_t worker_thread_scheduler`
Scheduler used for worker threads.
- `MHAParser::int_t worker_thread_priority`
Priority of worker threads.
- `MHAParser::string_mon_t framework_thread_scheduler`
Scheduler of the signal processing thread.
- `MHAParser::int_mon_t framework_thread_priority`
Priority of signal processing thread.
- `MHAParser::bool_t delay`
Switch to activate parallel processing of plugins at the cost of one block of additional delay.
- `std::vector< splitted_part_t * > chains`
Interfaces to parallel plugins.
- `MHASignal::waveform_t * wave_out`
Combined output waveforms structure.
- `MHASignal::spectrum_t * spec_out`
Combined output spectra structure.

Additional Inherited Members

5.256.1 Detailed Description

Implements split plugin.

An instance of class **split_t** (p. 678) implements the split plugin functionality: The audio channels are splitted and groups of audio channels are processed by different plugins in parallel.

5.256.2 Constructor & Destructor Documentation

5.256.2.1 `MHAPLugin_Split::split_t::split_t (`
 `algo_comm_t iac,`
 `const std::string & chain_name,`
 `const std::string & algo_name)`

Plugin constructor.

5.256.2.2 `MHAPLugin_Split::split_t::~~split_t ()`

Plugin destructor. Unloads nested plugins.

5.256.3 Member Function Documentation

5.256.3.1 `void MHAPLugin_Split::split_t::prepare_ (`
 `mhaconfig_t & signal_parameters)`

Check signal parameters, prepare chains, and allocate output signal holders.

5.256.3.2 `void MHAPLugin_Split::split_t::release_ ()`

Delete output signal holder and release chains.

5.256.3.3 `template<class SigTypeIn , class SigTypeOut > void MHAPLugin_Split::split_t::process (`
 `SigTypeIn * s_in,`
 `SigTypeOut ** s_out)`

Let the parallel plugins process channel groups of the input signal.

5.256.3.4 `void MHAPLugin_Split::split_t::update () [private]`

Load plugins in response to a value change in the algos variable.

5.256.3.5 `void MHAPLugin_Split::split_t::clear_chains () [private]`

Unload the plugins.

5.256.3.6 `mha_wave_t* MHAPLugin_Split::split_t::copy_output_wave () [private]`

5.256.3.7 `mha_spec_t* MHAPLugin_Split::split_t::copy_output_spec () [private]`

5.256.3.8 `template<class SigType > void MHAPLugin_Split::split_t::trigger_processing (`
 `SigType * s_in) [private]`

Split the argument input signal to groups of channels for the plugins and initiate signal processing.

5.256.3.9 `template<class SigType > void MHAPLugin_Split::split_t::collect_result (SigType * s_out) [private]`

Combine the output signal from the plugins.

5.256.3.10 `MHASignal::waveform_t* MHAPLugin_Split::split_t::signal_out (mha_wave_t **) [inline],[private]`

Waveform domain output signal structure accessor.

Parameter is only for domain disambiguation and is ignored.

5.256.3.11 `MHASignal::spectrum_t* MHAPLugin_Split::split_t::signal_out (mha_spec_t **) [inline],[private]`

Spectrum domain output signal structure. Parameter is ignored.

5.256.4 Member Data Documentation

5.256.4.1 `MHAEvents::patchbay_t<split_t> MHAPLugin_Split::split_t::patchbay [private]`

Reload plugins when the algos variable changes.

5.256.4.2 `MHAParser::vstring_t MHAPLugin_Split::split_t::algos [private]`

Vector of plugins to load in parallel.

5.256.4.3 `MHAParser::vint_t MHAPLugin_Split::split_t::channels [private]`

Number of channels to route through each plugin.

5.256.4.4 `MHAParser::kw_t MHAPLugin_Split::split_t::thread_platform [private]`

Thread platform chooser.

5.256.4.5 `MHAParser::kw_t MHAPLugin_Split::split_t::worker_thread_scheduler [private]`

Scheduler used for worker threads.

5.256.4.6 `MHAParser::int_t MHAPLugin_Split::split_t::worker_thread_priority [private]`

Priority of worker threads.

5.256.4.7 **MHAParser::string_mon_t** MHAPugin_Split::split_t::framework_thread_scheduler [private]

Scheduler of the signal processing thread.

5.256.4.8 **MHAParser::int_mon_t** MHAPugin_Split::split_t::framework_thread_priority [private]

Priority of signal processing thread.

5.256.4.9 **MHAParser::bool_t** MHAPugin_Split::split_t::delay [private]

Switch to activate parallel processing of plugins at the cost of one block of additional delay.

5.256.4.10 **std::vector<splitted_part_t*>** MHAPugin_Split::split_t::chains [private]

Interfaces to parallel plugins.

5.256.4.11 **MHASignal::waveform_t*** MHAPugin_Split::split_t::wave_out [private]

Combined output waveforms structure.

5.256.4.12 **MHASignal::spectrum_t*** MHAPugin_Split::split_t::spec_out [private]

Combined output spectra structure.

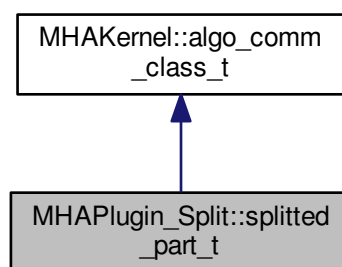
The documentation for this class was generated from the following file:

- **split.cpp**

5.257 MHAPugin_Split::splitted_part_t Class Reference

The **splitted_part_t** (p. 682) instance manages the plugin that performs processing on the reduced set of channels.

Inheritance diagram for MHAPugin_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. 686).*
- 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. 667) instance.*
- void **release** ()
*Delegates the release method to the plugin and deletes the **MHAPLugin_Split::domain_handler_t** (p. 667) instance.*
- std::string **parse** (const std::string &str)
Delegates parser incovation to plugin.
- template<class SigType >
unsigned **trigger_processing** (SigType *s_in, unsigned start_channel)
The domain handler copies the input signal channels.
- template<class SigType >
unsigned **collect_result** (SigType *s_out, unsigned start_channel)
Wait until processing is finished, then copy the output data.

Private Member Functions

- **splitted_part_t** (const **splitted_part_t** &)
Disallow copy constructor.
- **splitted_part_t** & **operator=** (const **splitted_part_t** &)
Disallow assignment operator.

Private Attributes

- **PluginLoader::fourway_processor_t** * **plug**
The plugin that performs the signal processing on the prepared channels.
- **domain_handler_t** * **domain**
The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.
- **thread_platform_t** * **thread**
The platform-dependent thread synchronization implementation.

Additional Inherited Members

5.257.1 Detailed Description

The **splitted_part_t** (p. 682) 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.

5.257.2 Constructor & Destructor Documentation

5.257.2.1 MHAPlugin_Split::splitted_part_t::splitted_part_t (const splitted_part_t &) [private]

Disallow copy constructor.

5.257.2.2 MHAPlugin_Split::splitted_part_t::splitted_part_t (const std::string & *plugname*, MHAParser::parser_t * *parent*)

Load the plugin for this partial signal path.

Loads the MHA plugin for a signal path of these audio channels.

Parameters

| | |
|-----------------|--|
| <i>plugname</i> | The name of the MHA plugin, optionally followed by a colon and the algorithm name. |
| <i>parent</i> | The parser node where the configuration of the new plugin is inserted. The plugin's parser name is the configured name (colon syntax). |

5.257.2.3 MHAPlugin_Split::splitted_part_t::splitted_part_t (PluginLoader::fourway_processor_t * *plugin*)

Create the handler for the partial signal.

The plugin is loaded by the caller, but it will be deleted by the destructor of this class. This constructor exists solely for testing purposes.

Parameters

| | |
|---------------|--|
| <i>plugin</i> | The plugin used for processing the signal. The new splitted_part_t (p. 684) instance will take ownership of this instance and release it in the destructor. |
|---------------|--|

5.257.2.4 MHAPLugin_Split::splitted_part_t::~splitted_part_t () throw ()

Destructor. Deletes the plugin **plug** (p. 686).

5.257.3 Member Function Documentation

5.257.3.1 splitted_part_t & MHAPLugin_Split::splitted_part_t::operator= (const splitted_part_t &) [private]

Disallow assignment operator.

5.257.3.2 void MHAPLugin_Split::splitted_part_t::prepare (mhaconfig_t & signal_parameters, const std::string & thread_platform, const std::string & thread_scheduler, int thread_priority)

Delegates the prepare method to the plugin and allocates a suitable **MHAPLugin_Split::domain_handler_t** (p. 667) instance.

Prepare the loaded plugin.

Plugin preparation.

Parameters

| | |
|--------------------------|--|
| <i>signal_parameters</i> | The signal description parameters for this path. |
| <i>thread_platform</i> | The name of the thread platform to use. Possible values: "posix", "win32", "dummy". |
| <i>thread_scheduler</i> | The name of the scheduler to use. Posix threads support "SCHED_OTHER", "SCHED_RR", "SCHED_FIFO". The other thread platforms do not support different thread schedulers. This value is not used for platforms other than "posix". |
| <i>thread_priority</i> | The new thread priority. Interpretation and permitted range depend on the thread platform and possibly on the scheduler. |

5.257.3.3 void MHAPLugin_Split::splitted_part_t::release (void)

Delegates the release method to the plugin and deletes the **MHAPLugin_Split::domain_handler_t** (p. 667) instance.

Release the loaded plugin.

Plugin release.

5.257.3.4 `std::string MHAPlugin_Split::splitted_part_t::parse (`
`const std::string & str) [inline]`

Delegates parser incovation to plugin.

5.257.3.5 `template<class SigType > unsigned MHAPlugin_Split::splitted_part_t::trigger_processing (`
`SigType * s_in,`
`unsigned start_channel) [inline]`

The domain handler copies the input signal channels.

Then, processing is initiated.

Parameters

| | |
|----------------------|---|
| <i>s_in</i> | The combined input signal. |
| <i>start_channel</i> | The index (0-based) of the first channel in <i>s_in</i> to be copied to the partial input signal. |

Returns

The number of channels that were copied from the input signal

5.257.3.6 `template<class SigType > unsigned MHAPlugin_Split::splitted_part_t::collect_result (`
`SigType * s_out,`
`unsigned start_channel) [inline]`

Wait until processing is finished, then copy the output data.

Parameters

| | |
|----------------------|--|
| <i>s_out</i> | The combined waveform output signal. |
| <i>start_channel</i> | The channel offset (0-based) in <i>s_out</i> . |

Returns

The number of channels that were copied to the output signal

5.257.4 Member Data Documentation

5.257.4.1 `PluginLoader::fourway_processor_t* MHAPlugin_Split::splitted_part_t::plug`
`[private]`

The plugin that performs the signal processing on the prepared channels.

5.257.4.2 domain_handler_t* MHAPLugin_Split::splitted_part_t::domain [private]

The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.

5.257.4.3 thread_platform_t* MHAPLugin_Split::splitted_part_t::thread [private]

The platform-dependent thread synchronization implementation.

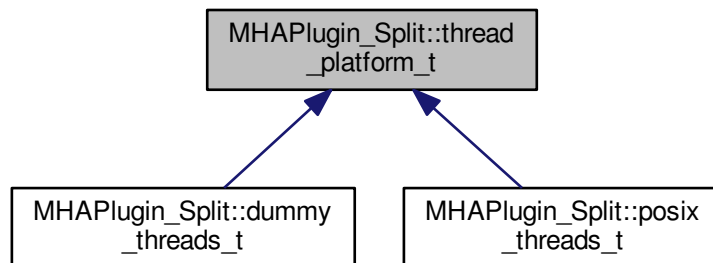
The documentation for this class was generated from the following file:

- **split.cpp**

5.258 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.

5.258.1 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.

5.258.2 Constructor & Destructor Documentation

5.258.2.1 **MHAPLugin_Split::thread_platform_t::thread_platform_t** (
 const **thread_platform_t** &) [*private*]

Disallow copy constructor.

5.258.2.2 **MHAPLugin_Split::thread_platform_t::thread_platform_t** (
 uni_processor_t * proc) [*inline*]

Constructor.

Derived classes create the thread in the constructor.

Parameters

| | |
|-------------|---|
| <i>proc</i> | 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. 682) instance. |
|-------------|---|

5.258.2.3 `virtual MHAPLugin_Split::thread_platform_t::~~thread_platform_t () [inline],
[virtual]`

Make derived classes destructable via pointer to this base class.

Derived classes' destructors notify the thread that it should terminate itself, and wait for the termination to occur.

5.258.3 Member Function Documentation

5.258.3.1 `thread_platform_t& MHAPLugin_Split::thread_platform_t::operator= (
const thread_platform_t &) [private]`

Disallow assignment operator.

5.258.3.2 `virtual void MHAPLugin_Split::thread_platform_t::kick_thread () [pure virtual]`

Derived classes notify their processing thread that it should call `processor->process()`.

Implemented in **MHAPLugin_Split::posix_threads_t** (p. 676), and **MHAPLugin_Split::dummy_threads_t** (p. 673).

5.258.3.3 `virtual void MHAPLugin_Split::thread_platform_t::catch_thread () [pure virtual]`

Derived classes wait for their signal processing thread to return from the call to `part->process()`.

Implemented in **MHAPLugin_Split::posix_threads_t** (p. 676), and **MHAPLugin_Split::dummy_threads_t** (p. 674).

5.258.4 Member Data Documentation

5.258.4.1 `uni_processor_t* MHAPLugin_Split::thread_platform_t::processor [protected]`

A pointer to the plugin loader that processes the sound data in the channels for which this thread was created.

Using the **MHAPLugin_Split::uni_processor_t** (p. 690) interface instead of the `mhaplugin-loader` class directly for testability (no need to load real plugins for testing the thread platform).

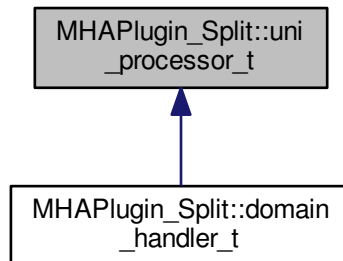
The documentation for this class was generated from the following file:

- **split.cpp**

5.259 MHAPugin_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 MHAPugin_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.

5.259.1 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.

5.259.2 Constructor & Destructor Documentation

5.259.2.1 virtual MHAPugin_Split::uni_processor_t::~~uni_processor_t() [inline], [virtual]

Classes containing virtual methods need virtual destructors.

5.259.3 Member Function Documentation

5.259.3.1 virtual void MHAPlugin_Split::uni_processor_t::process () [pure virtual]

This method uses some input signal, performs processing and stores the output signal somewhere.

This method also has to dispatch the process call based on the configured domains.

Signal transfer and domain configuration have to be done in derived class in different methods.

Implemented in **MHAPlugin_Split::domain_handler_t** (p. 671).

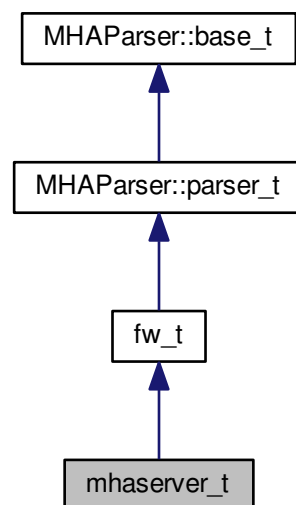
The documentation for this class was generated from the following file:

- **split.cpp**

5.260 mhaserver_t Class Reference

MHA Framework listening on TCP port for commands.

Inheritance diagram for mhaserver_t:



Public Member Functions

- **mhaserver_t** (const std::string &ao, const std::string &af, const std::string &lf)
- **~mhaserver_t** ()
- virtual std::string **received_group** (const std::string &line)
A line of text was received from network client.
- virtual void **acceptor_started** (int status)
Notification: "TCP port is open".
- virtual void **set_announce_port** (unsigned short **announce_port**)
If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.
- void **logstring** (const std::string &)
Log a message to log file.
- int **run** (unsigned short **port**, const std::string &_interface)
Accept network connections and act on commands.

Public Attributes

- **MHAParser::int_t port**

Private Attributes

- **MHA_TCP::Server * tcpserver**
- std::string **ack_ok**
- std::string **ack_fail**
- std::string **logfile**
- unsigned short **announce_port**
- **MHAParser::int_mon_t pid_mon**

Additional Inherited Members

5.260.1 Detailed Description

MHA Framework listening on TCP port for commands.

5.260.2 Constructor & Destructor Documentation

- #### 5.260.2.1 mhaserver_t::mhaserver_t (
- const std::string & ao,
const std::string & af,
const std::string & lf)

Parameters

| | |
|-----------|---|
| <i>ao</i> | Acknowledgement string at end of successful command responses |
| <i>af</i> | Acknowledgement string at end of failed command responses |
| <i>lf</i> | File system path of file to use as log file. MHA appends. |

5.260.2.2 mhaserver_t::~mhaserver_t ()

5.260.3 Member Function Documentation

5.260.3.1 std::string mhaserver_t::received_group (
 const std::string & *line*) [virtual]

A line of text was received from network client.

5.260.3.2 void mhaserver_t::acceptor_started (
 int *status*) [virtual]

Notification: "TCP port is open".

5.260.3.3 void mhaserver_t::set_announce_port (
 unsigned short *announce_port*) [virtual]

If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.

5.260.3.4 void mhaserver_t::logstring (
 const std::string & *s*) [inline]

Log a message to log file.

5.260.3.5 int mhaserver_t::run (
 unsigned short *port*,
 const std::string & *_interface*)

Accept network connections and act on commands.

Calls **acceptor_started()** (p. 693) when the TCP port is opened. Calls `received_group` for every line received.

Returns

exit code that can be used as process exit code

5.260.4 Member Data Documentation

5.260.4.1 `MHA_TCP::Server* mhaserver_t::tcpserver` [private]

5.260.4.2 `std::string mhaserver_t::ack_ok` [private]

5.260.4.3 `std::string mhaserver_t::ack_fail` [private]

5.260.4.4 `std::string mhaserver_t::logfile` [private]

5.260.4.5 `unsigned short mhaserver_t::announce_port` [private]

5.260.4.6 `MHAParser::int_mon_t mhaserver_t::pid_mon` [private]

5.260.4.7 `MHAParser::int_t mhaserver_t::port`

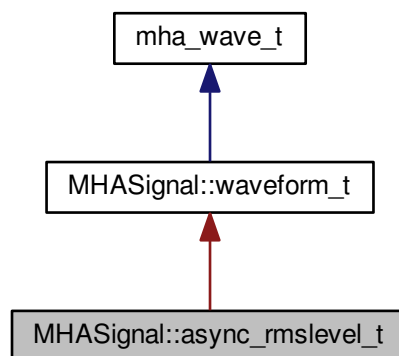
The documentation for this class was generated from the following file:

- **mhamain.cpp**

5.261 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

5.261.1 Detailed Description

Class for asynchronous level metering.

5.261.2 Constructor & Destructor Documentation

5.261.2.1 MHASignal::async_rmslevel_t::async_rmslevel_t (

unsigned int *frames*,
unsigned int *channels*)

Constructor for level metering class.

Allocate memory for metering. The RMS integration time corresponds to the number of frames in the buffer.

Parameters

| | |
|-----------------|---|
| <i>frames</i> | Number of frames to integrate. |
| <i>channels</i> | Number of channels used for level-metering. |

5.261.3 Member Function Documentation

5.261.3.1 `std::vector< float > MHASignal::async_rmslevel_t::rmslevel () const`

Read-only function for querying the current RMS level.

Returns

Vector of floats, one value for each channel, containing the RMS level in dB (SPL if calibrated properly).

5.261.3.2 `std::vector< float > MHASignal::async_rmslevel_t::peaklevel () const`

Read-only function for querying the current peak level.

Returns

Vector of floats, one value for each channel, containing the peak level in dB (SPL if calibrated properly).

5.261.3.3 `void MHASignal::async_rmslevel_t::process (mha_wave_t * s)`

Function to store a chunk of audio in the level meter.

Parameters

| | |
|----------------|---|
| <code>s</code> | Audio chunk (same number of channels required as given in the constructor). |
|----------------|---|

5.261.4 Member Data Documentation

5.261.4.1 `unsigned int MHASignal::async_rmslevel_t::pos [private]`

5.261.4.2 `unsigned int MHASignal::async_rmslevel_t::filled [private]`

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.262 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**

5.262.1 Constructor & Destructor Documentation

5.262.1.1 MHASignal::delay_spec_t::delay_spec_t (
 unsigned int *delay*,
 unsigned int *frames*,
 unsigned int *channels*)

5.262.1.2 MHASignal::delay_spec_t::~~delay_spec_t ()

5.262.2 Member Function Documentation

5.262.2.1 mha_spec_t * MHASignal::delay_spec_t::process (
 mha_spec_t * *s*)

5.262.3 Member Data Documentation

5.262.3.1 unsigned int MHASignal::delay_spec_t::delay [private]

5.262.3.2 MHASignal::spectrum_t** MHASignal::delay_spec_t::buffer [private]

5.262.3.3 unsigned int MHASignal::delay_spec_t::pos [private]

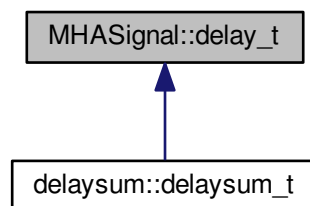
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.263 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**

5.263.1 Detailed Description

Class to realize a simple delay of waveform streams.

5.263.2 Constructor & Destructor Documentation

5.263.2.1 MHASignal::delay_t::delay_t (

std::vector< int > *delays*,
unsigned int *channels*)

Constructor.

Parameters

| | |
|-----------------|---|
| <i>delays</i> | Vector of delays, one entry for each channel. |
| <i>channels</i> | Number of channels expected. |

5.263.2.2 MHASignal::delay_t::~~delay_t ()

5.263.3 Member Function Documentation

5.263.3.1 mha_wave_t * MHASignal::delay_t::process (

mha_wave_t * s)

Processing method.

Parameters

| | |
|---|---|
| s | Input waveform fragment, with number of channels provided in constructor. |
|---|---|

Returns

Output waveform fragment.

5.263.3.2 `std::string MHASignal::delay_t::inspect () const` `[inline]`

5.263.4 Member Data Documentation

5.263.4.1 `unsigned int MHASignal::delay_t::channels` `[private]`

5.263.4.2 `unsigned int* MHASignal::delay_t::delays` `[private]`

5.263.4.3 `unsigned int* MHASignal::delay_t::pos` `[private]`

5.263.4.4 `mha_real_t** MHASignal::delay_t::buffer` `[private]`

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

- `mha_signal.hh`
- `mha_signal.cpp`

5.264 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**

5.264.1 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).

5.264.2 Constructor & Destructor Documentation

5.264.2.1 `MHASignal::delay_wave_t::delay_wave_t(`
 unsigned int *delay*,
 unsigned int *frames*,
 unsigned int *channels*)

5.264.2.2 `MHASignal::delay_wave_t::~~delay_wave_t()`

5.264.3 Member Function Documentation

5.264.3.1 `mha_wave_t * MHASignal::delay_wave_t::process(`
 mha_wave_t * s)

5.264.4 Member Data Documentation

5.264.4.1 unsigned int `MHASignal::delay_wave_t::delay` [private]

5.264.4.2 `MHASignal::waveform_t** MHASignal::delay_wave_t::buffer` [private]

5.264.4.3 unsigned int `MHASignal::delay_wave_t::pos` [private]

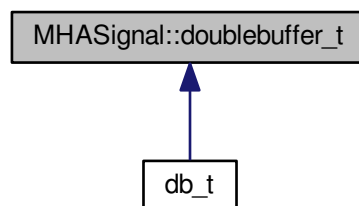
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

5.265 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**

5.265.1 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::outer_process()` callback. The pure virtual method **`doublebuffer_t::inner_process()`** (p. 702) is called whenever enough data is available.

5.265.2 Constructor & Destructor Documentation

5.265.2.1 MHASignal::doublebuffer_t::doublebuffer_t (
 unsigned int *nchannels_in*,
 unsigned int *nchannels_out*,
 unsigned int *outer_fragsize*,
 unsigned int *inner_fragsize*)

Constructor of double buffer.

Parameters

| | |
|-----------------------|---|
| <i>nchannels_in</i> | Number of channels at the input (both layers). |
| <i>nchannels_out</i> | Number of channels at the output (both layers). |
| <i>outer_fragsize</i> | Fragment size of the outer layer (e.g., hardware fragment size) |
| <i>inner_fragsize</i> | Fragment size of the inner layer (e.g., software fragment size) |

5.265.2.2 `MHASignal::doublebuffer_t::~doublebuffer_t()` `[virtual]`

5.265.3 Member Function Documentation

5.265.3.1 `mha_wave_t * MHASignal::doublebuffer_t::outer_process (mha_wave_t * s)`

Method to pass audio fragments into the inner layer.

Parameters

| | |
|----------|-------------------------------------|
| s | Pointer to input waveform fragment. |
|----------|-------------------------------------|

Returns

Pointer to output waveform fragment.

5.265.3.2 `virtual mha_wave_t * MHASignal::doublebuffer_t::inner_process (mha_wave_t * s)` `[protected], [pure virtual]`

Method to realize inner processing callback.

To be overwritten by derived classes.

Parameters

| | |
|----------|-------------------------------------|
| s | Pointer to input waveform fragment. |
|----------|-------------------------------------|

Returns

Pointer to output waveform fragment.

Implemented in **db_t** (p. [215](#)).

5.265.3.3 unsigned int MHASignal::doublebuffer_t::min (
 unsigned int *a*,
 unsigned int *b*) [inline], [private]

5.265.4 Member Data Documentation

5.265.4.1 waveform_t MHASignal::doublebuffer_t::outer_out [private]

5.265.4.2 mha_wave_t MHASignal::doublebuffer_t::this_outer_out [private]

5.265.4.3 waveform_t MHASignal::doublebuffer_t::inner_in [private]

5.265.4.4 waveform_t MHASignal::doublebuffer_t::inner_out [private]

5.265.4.5 unsigned int MHASignal::doublebuffer_t::k_inner [private]

5.265.4.6 unsigned int MHASignal::doublebuffer_t::k_outer [private]

5.265.4.7 unsigned int MHASignal::doublebuffer_t::ch [private]

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

- mha_signal.hh
- mha_signal.cpp

5.266 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**

5.266.1 Constructor & Destructor Documentation

5.266.1.1 **MHASignal::fft_t::fft_t** (
const unsigned int & *n*)

5.266.1.2 **MHASignal::fft_t::~~fft_t** ()

5.266.2 Member Function Documentation

5.266.2.1 void **MHASignal::fft_t::wave2spec** (
const **mha_wave_t** * *wave*,
mha_spec_t * *spec*,
bool *swap*)

fast fourier transform.

if swap is set, the buffer halves of the wave signal are exchanged before computing the fft.

5.266.2.2 void **MHASignal::fft_t::spec2wave** (
const **mha_spec_t** * *spec*,
mha_wave_t * *wave*)

5.266.2.3 void **MHASignal::fft_t::spec2wave** (
const **mha_spec_t** * *spec*,
mha_wave_t * *wave*,
unsigned int *offset*)

wave may have fewer number of frames than needed for a complete iFFT.

Only as many frames are written into wave as fit, starting with offset offset of the complete iFFT.

- 5.266.2.4 void MHASignal::fft_t::forward (
 mha_spec_t * *sIn*,
 mha_spec_t * *sOut*)
- 5.266.2.5 void MHASignal::fft_t::backward (
 mha_spec_t * *sIn*,
 mha_spec_t * *sOut*)
- 5.266.2.6 void MHASignal::fft_t::wave2spec_scale (
 const mha_wave_t * *wave*,
 mha_spec_t * *spec*,
 bool *swap*)
- 5.266.2.7 void MHASignal::fft_t::spec2wave_scale (
 const mha_spec_t * *spec*,
 mha_wave_t * *wave*)
- 5.266.2.8 void MHASignal::fft_t::forward_scale (
 mha_spec_t * *sIn*,
 mha_spec_t * *sOut*)
- 5.266.2.9 void MHASignal::fft_t::backward_scale (
 mha_spec_t * *sIn*,
 mha_spec_t * *sOut*)
- 5.266.2.10 void MHASignal::fft_t::sort_fftw2spec (
 fftw_real * *s_fftw*,
 mha_spec_t * *s_spec*,
 unsigned int *ch*) [private]

Arrange the order of an fftw spectrum to the internal order.

The fftw spectrum is arranged [r0 r1 r2 ... rn-1 in in-1 ... i1], while the internal order is [r0 – r1 i1 r2 i2 ... rn-1 in-1 rn –].

- 5.266.2.11 void MHASignal::fft_t::sort_spec2fftw (
 fftw_real * *s_fftw*,
 const mha_spec_t * *s_spec*,
 unsigned int *ch*) [private]

Arrange the order of an internal spectrum to the fftw order.

5.266.3 Member Data Documentation

- 5.266.3.1 `unsigned int MHASignal::fft_t::nfft` [private]
- 5.266.3.2 `unsigned int MHASignal::fft_t::n_re` [private]
- 5.266.3.3 `unsigned int MHASignal::fft_t::n_im` [private]
- 5.266.3.4 `mha_real_t MHASignal::fft_t::scale` [private]
- 5.266.3.5 `mha_real_t* MHASignal::fft_t::buf_in` [private]
- 5.266.3.6 `mha_real_t* MHASignal::fft_t::buf_out` [private]
- 5.266.3.7 `rfftw_plan MHASignal::fft_t::rfftw_plan_wave2spec` [private]
- 5.266.3.8 `rfftw_plan MHASignal::fft_t::rfftw_plan_spec2wave` [private]
- 5.266.3.9 `fftw_plan MHASignal::fft_t::fftw_plan_fft` [private]
- 5.266.3.10 `fftw_plan MHASignal::fft_t::fftw_plan_ifft` [private]

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

- `mha_signal_fft.h`
- `mha_signal.cpp`

5.267 MHASignal::hilbert_fftw_t Class Reference

Public Member Functions

- `hilbert_fftw_t` (unsigned int len)
- void `hilbert` (const `mha_wave_t *`, `mha_wave_t *`)

Private Attributes

- unsigned int `n`
- `rfftw_plan` `p1`
- `fftw_plan` `p2`
- `fftw_real *` `buf_r_in`
- `fftw_real *` `buf_r_out`
- `fftw_complex *` `buf_c_in`
- `fftw_complex *` `buf_c_out`
- `mha_real_t` `sc`

5.267.1 Constructor & Destructor Documentation

5.267.1.1 MHASignal::hilbert_fftw_t::hilbert_fftw_t (
 unsigned int *len*)

5.267.2 Member Function Documentation

5.267.2.1 void MHASignal::hilbert_fftw_t::hilbert (
 const mha_wave_t * *s_in*,
 mha_wave_t * *s_out*)

5.267.3 Member Data Documentation

5.267.3.1 unsigned int MHASignal::hilbert_fftw_t::n [private]

5.267.3.2 rfftw_plan MHASignal::hilbert_fftw_t::p1 [private]

5.267.3.3 fftw_plan MHASignal::hilbert_fftw_t::p2 [private]

5.267.3.4 fftw_real* MHASignal::hilbert_fftw_t::buf_r_in [private]

5.267.3.5 fftw_real* MHASignal::hilbert_fftw_t::buf_r_out [private]

5.267.3.6 fftw_complex* MHASignal::hilbert_fftw_t::buf_c_in [private]

5.267.3.7 fftw_complex* MHASignal::hilbert_fftw_t::buf_c_out [private]

5.267.3.8 mha_real_t MHASignal::hilbert_fftw_t::sc [private]

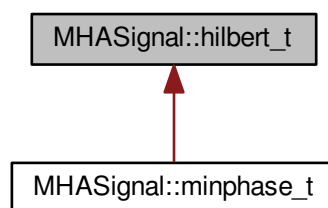
The documentation for this class was generated from the following file:

- mha_signal.cpp

5.268 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**

5.268.1 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.

5.268.2 Constructor & Destructor Documentation

5.268.2.1 MHASignal::hilbert_t::hilbert_t (unsigned int *len*)

Parameters

| | |
|------------|----------------------------|
| <i>len</i> | Length of waveform segment |
|------------|----------------------------|

5.268.2.2 MHASignal::hilbert_t::~~hilbert_t ()

5.268.3 Member Function Documentation

5.268.3.1 void MHASignal::hilbert_t::operator() (const **mha_wave_t** * *s_in*, **mha_wave_t** * *s_out*)

Apply Hilbert transformation on a waveform segment.

5.268.4 Member Data Documentation

5.268.4.1 void* MHASignal::hilbert_t::h [private]

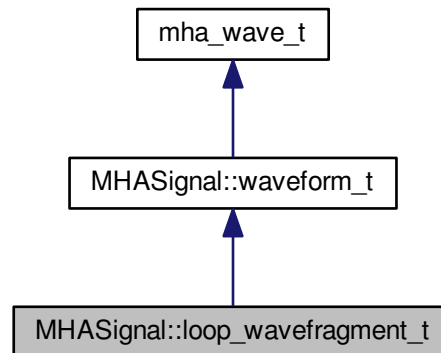
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.269 MHASignal::loop_wavefragment_t Class Reference

Copy a fixed waveform fragment to a series of waveform fragments of other size.

Inheritance diagram for MHASignal::loop_wavefragment_t:



Public Types

Public Member Functions

- **loop_wavefragment_t** (const **mha_wave_t** &src, bool loop, **level_mode_t** level_mode, std::vector< int > **channels**, unsigned int startpos=0)
*Constructor to create an instance of **loop_wavefragment_t** (p. 709) 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

5.269.1 Detailed Description

Copy a fixed waveform fragment to a series of waveform fragments of other size.

This class is designed to continuously play back a waveform to an output stream, with variable output block size.

5.269.2 Member Enumeration Documentation

5.269.2.1 `enum MHASignal::loop_wavefragment_t::level_mode_t`

Switch for playback level mode.

Enumerator

relative The nominal level is applied as a gain to the source signal.

peak The nominal level is the peak level of source signal in Pascal.

rms The nominal level is the RMS level of the source signal in Pascal.

rms_limit40

5.269.2.2 `enum MHASignal::loop_wavefragment_t::playback_mode_t`

Switch for playback mode.

Enumerator

add Add source signal to output stream.

replace Replace output stream by source signal.

input Do nothing, keep output stream (source position is unchanged).

mute Mute output stream (source position is unchanged).

5.269.3 Constructor & Destructor Documentation

5.269.3.1 `MHASignal::loop_wavefragment_t::loop_wavefragment_t (` `const mha_wave_t & src,` `bool loop,` `level_mode_t level_mode,` `std::vector< int > channels,` `unsigned int startpos = 0)`

Constructor to create an instance of `loop_wavefragment_t` (p. 709) based on an existing waveform block.

Parameters

| | |
|-------------------|--|
| <i>src</i> | Waveform block to copy data from. |
| <i>loop</i> | Flag whether the block should be looped or played once. |
| <i>level_mode</i> | Configuration of playback level (see MHASignal::loop_wavefragment_t::level_mode_t (p. 710) for details) |
| <i>channels</i> | Mapping of input to output channels. |
| <i>startpos</i> | Starting position |

5.269.4 Member Function Documentation

5.269.4.1 `std::vector< int > MHASignal::loop_wavefragment_t::get_mapping (unsigned int channels)`

5.269.4.2 `void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode, mha_wave_t * level_pa, const std::vector< int > & channels)`

Add source waveform block to an output block.

Parameters

| | |
|-----------------|---|
| <i>s</i> | Output block (streamed signal). |
| <i>pmode</i> | Playback mode (add, replace, input, mute). |
| <i>level_pa</i> | Linear output level/gain (depending on level_mode parameter in constructor); one value for each sample in output block. |
| <i>channels</i> | Output channels |

5.269.4.3 `void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode, mha_wave_t * level_pa)`

Add source waveform block to an output block.

Parameters

| | |
|-----------------|---|
| <i>s</i> | Output block (streamed signal). |
| <i>pmode</i> | Playback mode (add, replace, input, mute). |
| <i>level_pa</i> | Linear output level/gain (depending on level_mode parameter in constructor); one value for each sample in output block. |

5.269.4.4 void MHASignal::loop_wavefragment_t::playback (
 mha_wave_t * s,
 playback_mode_t pmode)

Add source waveform block to an output block.

Parameters

| | |
|--------------|--|
| <i>s</i> | Output block (streamed signal). |
| <i>pmode</i> | Playback mode (add, replace, input, mute). |

5.269.4.5 void MHASignal::loop_wavefragment_t::set_level_lin (
 mha_real_t /)

5.269.4.6 void MHASignal::loop_wavefragment_t::set_level_db (
 mha_real_t /)

5.269.4.7 void MHASignal::loop_wavefragment_t::rewind () [inline]

5.269.4.8 void MHASignal::loop_wavefragment_t::locate_end () [inline]

5.269.4.9 bool MHASignal::loop_wavefragment_t::is_playback_active () const [inline]

5.269.5 Member Data Documentation

5.269.5.1 std::vector<int> MHASignal::loop_wavefragment_t::playback_channels [private]

5.269.5.2 bool MHASignal::loop_wavefragment_t::b_loop [private]

5.269.5.3 unsigned int MHASignal::loop_wavefragment_t::pos [private]

5.269.5.4 MHASignal::waveform_t MHASignal::loop_wavefragment_t::intern_level [private]

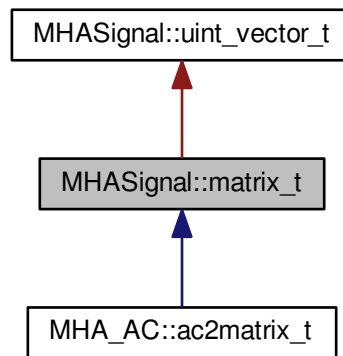
The documentation for this class was generated from the following files:

- mha_signal.hh
- mha_signal.cpp

5.270 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, described by size argument.
- **matrix_t** (const **MHASignal::matrix_t** &)
- **matrix_t** (const uint8_t *buf, unsigned int len)
Construct from memory area.
- **~matrix_t** ()
- **MHASignal::matrix_t &operator=** (const **MHASignal::matrix_t** &)
- **MHASignal::matrix_t &operator=** (const **comm_var_t** &v)
Fill matrix with data of an AC variable object.
- **comm_var_t get_comm_var** ()
Return a AC communication variable pointing to the data of the current matrix.
- unsigned int **dimension** () const
Return the dimension of the matrix.
- unsigned int **size** (unsigned int k) const
Return the size of the matrix.
- unsigned int **get_nelements** () const
Return total number of elements.

- **bool is_same_size** (const **MHASignal::matrix_t** &)
Test if matrix has same size as other.
- **bool iscomplex** () const
Return information about complexity.
- **mha_real_t & real** (const **MHASignal::uint_vector_t** &index)
Access real part of an element in a n-dimensional matrix.
- **mha_real_t & imag** (const **MHASignal::uint_vector_t** &index)
Access imaginary part of an element in a n-dimensional matrix.
- **mha_complex_t & operator()** (const **MHASignal::uint_vector_t** &index)
Access complex value of an element in a n-dimensional matrix.
- **const mha_real_t & real** (const **MHASignal::uint_vector_t** &index) const
Access real part of an element in a n-dimensional matrix.
- **const mha_real_t & imag** (const **MHASignal::uint_vector_t** &index) const
Access imaginary part of an element in a n-dimensional matrix.
- **const mha_complex_t & operator()** (const **MHASignal::uint_vector_t** &index) const
Access complex value of an element in a n-dimensional matrix.
- **mha_real_t & real** (unsigned int row, unsigned int col)
Access real part of an element in a two-dimensional matrix.
- **mha_real_t & imag** (unsigned int row, unsigned int col)
Access imaginary part of an element in a two-dimensional matrix.
- **mha_complex_t & operator()** (unsigned int row, unsigned int col)
Access complex value of an element in a two-dimensional matrix.
- **const mha_real_t & real** (unsigned int row, unsigned int col) const
Access real part of an element in a two-dimensional matrix.
- **const mha_real_t & imag** (unsigned int row, unsigned int col) const
Access imaginary part of an element in a two-dimensional matrix.
- **const mha_complex_t & operator()** (unsigned int row, unsigned int col) const
Access complex value of an element in a two-dimensional matrix.
- unsigned int **get_nreals** () const
- unsigned int **get_index** (unsigned int row, unsigned int col) const
- unsigned int **get_index** (const **MHASignal::uint_vector_t** &index) const
- unsigned int **numbytes** () const
Return number of bytes needed to store into memory.
- unsigned int **write** (uint8_t *buf, unsigned int len) const
Copy to memory area.
- **const mha_real_t * get_rdata** () const
Return pointer of real data.
- **const mha_complex_t * get_cdata** () const
Return pointer of complex data.

Private Attributes

- uint32_t **complex_ofs**
- uint32_t **nelements**
- union {
 mha_real_t * **rdata**
 mha_complex_t * **cdata**
 };

Additional Inherited Members

5.270.1 Detailed Description

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

Warning

The member functions **imag()** (p. 718) and **operator()** should only be called if the matrix is defined to hold complex values.

5.270.2 Constructor & Destructor Documentation

5.270.2.1 MHASignal::matrix_t::matrix_t (
 unsigned int *nrows*,
 unsigned int *ncols*,
 bool *b_is_complex* = true)

Create a two-dimensional matrix.

Parameters

| | |
|---------------------|------------------------------|
| <i>nrows</i> | Number of rows |
| <i>ncols</i> | Number of columns |
| <i>b_is_complex</i> | Add space for complex values |

5.270.2.2 MHASignal::matrix_t::matrix_t (
 const mha_spec_t & *spec*)

Create a two-dimensional matrix from a spectrum, copy values.

Parameters

| | |
|-------------|---------------------------|
| <i>spec</i> | Source spectrum structure |
|-------------|---------------------------|

5.270.2.3 **MHASignal::matrix_t::matrix_t (**
 const MHASignal::uint_vector_t & size,
 bool *b_is_complex* = true)

Create n-dimensional matrix, described by size argument.

Parameters

| | |
|---------------------|------------------------------|
| <i>size</i> | Size vector |
| <i>b_is_complex</i> | Add space for complex values |

5.270.2.4 **MHASignal::matrix_t::matrix_t (**
 const MHASignal::matrix_t & src)

5.270.2.5 **MHASignal::matrix_t::matrix_t (**
 const uint8_t * *buf*,
 unsigned int *len*)

Construct from memory area.

Warning

This constructor is not real time safe

5.270.2.6 **MHASignal::matrix_t::~~matrix_t ()**

5.270.3 Member Function Documentation

5.270.3.1 **matrix_t & MHASignal::matrix_t::operator= (**
 const MHASignal::matrix_t & src)

5.270.3.2 **MHASignal::matrix_t & MHASignal::matrix_t::operator= (**
 const comm_var_t & v)

Fill matrix with data of an AC variable object.

Parameters

| | |
|----------|---|
| <i>v</i> | Source AC variable (comm_var_t (p. 209)) |
|----------|---|

Note

The type and dimension of the AC variable must match the type and dimension of the matrix.

5.270.3.3 `comm_var_t MHASignal::matrix_t::get_comm_var ()`

Return a AC communication variable pointing to the data of the current matrix.

Returns

AC variable object (`comm_var_t` (p. 209)), valid for the life time of the matrix.

5.270.3.4 `unsigned int MHASignal::matrix_t::dimension () const` `[inline]`

Return the dimension of the matrix.

Returns

Dimension of the matrix

5.270.3.5 `unsigned int MHASignal::matrix_t::size (`
`unsigned int k) const` `[inline]`

Return the size of the matrix.

Parameters

| | |
|----------|-----------|
| <i>k</i> | Dimension |
|----------|-----------|

Returns

Size of the matrix in dimension *k*

5.270.3.6 `unsigned int MHASignal::matrix_t::get_nelements () const`

Return total number of elements.

5.270.3.7 `bool MHASignal::matrix_t::is_same_size (`
`const MHASignal::matrix_t & src)`

Test if matrix has same size as other.

5.270.3.8 `bool MHASignal::matrix_t::iscomplex () const` `[inline]`

Return information about complexity.

5.270.3.9 `mha_real_t& MHASignal::matrix_t::real (`
`const MHASignal::uint_vector_t & index)` `[inline]`

Access real part of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.10 `mha_real_t& MHASignal::matrix_t::imag (`
`const MHASignal::uint_vector_t & index)` `[inline]`

Access imaginary part of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.11 `mha_complex_t& MHASignal::matrix_t::operator() (`
`const MHASignal::uint_vector_t & index)` `[inline]`

Access complex value of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.12 `const mha_real_t& MHASignal::matrix_t::real (`
`const MHASignal::uint_vector_t & index) const` `[inline]`

Access real part of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.13 `const mha_real_t& MHASignal::matrix_t::imag (`
`const MHASignal::uint_vector_t & index) const` `[inline]`

Access imaginary part of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.14 `const mha_complex_t& MHASignal::matrix_t::operator() (`
`const MHASignal::uint_vector_t & index) const` `[inline]`

Access complex value of an element in a n-dimensional matrix.

Parameters

| | |
|--------------|--------------|
| <i>index</i> | Index vector |
|--------------|--------------|

5.270.3.15 `mha_real_t& MHASignal::matrix_t::real (`
`unsigned int row,`
`unsigned int col)` `[inline]`

Access real part of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

5.270.3.16 `mha_real_t& MHASignal::matrix_t::imag (`
`unsigned int row,`
`unsigned int col)` `[inline]`

Access imaginary part of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

5.270.3.17 `mha_complex_t& MHASignal::matrix_t::operator() (`
`unsigned int row,`
`unsigned int col)` `[inline]`

Access complex value of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

```
5.270.3.18  const mha_real_t& MHASignal::matrix_t::real (
                unsigned int row,
                unsigned int col ) const  [inline]
```

Access real part of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

```
5.270.3.19  const mha_real_t& MHASignal::matrix_t::imag (
                unsigned int row,
                unsigned int col ) const  [inline]
```

Access imaginary part of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

```
5.270.3.20  const mha_complex_t& MHASignal::matrix_t::operator() (
                unsigned int row,
                unsigned int col ) const  [inline]
```

Access complex value of an element in a two-dimensional matrix.

Parameters

| | |
|------------|--------------------------|
| <i>row</i> | Row number of element |
| <i>col</i> | Column number of element |

```
5.270.3.21  unsigned int MHASignal::matrix_t::get_nreals ( ) const  [inline]
```

```
5.270.3.22  unsigned int MHASignal::matrix_t::get_index (
                unsigned int row,
                unsigned int col ) const
```

```
5.270.3.23  unsigned int MHASignal::matrix_t::get_index (
                const MHASignal::uint_vector_t & index ) const
```

```
5.270.3.24  unsigned int MHASignal::matrix_t::numbytes ( ) const
```

Return number of bytes needed to store into memory.

5.270.3.25 unsigned int MHASignal::matrix_t::write (
 uint8_t * *buf*,
 unsigned int *len*) const

Copy to memory area.

5.270.3.26 const mha_real_t* MHASignal::matrix_t::get_rdata () const [inline]

Return pointer of real data.

5.270.3.27 const mha_complex_t* MHASignal::matrix_t::get_cdata () const [inline]

Return pointer of complex data.

5.270.4 Member Data Documentation

5.270.4.1 uint32_t MHASignal::matrix_t::complex_ofs [private]

5.270.4.2 uint32_t MHASignal::matrix_t::nelements [private]

5.270.4.3 mha_real_t* MHASignal::matrix_t::rdata

5.270.4.4 mha_complex_t* MHASignal::matrix_t::cdata

5.270.4.5 union { ... } [private]

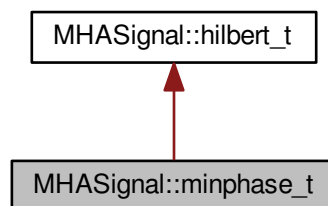
The documentation for this class was generated from the following files:

- mha_signal.hh
- mha_signal.cpp

5.271 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

5.271.1 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 $\mathcal{H}\{\dots\}$.

5.271.2 Constructor & Destructor Documentation

5.271.2.1 MHASignal::minphase_t::minphase_t (unsigned int *fftlen*, unsigned int *ch*)

Constructor.

Parameters

| | |
|---------------|--------------------|
| <i>fftlen</i> | FFT length |
| <i>ch</i> | Number of channels |

5.271.3 Member Function Documentation

5.271.3.1 void MHASignal::minphase_t::operator() (**mha_spec_t** * *s*)

Transform input spectrum to a minimal-phase spectrum, discarding the original phase.

Parameters

| | |
|---|-------------------------|
| s | Spectrum to operate on. |
|---|-------------------------|

5.271.4 Member Data Documentation

5.271.4.1 MHASignal::waveform_t MHASignal::minphase_t::phase [private]

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

- mha_signal.hh
- mha_signal.cpp

5.272 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**

5.272.1 Detailed Description

Simple simulation of fixpoint quantization.

5.272.2 Constructor & Destructor Documentation

5.272.2.1 MHASignal::quantizer_t::quantizer_t (
unsigned int num_bits)

Constructor.

Parameters

| | |
|-----------------|--|
| <i>num_bits</i> | Number of bits to simulate, or zero for limiting to [-1,1] only. |
|-----------------|--|

5.272.3 Member Function Documentation

5.272.3.1 `void MHASignal::quantizer_t::operator() (mha_wave_t & s)`

Quantization of a waveform fragment.

Parameters

| | |
|----------|------------------------------------|
| <i>s</i> | Waveform fragment to be quantized. |
|----------|------------------------------------|

5.272.4 Member Data Documentation

5.272.4.1 `bool MHASignal::quantizer_t::limit` [private]

5.272.4.2 `mha_real_t MHASignal::quantizer_t::upscale` [private]

5.272.4.3 `mha_real_t MHASignal::quantizer_t::downscale` [private]

5.272.4.4 `mha_real_t MHASignal::quantizer_t::up_limit` [private]

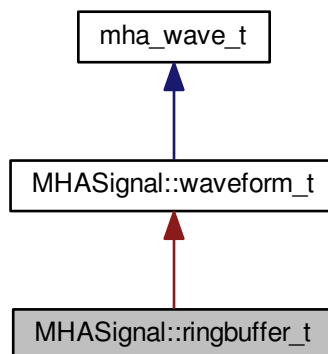
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.273 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

5.273.1 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. 727) method. Individual audio samples can be accessed and altered using the **value** (p. 726) method. Blocks of audio data can be deleted from the ringbuffer using the **discard** (p. 727) method.

5.273.2 Constructor & Destructor Documentation

5.273.2.1 `ringbuffer_t::ringbuffer_t (`
 `unsigned frames,`
 `unsigned channels,`
 `unsigned prefilled_frames)`

Creates new ringbuffer for time domain signal.

Constructor allocates enough storage so that *frames* audio samples can be stored in the ringbuffer.

Parameters

| | |
|-------------------------|--|
| <i>frames</i> | Size of ringbuffer in samples per channel. Maximum number of frames that can be stored in the ringbuffer at one time. This number cannot be changed after instance creation. |
| <i>channels</i> | Number of audio channels. |
| <i>prefilled_frames</i> | Number of frames to be prefilled with zero values. Many applications of a ringbuffer require the introduction of a delay. In practice, this delay is achieved by inserting silence audio samples (zeros) into the ringbuffer before the start of the actual signal is inserted for the first time. |

Exceptions

| | |
|----------------------------------|--|
| <i>MHA_Error</i> (p. 387) | if <code>prefilled_frames > frames</code> |
|----------------------------------|--|

5.273.3 Member Function Documentation

5.273.3.1 `unsigned MHASignal::ringbuffer_t::contained_frames () const` `[inline]`

number of currently contained frames

5.273.3.2 `mha_real_t& MHASignal::ringbuffer_t::value (`
 `unsigned frame,`
 `unsigned channel)` `[inline]`

Access to value stored in ringbuffer.

frame index is relative to the oldest frame stored in the ringbuffer, therefore, the meaning of the *frame* changes when the **discard** (p. 727) method is called.

Parameters

| | |
|----------------|--|
| <i>frame</i> | frame index, 0 corresponds to oldest frame stored. |
| <i>channel</i> | audio channel |

Returns

reference to contained sample value

Exceptions

| | |
|---------------------------|------------------------------------|
| MHA_Error (p. 387) | if channel or frame out of bounds. |
|---------------------------|------------------------------------|

5.273.3.3 void MHASignal::ringbuffer_t::discard (
 unsigned frames) [inline]

Discards the oldest frames.

Makes room for new **write** (p. 727), alters base frame index for **value** (p. 726)

Parameters

| | |
|--------|-----------------------------|
| frames | how many frames to discard. |
|--------|-----------------------------|

Exceptions

| | |
|---------------------------|--|
| MHA_Error (p. 387) | if frames > contained_frames (p. 726) |
|---------------------------|--|

5.273.3.4 void MHASignal::ringbuffer_t::write (
 mha_wave_t & signal) [inline]

Copies the contents of the signal into the ringbuffer if there is enough space.

Parameters

| | |
|--------|---|
| signal | New signal to be appended to the signal already present in the ringbuffer |
|--------|---|

Exceptions

| | |
|---------------------------|---|
| MHA_Error (p. 387) | if there is not enough space or if the channel count mismatches. Nothing is copied if the space is insufficient. |
|---------------------------|---|

5.273.4 Member Data Documentation

5.273.4.1 unsigned MHASignal::ringbuffer_t::next_read_frame_index [private]

Index of oldest frame in underlying storage for the ringbuffer.

This value is added to the frame parameter of the **value** (p. 726) method, and this value is altered when **discard** (p. 727) is called.

5.273.4.2 unsigned MHASignal::ringbuffer_t::next_write_frame_index [private]

Index of first free frame in underlying storage.

Next frame to be stored will be placed here.

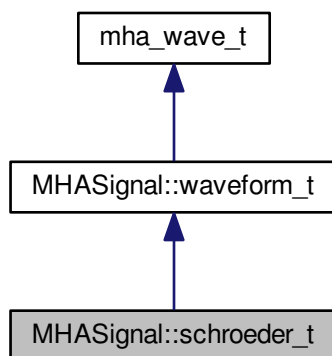
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

5.274 MHASignal::schroeder_t Class Reference

Schroeder tone complex class.

Inheritance diagram for MHASignal::schroeder_t:



Public Types

- typedef float(* **groupdelay_t**) (float f, float fmin, float fmax)
Function type for group delay definition.

Public Member Functions

- **schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::sign_t** sign=**up**, **mha_real_t** speed=1)
Constructor.
- **schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::groupdelay_t** freqfun=**MHASignal::schroeder_t::identity**, float fmin=0, float fmax=1, float eps=1e-10)
Construct create Schroeder tone complex from a given frequency function.

Static Public Member Functions

- static float **identity** (float x, float, float)
- static float **log_up** (float x, float fmin, float fmax)
- static float **log_down** (float x, float fmin, float fmax)

Additional Inherited Members

5.274.1 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.

5.274.2 Member Typedef Documentation

5.274.2.1 typedef float(* MHASignal::schroeder_t::groupdelay_t) (float f, float fmin, float fmax)

Function type for group delay definition.

Parameters

| | |
|--------|--|
| f | Frequency relative to Nyquist frequency. |
| $fmin$ | Minimum frequency relative to Nyquist frequency. |
| $fmax$ | Maximum frequency relative to Nyquist frequency. |

5.274.3 Member Enumeration Documentation

5.274.3.1 enum MHASignal::schroeder_t::sign_t

Enumerator for sign of Schroeder tone complex sweep direction.

Enumerator

- up** Sweep from zero to Nyquist frequency ($\sigma = -1$)
- down** Sweep from Nyquist frequency to zero ($\sigma = +1$)

5.274.4 Constructor & Destructor Documentation

5.274.4.1 MHASignal::schroeder_t::schroeder_t (

```

    unsigned int len,
    unsigned int channels = 1,
    schroeder_t::sign_t sign = up,
    mha_real_t speed = 1 )

```

Constructor.

Parameters of the Schroeder tone complex are configured in the constructor.

Parameters

| | |
|-----------------|--|
| <i>len</i> | Length τ of the Schroeder tone complex in samples |
| <i>channels</i> | Number of channels |
| <i>sign</i> | Sign σ of Schroeder sweep |
| <i>speed</i> | Relative speed α (curvature of phase function) |

5.274.4.2 MHASignal::schroeder_t::schroeder_t (

```

    unsigned int len,
    unsigned int channels = 1,
    schroeder_t::groupdelay_t freqfun = MHASignal::schroeder_t::identity,
    float fmin = 0,
    float fmax = 1,
    float eps = 1e-10 )

```

Construct create Schroeder tone complex from a given frequency function.

The frequency function $g(f)$ defines the sweep speed and sign (based on the group delay). It must be defined in the interval $[0,1)$ and should return values in the interval $[0,1]$.

$$\Phi(f) = -4\pi\tau \int_0^\tau g(f) \, df, \quad S(f) = e^{i\Phi(f)}$$

Parameters

| | |
|-----------------|--|
| <i>len</i> | Length τ of the Schroeder tone complex in samples. |
| <i>channels</i> | Number of channels. |
| <i>freqfun</i> | Frequency function $g(f)$. |
| <i>fmin</i> | Start frequency (relative to Nyquist frequency). |
| <i>fmax</i> | End frequency (relative to Nyquist frequency). |
| <i>eps</i> | Stability constant for frequency ranges not covered by Schroeder tone complex. |

5.274.5 Member Function Documentation

5.274.5.1 static float MHASignal::schroeder_t::identity (
float *x*,
float ,
float) [inline],[static]

5.274.5.2 static float MHASignal::schroeder_t::log_up (
float *x*,
float *fmin*,
float *fmax*) [inline],[static]

5.274.5.3 static float MHASignal::schroeder_t::log_down (
float *x*,
float *fmin*,
float *fmax*) [inline],[static]

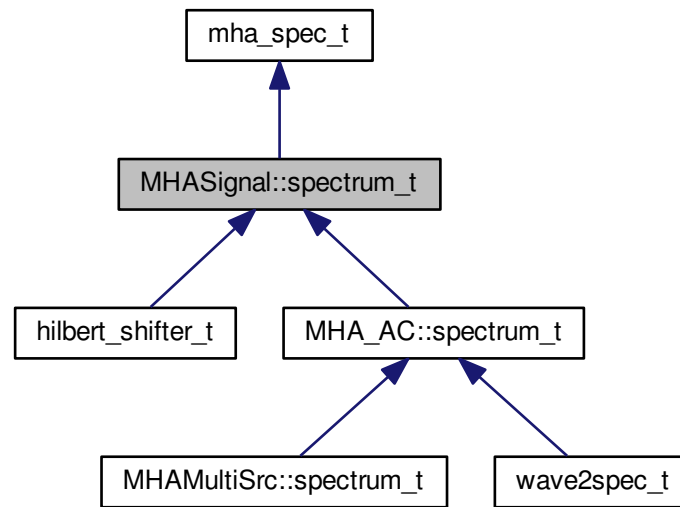
The documentation for this class was generated from the following files:

- mha_signal.hh
- mha_signal.cpp

5.275 MHASignal::spectrum_t Class Reference

a signal processing class for spectral data (based on **mha_spec_t** (p. [406](#)))

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

5.275.1 Detailed Description

a signal processing class for spectral data (based on **mha_spec_t** (p. 406))

5.275.2 Constructor & Destructor Documentation

5.275.2.1 **spectrum_t::spectrum_t** (
 const unsigned int & *frames*,
 const unsigned int & *channels*)

constructor of spectrum class

Allocates buffers and initializes memory to zeros.

Parameters

| | |
|-----------------|--|
| <i>frames</i> | number of frames (fft bins) in one channel. Number of Frames is usually fftlen / 2 + 1 |
| <i>channels</i> | number of channels |

5.275.2.2 **spectrum_t::spectrum_t** (
 const **mha_spec_t** & *src*) [explicit]

Copy constructor.

5.275.2.3 **spectrum_t::spectrum_t** (
 const **MHASignal::spectrum_t** & *src*)

Copy constructor.

5.275.2.4 **spectrum_t::spectrum_t** (
 const std::vector< **mha_complex_t** > & *src*)

5.275.2.5 **spectrum_t::~~spectrum_t** (
 void) [virtual]

Reimplemented in **MHA_AC::spectrum_t** (p. 366).

5.275.3 Member Function Documentation

5.275.3.1 `mha_complex_t& MHASignal::spectrum_t::operator() (`
 `unsigned int f,`
 `unsigned int ch) [inline]`

Access to element.

Parameters

| | |
|-----------|----------------|
| <i>f</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

5.275.3.2 `mha_complex_t& MHASignal::spectrum_t::operator[] (`
 `unsigned int k) [inline]`

Access to a single element, direct index into data buffer.

Parameters

| | |
|----------|--------------|
| <i>k</i> | Buffer index |
|----------|--------------|

Returns

Reference to element

5.275.3.3 `mha_complex_t& MHASignal::spectrum_t::value (`
 `unsigned int f,`
 `unsigned int ch) [inline]`

Access to element.

Parameters

| | |
|-----------|----------------|
| <i>f</i> | Bin number |
| <i>ch</i> | Channel number |

Returns

Reference to element

5.275.3.4 void spectrum_t::copy (
 const mha_spec_t & src)

copy all elements from a spectrum

Parameters

| | |
|------------|----------------|
| <i>src</i> | input spectrum |
|------------|----------------|

5.275.3.5 void spectrum_t::copy_channel (
 const mha_spec_t & s,
 unsigned sch,
 unsigned dch)

Copy one channel of a given spectrum signal to a target channel.

Parameters

| | |
|------------|--|
| <i>s</i> | Input spectrum signal |
| <i>sch</i> | Channel index in source signal |
| <i>dch</i> | Channel index in destination (this) signal |

5.275.3.6 void spectrum_t::export_to (
 mha_spec_t & dest)

copy elements to spectrum structure

Parameters

| | |
|-------------|--------------------------------|
| <i>dest</i> | destination spectrum structure |
|-------------|--------------------------------|

5.275.3.7 void spectrum_t::scale (
 const unsigned int & a,
 const unsigned int & b,
 const unsigned int & ch,
 const mha_real_t & val)

scale section [a,b) in channel "ch" by "val"

Parameters

| | |
|------------|----------------------|
| <i>a</i> | starting frame |
| <i>b</i> | end frame (excluded) |
| <i>ch</i> | channel number |
| <i>val</i> | scale factor |

5.275.3.8 void spectrum_t::scale_channel (
 const unsigned int & *ch*,
 const mha_real_t & *src*)

scale all elements in one channel

Parameters

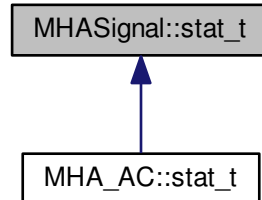
| | |
|------------|----------------|
| <i>ch</i> | channel number |
| <i>src</i> | scale factor |

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

- **mha_signal.hh**
- **mha_signal.cpp**

5.276 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

5.276.1 Constructor & Destructor Documentation

5.276.1.1 MHASignal::stat_t::stat_t (
 const unsigned int & *frames*,
 const unsigned int & *channels*)

5.276.2 Member Function Documentation

5.276.2.1 void MHASignal::stat_t::mean (
 mha_wave_t & *m*)

5.276.2.2 void MHASignal::stat_t::mean_std (
 mha_wave_t & *m*,
 mha_wave_t & *s*)

5.276.2.3 void MHASignal::stat_t::push (
 const mha_wave_t & *x*)

5.276.2.4 void MHASignal::stat_t::push (
 const mha_real_t & *x*,
 const unsigned int & *k*,
 const unsigned int & *ch*)

5.276.3 Member Data Documentation

5.276.3.1 MHASignal::waveform_t MHASignal::stat_t::n [private]

5.276.3.2 MHASignal::waveform_t MHASignal::stat_t::sum [private]

5.276.3.3 MHASignal::waveform_t MHASignal::stat_t::sum2 [private]

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

- mha_signal.hh
- mha_signal.cpp

5.277 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 phase 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.

5.277.1 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.

5.277.2 Constructor & Destructor Documentation

5.277.2.1 MHASignal::subsample_delay_t::subsample_delay_t (
const std::vector< float > & subsample_delay,
unsigned fftlen)

Constructor computes complex phase factors to apply to achieve subsample delay.

Parameters

| | |
|------------------------|--|
| <i>subsample_delay</i> | The subsample delay to apply. -0.5 <= subsample_delay <= 0.5 |
| <i>fftlen</i> | FFT length |

Exceptions

| | |
|---------------------------|------------------------------------|
| MHA_Error (p. 387) | if the parameters are out of range |
|---------------------------|------------------------------------|

5.277.3 Member Function Documentation

5.277.3.1 void MHASignal::subsample_delay_t::process (
mha_spec_t * s)

Apply the phase_gains to s to achieve the subsample delay.

5.277.3.2 void MHASignal::subsample_delay_t::process (
mha_spec_t * s,
unsigned idx)

Apply the phase 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. 387) | if idx >= s->num_channels |
|--|---------------------------|

5.277.4 Member Data Documentation

5.277.4.1 spectrum_t MHASignal::subsample_delay_t::phase_gains

The complex factors to apply to achieve the necessary phase shift.

5.277.4.2 unsigned MHASignal::subsample_delay_t::last_complex_bin [private]

index of the last complex fft bin for the used fft length.

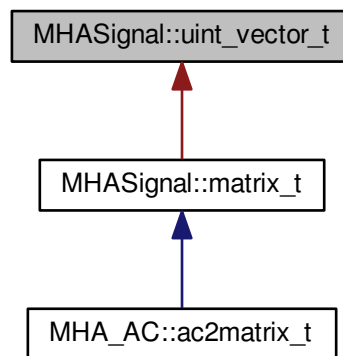
The documentation for this class was generated from the following files:

- mha_signal.hh
- mha_signal.cpp

5.278 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. 740).*
- 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**

5.278.1 Detailed Description

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

5.278.2 Constructor & Destructor Documentation

5.278.2.1 MHASignal::uint_vector_t::uint_vector_t (
 unsigned int *len*)

Constructor, initializes all elements to zero.

Parameters

| | |
|------------|-------------------|
| <i>len</i> | Length of vector. |
|------------|-------------------|

5.278.2.2 MHASignal::uint_vector_t::uint_vector_t (
 const uint_vector_t & *src*)5.278.2.3 MHASignal::uint_vector_t::uint_vector_t (
 const uint8_t * *buf*,
 unsigned int *len*)

Construct from memory area.

Warning

This constructor is not real time safe

5.278.2.4 MHASignal::uint_vector_t::~~uint_vector_t ()

5.278.3 Member Function Documentation

5.278.3.1 bool MHASignal::uint_vector_t::operator== (
 const uint_vector_t & *src*) const

Check for equality.

5.278.3.2 `uint_vector_t & MHASignal::uint_vector_t::operator= (`
`const uint_vector_t & src)`

Assign from other `uint_vector_t` (p. 740).

Warning

This assignment will fail if the lengths mismatch.

5.278.3.3 `unsigned int MHASignal::uint_vector_t::get_length () const` `[inline]`

Return the length of the vector.

5.278.3.4 `const uint32_t& MHASignal::uint_vector_t::operator[] (`
`unsigned int k) const` `[inline]`

Read-only access to elements.

5.278.3.5 `uint32_t& MHASignal::uint_vector_t::operator[] (`
`unsigned int k)` `[inline]`

Access to elements.

5.278.3.6 `unsigned int MHASignal::uint_vector_t::numbytes () const`

Return number of bytes needed to store into memory.

5.278.3.7 `unsigned int MHASignal::uint_vector_t::write (`
`uint8_t * buf,`
`unsigned int len) const`

Copy to memory area.

5.278.3.8 `const uint32_t* MHASignal::uint_vector_t::getdata () const` `[inline]`

Return pointer to the data field.

5.278.4 Member Data Documentation

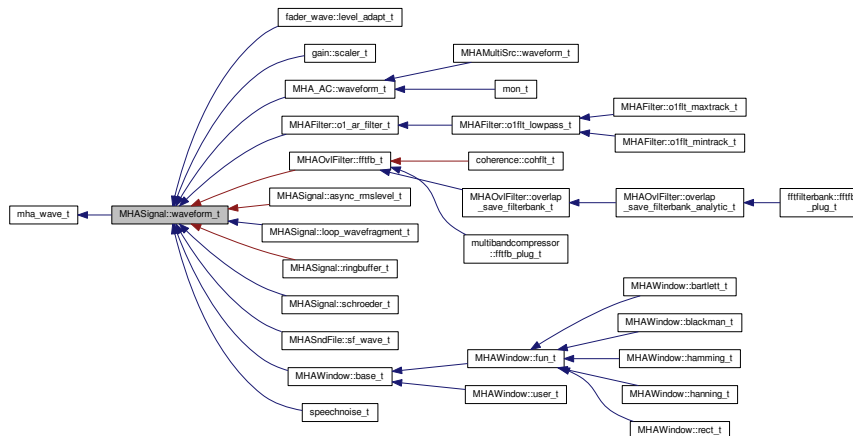
5.278.4.1 `uint32_t MHASignal::uint_vector_t::length` `[protected]`

5.278.4.2 `uint32_t* MHASignal::uint_vector_t::data` `[protected]`

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

- `mha_signal.hh`
- `mha_signal.cpp`

Inheritance diagram for MHASignal::waveform_t:



- **waveform_t** (const unsigned int &frames, const unsigned int &channels)
*constructor of **waveform_t** (p. 743)*
- **waveform_t** (const **mhaconfig_t** &cf)
Constructor to create a waveform from plugin configuration.
- **waveform_t** (const **mha_wave_t** &src)
*Copy constructor for **mha_wave_t** (p. 436) source.*
- **waveform_t** (const **MHASignal::waveform_t** &src)
Copy constructor.
- **waveform_t** (const std::vector< **mha_real_t** > &src)
Copy constructor for std::vector<mha_real_t> source.
- virtual ~**waveform_t** (void)
- void **operator=** (const **mha_real_t** &v)
- **mha_real_t** & **operator[]** (unsigned int k)
- const **mha_real_t** & **operator[]** (unsigned int k) const
- **mha_real_t** & **value** (unsigned int t, unsigned int ch)
Element accessor.
- **mha_real_t** & **operator()** (unsigned int t, unsigned int ch)
Element accessor.
- const **mha_real_t** & **value** (unsigned int t, unsigned int ch) const
Constant element accessor.
- const **mha_real_t** & **operator()** (unsigned int t, unsigned int ch) const
Constant element accessor.
- **mha_real_t** **sum** (const unsigned int &a, const unsigned int &b)

- sum of all elements between [a,b) in all channels*
- **mha_real_t sum** (const unsigned int &a, const unsigned int &b, const unsigned int &ch)
sum of all elements between [a,b) in channel ch
- **mha_real_t sum** ()
sum of all elements
- **mha_real_t sumsqr** ()
sum of square of all elements
- **mha_real_t sum_channel** (const unsigned int &)
return sum of all elements in one channel
- void **assign** (const unsigned int &k, const unsigned int &ch, const **mha_real_t** &val)
set frame "k" in channel "ch" to value "val"
- void **assign** (const **mha_real_t** &)
set all elements to value
- void **assign_frame** (const unsigned int &k, const **mha_real_t** &val)
assign value "val" to frame k in all channels
- void **assign_channel** (const unsigned int &c, const **mha_real_t** &val)
assign value "val" to channel ch in all frames
- void **copy** (const std::vector< **mha_real_t** > &v)
- void **copy** (const **mha_wave_t** &)
copy data from source into current waveform
- void **copy** (const **mha_wave_t** *)
- void **copy_channel** (const **mha_wave_t** &, unsigned int, unsigned int)
Copy one channel of a given waveform signal to a target channel.
- void **copy_from_at** (unsigned int, unsigned int, const **mha_wave_t** &, unsigned int)
Copy part of the source signal into part of this waveform object.
- void **export_to** (**mha_wave_t** &)
*copy data into allocated **mha_wave_t** (p. 436) 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/m²)
- void **powspec** (const **mha_spec_t** &)
get the power spectrum (in W/m²) 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

5.279.1 Detailed Description

signal processing class for waveform data (based on **mha_wave_t** (p. 436))

5.279.2 Constructor & Destructor Documentation

5.279.2.1 **waveform_t::waveform_t** (
 const unsigned int & *frames*,
 const unsigned int & *channels*)

constructor of **waveform_t** (p. 743)

Allocates buffer memory and initializes values to zero.

Parameters

| | |
|-----------------|----------------------------------|
| <i>frames</i> | number of frames in each channel |
| <i>channels</i> | number of channels |

5.279.2.2 **waveform_t::waveform_t** (
 const **mhaconfig_t** & *cf*) [explicit]

Constructor to create a waveform from plugin configuration.

Parameters

| | |
|-----------|----------------------|
| <i>cf</i> | Plugin configuration |
|-----------|----------------------|

5.279.2.3 **waveform_t::waveform_t** (
 const **mha_wave_t** & *src*) [explicit]

Copy constructor for **mha_wave_t** (p. 436) source.

5.279.2.4 **waveform_t::waveform_t** (
 const **MHASignal::waveform_t** & *src*)

Copy constructor.

5.279.2.5 **waveform_t::waveform_t** (
 const std::vector< **mha_real_t** > & *src*)

Copy constructor 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.

5.279.2.6 `waveform_t::~~waveform_t (`
`void) [virtual]`

Reimplemented in **MHA_AC::waveform_t** (p. 370).

5.279.3 Member Function Documentation

5.279.3.1 `void MHASignal::waveform_t::operator= (`
`const mha_real_t & v) [inline]`

5.279.3.2 `mha_real_t& MHASignal::waveform_t::operator[] (`
`unsigned int k) [inline]`

5.279.3.3 `const mha_real_t& MHASignal::waveform_t::operator[] (`
`unsigned int k) const [inline]`

5.279.3.4 `mha_real_t& MHASignal::waveform_t::value (`
`unsigned int t,`
`unsigned int ch) [inline]`

Element accessor.

Parameters

| | |
|-----------|----------------|
| <i>t</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

5.279.3.5 `mha_real_t& MHASignal::waveform_t::operator() (`
`unsigned int t,`
`unsigned int ch) [inline]`

Element accessor.

Parameters

| | |
|-----------|----------------|
| <i>t</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

```
5.279.3.6  const mha_real_t& MHASignal::waveform_t::value (
            unsigned int t,
            unsigned int ch ) const  [inline]
```

Constant element accessor.

Parameters

| | |
|-----------|----------------|
| <i>t</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

```
5.279.3.7  const mha_real_t& MHASignal::waveform_t::operator() (
            unsigned int t,
            unsigned int ch ) const  [inline]
```

Constant element accessor.

Parameters

| | |
|-----------|----------------|
| <i>t</i> | Frame number |
| <i>ch</i> | Channel number |

Returns

Reference to element

```
5.279.3.8  mha_real_t waveform_t::sum (
            const unsigned int & a,
            const unsigned int & b )
```

sum of all elements between [a,b) in all channels

Parameters

| | |
|----------|----------------------|
| <i>a</i> | starting frame |
| <i>b</i> | end frame (excluded) |

Returns

sum

5.279.3.9 mha_real_t waveform_t::sum (
 const unsigned int & *a*,
 const unsigned int & *b*,
 const unsigned int & *ch*)

sum of all elements between [a,b) in channel *ch*

Parameters

| | |
|-----------|---------------------|
| <i>a</i> | starting frame |
| <i>b</i> | end frame (exluded) |
| <i>ch</i> | channel number |

Returns

sum

5.279.3.10 mha_real_t waveform_t::sum ()

sum of all elements

Returns

sum of all elements

5.279.3.11 mha_real_t waveform_t::sumsq ()

sum of square of all elements

Returns

sum of square of all elements

5.279.3.12 mha_real_t waveform_t::sum_channel (
 const unsigned int & *ch*)

return sum of all elements in one channel

Parameters

| | |
|-----------|----------------|
| <i>ch</i> | channel number |
|-----------|----------------|

Returns

sum

5.279.3.13 void waveform_t::assign (
 const unsigned int & *k*,
 const unsigned int & *ch*,
 const mha_real_t & *val*)

set frame "k" in channel "ch" to value "val"

Parameters

| | |
|------------|----------------|
| <i>k</i> | frame number |
| <i>ch</i> | channel number |
| <i>val</i> | new value |

5.279.3.14 void waveform_t::assign (
 const mha_real_t & *val*)

set all elements to value

Parameters

| | |
|------------|-----------|
| <i>val</i> | new value |
|------------|-----------|

5.279.3.15 void waveform_t::assign_frame (
 const unsigned int & *k*,
 const mha_real_t & *val*)

assign value "val" to frame k in all channels

Parameters

| | |
|------------|--------------|
| <i>k</i> | frame number |
| <i>val</i> | new value |

5.279.3.16 void waveform_t::assign_channel (
 const unsigned int & *ch*,
 const mha_real_t & *val*)

assign value "val" to channel *ch* in all frames

Parameters

| | |
|------------|----------------|
| <i>ch</i> | channel number |
| <i>val</i> | new value |

5.279.3.17 void waveform_t::copy (
 const std::vector< mha_real_t > & *v*)

5.279.3.18 void waveform_t::copy (
 const mha_wave_t & *src*)

copy data from source into current waveform

Parameters

| | |
|------------|---|
| <i>src</i> | input data (need to be same size as target) |
|------------|---|

5.279.3.19 void waveform_t::copy (
 const mha_wave_t * *src*)

5.279.3.20 void waveform_t::copy_channel (
 const mha_wave_t & *src*,
 unsigned int *src_channel*,
 unsigned int *dest_channel*)

Copy one channel of a given waveform signal to a target channel.

Parameters

| | |
|---------------------|--------------------------------------|
| <i>src</i> | Input waveform signal |
| <i>src_channel</i> | Channel in source signal |
| <i>dest_channel</i> | Channel number in destination signal |

5.279.3.21 void waveform_t::copy_from_at (
 unsigned int *to_pos*,
 unsigned int *len*,
 const mha_wave_t & *src*,
 unsigned int *from_pos*)

Copy part of the source signal into part of this waveform object.

Source and target have to have the same number of channels.

Parameters

| | |
|-----------------|-------------------------|
| <i>to_pos</i> | Offset in target |
| <i>len</i> | Number of frames copied |
| <i>src</i> | Source |
| <i>from_pos</i> | Offset in source |

5.279.3.22 void waveform_t::export_to (
 mha_wave_t & *dest*)

copy data into allocated **mha_wave_t** (p. 436) structure

Parameters

| | |
|-------------|-----------------------|
| <i>dest</i> | destination structure |
|-------------|-----------------------|

5.279.3.23 void waveform_t::limit (
 const mha_real_t & *min*,
 const mha_real_t & *max*)

limit target to range [min,max]

Parameters

| | |
|------------|-------------|
| <i>min</i> | lower limit |
| <i>max</i> | upper limit |

5.279.3.24 void waveform_t::power (
 const waveform_t & *src*)

transform waveform signal (in Pa) to squared signal (in W/m²)

Parameters

| | |
|------------|--------------------------------|
| <i>src</i> | linear waveform signal (in Pa) |
|------------|--------------------------------|

5.279.3.25 void waveform_t::powspec (
 const mha_spec_t & src)

get the power spectrum (in W/m²) from a complex spectrum

Parameters

| | |
|------------|-------------------------------------|
| <i>src</i> | complex spectrum (normalized to Pa) |
|------------|-------------------------------------|

5.279.3.26 void waveform_t::scale (
 const unsigned int & a,
 const unsigned int & b,
 const unsigned int & ch,
 const mha_real_t & val)

scale section [a,b) in channel "ch" by "val"

Parameters

| | |
|------------|----------------------|
| <i>a</i> | starting frame |
| <i>b</i> | end frame (excluded) |
| <i>ch</i> | channel number |
| <i>val</i> | scale factor |

5.279.3.27 void waveform_t::scale (
 const unsigned int & k,
 const unsigned int & ch,
 const mha_real_t & val)

scale one element

Parameters

| | |
|------------|----------------|
| <i>k</i> | frame number |
| <i>ch</i> | channel number |
| <i>val</i> | scale factor |

5.279.3.28 void waveform_t::scale_channel (
 const unsigned int & ch,
 const mha_real_t & src)

scale one channel of target with a scalar

Parameters

| | |
|------------|----------------|
| <i>ch</i> | channel number |
| <i>src</i> | factor |

5.279.3.29 void waveform_t::scale_frame (
const unsigned int & *frame*,
const mha_real_t & *val*)

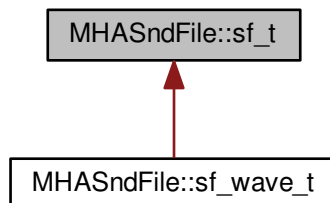
5.279.3.30 unsigned int MHASignal::waveform_t::get_size () const [inline]

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

- mha_signal.hh
- mha_signal.cpp

5.280 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**

5.280.1 Constructor & Destructor Documentation

5.280.1.1 `MHASndFile::sf_t::sf_t (`
 `const std::string & fname)`

5.280.1.2 `MHASndFile::sf_t::~sf_t ()`

5.280.2 Member Data Documentation

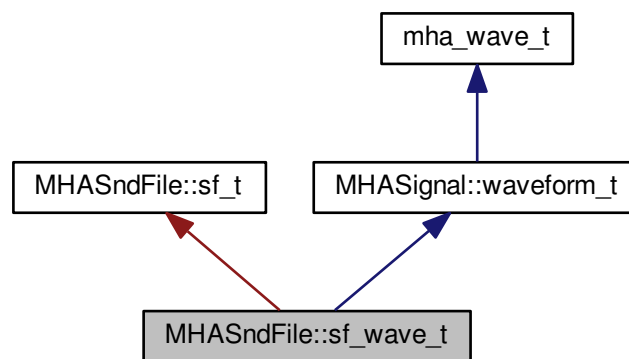
5.280.2.1 `SNDFILE* MHASndFile::sf_t::sf`

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

- **mhasndfile.h**
- **mhasndfile.cpp**

5.281 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

5.281.1 Constructor & Destructor Documentation

```

5.281.1.1 MHASndFile::sf_wave_t::sf_wave_t(
    const std::string & fname,
    mha_real_t peaklevel_db,
    unsigned int maxlen = std::numeric_limits<unsigned int>::max(),
    unsigned int startpos = 0,
    std::vector<int> channel_map = std::vector<int>() )

```

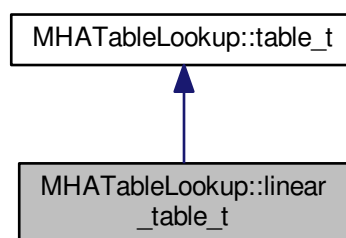
The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

5.282 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)
*constructor creates an empty **linear_table_t** (p. 755) 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

5.282.1 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.

5.282.2 Constructor & Destructor Documentation

5.282.2.1 linear_table_t::linear_table_t (void)

constructor creates an empty **linear_table_t** (p. 755) 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.

5.282.2.2 linear_table_t::~linear_table_t (void)

destructor

5.282.3 Member Function Documentation

5.282.3.1 mha_real_t linear_table_t::lookup (mha_real_t x) const [virtual]

look up the y value that is stored for the mesh point where x is lower than or equal to the x value given here.

This method does not extrapolate, so for $x < x_{\min}$, the y value for x_{\min} 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. 760).

5.282.3.2 mha_real_t linear_table_t::interp (mha_real_t x) const [virtual]

interpolate y value for the given x value.

The y values for the neighbouring mesh points are looked up and linearly interpolated. For x values outside the range of mesh points, the y value is extrapolated from the nearest two mesh points.

Precondition

prepare must have been called before interp may be called.

Implements **MHATableLookup::table_t** (p. 760).

5.282.3.3 void linear_table_t::set_xmin (
 mha_real_t xmin)

set the x value for the first mesh point.

Must be called before prepare can be called.

5.282.3.4 void linear_table_t::add_entry (
 mha_real_t y)

set the y value for the next mesh point.

Must be called at least twice before prepare can be called.

5.282.3.5 void linear_table_t::set_xmax (
 mha_real_t xmax)

this sets the x value for a past-the-end, not added mesh point.

Example:

```
t.set_xmin(100);  
t.add_entry(0); // mesh point {100,0}  
t.add_entry(1); // mesh point {110,1}  
// the next mesh point would be at x=120, but we do not add this  
t.set_xmax(120); // the x where the next mesh point would be  
t.prepare();
```

now, t.interp(100) == 0; t.interp(110) == 1; t.interp(105) == 0.5;

5.282.3.6 void linear_table_t::prepare (
 void)

prepare computes the x distance of the mesh points based on the values given to set_xmin, set_xmax, and the number of times that add_entry was called.

Precondition

set_xmin, set_xmax, add_entry functions must have been called before calling prepare, add_entry must have been called at least twice.

Only after this method has been called, interp or lookup may be called.

5.282.3.7 `void linear_table_t::clear (`
`void) [virtual]`

clear resets the state of this object to the state directly after construction.

mesh entries and x range are deleted.

interp and lookup may not be called after this function has been called unless prepare and before that its precondition methods are called again.

Implements **MHATableLookup::table_t** (p. 760).

5.282.4 Member Data Documentation

5.282.4.1 `mha_real_t* MHATableLookup::linear_table_t::vy [protected]`

5.282.4.2 `unsigned int MHATableLookup::linear_table_t::len [protected]`

5.282.4.3 `vector<mha_real_t> MHATableLookup::linear_table_t::vec_y [private]`

5.282.4.4 `mha_real_t MHATableLookup::linear_table_t::xmin [private]`

5.282.4.5 `mha_real_t MHATableLookup::linear_table_t::xmax [private]`

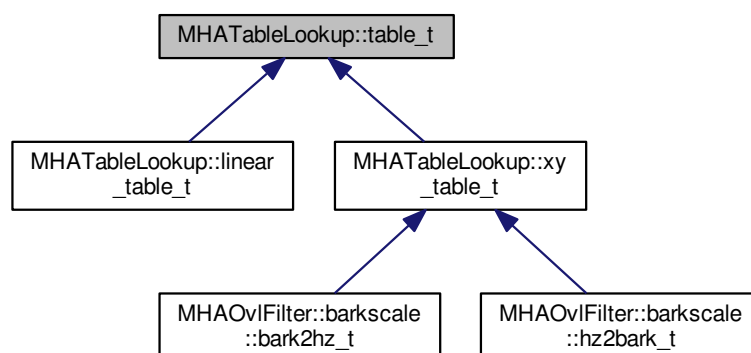
5.282.4.6 `mha_real_t MHATableLookup::linear_table_t::scalefac [private]`

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

- `mha_tablelookup.hh`
- `mha_tablelookup.cpp`

5.283 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

5.283.1 Constructor & Destructor Documentation

5.283.1.1 **table_t::table_t** (
void)

5.283.1.2 **table_t::~~table_t** (
void) [virtual]

5.283.2 Member Function Documentation

5.283.2.1 virtual mha_real_t MHA_{TableLookup}::table_t::lookup (
mha_real_t) const [pure virtual]

Implemented in **MHA_{TableLookup}::xy_table_t** (p. 762), and **MHA_{TableLookup}::linear_↔table_t** (p. 757).

5.283.2.2 virtual mha_real_t MHA_{TableLookup}::table_t::interp (
mha_real_t) const [pure virtual]

Implemented in **MHA_{TableLookup}::xy_table_t** (p. 763), and **MHA_{TableLookup}::linear_↔table_t** (p. 757).

5.283.2.3 virtual void MHA_{TableLookup}::table_t::clear (
void) [protected],[pure virtual]

Implemented in **MHA_{TableLookup}::xy_table_t** (p. 763), and **MHA_{TableLookup}::linear_↔table_t** (p. 759).

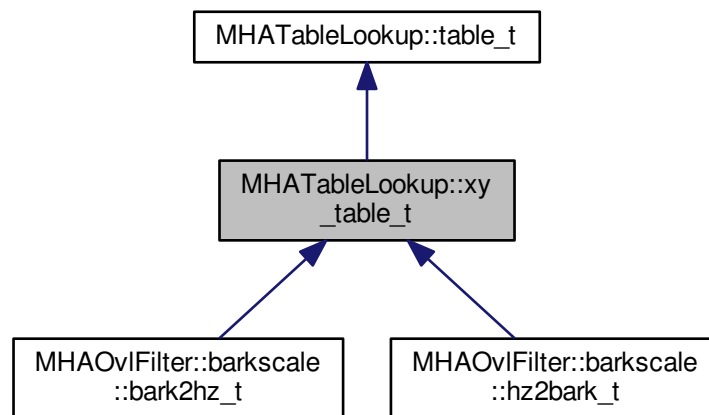
The documentation for this class was generated from the following files:

- mha_tablelookup.hh
- mha_tablelookup.cpp

5.284 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

5.284.1 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. 763) and `xy_table_t::lookup()` (p. 762). 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.

5.284.2 Constructor & Destructor Documentation

5.284.2.1 `xy_table_t::xy_table_t()`

5.284.3 Member Function Documentation

5.284.3.1 `mha_real_t xy_table_t::lookup(mha_real_t x) const` [virtual]

Return the y-value at the position of the nearest x value below input.

Parameters

| | |
|----------------|-------------|
| <code>x</code> | Input value |
|----------------|-------------|

Returns

y value at nearest x value below input.

Implements `MHATableLookup::table_t` (p. 760).

5.284.3.2 `mha_real_t xy_table_t::interp (`
`mha_real_t x) const` [virtual]

Linear interpolation function.

Parameters

| | |
|----------------|---------|
| <code>x</code> | x value |
|----------------|---------|

Returns

interpolated y value

Implements **MHATableLookup::table_t** (p. 760).

5.284.3.3 `void xy_table_t::add_entry (`
`mha_real_t x,`
`mha_real_t y)`

Add a single x-y pair entry.

Parameters

| | |
|----------------|-----------------------|
| <code>x</code> | x value |
| <code>y</code> | corresponding y value |

5.284.3.4 `void xy_table_t::add_entry (`
`mha_real_t * pVX,`
`mha_real_t * pVY,`
`unsigned int uLength)`

Add multiple entries at once.

Parameters

| | |
|----------------------|--------------------------|
| <code>pVX</code> | array of x values |
| <code>pVY</code> | array of y values |
| <code>uLength</code> | Length of x and y arrays |

5.284.3.5 `void xy_table_t::clear (`
`void)` [virtual]

Clear the table and transformation functions.

Implements **MHATableLookup::table_t** (p. 760).

5.284.3.6 `void xy_table_t::set_xfun (`
`float(*) (float) fun)`

Set transformation function for x values.

Parameters

| | |
|------------|--------------------------|
| <i>fun</i> | Transformation function. |
|------------|--------------------------|

5.284.3.7 `void xy_table_t::set_yfun (`
`float(*) (float) fun)`

Set transformation function for y values during insertion.

Parameters

| | |
|------------|--------------------------|
| <i>fun</i> | Transformation function. |
|------------|--------------------------|

5.284.3.8 `void xy_table_t::set_xyfun (`
`float(*) (float, float) fun)`

Set transformation function for y values during insertion, based on x and y values.

Parameters

| | |
|------------|--------------------------|
| <i>fun</i> | Transformation function. |
|------------|--------------------------|

5.284.3.9 `std::pair<mha_real_t,mha_real_t> MHATableLookup::xy_table_t::get_xlimits () const`
`[inline]`

returns the min and max x of all mesh points that are stored in the lookup table, i.e.
after transformation with xfun, if any. Not real-time safe

5.284.4 Member Data Documentation

5.284.4.1 `std::map<mha_real_t,mha_real_t> MHATableLookup::xy_table_t::mXY` `[private]`

5.284.4.2 `float(* MHATableLookup::xy_table_t::xfun) (float)` `[private]`

5.284.4.3 `float(* MHATableLookup::xy_table_t::yfun) (float)` `[private]`

5.284.4.4 `float(* MHATableLookup::xy_table_t::xyfun) (float, float)` `[private]`

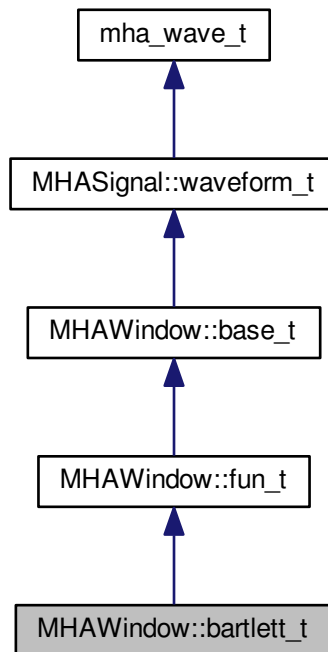
The documentation for this class was generated from the following files:

- `mha_tablelookup.hh`
- `mha_tablelookup.cpp`

5.285 MHAWindow::bartlett_t Class Reference

Bartlett window.

Inheritance diagram for MHAWindow::bartlett_t:



Public Member Functions

- **bartlett_t** (unsigned int n)

Additional Inherited Members

5.285.1 Detailed Description

Bartlett window.

5.285.2 Constructor & Destructor Documentation

5.285.2.1 MHAWindow::bartlett_t::bartlett_t (unsigned int n) [inline]

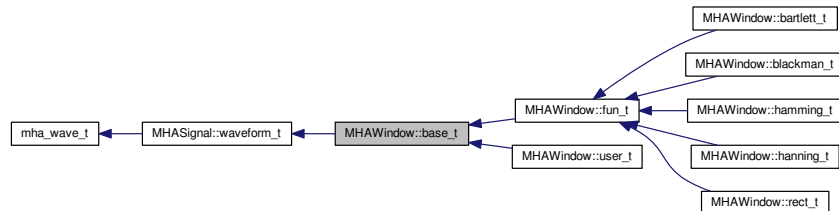
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.286 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 beginning.
- void **ramp_end** (**mha_wave_t** &) const
Apply a ramp at the end.

Additional Inherited Members

5.286.1 Detailed Description

Common base for window types.

5.286.2 Constructor & Destructor Documentation

5.286.2.1 MHAWindow::base_t::base_t (unsigned int len)

Constructor.

Parameters

| | |
|------------|---------------------------|
| <i>len</i> | Window length in samples. |
|------------|---------------------------|

5.286.2.2 MHAWindow::base_t::base_t (
const MHAWindow::base_t & src)

Copy constructor.

Parameters

| | |
|------------|---------------------|
| <i>src</i> | Source to be copied |
|------------|---------------------|

5.286.3 Member Function Documentation

5.286.3.1 void MHAWindow::base_t::operator() (
mha_wave_t & s) const

Apply window to waveform segment (reference)

5.286.3.2 void MHAWindow::base_t::operator() (
mha_wave_t * s) const

Apply window to waveform segment (pointer)

5.286.3.3 void MHAWindow::base_t::ramp_begin (
mha_wave_t & s) const

Apply a ramp at the beginning.

5.286.3.4 void MHAWindow::base_t::ramp_end (
mha_wave_t & s) const

Apply a ramp at the end.

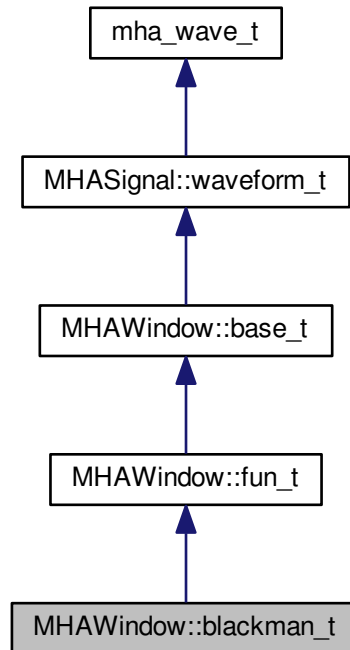
The documentation for this class was generated from the following files:

- mha_windowparser.h
- mha_windowparser.cpp

5.287 MHAWindow::blackman_t Class Reference

Blackman window.

Inheritance diagram for MHAWindow::blackman_t:



Public Member Functions

- **blackman_t** (unsigned int n)

Additional Inherited Members

5.287.1 Detailed Description

Blackman window.

5.287.2 Constructor & Destructor Documentation

5.287.2.1 MHAWindow::blackman_t::blackman_t (unsigned int *n*) [inline]

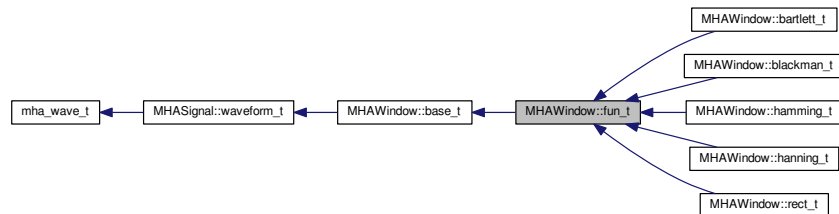
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.288 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

5.288.1 Detailed Description

Generic window based on a generator function.

The generator function should return a valid window function in the interval [-1,1].

5.288.2 Constructor & Destructor Documentation

5.288.2.1 MHAWindow::fun_t::fun_t (
 unsigned int n,
 float(*) (float) fun,
 float xmin = -1,
 float xmax = 1,
 bool min_included = true,
 bool max_included = false)

Constructor.

Parameters

| | |
|---------------------|--|
| <i>n</i> | Window length |
| <i>fun</i> | Generator function, i.e. MHAWindow::hanning() (p. 122) |
| <i>xmin</i> | Start value of window, i.e. -1 for full window or 0 for fade-out ramp. |
| <i>xmax</i> | Last value of window, i.e. 1 for full window |
| <i>min_included</i> | Flag if minimum value is included |
| <i>max_included</i> | Flag if maximum value is included |

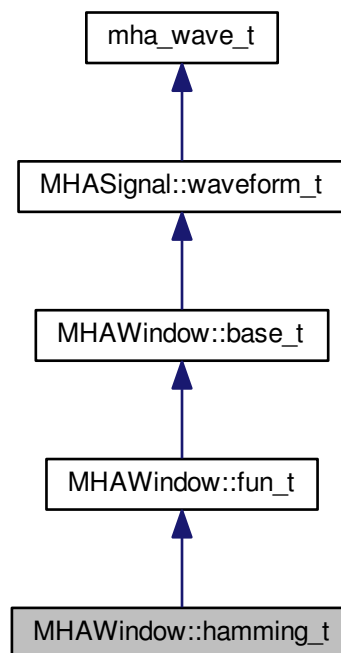
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.289 MHAWindow::hamming_t Class Reference

Hamming window.

Inheritance diagram for MHAWindow::hamming_t:



Public Member Functions

- **hamming_t** (unsigned int *n*)

Additional Inherited Members

5.289.1 Detailed Description

Hamming window.

5.289.2 Constructor & Destructor Documentation

5.289.2.1 MHAWindow::hamming_t::hamming_t(unsigned int *n*) [inline]

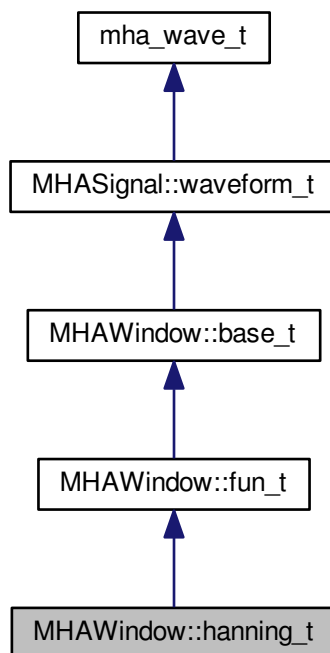
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.290 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

5.290.1 Detailed Description

von-Hann window

5.290.2 Constructor & Destructor Documentation

5.290.2.1 MHAWindow::hanning_t::hanning_t (unsigned int *n*) [inline]

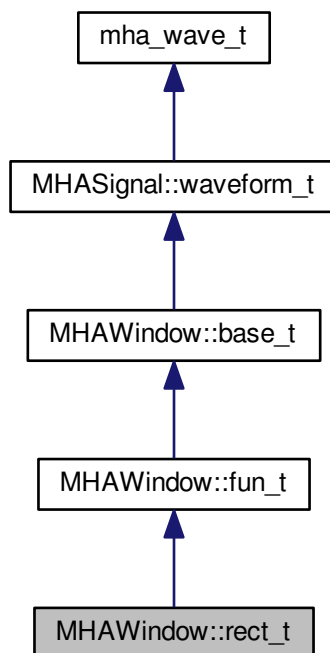
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.291 MHAWindow::rect_t Class Reference

Rectangular window.

Inheritance diagram for MHAWindow::rect_t:



Public Member Functions

- **rect_t** (unsigned int n)

Additional Inherited Members

5.291.1 Detailed Description

Rectangular window.

5.291.2 Constructor & Destructor Documentation

5.291.2.1 MHAWindow::rect_t::rect_t (
 unsigned int *n*) [inline]

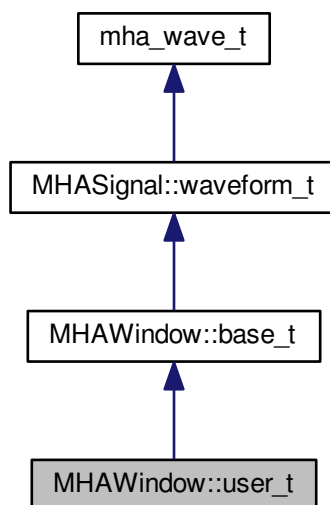
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

5.292 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

5.292.1 Detailed Description

User defined window.

5.292.2 Constructor & Destructor Documentation

5.292.2.1 MHAWindow::user_t::user_t (const std::vector< **mha_real_t** > & *wnd*)

Constructor.

Parameters

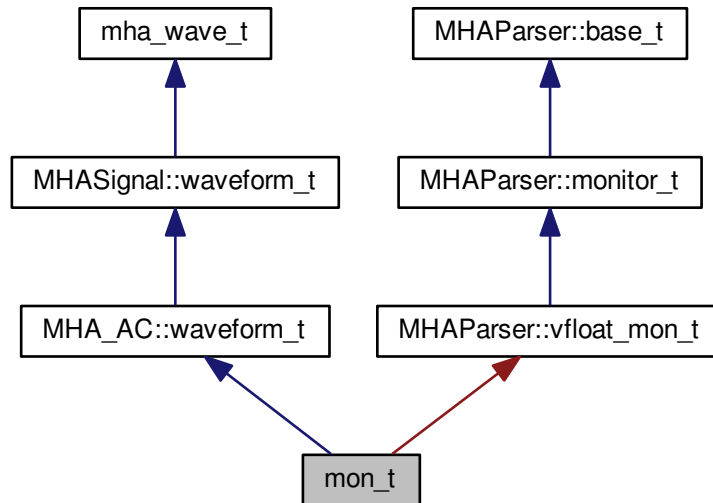
| | |
|------------|---------------------|
| <i>wnd</i> | User defined window |
|------------|---------------------|

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

- **mha_windowparser.h**
- **mha_windowparser.cpp**

5.293 mon_t Class Reference

Inheritance diagram for mon_t:



Public Member Functions

- **mon_t** (unsigned int *nch*, std::string *name*, algo_comm_t *ac*, std::string *base*, MHA↔Parser::parser_t &*p*, std::string *help*)
- void **store** ()

Additional Inherited Members

5.293.1 Constructor & Destructor Documentation

5.293.1.1 mon_t::mon_t (
 unsigned int *nch*,
 std::string *name*,
 algo_comm_t *ac*,
 std::string *base*,
 MHAParser::parser_t & *p*,
 std::string *help*)

5.293.2 Member Function Documentation

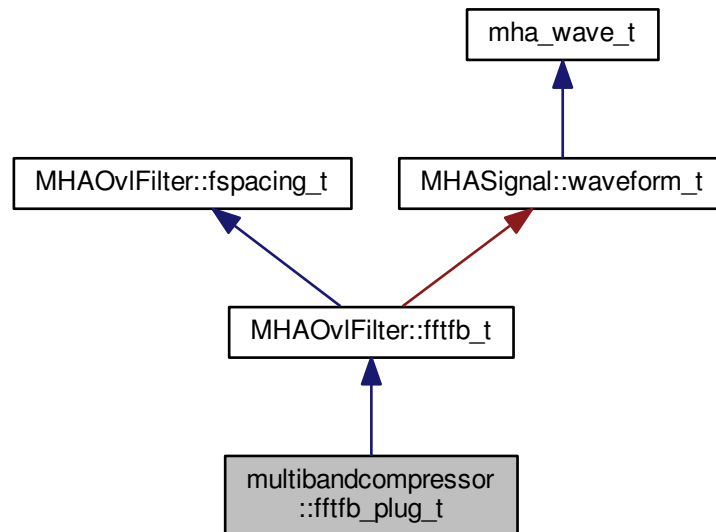
5.293.2.1 void mon_t::store ()

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

- **rmslevel.cpp**

5.294 multibandcompressor::fftfb_plug_t Class Reference

Inheritance diagram for multibandcompressor::fftfb_plug_t:



Public Member Functions

- **fftfb_plug_t** (**MHAOvfFilter::fftfb_vars_t** &, const **mhaconfig_t** &cfg, **algo_comm_t** ac, std::string alg)
- void **insert** ()

Private Attributes

- **MHA_AC::waveform_t cfv**
vector of nominal center frequencies / Hz
- **MHA_AC::waveform_t efv**
vector of edge frequencies / Hz
- **MHA_AC::waveform_t bwv**
vector of band-weights (sum of squared fft-bin-weights)/num_frames

Additional Inherited Members

5.294.1 Constructor & Destructor Documentation

5.294.1.1 multibandcompressor::fftfb_plug_t::fftfb_plug_t (
MHAOvIFilter::fftfb_vars_t & vars,
const mhaconfig_t & cfg,
algo_comm_t ac,
std::string alg)

5.294.2 Member Function Documentation

5.294.2.1 void multibandcompressor::fftfb_plug_t::insert ()

5.294.3 Member Data Documentation

5.294.3.1 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::cfv [private]

vector of nominal center frequencies / Hz

5.294.3.2 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::efv [private]

vector of edge frequencies / Hz

5.294.3.3 MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::bwv [private]

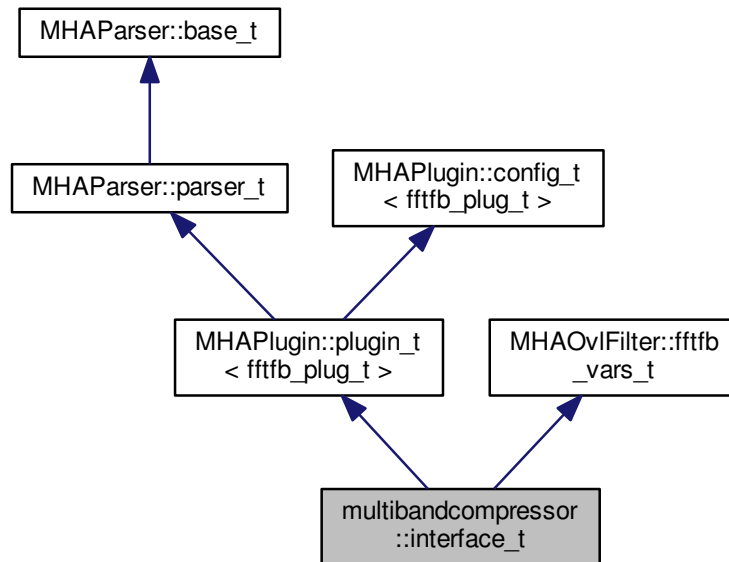
vector of band-weights (sum of squared fft-bin-weights)/num_frames

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

- multibandcompressor.cpp

5.295 multibandcompressor::interface_t Class Reference

Inheritance diagram for multibandcompressor::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- int **num_channels**
- **DynComp::dc_afterburn_t** burn
- **MHAEvents::patchbay_t** < **interface_t** > patchbay
- std::string **algo**
- **MHAParser::mhapluginloader_t** plug
- **plugin_signals_t** * plug_sigs

Additional Inherited Members

5.295.1 Constructor & Destructor Documentation

5.295.1.1 multibandcompressor::interface_t::interface_t (
 const algo_comm_t & ac,
 const std::string & th,
 const std::string & al)

Default values are set and MHA configuration variables registered into the parser.

Parameters

| | |
|-------------|--------------------------------|
| <i>ac</i> ↔ | algorithm communication handle |
| <i>th</i> | chain name |
| <i>al</i> | algorithm name |

5.295.2 Member Function Documentation

5.295.2.1 void multibandcompressor::interface_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t**< **fftfb_plug_t** > (p. [661](#)).

5.295.2.2 void multibandcompressor::interface_t::release (
 void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **fftfb_plug_t** > (p. [661](#)).

5.295.2.3 mha_spec_t * multibandcompressor::interface_t::process (
 mha_spec_t * s)

5.295.2.4 void multibandcompressor::interface_t::update_cfg () [private]

5.295.3 Member Data Documentation

5.295.3.1 int multibandcompressor::interface_t::num_channels [private]

5.295.3.2 DynComp::dc_afterburn_t multibandcompressor::interface_t::burn [private]

5.295.3.3 MHAEvents::patchbay_t<interface_t> multibandcompressor::interface_t::patchbay
 [private]

5.295.3.4 `std::string multibandcompressor::interface_t::algo` [private]

5.295.3.5 `MHAParser::mhapluginloader_t multibandcompressor::interface_t::plug` [private]

5.295.3.6 `plugin_signals_t*` `multibandcompressor::interface_t::plug_sigs` [private]

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

- `multibandcompressor.cpp`

5.296 `multibandcompressor::plugin_signals_t` Class Reference

Public Member Functions

- `plugin_signals_t` (unsigned int **channels**, unsigned int **bands**)
- void `update_levels` (`MHAOvIFilter::fftfb_t *`, `mha_spec_t *s_in`)
- void `apply_gains` (`MHAOvIFilter::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`

5.296.1 Constructor & Destructor Documentation

5.296.1.1 `multibandcompressor::plugin_signals_t::plugin_signals_t (`
 unsigned int *channels*,
 unsigned int *bands*)

5.296.2 Member Function Documentation

5.296.2.1 void `multibandcompressor::plugin_signals_t::update_levels (`
 `MHAOvIFilter::fftfb_t * pFb`,
 `mha_spec_t * s_in`)

5.296.2.2 void multibandcompressor::plugin_signals_t::apply_gains (
MHAOvfFilter::fftfb_t * *pFb*,
DynComp::dc_afterburn_t & *burn*,
mha_spec_t * *s_out*)

5.296.3 Member Data Documentation

5.296.3.1 MHASignal::waveform_t multibandcompressor::plugin_signals_t::plug_level
[private]

5.296.3.2 MHASignal::waveform_t multibandcompressor::plugin_signals_t::gain [private]

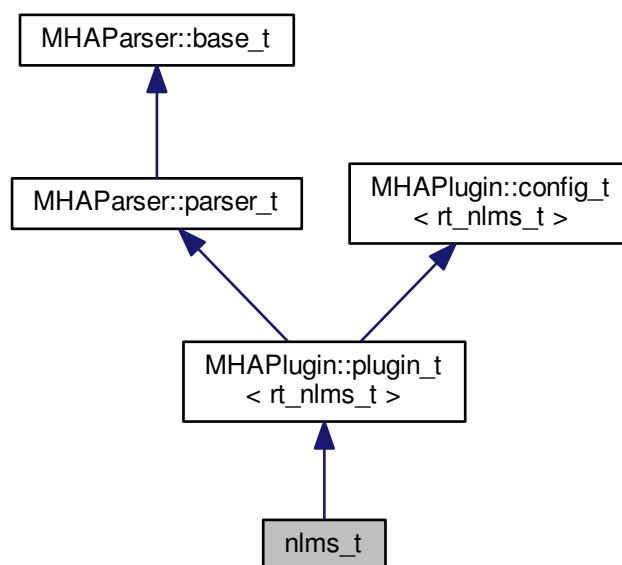
5.296.3.3 mha_wave_t* multibandcompressor::plugin_signals_t::plug_output

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

- multibandcompressor.cpp

5.297 nlms_t Class Reference

Inheritance diagram for nlms_t:



Public Member Functions

- **nlms_t** (**algo_comm_t**, const char *, const char *)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAParser::float_t** rho
- **MHAParser::float_t** c
- **MHAParser::int_t** ntaps
- **MHAParser::string_t** name_u
- **MHAParser::string_t** name_d
- **MHAParser::kw_t** normtype
- **MHAParser::kw_t** estimtype
- **MHAParser::float_t** lambda_smoothing_power
- **MHAParser::string_t** name_e
- **MHAParser::string_t** name_f
- **MHAParser::int_t** n_no_update
- std::string algo
- **MHAEvents::patchbay_t** < **nlms_t** > patchbay

Additional Inherited Members

5.297.1 Constructor & Destructor Documentation

5.297.1.1 **nlms_t::nlms_t** (
 algo_comm_t ac,
 const char *,
 const char * *ialg*)

5.297.2 Member Function Documentation

5.297.2.1 void **nlms_t::prepare** (
 mhaconfig_t & cf) [virtual]

Implements **MHAPLugin::plugin_t** < **rt_nlms_t** > (p. 661).

5.297.2.2 void nlms_t::release (
void) [virtual]

Reimplemented from **MHAPLugin::plugin_t< rt_nlms_t >** (p. 661).

5.297.2.3 mha_wave_t * nlms_t::process (
mha_wave_t * s)

5.297.2.4 void nlms_t::update () [private]

5.297.3 Member Data Documentation

5.297.3.1 MHAParser::float_t nlms_t::rho [private]

5.297.3.2 MHAParser::float_t nlms_t::c [private]

5.297.3.3 MHAParser::int_t nlms_t::ntaps [private]

5.297.3.4 MHAParser::string_t nlms_t::name_u [private]

5.297.3.5 MHAParser::string_t nlms_t::name_d [private]

5.297.3.6 MHAParser::kw_t nlms_t::normtype [private]

5.297.3.7 MHAParser::kw_t nlms_t::estimtype [private]

5.297.3.8 MHAParser::float_t nlms_t::lambda_smoothing_power [private]

5.297.3.9 MHAParser::string_t nlms_t::name_e [private]

5.297.3.10 MHAParser::string_t nlms_t::name_f [private]

5.297.3.11 MHAParser::int_t nlms_t::n_no_update [private]

5.297.3.12 std::string nlms_t::algo [private]

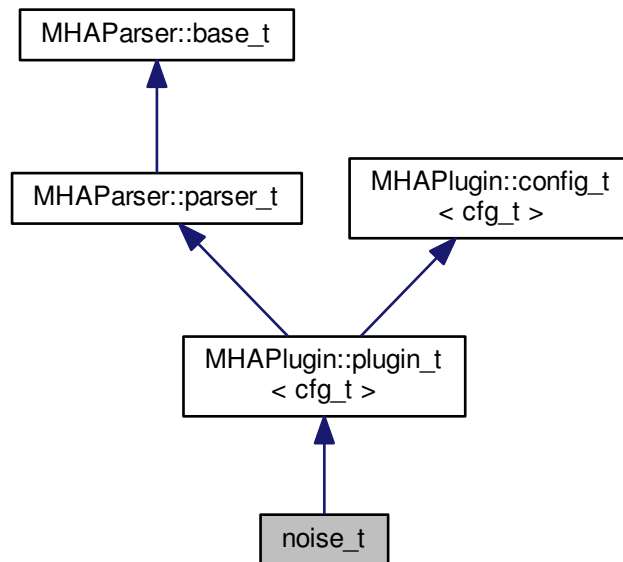
5.297.3.13 MHAEvents::patchbay_t<nlms_t> nlms_t::patchbay [private]

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

- nlms_wave.cpp

5.298 noise_t Class Reference

Inheritance diagram for noise_t:



Public Member Functions

- **noise_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t** lev
- **MHAParser::kw_t** mode
- **MHAParser::float_t** frozennoise_length
- **MHAEvents::patchbay_t** < **noise_t** > patchbay

Additional Inherited Members

5.298.1 Constructor & Destructor Documentation

5.298.1.1 noise_t::noise_t (
 const algo_comm_t & iac,
 const std::string & ,
 const std::string &)

5.298.2 Member Function Documentation

5.298.2.1 mha_wave_t * noise_t::process (
 mha_wave_t * s)

5.298.2.2 mha_spec_t * noise_t::process (
 mha_spec_t * s)

5.298.2.3 void noise_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAParser::plugin_t**< **cfg_t** > (p. 661).

5.298.2.4 void noise_t::update_cfg ()

5.298.3 Member Data Documentation

5.298.3.1 MHAParser::float_t noise_t::lev [private]

5.298.3.2 MHAParser::kw_t noise_t::mode [private]

5.298.3.3 MHAParser::float_t noise_t::frozennoise_length [private]

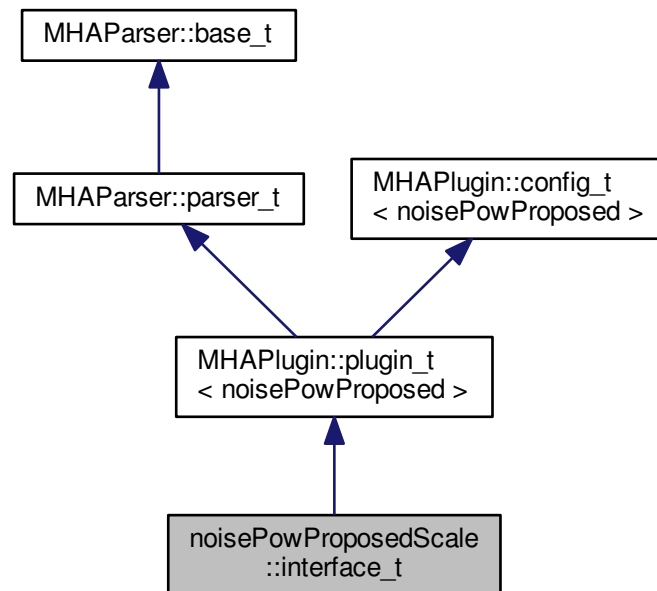
5.298.3.4 MHAEvents::patchbay_t<noise_t> noise_t::patchbay [private]

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

- noise.cpp

5.299 noisePowProposedScale::interface_t Class Reference

Inheritance diagram for noisePowProposedScale::interface_t:



Public Member Functions

- **interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t** **alphaPH1mean**
- **MHAParser::float_t** **alphaPSD**
- **MHAParser::float_t** **q**
- **MHAParser::float_t** **xiOptDb**
- std::string **name**
- **MHAEvents::patchbay_t** < **interface_t** > **patchbay**

Additional Inherited Members

5.299.1 Constructor & Destructor Documentation

5.299.1.1 `noisePowProposedScale::interface_t::interface_t (`
`const algo_comm_t & iac,`
`const std::string & ,`
`const std::string & iname)`

5.299.2 Member Function Documentation

5.299.2.1 `mha_spec_t * noisePowProposedScale::interface_t::process (`
`mha_spec_t * s)`

5.299.2.2 `void noisePowProposedScale::interface_t::prepare (`
`mhaconfig_t & cf) [virtual]`

Implements **MHAPLugin::plugin_t< noisePowProposed >** (p. 661).

5.299.2.3 `void noisePowProposedScale::interface_t::update_cfg (`
`void) [private]`

5.299.3 Member Data Documentation

5.299.3.1 `MHAParser::float_t noisePowProposedScale::interface_t::alphaPH1mean [private]`

5.299.3.2 `MHAParser::float_t noisePowProposedScale::interface_t::alphaPSD [private]`

5.299.3.3 `MHAParser::float_t noisePowProposedScale::interface_t::q [private]`

5.299.3.4 `MHAParser::float_t noisePowProposedScale::interface_t::xiOptDb [private]`

5.299.3.5 `std::string noisePowProposedScale::interface_t::name [private]`

5.299.3.6 `MHAEvents::patchbay_t<interface_t> noisePowProposedScale::interface_t::patchbay`
`[private]`

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

- **noisePowProposedScale.cpp**

5.300 noisePowProposedScale::noisePowProposed Class Reference

Public Member Functions

- **noisePowProposed** (const **mhaconfig_t** &cf, **algo_comm_t** ac, const std::string &name, float alphaPH1mean, float alphaPSD, float q, float xiOptDb)
- void **process** (**mha_spec_t** *noisyDftFrame)
- void **insert** ()

Private Attributes

- **MHASignal::waveform_t** noisyPer
- **MHASignal::waveform_t** PH1mean
- **MHA_AC::waveform_t** noisePow
- **MHA_AC::waveform_t** inputPow
- **MHA_AC::waveform_t** snrPost1Debug
- **MHA_AC::waveform_t** GLRDebug
- **MHA_AC::waveform_t** PH1Debug
- **MHA_AC::waveform_t** estimateDebug
- **MHA_AC::spectrum_t** inputSpec
- float **alphaPH1mean_**
- float **alphaPSD_**
- float **priorFact**
- float **xiOpt**
- float **logGLRFact**
- float **GLRexp**
- int **frameno**

5.300.1 Constructor & Destructor Documentation

5.300.1.1 noisePowProposedScale::noisePowProposed::noisePowProposed (const **mhaconfig_t** & cf, **algo_comm_t** ac, const std::string & name, float *alphaPH1mean*, float *alphaPSD*, float *q*, float *xiOptDb*)

5.300.2 Member Function Documentation

5.300.2.1 void noisePowProposedScale::noisePowProposed::process (**mha_spec_t** * *noisyDftFrame*)

5.300.2.2 void noisePowProposedScale::noisePowProposed::insert () [inline]

5.300.3 Member Data Documentation

5.300.3.1 **MHASignal::waveform_t** noisePowProposedScale::noisePowProposed::noisyPer
[private]

5.300.3.2 **MHASignal::waveform_t** noisePowProposedScale::noisePowProposed::PH1mean
[private]

5.300.3.3 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::noisePow
[private]

5.300.3.4 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::inputPow
[private]

5.300.3.5 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::snrPost1Debug
[private]

5.300.3.6 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::GLRDebug
[private]

5.300.3.7 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::PH1Debug
[private]

5.300.3.8 **MHA_AC::waveform_t** noisePowProposedScale::noisePowProposed::estimateDebug
[private]

5.300.3.9 **MHA_AC::spectrum_t** noisePowProposedScale::noisePowProposed::inputSpec
[private]

5.300.3.10 float noisePowProposedScale::noisePowProposed::alphaPH1mean_ [private]

5.300.3.11 float noisePowProposedScale::noisePowProposed::alphaPSD_ [private]

5.300.3.12 float noisePowProposedScale::noisePowProposed::priorFact [private]

5.300.3.13 float noisePowProposedScale::noisePowProposed::xiOpt [private]

5.300.3.14 float noisePowProposedScale::noisePowProposed::logGLRFact [private]

5.300.3.15 float noisePowProposedScale::noisePowProposed::GLRexp [private]

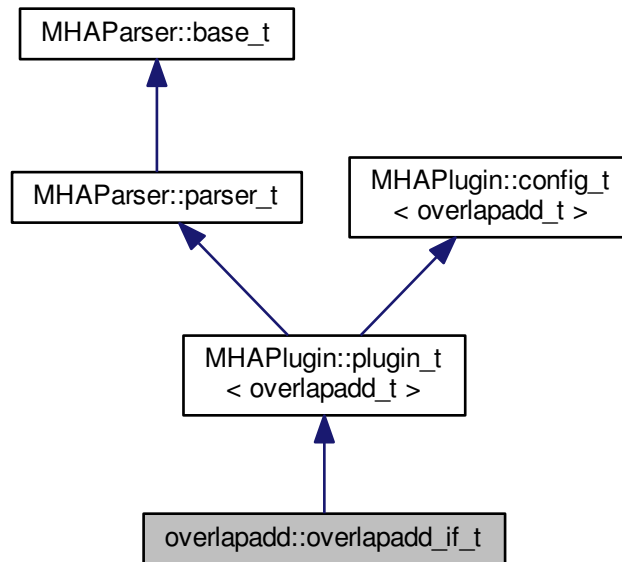
5.300.3.16 int noisePowProposedScale::noisePowProposed::frameno [private]

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

- **noisePowProposedScale.cpp**

5.301 overlapadd::overlapadd_if_t Class Reference

Inheritance diagram for overlapadd::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **~overlapadd_if_t** ()
- void **prepare** (mhaconfig_t &)
- void **release** ()
- **mha_wave_t * process** (mha_wave_t *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t< overlapadd_if_t > patchbay**
- **MHAParser::int_t nfft**
FFT length to be used, zero-padding is FFT length-wndlength.
- **MHAParser::int_t nwnd**
Window length to be used (overlap is 1-fragsize/wndlength)

- **MHAParser::float_t wndpos**
Relative position of zero padding (0 end, 0.5 center, 1 start)
- **MHAParser::window_t window**
- **MHAParser::float_t wndexp**
- **MHAParser::window_t zerowindow**
- **MHAParser::mhapluginloader_t plugloader**
- **MHAParser::float_mon_t prescale**
- **MHAParser::float_mon_t postscale**
- **std::string algo**
- **mhaconfig_t cf_in**
- **mhaconfig_t cf_out**

Additional Inherited Members

5.301.1 Constructor & Destructor Documentation

5.301.1.1 `overlapadd::overlapadd_if_t::overlapadd_if_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string & ialg)`

5.301.1.2 `overlapadd::overlapadd_if_t::~~overlapadd_if_t ()`

5.301.2 Member Function Documentation

5.301.2.1 `void overlapadd::overlapadd_if_t::prepare (`
 `mhaconfig_t & t) [virtual]`

Implements **MHAPLugin::plugin_t< overlapadd_t >** (p. 661).

5.301.2.2 `void overlapadd::overlapadd_if_t::release (`
 `void) [virtual]`

Reimplemented from **MHAPLugin::plugin_t< overlapadd_t >** (p. 661).

5.301.2.3 `mha_wave_t * overlapadd::overlapadd_if_t::process (`
 `mha_wave_t * wave_in)`

5.301.2.4 `void overlapadd::overlapadd_if_t::update () [private]`

5.301.3 Member Data Documentation

5.301.3.1 `MHAEvents::patchbay_t<overlapadd_if_t> overlapadd::overlapadd_if_t::patchbay`
 `[private]`

5.301.3.2 `MHAParser::int_t overlapadd::overlapadd_if_t::nfft [private]`

FFT length to be used, zero-padding is FFT length-wndlength.

5.301.3.3 **MHAParser::int_t overlapadd::overlapadd_if_t::nwnd** [private]

Window length to be used (overlap is 1-fragsize/wndlength)

5.301.3.4 **MHAParser::float_t overlapadd::overlapadd_if_t::wndpos** [private]

Relative position of zero padding (0 end, 0.5 center, 1 start)

5.301.3.5 **MHAParser::window_t overlapadd::overlapadd_if_t::window** [private]

5.301.3.6 **MHAParser::float_t overlapadd::overlapadd_if_t::wndexp** [private]

5.301.3.7 **MHAParser::window_t overlapadd::overlapadd_if_t::zerowindow** [private]

5.301.3.8 **MHAParser::mhapluginloader_t overlapadd::overlapadd_if_t::plugloader** [private]

5.301.3.9 **MHAParser::float_mon_t overlapadd::overlapadd_if_t::prescale** [private]

5.301.3.10 **MHAParser::float_mon_t overlapadd::overlapadd_if_t::postscale** [private]

5.301.3.11 **std::string overlapadd::overlapadd_if_t::algo** [private]

5.301.3.12 **mhaconfig_t overlapadd::overlapadd_if_t::cf_in** [private]

5.301.3.13 **mhaconfig_t overlapadd::overlapadd_if_t::cf_out** [private]

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

- **overlapadd.cpp**

5.302 overlapadd::overlapadd_t Class Reference

Public Member Functions

- **overlapadd_t** (**mhaconfig_t** spar_in, **mhaconfig_t** spar_out, float wexp, float wndpos, const **MHAParser::window_t** &window, const **MHAParser::window_t** &zerowindow, float &prescale_fac, float &postscale_fac)
- **~overlapadd_t** ()
- **mha_spec_t * ola1** (**mha_wave_t** *)
- **mha_wave_t * ola2** (**mha_spec_t** *)

Private Attributes

- **mha_fft_t** *fft*
- **MHAWindow::base_t** *prewnd*
- **MHAWindow::base_t** *postwnd*
- **MHASignal::waveform_t** *wave_in1*
- **MHASignal::waveform_t** *wave_out1*
- **MHASignal::spectrum_t** *spec_in*
- **MHASignal::waveform_t** *calc_out*
- **MHASignal::waveform_t** *out_buf*
- **MHASignal::waveform_t** *write_buf*
- unsigned int **n_zero**
- unsigned int **n_pad1**
- unsigned int **n_pad2**

5.302.1 Constructor & Destructor Documentation

5.302.1.1 **overlapadd::overlapadd_t::overlapadd_t** (
 mhaconfig_t *spar_in*,
 mhaconfig_t *spar_out*,
 float *wexp*,
 float *wndpos*,
 const **MHAParser::window_t** & *window*,
 const **MHAParser::window_t** & *zerowindow*,
 float & *prescale_fac*,
 float & *postscale_fac*)

5.302.1.2 **overlapadd::overlapadd_t::~~overlapadd_t** ()

5.302.2 Member Function Documentation

5.302.2.1 **mha_spec_t** * **overlapadd::overlapadd_t::ola1** (
 mha_wave_t * *s*)

5.302.2.2 **mha_wave_t** * **overlapadd::overlapadd_t::ola2** (
 mha_spec_t * *s*)

5.302.3 Member Data Documentation

5.302.3.1 **mha_fft_t** **overlapadd::overlapadd_t::fft** [private]

5.302.3.2 **MHAWindow::base_t** **overlapadd::overlapadd_t::prewnd** [private]

5.302.3.3 **MHAWindow::base_t** **overlapadd::overlapadd_t::postwnd** [private]

5.302.3.4 **MHASignal::waveform_t overlapadd::overlapadd_t::wave_in1** [private]

5.302.3.5 **MHASignal::waveform_t overlapadd::overlapadd_t::wave_out1** [private]

5.302.3.6 **MHASignal::spectrum_t overlapadd::overlapadd_t::spec_in** [private]

5.302.3.7 **MHASignal::waveform_t overlapadd::overlapadd_t::calc_out** [private]

5.302.3.8 **MHASignal::waveform_t overlapadd::overlapadd_t::out_buf** [private]

5.302.3.9 **MHASignal::waveform_t overlapadd::overlapadd_t::write_buf** [private]

5.302.3.10 **unsigned int overlapadd::overlapadd_t::n_zero** [private]

5.302.3.11 **unsigned int overlapadd::overlapadd_t::n_pad1** [private]

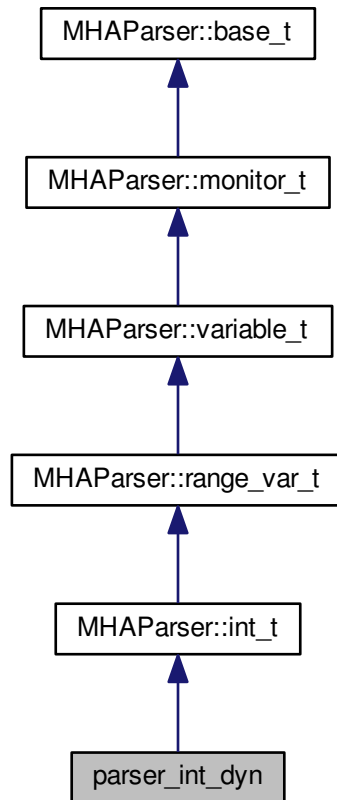
5.302.3.12 **unsigned int overlapadd::overlapadd_t::n_pad2** [private]

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

- **overlapadd.cpp**

5.303 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

5.303.1 Constructor & Destructor Documentation

5.303.1.1 `parser_int_dyn::parser_int_dyn (`
 const std::string & *help_text*,
 const std::string & *initial_value*,
 const std::string & *range*) `[inline]`

5.303.2 Member Function Documentation

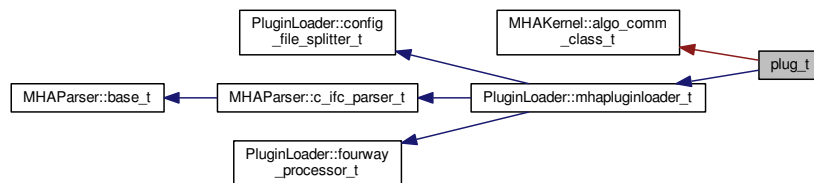
5.303.2.1 void parser_int_dyn::set_max_angle_ind (unsigned int *max_ind*) [inline]

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

- **steerbf.h**

5.304 plug_t Class Reference

Inheritance diagram for plug_t:



Public Member Functions

- **plug_t** (const std::string &**libname**, const std::string &chain, const std::string &algo)
- **~plug_t** () throw ()
- **MHAProc_wave2wave_t** get_process_wave ()
- **MHAProc_wave2spec_t** get_process_spec ()
- void * **get_handle** ()
- **algo_comm_t** get_ac ()

Additional Inherited Members

5.304.1 Constructor & Destructor Documentation

5.304.1.1 plug_t::plug_t (const std::string & *libname*, const std::string & *chain*, const std::string & *algo*)

5.304.1.2 plug_t::~~plug_t () throw) [inline]

5.304.2 Member Function Documentation

5.304.2.1 MHAProc_wave2wave_t plug_t::get_process_wave ()

5.304.2.2 MHAProc_wave2spec_t plug_t::get_process_spec ()

5.304.2.3 void * plug_t::get_handle ()

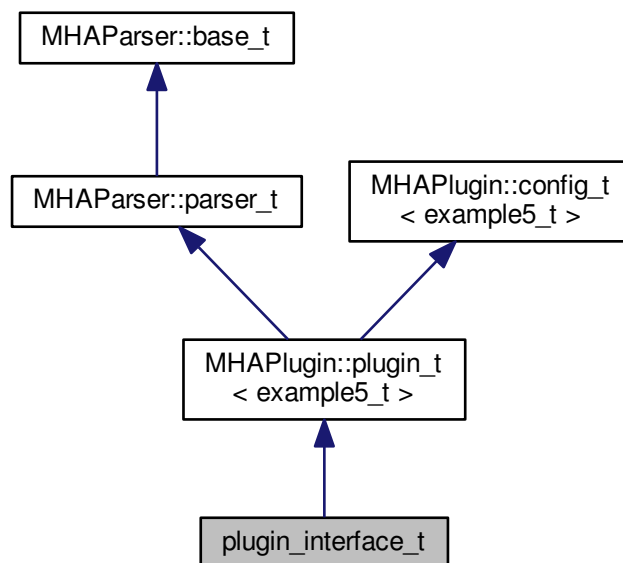
5.304.2.4 algo_comm_t plug_t::get_ac () [inline]

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

- **analysispath.cpp**

5.305 plugin_interface_t Class Reference

Inheritance diagram for plugin_interface_t:



Public Member Functions

- **plugin_interface_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::int_t** **scale_ch**
- **MHAParser::float_t** **factor**
- **MHAEvents::patchbay_t**< **plugin_interface_t** > **patchbay**

Additional Inherited Members

5.305.1 Constructor & Destructor Documentation

5.305.1.1 **plugin_interface_t::plugin_interface_t** (
 const **algo_comm_t** & *iac*,
 const std::string & ,
 const std::string &)

5.305.2 Member Function Documentation

5.305.2.1 **mha_spec_t** * **plugin_interface_t::process** (
 mha_spec_t * *spec*)

5.305.2.2 void **plugin_interface_t::prepare** (
 mhaconfig_t & *tfcfg*) [virtual]

Implements **MHAPLugin::plugin_t**< **example5_t** > (p. 661).

5.305.2.3 void **plugin_interface_t::update_cfg** () [private]

5.305.3 Member Data Documentation

5.305.3.1 **MHAParser::int_t** **plugin_interface_t::scale_ch** [private]

5.305.3.2 **MHAParser::float_t** **plugin_interface_t::factor** [private]

5.305.3.3 **MHAEvents::patchbay_t**<**plugin_interface_t**> **plugin_interface_t::patchbay**
 [private]

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

- **example5.cpp**

5.306 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**

5.306.1 Constructor & Destructor Documentation

5.306.1.1 pluginbrowser_t::pluginbrowser_t ()

5.306.2 Member Function Documentation

5.306.2.1 void pluginbrowser_t::get_paths ()

5.306.2.2 plugindescription_t pluginbrowser_t::scan_plugin (
const std::string & *name*)

5.306.2.3 void pluginbrowser_t::add_plugins ()

5.306.2.4 void pluginbrowser_t::clear_plugins ()

5.306.2.5 void pluginbrowser_t::scan_plugins ()

5.306.2.6 void pluginbrowser_t::add_plugin (
const std::string & *name*)

5.306.2.7 std::list<plugindescription_t> pluginbrowser_t::get_plugins () const [inline]

5.306.3 Member Data Documentation

- 5.306.3.1 `std::string pluginbrowser_t::plugin_extension` [private]
- 5.306.3.2 `std::list<std::string> pluginbrowser_t::library_paths` [private]
- 5.306.3.3 `std::list<plugindescription_t> pluginbrowser_t::plugins` [private]
- 5.306.3.4 `std::map<std::string,pluginloader_t*> pluginbrowser_t::p` [private]

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

- **pluginbrowser.h**
- **pluginbrowser.cpp**

5.307 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**

5.307.1 Member Data Documentation

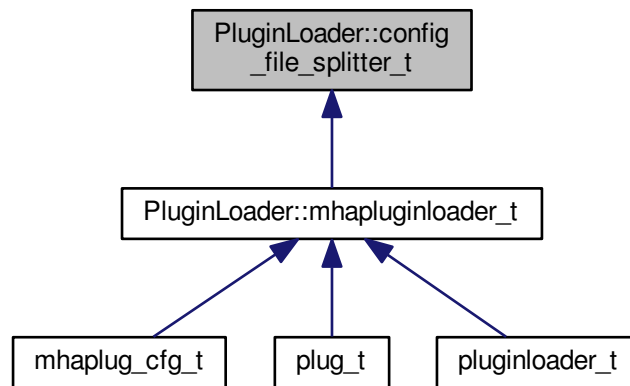
- 5.307.1.1 `std::string plugindescription_t::name`
- 5.307.1.2 `std::string plugindescription_t::fullname`
- 5.307.1.3 `std::string plugindescription_t::documentation`
- 5.307.1.4 `std::vector<std::string> plugindescription_t::categories`
- 5.307.1.5 `bool plugindescription_t::wave2wave`
- 5.307.1.6 `bool plugindescription_t::wave2spec`
- 5.307.1.7 `bool plugindescription_t::spec2wave`
- 5.307.1.8 `bool plugindescription_t::spec2spec`
- 5.307.1.9 `std::vector<std::string> plugindescription_t::query_cmds`
- 5.307.1.10 `std::map<std::string,std::string> plugindescription_t::queries`

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

- **pluginbrowser.h**

5.308 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_oriname** () const
- const std::string & **get_configfile** () const

Private Attributes

- std::string **libname**
- std::string **configname**
- std::string **oriname**
- std::string **configfile**

5.308.1 Constructor & Destructor Documentation

5.308.1.1 `PluginLoader::config_file_splitter_t::config_file_splitter_t (const std::string & name)`

5.308.2 Member Function Documentation

- 5.308.2.1 `const std::string& PluginLoader::config_file_splitter_t::get_configname () const` `[inline]`
- 5.308.2.2 `const std::string& PluginLoader::config_file_splitter_t::get_libname () const` `[inline]`
- 5.308.2.3 `const std::string& PluginLoader::config_file_splitter_t::get_oriname () const` `[inline]`
- 5.308.2.4 `const std::string& PluginLoader::config_file_splitter_t::get_configfile () const` `[inline]`

5.308.3 Member Data Documentation

- 5.308.3.1 `std::string PluginLoader::config_file_splitter_t::libname` `[private]`
- 5.308.3.2 `std::string PluginLoader::config_file_splitter_t::configname` `[private]`
- 5.308.3.3 `std::string PluginLoader::config_file_splitter_t::oriname` `[private]`
- 5.308.3.4 `std::string PluginLoader::config_file_splitter_t::configfile` `[private]`

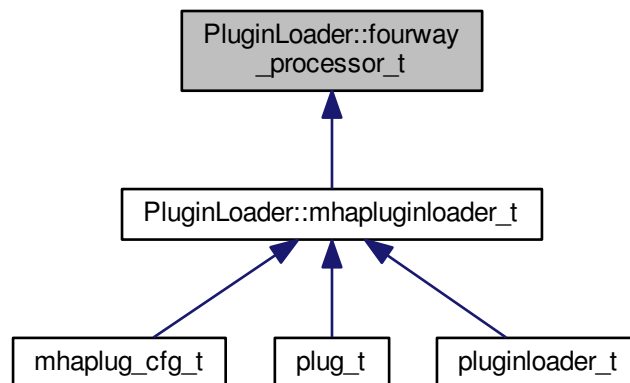
The documentation for this class was generated from the following files:

- **mhappluginloader.h**
- **mhappluginloader.cpp**

5.309 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. 805) are released here in **fourway_processor_t::release** (p. 805).*
- virtual std::string **parse** (const std::string &query)=0
Parser interface.
- virtual ~**fourway_processor_t** ()
Classes with virtual methods need virtual destructor.

5.309.1 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.

5.309.2 Constructor & Destructor Documentation

5.309.2.1 virtual PluginLoader::fourway_processor_t::~fourway_processor_t() [inline],
[virtual]

Classes with virtual methods need virtual destructor.

This destructor is empty.

5.309.3 Member Function Documentation

5.309.3.1 virtual void PluginLoader::fourway_processor_t::process (
 mha_wave_t *s_in,
 mha_wave_t **s_out) [pure virtual]

Pure waveform processing.

Parameters

| | |
|--------------|------------------------|
| <i>s_in</i> | input waveform signal |
| <i>s_out</i> | output waveform signal |

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

```
5.309.3.2  virtual void PluginLoader::fourway_processor_t::process (
            mha_spec_t * s_in,
            mha_spec_t ** s_out ) [pure virtual]
```

Pure spectrum processing.

Parameters

| | |
|--------------|------------------------|
| <i>s_in</i> | input spectrum signal |
| <i>s_out</i> | output spectrum signal |

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

```
5.309.3.3  virtual void PluginLoader::fourway_processor_t::process (
            mha_wave_t * s_in,
            mha_spec_t ** s_out ) [pure virtual]
```

Signal processing with domain transformation from waveform to spectrum.

Parameters

| | |
|--------------|------------------------|
| <i>s_in</i> | input waveform signal |
| <i>s_out</i> | output spectrum signal |

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

```
5.309.3.4  virtual void PluginLoader::fourway_processor_t::process (
            mha_spec_t * s_in,
            mha_wave_t ** s_out ) [pure virtual]
```

Signal processing with domain transformation from spectrum to waveform.

Parameters

| | |
|--------------|------------------------|
| <i>s_in</i> | input spectrum signal |
| <i>s_out</i> | output waveform signal |

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

5.309.3.5 `virtual void PluginLoader::fourway_processor_t::prepare (mhaconfig_t & settings) [pure virtual]`

Prepares the processor for signal processing.

Parameters

| | |
|-----------------|---|
| <i>settings</i> | domain and dimensions of the signal. The contents of settings may be modified by the prepare implementation. Upon calling fourway_processor_t::prepare (p. 805), settings reflects domain and dimensions of the input signal. When fourway_processor_t::prepare (p. 805) returns, settings reflects domain and dimensions of the output signal. |
|-----------------|---|

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

5.309.3.6 `virtual void PluginLoader::fourway_processor_t::release () [pure virtual]`

Resources allocated for signal processing in **fourway_processor_t::prepare** (p. 805) are released here in **fourway_processor_t::release** (p. 805).

Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

5.309.3.7 `virtual std::string PluginLoader::fourway_processor_t::parse (const std::string & query) [pure virtual]`

Parser interface.

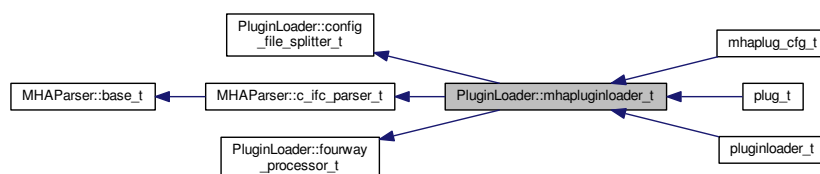
Implemented in **PluginLoader::mhapluginloader_t** (p. 808).

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

- **mhapluginloader.h**

5.310 PluginLoader::mhapluginloader_t Class Reference

Inheritance diagram for PluginLoader::mhapluginloader_t:



Public Member Functions

- `std::string parse` (const `std::string` &str)
- `mhapluginloader_t` (`algo_comm_t` iac, const `std::string` &libname, bool check_↵
version=true)
- Loads and initializes mha plugin and establishes interface.*
- `~mhapluginloader_t` () throw ()
- `bool has_process` (`mha_domain_t` in, `mha_domain_t` out) const
- `bool has_parser` () const
- `mha_domain_t input_domain` () const
- `mha_domain_t output_domain` () const
- `void prepare` (`mhaconfig_t` &)
- `void release` ()
- `void process` (`mha_wave_t` *, `mha_wave_t` **)
- `void process` (`mha_spec_t` *, `mha_spec_t` **)
- `void process` (`mha_wave_t` *, `mha_spec_t` **)
- `void process` (`mha_spec_t` *, `mha_wave_t` **)
- `std::string getfullname` () const
- `std::string get_documentation` () const
- `std::vector< std::string > get_categories` () const
- `bool is_prepared` () const

Protected Member Functions

- `void test_error` ()
- `void test_version` ()
- `void mha_test_struct_size` (unsigned int s)
- `void resolve_and_init` ()

Protected Attributes

- `int lib_err`
- `algo_comm_t ac`
- `dynamiclib_t lib_handle`
- `void * lib_data`
- `MHAGetVersion_t MHAGetVersion_cb`
- `MHAInit_t MHAInit_cb`
- `MHADestroy_t MHADestroy_cb`
- `MHAPrepere_t MHAPrepere_cb`
- `MHARelease_t MHARelease_cb`
- `MHAProc_wave2wave_t MHAProc_wave2wave_cb`
- `MHAProc_spec2spec_t MHAProc_spec2spec_cb`
- `MHAProc_wave2spec_t MHAProc_wave2spec_cb`
- `MHAProc_spec2wave_t MHAProc_spec2wave_cb`
- `MHASet_t MHASet_cb`
- `MHAStrError_t MHAStrError_cb`

- **mhaconfig_t cf_input**
- **mhaconfig_t cf_output**
- **std::string plugin_documentation**
- **std::vector< std::string > plugin_categories**
- **bool b_check_version**
- **bool b_is_prepared**

Additional Inherited Members

5.310.1 Constructor & Destructor Documentation

5.310.1.1 **PluginLoader::mhapluginloader_t::mhapluginloader_t (**
 algo_comm_t iac,
 const std::string & libname,
 bool check_version = true)

Loads and initializes mha plugin and establishes interface.

Parameters

| | |
|----------------|---|
| <i>iac</i> | AC space (algorithm communication variables) |
| <i>libname</i> | 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

| | |
|----------------------|--|
| <i>check_version</i> | 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. |
|----------------------|--|

5.310.1.2 **PluginLoader::mhapluginloader_t::~~mhapluginloader_t () throw)**

5.310.2 Member Function Documentation

5.310.2.1 `std::string PluginLoader::mhapluginloader_t::parse (`
`const std::string & str) [inline], [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 805).

5.310.2.2 `bool PluginLoader::mhapluginloader_t::has_process (`
`mha_domain_t in,`
`mha_domain_t out) const`

5.310.2.3 `bool PluginLoader::mhapluginloader_t::has_parser () const`

5.310.2.4 `mha_domain_t PluginLoader::mhapluginloader_t::input_domain () const`

5.310.2.5 `mha_domain_t PluginLoader::mhapluginloader_t::output_domain () const`

5.310.2.6 `void PluginLoader::mhapluginloader_t::prepare (`
`mhaconfig_t & tf) [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 805).

5.310.2.7 `void PluginLoader::mhapluginloader_t::release () [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 805).

5.310.2.8 `void PluginLoader::mhapluginloader_t::process (`
`mha_wave_t * s_in,`
`mha_wave_t ** s_out) [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 803).

5.310.2.9 `void PluginLoader::mhapluginloader_t::process (`
`mha_spec_t * s_in,`
`mha_spec_t ** s_out) [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 804).

5.310.2.10 `void PluginLoader::mhapluginloader_t::process (`
`mha_wave_t * s_in,`
`mha_spec_t ** s_out) [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 804).

5.310.2.11 `void PluginLoader::mhapluginloader_t::process (`
`mha_spec_t * s_in,`
`mha_wave_t ** s_out) [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 804).

- 5.310.2.12 `std::string PluginLoader::mhapluginloader_t::getfullname () const` [inline]
- 5.310.2.13 `std::string PluginLoader::mhapluginloader_t::get_documentation () const` [inline]
- 5.310.2.14 `std::vector<std::string> PluginLoader::mhapluginloader_t::get_categories () const`
[inline]
- 5.310.2.15 `bool PluginLoader::mhapluginloader_t::is_prepared () const` [inline]
- 5.310.2.16 `void PluginLoader::mhapluginloader_t::test_error ()` [protected]
- 5.310.2.17 `void PluginLoader::mhapluginloader_t::test_version ()` [protected]
- 5.310.2.18 `void PluginLoader::mhapluginloader_t::mha_test_struct_size (`
`unsigned int s)` [protected]
- 5.310.2.19 `void PluginLoader::mhapluginloader_t::resolve_and_init ()` [protected]

5.310.3 Member Data Documentation

- 5.310.3.1 `int PluginLoader::mhapluginloader_t::lib_err` [protected]
- 5.310.3.2 `algo_comm_t PluginLoader::mhapluginloader_t::ac` [protected]
- 5.310.3.3 `dynamiclib_t PluginLoader::mhapluginloader_t::lib_handle` [protected]
- 5.310.3.4 `void* PluginLoader::mhapluginloader_t::lib_data` [protected]
- 5.310.3.5 `MHAGetVersion_t PluginLoader::mhapluginloader_t::MHAGetVersion_cb`
[protected]
- 5.310.3.6 `MHAINit_t PluginLoader::mhapluginloader_t::MHAINit_cb` [protected]
- 5.310.3.7 `MHADestroy_t PluginLoader::mhapluginloader_t::MHADestroy_cb` [protected]
- 5.310.3.8 `MHAPrepate_t PluginLoader::mhapluginloader_t::MHAPrepate_cb` [protected]
- 5.310.3.9 `MHARelease_t PluginLoader::mhapluginloader_t::MHARelease_cb` [protected]
- 5.310.3.10 `MHAProc_wave2wave_t PluginLoader::mhapluginloader_t::MHAProc_wave2wave_cb`
[protected]
- 5.310.3.11 `MHAProc_spec2spec_t PluginLoader::mhapluginloader_t::MHAProc_spec2spec_cb`
[protected]

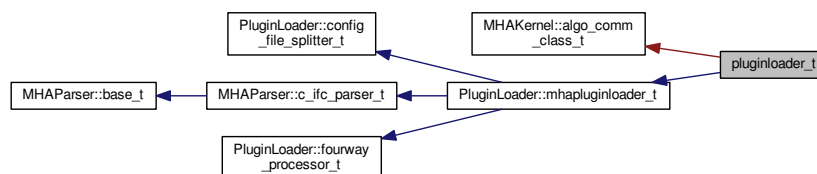
- 5.310.3.12 **MHAProc_wave2spec_t** PluginLoader::mhapluginloader_t::MHAProc_wave2spec_cb
[protected]
- 5.310.3.13 **MHAProc_spec2wave_t** PluginLoader::mhapluginloader_t::MHAProc_spec2wave_cb
[protected]
- 5.310.3.14 **MHASet_t** PluginLoader::mhapluginloader_t::MHASet_cb [protected]
- 5.310.3.15 **MHASTrError_t** PluginLoader::mhapluginloader_t::MHASTrError_cb [protected]
- 5.310.3.16 **mhaconfig_t** PluginLoader::mhapluginloader_t::cf_input [protected]
- 5.310.3.17 **mhaconfig_t** PluginLoader::mhapluginloader_t::cf_output [protected]
- 5.310.3.18 **std::string** PluginLoader::mhapluginloader_t::plugin_documentation [protected]
- 5.310.3.19 **std::vector<std::string>** PluginLoader::mhapluginloader_t::plugin_categories
[protected]
- 5.310.3.20 **bool** PluginLoader::mhapluginloader_t::b_check_version [protected]
- 5.310.3.21 **bool** PluginLoader::mhapluginloader_t::b_is_prepared [protected]

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

- **mhapluginloader.h**
- **mhapluginloader.cpp**

5.311 pluginloader_t Class Reference

Inheritance diagram for pluginloader_t:



Public Member Functions

- **pluginloader_t** (const std::string &name)
- **~pluginloader_t** () throw ()

Additional Inherited Members

5.311.1 Constructor & Destructor Documentation

5.311.1.1 pluginloader_t::pluginloader_t (
const std::string & name)

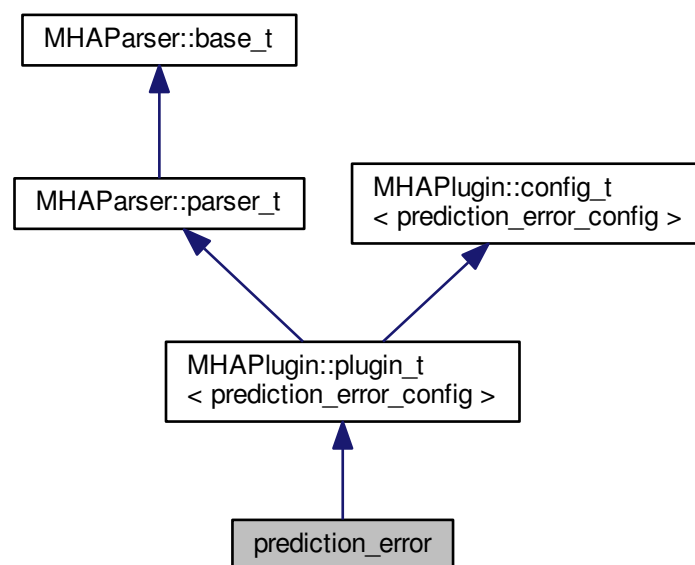
5.311.1.2 pluginloader_t::~~pluginloader_t () throw)

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

- pluginbrowser.h
- pluginbrowser.cpp

5.312 prediction_error Class Reference

Inheritance diagram for prediction_error:



Public Member Functions

- **prediction_error** (**algo_comm_t** &**ac**, const std::string &chain_name, const std::string &algo_name)
Constructs our plugin.
- **~prediction_error** ()
- **mha_wave_t * process** (**mha_wave_t ***)
Checks for the most recent configuration and defers processing to it.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::float_t rho**
- **MHAParser::float_t c**
- **MHAParser::int_t ntaps**
- **MHAParser::vfloat_t gains**
- **MHAParser::string_t name_e**
- **MHAParser::string_t name_f**
- **MHAParser::string_t name_lpc**
- **MHAParser::int_t lpc_order**
- **MHAParser::vint_t pred_err_delay**
- **MHAParser::vint_t delay_w**
- **MHAParser::vint_t delay_d**
- **MHAParser::int_t n_no_update**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t< prediction_error > patchbay**

Additional Inherited Members

5.312.1 Constructor & Destructor Documentation

- 5.312.1.1 **prediction_error::prediction_error** (
 algo_comm_t &*ac*,
 const std::string &*chain_name*,
 const std::string &*algo_name*)

Constructs our plugin.

5.312.1.2 prediction_error::~prediction_error ()

5.312.2 Member Function Documentation

5.312.2.1 mha_wave_t * prediction_error::process (mha_wave_t * signal)

Checks for the most recent configuration and defers processing to it.

5.312.2.2 void prediction_error::prepare (mhaconfig_t & signal_info) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate). |
|--------------------|---|

Implements **MHAParser::plugin_t< prediction_error_config >** (p. 661).

5.312.2.3 void prediction_error::release (void) [inline], [virtual]

Reimplemented from **MHAParser::plugin_t< prediction_error_config >** (p. 661).

5.312.2.4 void prediction_error::update_cfg () [private]

5.312.3 Member Data Documentation

5.312.3.1 MHAParser::float_t prediction_error::rho

5.312.3.2 MHAParser::float_t prediction_error::c

5.312.3.3 MHAParser::int_t prediction_error::ntaps

5.312.3.4 MHAParser::vfloat_t prediction_error::gains

5.312.3.5 MHAParser::string_t prediction_error::name_e

5.312.3.6 MHAParser::string_t prediction_error::name_f

5.312.3.7 **MHAParser::string_t prediction_error::name_lpc**

5.312.3.8 **MHAParser::int_t prediction_error::lpc_order**

5.312.3.9 **MHAParser::vint_t prediction_error::pred_err_delay**

5.312.3.10 **MHAParser::vint_t prediction_error::delay_w**

5.312.3.11 **MHAParser::vint_t prediction_error::delay_d**

5.312.3.12 **MHAParser::int_t prediction_error::n_no_update**

5.312.3.13 **MHAEvents::patchbay_t<prediction_error> prediction_error::patchbay**
[private]

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

- **prediction_error.h**
- **prediction_error.cpp**

5.313 prediction_error_config Class Reference

Public Member Functions

- **prediction_error_config (algo_comm_t &ac, const mhaconfig_t in_cfg, prediction_↔
_error *pred_err)**
- **~prediction_error_config ()**
- **mha_wave_t * process (mha_wave_t *s_Y, mha_real_t rho, mha_real_t c)**
- **void insert ()**

Private Attributes

- **algo_comm_t ac**
- unsigned int **ntaps**
- unsigned int **frames**
- unsigned int **channels**
- **MHA_AC::waveform_t s_E**
- **MHA_AC::waveform_t F**
- **MHASignal::waveform_t Pu**
Power of input signal delayline.
- std::string **name_d_**
- std::string **name_lpc_**
- int **n_no_update_**
- int **no_iter**

- int iter
- MHASignal::waveform_t v_G
- MHASignal::waveform_t s_U
- MHASignal::delay_t s_E_pred_err_delay
- MHASignal::delay_t s_W
- MHASignal::ringbuffer_t s_Wflt
- MHASignal::delay_t s_U_delay
- MHASignal::ringbuffer_t s_U_delayflt
- MHASignal::waveform_t F_Uflt
- MHASignal::delay_t s_Y_delay
- MHASignal::ringbuffer_t s_Y_delayflt
- MHASignal::ringbuffer_t UbufferPrew
- mha_wave_t s_LPC
- mha_wave_t UPrew
- mha_wave_t YPrew
- mha_wave_t EPrew
- mha_wave_t UPrewW
- mha_wave_t smpl
- mha_wave_t * s_Usmpl

5.313.1 Constructor & Destructor Documentation

5.313.1.1 prediction_error_config::prediction_error_config (
 algo_comm_t & ac,
 const mhaconfig_t in_cfg,
 prediction_error * pred_err)

5.313.1.2 prediction_error_config::~prediction_error_config ()

5.313.2 Member Function Documentation

5.313.2.1 mha_wave_t * prediction_error_config::process (
 mha_wave_t * s_Y,
 mha_real_t rho,
 mha_real_t c)

5.313.2.2 void prediction_error_config::insert ()

5.313.3 Member Data Documentation

5.313.3.1 algo_comm_t prediction_error_config::ac [private]

5.313.3.2 unsigned int prediction_error_config::ntaps [private]

- 5.313.3.3 `unsigned int prediction_error_config::frames` [private]
- 5.313.3.4 `unsigned int prediction_error_config::channels` [private]
- 5.313.3.5 `MHA_AC::waveform_t prediction_error_config::s_E` [private]
- 5.313.3.6 `MHA_AC::waveform_t prediction_error_config::F` [private]
- 5.313.3.7 `MHASignal::waveform_t prediction_error_config::Pu` [private]

Power of input signal delayline.

- 5.313.3.8 `std::string prediction_error_config::name_d_` [private]
- 5.313.3.9 `std::string prediction_error_config::name_lpc_` [private]
- 5.313.3.10 `int prediction_error_config::n_no_update_` [private]
- 5.313.3.11 `int prediction_error_config::no_iter` [private]
- 5.313.3.12 `int prediction_error_config::iter` [private]
- 5.313.3.13 `MHASignal::waveform_t prediction_error_config::v_G` [private]
- 5.313.3.14 `MHASignal::waveform_t prediction_error_config::s_U` [private]
- 5.313.3.15 `MHASignal::delay_t prediction_error_config::s_E_pred_err_delay` [private]
- 5.313.3.16 `MHASignal::delay_t prediction_error_config::s_W` [private]
- 5.313.3.17 `MHASignal::ringbuffer_t prediction_error_config::s_Wflt` [private]
- 5.313.3.18 `MHASignal::delay_t prediction_error_config::s_U_delay` [private]
- 5.313.3.19 `MHASignal::ringbuffer_t prediction_error_config::s_U_delayflt` [private]
- 5.313.3.20 `MHASignal::waveform_t prediction_error_config::F_Uflt` [private]
- 5.313.3.21 `MHASignal::delay_t prediction_error_config::s_Y_delay` [private]
- 5.313.3.22 `MHASignal::ringbuffer_t prediction_error_config::s_Y_delayflt` [private]
- 5.313.3.23 `MHASignal::ringbuffer_t prediction_error_config::UbufferPrew` [private]

5.313.3.24 mha_wave_t prediction_error_config::s_LPC [private]

5.313.3.25 mha_wave_t prediction_error_config::UPrew [private]

5.313.3.26 mha_wave_t prediction_error_config::YPrew [private]

5.313.3.27 mha_wave_t prediction_error_config::EPrew [private]

5.313.3.28 mha_wave_t prediction_error_config::UPrewW [private]

5.313.3.29 mha_wave_t prediction_error_config::smpl [private]

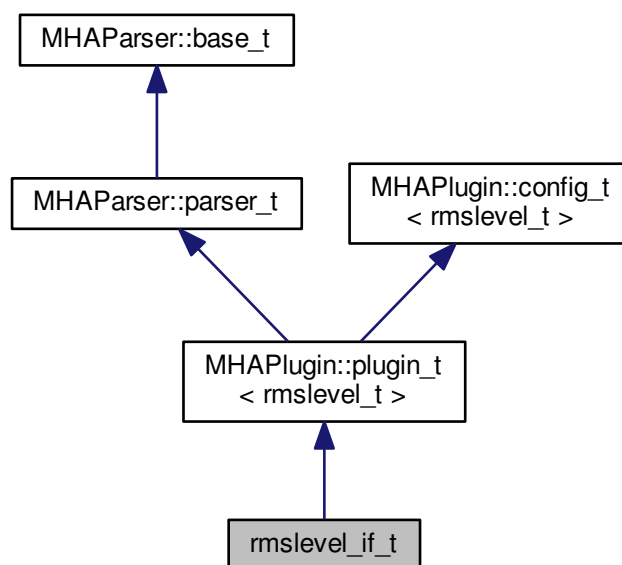
5.313.3.30 mha_wave_t* prediction_error_config::s_Usmpl [private]

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

- prediction_error.h
- prediction_error.cpp

5.314 rmslevel_if_t Class Reference

Inheritance diagram for rmslevel_if_t:



Public Member Functions

- **rmslevel_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t * process** (**mha_spec_t** *)
- **mha_wave_t * process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- std::string **name**

Additional Inherited Members

5.314.1 Constructor & Destructor Documentation

5.314.1.1 **rmslevel_if_t::rmslevel_if_t** (
 const **algo_comm_t** & *iac*,
 const std::string & *ith*,
 const std::string & *ial*)

5.314.2 Member Function Documentation

5.314.2.1 **mha_spec_t * rmslevel_if_t::process** (
 mha_spec_t * *s*)

5.314.2.2 **mha_wave_t * rmslevel_if_t::process** (
 mha_wave_t * *s*)

5.314.2.3 void **rmslevel_if_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Implements **MHAPLugin::plugin_t**< **rmslevel_t** > (p. 661).

5.314.3 Member Data Documentation

5.314.3.1 std::string **rmslevel_if_t::name** [private]

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

- **rmslevel.cpp**

5.315 rmslevel_t Class Reference

Public Member Functions

- **rmslevel_t** (unsigned int *nch*, **algo_comm_t** *ac*, std::string *name*, **MHAParser::parser**↔
t &*p*, unsigned int *fftlen*)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- void **insert** ()

Private Attributes

- **mon_t** *level_db*
- **mon_t** *peak_db*
- **mon_t** *level*
- **mon_t** *peak*
- unsigned int **fftlen**

5.315.1 Constructor & Destructor Documentation

5.315.1.1 **rmslevel_t::rmslevel_t** (
 unsigned int *nch*,
 algo_comm_t *ac*,
 std::string *name*,
 MHAParser::parser_t & *p*,
 unsigned int *fftlen_*)

5.315.2 Member Function Documentation

5.315.2.1 **mha_spec_t** * **rmslevel_t::process** (
 mha_spec_t * *s*)

5.315.2.2 **mha_wave_t** * **rmslevel_t::process** (
 mha_wave_t * *s*)

5.315.2.3 void **rmslevel_t::insert** ()

5.315.3 Member Data Documentation

5.315.3.1 **mon_t** **rmslevel_t::level_db** [private]

5.315.3.2 **mon_t** **rmslevel_t::peak_db** [private]

5.315.3.3 **mon_t** **rmslevel_t::level** [private]

5.315.3.4 **mon_t** **rmslevel_t::peak** [private]

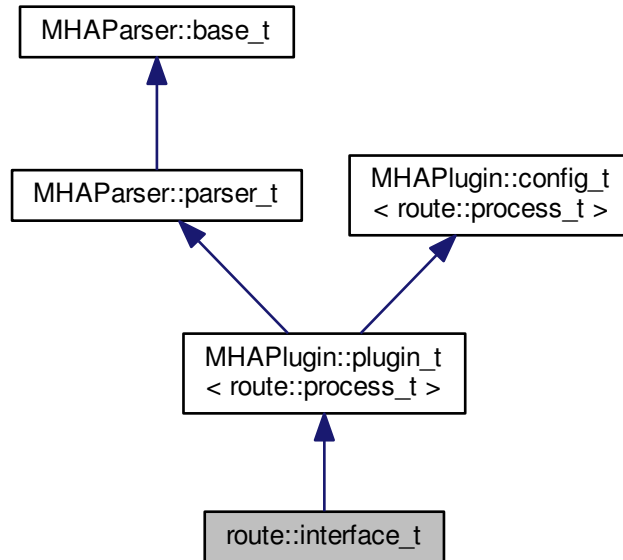
5.315.3.5 unsigned int **rmslevel_t::fftlen** [private]

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

- **rmslevel.cpp**

5.316 route::interface_t Class Reference

Inheritance diagram for route::interface_t:



Public Member Functions

- **interface_t** (algo_comm_t iac, const std::string &, const std::string &)
- void **prepare** (mhaconfig_t &)
- void **release** ()
- mha_wave_t * **process** (mha_wave_t *)
- mha_spec_t * **process** (mha_spec_t *)

Private Member Functions

- void **update** ()

Private Attributes

- MHAEvents::patchbay_t< route::interface_t > patchbay
- MHAParser::vstring_t route_out
- MHAParser::vstring_t route_ac
- mhaconfig_t cfin
- mhaconfig_t cfout
- mhaconfig_t cfac
- bool **prepared**
- bool **stopped**
- std::string algo

Additional Inherited Members

5.316.1 Constructor & Destructor Documentation

5.316.1.1 route::interface_t::interface_t (
 algo_comm_t iac,
 const std::string & ,
 const std::string & ialg)

5.316.2 Member Function Documentation

5.316.2.1 void route::interface_t::prepare (
 mhaconfig_t & cf) [virtual]

Implements **MHAPLugin::plugin_t**< route::process_t > (p. 661).

5.316.2.2 void route::interface_t::release (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t**< route::process_t > (p. 661).

5.316.2.3 mha_wave_t * route::interface_t::process (
 mha_wave_t * s)

5.316.2.4 mha_spec_t * route::interface_t::process (
 mha_spec_t * s)

5.316.2.5 void route::interface_t::update () [private]

5.316.3 Member Data Documentation

5.316.3.1 MHAEvents::patchbay_t<route::interface_t> route::interface_t::patchbay
 [private]

5.316.3.2 MHAParser::vstring_t route::interface_t::route_out [private]

5.316.3.3 MHAParser::vstring_t route::interface_t::route_ac [private]

5.316.3.4 mhaconfig_t route::interface_t::cfin [private]

5.316.3.5 mhaconfig_t route::interface_t::cfout [private]

5.316.3.6 mhaconfig_t route::interface_t::cfac [private]

5.316.3.7 bool route::interface_t::prepared [private]

5.316.3.8 bool route::interface_t::stopped [private]

5.316.3.9 std::string route::interface_t::algo [private]

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

- route.cpp

5.317 route::process_t Class Reference

Public Member Functions

- **process_t** (**algo_comm_t** iac, const std::string acname, const std::vector< std::string > &r_out, const std::vector< std::string > &r_ac, const **mhaconfig_t** &cf_in, const **mhaconfig_t** &cf_out, const **mhaconfig_t** &cf_ac, bool sync)
- **mha_wave_t** * **process** (**mha_wave_t** *)
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Attributes

- **MHAMultiSrc::waveform_t** wout
- **MHAMultiSrc::spectrum_t** sout
- **MHAMultiSrc::waveform_t** wout_ac
- **MHAMultiSrc::spectrum_t** sout_ac

5.317.1 Constructor & Destructor Documentation

5.317.1.1 route::process_t::process_t (
 algo_comm_t iac,
 const std::string acname,
 const std::vector< std::string > & r_out,
 const std::vector< std::string > & r_ac,
 const **mhaconfig_t** & cf_in,
 const **mhaconfig_t** & cf_out,
 const **mhaconfig_t** & cf_ac,
 bool sync)

5.317.2 Member Function Documentation

5.317.2.1 **mha_wave_t** * route::process_t::process (
 mha_wave_t * s)

5.317.2.2 **mha_spec_t** * route::process_t::process (
 mha_spec_t * s)

5.317.3 Member Data Documentation

5.317.3.1 **MHAMultiSrc::waveform_t** route::process_t::wout [private]

5.317.3.2 **MHAMultiSrc::spectrum_t** route::process_t::sout [private]

5.317.3.3 **MHAMultiSrc::waveform_t** route::process_t::wout_ac [private]

5.317.3.4 **MHAMultiSrc::spectrum_t** route::process_t::sout_ac [private]

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

- **route.cpp**

5.318 rt_nlms_t Class Reference

Public Member Functions

- **rt_nlms_t** (**algo_comm_t** iac, const std::string &name, const **mhaconfig_t** &cfg, unsigned int ntaps_, const std::string &name_u, const std::string &name_d, const std::string &name_e, const std::string &name_f, const int n_no_update)
- **~rt_nlms_t** ()
- **mha_wave_t** * **process** (**mha_wave_t** *sUD, **mha_real_t** rho, **mha_real_t** c, unsigned int norm_type, unsigned int estim_type, **mha_real_t** lambda_smooth)
- void **insert** ()

Private Attributes

- **algo_comm_t** ac
- unsigned int **ntaps**
- unsigned int **frames**
- unsigned int **channels**
- **MHA_AC::waveform_t** F
- **MHASignal::waveform_t** U
Input signal cache.
- **MHASignal::waveform_t** Uflt
Input signal cache (second filter)
- **MHASignal::waveform_t** Pu
Power of input signal delayline.
- **MHASignal::waveform_t** fu
Filtered input signal.
- **MHASignal::waveform_t** fuflt
Filtered input signal.
- **MHASignal::waveform_t** fu_previous
- **MHASignal::waveform_t** y_previous
- **MHASignal::waveform_t** P_Sum
- std::string **name_u**
- std::string **name_d**
- std::string **name_e**
- int **n_no_update**
- int **no_iter**
- **mha_wave_t** s_E

5.318.1 Constructor & Destructor Documentation

5.318.1.1 `rt_nlms_t::rt_nlms_t (`
 `algo_comm_t iac,`
 `const std::string & name,`
 `const mhaconfig_t & cfg,`
 `unsigned int ntaps,`
 `const std::string & name_u,`
 `const std::string & name_d,`
 `const std::string & name_e,`
 `const std::string & name_f,`
 `const int n_no_update)`

5.318.1.2 `rt_nlms_t::~~rt_nlms_t() [inline]`

5.318.2 Member Function Documentation

5.318.2.1 `mha_wave_t * rt_nlms_t::process (`
 `mha_wave_t * sUD,`
 `mha_real_t rho,`
 `mha_real_t c,`
 `unsigned int norm_type,`
 `unsigned int estim_type,`
 `mha_real_t lambda_smooth)`

5.318.2.2 `void rt_nlms_t::insert ()`

5.318.3 Member Data Documentation

5.318.3.1 `algo_comm_t rt_nlms_t::ac [private]`

5.318.3.2 `unsigned int rt_nlms_t::ntaps [private]`

5.318.3.3 `unsigned int rt_nlms_t::frames [private]`

5.318.3.4 `unsigned int rt_nlms_t::channels [private]`

5.318.3.5 `MHA_AC::waveform_t rt_nlms_t::F [private]`

5.318.3.6 `MHASignal::waveform_t rt_nlms_t::U [private]`

Input signal cache.

5.318.3.7 `MHASignal::waveform_t rt_nlms_t::Uflt [private]`

Input signal cache (second filter)

5.318.3.8 MHASignal::waveform_t rt_nlms_t::Pu [private]

Power of input signal delayline.

5.318.3.9 MHASignal::waveform_t rt_nlms_t::fu [private]

Filtered input signal.

5.318.3.10 MHASignal::waveform_t rt_nlms_t::fufilt [private]

Filtered input signal.

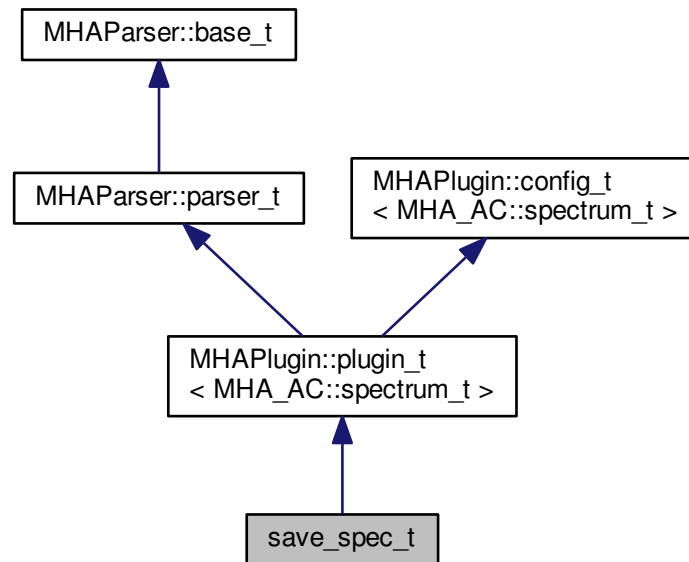
5.318.3.11 MHASignal::waveform_t rt_nlms_t::fu_previous [private]**5.318.3.12 MHASignal::waveform_t rt_nlms_t::y_previous** [private]**5.318.3.13 MHASignal::waveform_t rt_nlms_t::P_Sum** [private]**5.318.3.14 std::string rt_nlms_t::name_u_** [private]**5.318.3.15 std::string rt_nlms_t::name_d_** [private]**5.318.3.16 std::string rt_nlms_t::name_e_** [private]**5.318.3.17 int rt_nlms_t::n_no_update_** [private]**5.318.3.18 int rt_nlms_t::no_iter** [private]**5.318.3.19 mha_wave_t rt_nlms_t::s_E** [private]

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

- **nlms_wave.cpp**

5.319 save_spec_t Class Reference

Inheritance diagram for save_spec_t:



Public Member Functions

- **save_spec_t** (const **algo_comm_t** &iac, const std::string &ith, const std::string &ial)
- **mha_spec_t** * **process** (mha_spec_t *s)
- void **prepare** (mhaconfig_t &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.319.1 Constructor & Destructor Documentation

5.319.1.1 **save_spec_t::save_spec_t** (
 const **algo_comm_t** & iac,
 const std::string & ith,
 const std::string & ial) [inline]

5.319.2 Member Function Documentation

5.319.2.1 mha_spec_t* save_spec_t::process (
mha_spec_t* s) [inline]

5.319.2.2 void save_spec_t::prepare (
mhaconfig_t & tf) [inline],[virtual]

Implements **MHAPLugin::plugin_t**< **MHA_AC::spectrum_t** > (p. 661).

5.319.3 Member Data Documentation

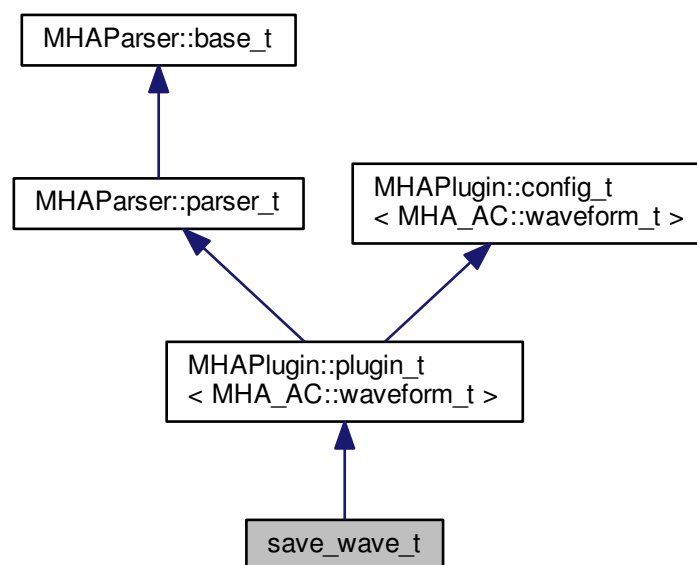
5.319.3.1 std::string save_spec_t::basename [private]

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

- **save_spec.cpp**

5.320 save_wave_t Class Reference

Inheritance diagram for save_wave_t:



Public Member Functions

- **save_wave_t** (const **algo_comm_t** &iac, const std::string &ith, const std::string &ial)
- **mha_wave_t * process** (**mha_wave_t** *s)
- void **prepare** (**mhaconfig_t** &tf)

Private Attributes

- std::string **basename**

Additional Inherited Members

5.320.1 Constructor & Destructor Documentation

5.320.1.1 **save_wave_t::save_wave_t** (
 const **algo_comm_t** & iac,
 const std::string & ith,
 const std::string & ial) [inline]

5.320.2 Member Function Documentation

5.320.2.1 **mha_wave_t* save_wave_t::process** (
 mha_wave_t * s) [inline]

5.320.2.2 void **save_wave_t::prepare** (
 mhaconfig_t & tf) [inline], [virtual]

Implements **MHAPlugin::plugin_t**< **MHA_AC::waveform_t** > (p. 661).

5.320.3 Member Data Documentation

5.320.3.1 std::string **save_wave_t::basename** [private]

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

- **save_wave.cpp**

5.321 shadowfilter_begin::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft, int inch, int outch, **algo_comm_t** ac, std::string name)
- **mha_spec_t * process** (**mha_spec_t** *)

Private Attributes

- **MHA_AC::spectrum_t** in_spec_copy
- **MHASignal::spectrum_t** out_spec
- **MHA_AC::int_t** nch
- **MHA_AC::int_t** ntracks

5.321.1 Constructor & Destructor Documentation

5.321.1.1 `cfg_t::cfg_t (`
 `int nfft,`
 `int inch,`
 `int outch,`
 `algo_comm_t ac,`
 `std::string name)`

5.321.2 Member Function Documentation

5.321.2.1 `mha_spec_t * cfg_t::process (`
 `mha_spec_t * s)`

5.321.3 Member Data Documentation

5.321.3.1 **MHA_AC::spectrum_t** shadowfilter_begin::cfg_t::in_spec_copy [private]

5.321.3.2 **MHASignal::spectrum_t** shadowfilter_begin::cfg_t::out_spec [private]

5.321.3.3 **MHA_AC::int_t** shadowfilter_begin::cfg_t::nch [private]

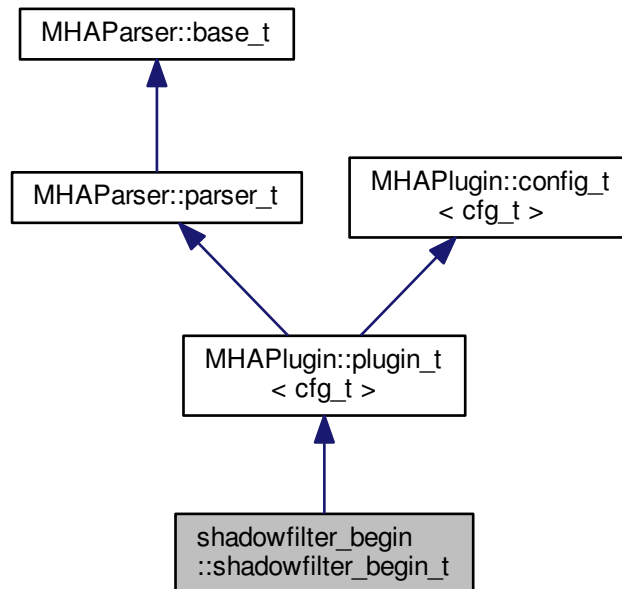
5.321.3.4 **MHA_AC::int_t** shadowfilter_begin::cfg_t::ntracks [private]

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

- **shadowfilter_begin.cpp**

5.322 shadowfilter_begin::shadowfilter_begin_t Class Reference

Inheritance diagram for shadowfilter_begin::shadowfilter_begin_t:



Public Member Functions

- **shadowfilter_begin_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t * process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- std::string **basename**
- **MHAParser::int_t nch**
- **MHAParser::int_t ntracks**

Additional Inherited Members

5.322.1 Constructor & Destructor Documentation

5.322.1.1 shadowfilter_begin::shadowfilter_begin_t::shadowfilter_begin_t (
 const algo_comm_t & iac,
 const std::string & ith,
 const std::string & ial)

5.322.2 Member Function Documentation

5.322.2.1 mha_spec_t * shadowfilter_begin::shadowfilter_begin_t::process (
 mha_spec_t * s)

5.322.2.2 void shadowfilter_begin::shadowfilter_begin_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPLugin::plugin_t**< **cfg_t** > (p. 661).

5.322.3 Member Data Documentation

5.322.3.1 std::string shadowfilter_begin::shadowfilter_begin_t::basename [private]

5.322.3.2 MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::nch [private]

5.322.3.3 MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::ntracks [private]

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

- shadowfilter_begin.cpp

5.323 shadowfilter_end::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int nfft_, algo_comm_t ac_, std::string name_)
- mha_spec_t * **process** (mha_spec_t *)

Private Attributes

- **algo_comm_t** ac
- std::string **name**
- int **nfft**
- int **ntracks**
- int **nch_out**
- **mha_spec_t** in_spec
- **MHASignal::spectrum_t** out_spec
- **MHA_AC::spectrum_t** gains

5.323.1 Constructor & Destructor Documentation

5.323.1.1 `cfg_t::cfg_t (`
 `int nfft,`
 `algo_comm_t ac,`
 `std::string name_)`

5.323.2 Member Function Documentation

5.323.2.1 `mha_spec_t * cfg_t::process (`
 `mha_spec_t * s)`

5.323.3 Member Data Documentation

5.323.3.1 `algo_comm_t shadowfilter_end::cfg_t::ac` [private]

5.323.3.2 `std::string shadowfilter_end::cfg_t::name` [private]

5.323.3.3 `int shadowfilter_end::cfg_t::nfft` [private]

5.323.3.4 `int shadowfilter_end::cfg_t::ntracks` [private]

5.323.3.5 `int shadowfilter_end::cfg_t::nch_out` [private]

5.323.3.6 `mha_spec_t shadowfilter_end::cfg_t::in_spec` [private]

5.323.3.7 `MHASignal::spectrum_t shadowfilter_end::cfg_t::out_spec` [private]

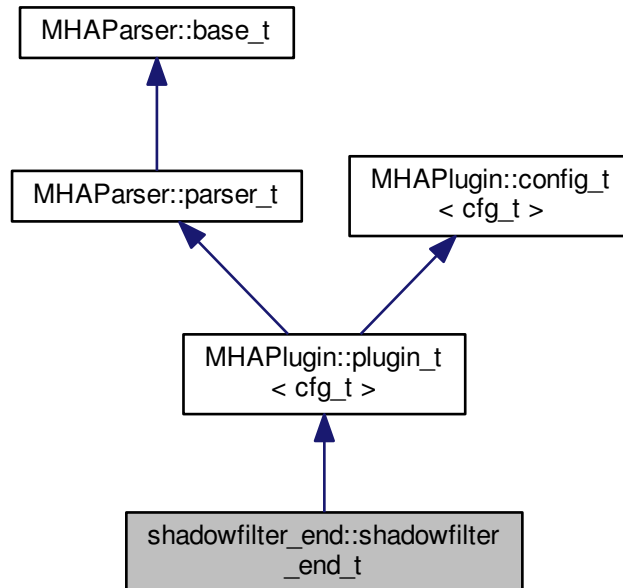
5.323.3.8 `MHA_AC::spectrum_t shadowfilter_end::cfg_t::gains` [private]

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

- `shadowfilter_end.cpp`

5.324 shadowfilter_end::shadowfilter_end_t Class Reference

Inheritance diagram for shadowfilter_end::shadowfilter_end_t:



Public Member Functions

- **shadowfilter_end_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **prepare** (**mhaconfig_t** &)

Private Attributes

- **MHAParser::string_t** **basename**

Additional Inherited Members

5.324.1 Constructor & Destructor Documentation

5.324.1.1 shadowfilter_end::shadowfilter_end_t::shadowfilter_end_t (const **algo_comm_t** & *iac*, const std::string & *ith*, const std::string & *ial*)

5.324.2 Member Function Documentation

5.324.2.1 `mha_spec_t * shadowfilter_end::shadowfilter_end_t::process (mha_spec_t * s)`

5.324.2.2 `void shadowfilter_end::shadowfilter_end_t::prepare (mhaconfig_t & tf) [virtual]`

Implements `MHAPLugin::plugin_t< cfg_t >` (p. 661).

5.324.3 Member Data Documentation

5.324.3.1 `MHAParser::string_t shadowfilter_end::shadowfilter_end_t::basename [private]`

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

- `shadowfilter_end.cpp`

5.325 sine_cfg_t Struct Reference

Public Member Functions

- `sine_cfg_t` (double `sampling_rate`, `mha_real_t` `frequency`, `mha_real_t` `newlev`, int `_mix`, const std::vector< int > & `_channels`)

Public Attributes

- double `phase_increment_div_2pi`
- double `amplitude`
- int `mix`
- const std::vector< int > `channels`

5.325.1 Constructor & Destructor Documentation

5.325.1.1 `sine_cfg_t::sine_cfg_t(`
 double sampling_rate,
 mha_real_t frequency,
 mha_real_t newlev,
 int _mix,
 const std::vector< int > &_channels) `[inline]`

5.325.2 Member Data Documentation

5.325.2.1 `double sine_cfg_t::phase_increment_div_2pi`

5.325.2.2 `double sine_cfg_t::amplitude`

5.325.2.3 `int sine_cfg_t::mix`

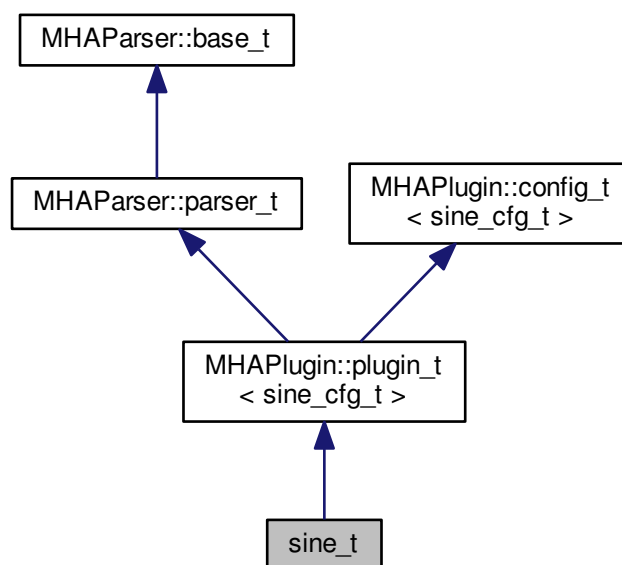
5.325.2.4 `const std::vector<int> sine_cfg_t::channels`

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

- **sine.cpp**

5.326 sine_t Class Reference

Inheritance diagram for `sine_t`:



Public Member Functions

- **sine_t** (const **algo_comm_t** &, const std::string &chain_name, const std::string &algo_name)
- **~sine_t** ()
- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &)

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAParser::float_t** lev
- **MHAParser::float_t** frequency
- **MHAParser::kw_t** mode
- **MHAParser::vint_t** channels
- double **phase_div_2pi**
- **MHAEvents::patchbay_t** < **sine_t** > patchbay

Additional Inherited Members

5.326.1 Constructor & Destructor Documentation

5.326.1.1 **sine_t::sine_t** (
 const **algo_comm_t** & *iac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

5.326.1.2 **sine_t::~sine_t** ()

5.326.2 Member Function Documentation

5.326.2.1 **mha_wave_t * sine_t::process** (
 mha_wave_t * *s*)

5.326.2.2 void **sine_t::prepare** (
 mhaconfig_t & *tf*) [virtual]

Implements **MHAPLugin::plugin_t** < **sine_cfg_t** > (p. 661).

5.326.2.3 void sine_t::update_cfg () [private]

5.326.3 Member Data Documentation

5.326.3.1 MHAParser::float_t sine_t::lev [private]

5.326.3.2 MHAParser::float_t sine_t::frequency [private]

5.326.3.3 MHAParser::kw_t sine_t::mode [private]

5.326.3.4 MHAParser::vint_t sine_t::channels [private]

5.326.3.5 double sine_t::phase_div_2pi [private]

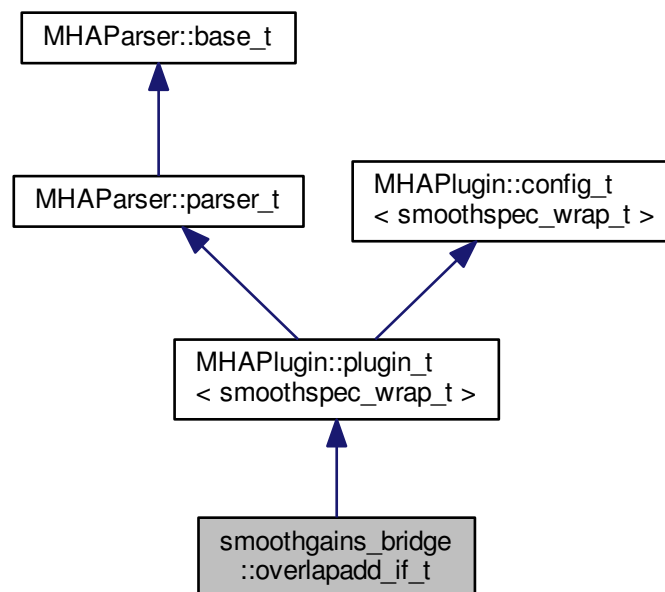
5.326.3.6 MHAEvents::patchbay_t<sine_t> sine_t::patchbay [private]

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

- **sine.cpp**

5.327 smoothgains_bridge::overlapadd_if_t Class Reference

Inheritance diagram for smoothgains_bridge::overlapadd_if_t:



Public Member Functions

- **overlapadd_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **~overlapadd_if_t** ()
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t** < **overlapadd_if_t** > **patchbay**
- **MHAParser::kw_t** **mode**
- **MHAParser::window_t** **irswnd**
- **MHAParser::float_t** **epsilon**
- **MHAParser::mhapluginloader_t** **plugloader**
- std::string **algo**
- **mhaconfig_t** **cf_in**
- **mhaconfig_t** **cf_out**

Additional Inherited Members

5.327.1 Constructor & Destructor Documentation

5.327.1.1 **smoothgains_bridge::overlapadd_if_t::overlapadd_if_t** (
 const **algo_comm_t** & *iac*,
 const std::string & ,
 const std::string & *ialg*)

5.327.1.2 **smoothgains_bridge::overlapadd_if_t::~~overlapadd_if_t** ()

5.327.2 Member Function Documentation

5.327.2.1 void **smoothgains_bridge::overlapadd_if_t::prepare** (
 mhaconfig_t & *t*) [virtual]

Implements **MHAPLugin::plugin_t** < **smoothspec_wrap_t** > (p. 661).

5.327.2.2 void **smoothgains_bridge::overlapadd_if_t::release** (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t** < **smoothspec_wrap_t** > (p. 661).

5.328 smoothgains_bridge::smoothspec_wrap_t Class Reference

5.327.2.3 **mha_spec_t** * smoothgains_bridge::overlapadd_if_t::process (
 mha_spec_t * *spec*)

5.327.2.4 **void** smoothgains_bridge::overlapadd_if_t::update () [private]

5.327.3 Member Data Documentation

5.327.3.1 **MHAEvents::patchbay_t**<**overlapadd_if_t**> smoothgains_bridge::overlapadd_if_t↵
::patchbay [private]

5.327.3.2 **MHAParser::kw_t** smoothgains_bridge::overlapadd_if_t::mode [private]

5.327.3.3 **MHAParser::window_t** smoothgains_bridge::overlapadd_if_t::irswnd [private]

5.327.3.4 **MHAParser::float_t** smoothgains_bridge::overlapadd_if_t::epsilon [private]

5.327.3.5 **MHAParser::mhapluginloader_t** smoothgains_bridge::overlapadd_if_t::plugloader
[private]

5.327.3.6 **std::string** smoothgains_bridge::overlapadd_if_t::algo [private]

5.327.3.7 **mhaconfig_t** smoothgains_bridge::overlapadd_if_t::cf_in [private]

5.327.3.8 **mhaconfig_t** smoothgains_bridge::overlapadd_if_t::cf_out [private]

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

- **smoothgains_bridge.cpp**

5.328 smoothgains_bridge::smoothspec_wrap_t Class Reference

Public Member Functions

- **smoothspec_wrap_t** (**mhaconfig_t** spar_in, **mhaconfig_t** spar_out, const **MHA**↵
Parser::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**

5.328.1 Constructor & Destructor Documentation

5.328.1.1 `smoothgains_bridge::smoothspec_wrap_t::smoothspec_wrap_t (`
 `mhaconfig_t spar_in,`
 `mhaconfig_t spar_out,`
 `const MHAParser::kw_t & mode,`
 `const MHAParser::window_t & irswnd,`
 `const MHAParser::float_t & epsilon)`

5.328.2 Member Function Documentation

5.328.2.1 `mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_1 (`
 `mha_spec_t * s)`

5.328.2.2 `mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_2 (`
 `mha_spec_t * s)`

5.328.3 Member Data Documentation

5.328.3.1 `MHASignal::spectrum_t smoothgains_bridge::smoothspec_wrap_t::spec_in_copy`
 `[private]`

Copy of input spectrum for smoothspec.

5.328.3.2 `MHAFilter::smoothspec_t smoothgains_bridge::smoothspec_wrap_t::smoothspec`
 `[private]`

Smoothspec calculator.

5.328.3.3 `bool smoothgains_bridge::smoothspec_wrap_t::use_smoothspec` `[private]`

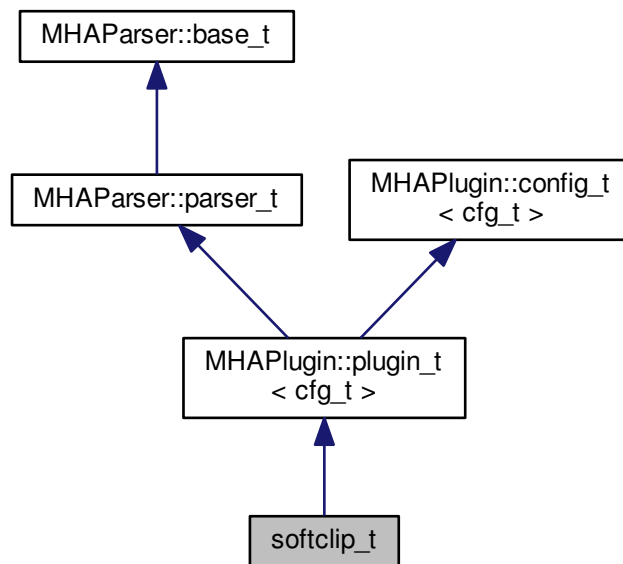
5.328.3.4 `float smoothgains_bridge::smoothspec_wrap_t::smoothspec_epsilon` `[private]`

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

- `smoothgains_bridge.cpp`

5.329 softclip_t Class Reference

Inheritance diagram for softclip_t:



Public Member Functions

- **softclip_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- **mha_wave_t * process** (**mha_wave_t** *)
- void **prepare** (**mhaconfig_t** &)
- void **update** ()

Private Attributes

- **mhaconfig_t** tftype
- **MHAParser::float_t** attack
- **MHAParser::float_t** decay
- **MHAParser::float_t** start_limit
- **MHAParser::float_t** slope_db
- **MHAEvents::patchbay_t** < **softclip_t** > patchbay

Additional Inherited Members

5.329.1 Constructor & Destructor Documentation

5.329.1.1 `softclip_t::softclip_t (`
 `const algo_comm_t & iac,`
 `const std::string & chain,`
 `const std::string & name)`

5.329.2 Member Function Documentation

5.329.2.1 `mha_wave_t * softclip_t::process (`
 `mha_wave_t * s)`

5.329.2.2 `void softclip_t::prepare (`
 `mhaconfig_t & tf)` [virtual]

Implements **MHAParser::plugin_t< cfg_t >** (p. 661).

5.329.2.3 `void softclip_t::update ()`

5.329.3 Member Data Documentation

5.329.3.1 `mhaconfig_t softclip_t::tftype` [private]

5.329.3.2 `MHAParser::float_t softclip_t::attack` [private]

5.329.3.3 `MHAParser::float_t softclip_t::decay` [private]

5.329.3.4 `MHAParser::float_t softclip_t::start_limit` [private]

5.329.3.5 `MHAParser::float_t softclip_t::slope_db` [private]

5.329.3.6 `MHAEvents::patchbay_t<softclip_t> softclip_t::patchbay` [private]

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

- **softclip.cpp**

5.330 softclipper_t Class Reference

Public Member Functions

- **softclipper_t** (const **softclipper_variables_t** &v, const **mhaconfig_t** &)
- **mha_real_t process** (**mha_wave_t** *)

Private Attributes

- **MHAFilter::o1flt_lowpass_t** attack
- **MHAFilter::o1flt_maxtrack_t** decay
- **MHAFilter::o1flt_lowpass_t** clipmeter
- **mha_real_t** threshold
- **mha_real_t** hardlimit
- **mha_real_t** slope
- **bool** linear

5.330.1 Constructor & Destructor Documentation

5.330.1.1 `softclipper_t::softclipper_t (`
 `const softclipper_variables_t & v,`
 `const mhaconfig_t & tf)`

5.330.2 Member Function Documentation

5.330.2.1 `mha_real_t softclipper_t::process (`
 `mha_wave_t * s)`

5.330.3 Member Data Documentation

5.330.3.1 `MHAFilter::o1flt_lowpass_t softclipper_t::attack` [private]

5.330.3.2 `MHAFilter::o1flt_maxtrack_t softclipper_t::decay` [private]

5.330.3.3 `MHAFilter::o1flt_lowpass_t softclipper_t::clipmeter` [private]

5.330.3.4 `mha_real_t softclipper_t::threshold` [private]

5.330.3.5 `mha_real_t softclipper_t::hardlimit` [private]

5.330.3.6 `mha_real_t softclipper_t::slope` [private]

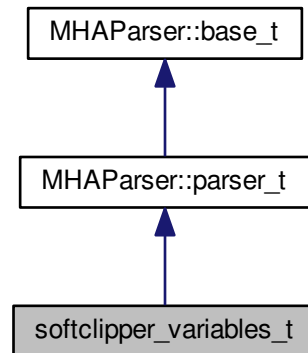
5.330.3.7 `bool softclipper_t::linear` [private]

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

- **transducers.cpp**

5.331 softclipper_variables_t Class Reference

Inheritance diagram for softclipper_variables_t:



Public Member Functions

- **softclipper_variables_t ()**

Public Attributes

- **MHAParser::float_t tau_attack**
- **MHAParser::float_t tau_decay**
- **MHAParser::float_t tau_clip**
- **MHAParser::float_t threshold**
- **MHAParser::float_t hardlimit**
- **MHAParser::float_t slope**
- **MHAParser::bool_t linear**
- **MHAParser::float_mon_t clipped**
- **MHAParser::float_t max_clipped**

Additional Inherited Members

5.331.1 Constructor & Destructor Documentation

5.331.1.1 softclipper_variables_t::softclipper_variables_t ()

5.331.2 Member Data Documentation

5.331.2.1 MHAParser::float_t softclipper_variables_t::tau_attack

5.331.2.2 MHAParser::float_t softclipper_variables_t::tau_decay

5.331.2.3 MHAParser::float_t softclipper_variables_t::tau_clip

5.331.2.4 MHAParser::float_t softclipper_variables_t::threshold

5.331.2.5 MHAParser::float_t softclipper_variables_t::hardlimit

5.331.2.6 MHAParser::float_t softclipper_variables_t::slope

5.331.2.7 MHAParser::bool_t softclipper_variables_t::linear

5.331.2.8 MHAParser::float_mon_t softclipper_variables_t::clipped

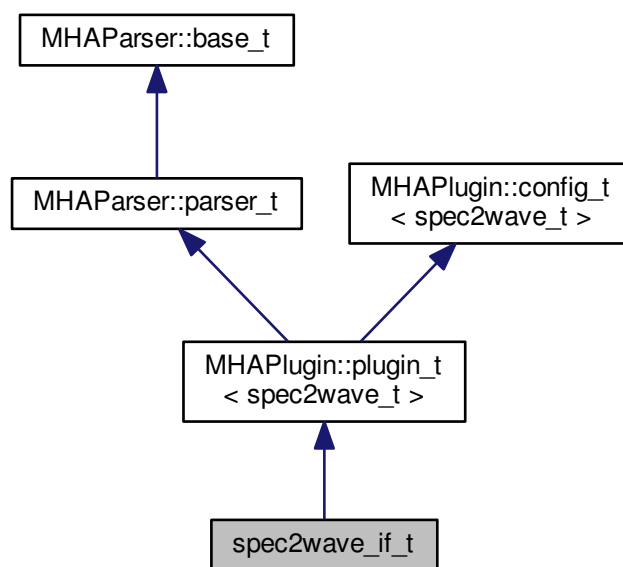
5.331.2.9 MHAParser::float_t softclipper_variables_t::max_clipped

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

- **transducers.cpp**

5.332 spec2wave_if_t Class Reference

Inheritance diagram for spec2wave_if_t:



Public Member Functions

- **spec2wave_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- **mha_wave_t** * **process** (**mha_spec_t** *)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t**< **spec2wave_if_t** > **patchbay**
- **MHAParser::float_t** **ramplen**
- **windowselector_t** **window_config**

Additional Inherited Members

5.332.1 Constructor & Destructor Documentation

5.332.1.1 **spec2wave_if_t::spec2wave_if_t** (
 const **algo_comm_t** & *iac*,
 const std::string & ,
 const std::string &)

5.332.2 Member Function Documentation

5.332.2.1 void **spec2wave_if_t::prepare** (
 mhaconfig_t & *t*) [virtual]

Implements **MHAPlugin::plugin_t**< **spec2wave_t** > (p. 661).

5.332.2.2 **mha_wave_t** * **spec2wave_if_t::process** (
 mha_spec_t * *spec_in*)

5.332.2.3 void **spec2wave_if_t::update** () [private]

5.332.3 Member Data Documentation

5.332.3.1 **MHAEvents::patchbay_t**<**spec2wave_if_t**> **spec2wave_if_t::patchbay** [private]

5.332.3.2 **MHAParser::float_t** **spec2wave_if_t::ramplen** [private]

5.332.3.3 **windowselector_t** **spec2wave_if_t::window_config** [private]

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

- **spec2wave.cpp**

5.333 spec2wave_t Class Reference

Public Member Functions

- **spec2wave_t** (unsigned int *nfft_*, unsigned int *nwnd_*, unsigned int *nwndshift_*, unsigned int *nch*, **mha_real_t** *ramplen*, const **MHAWindow::base_t** &*postwin*)
- **~spec2wave_t** ()
- **mha_wave_t * process** (**mha_spec_t** *)

Private Attributes

- **mha_fft_t** *ft*
FFT class.
- unsigned int **npad1**
length of zero padding before window
- unsigned int **npad2**
length of zero padding after window
- **hanning_ramps_t** *ramps*
- **MHASignal::waveform_t** *calc_out*
- **MHASignal::waveform_t** *out_buf*
- **MHASignal::waveform_t** *write_buf*
- **mha_real_t** *sc*
- unsigned int **nfft**
- unsigned int **nwndshift**
- **MHAWindow::base_t** *postwindow*

5.333.1 Constructor & Destructor Documentation

5.333.1.1 **spec2wave_t::spec2wave_t** (
 unsigned int *nfft_*,
 unsigned int *nwnd_*,
 unsigned int *nwndshift_*,
 unsigned int *nch*,
 mha_real_t *ramplen*,
 const **MHAWindow::base_t** & *postwin*)

5.333.1.2 **spec2wave_t::~~spec2wave_t** ()

5.333.2 Member Function Documentation

5.333.2.1 **mha_wave_t * spec2wave_t::process** (
 mha_spec_t * *spec_in*)

5.333.3 Member Data Documentation

5.333.3.1 **mha_fft_t** **spec2wave_t::ft** [private]

FFT class.

5.333.3.2 `unsigned int spec2wave_t::npad1` [private]

length of zero padding before window

5.333.3.3 `unsigned int spec2wave_t::npad2` [private]

length of zero padding after window

5.333.3.4 `hanning_ramps_t spec2wave_t::ramps` [private]

5.333.3.5 `MHASignal::waveform_t spec2wave_t::calc_out` [private]

5.333.3.6 `MHASignal::waveform_t spec2wave_t::out_buf` [private]

5.333.3.7 `MHASignal::waveform_t spec2wave_t::write_buf` [private]

5.333.3.8 `mha_real_t spec2wave_t::sc` [private]

5.333.3.9 `unsigned int spec2wave_t::nfft` [private]

5.333.3.10 `unsigned int spec2wave_t::nwndshift` [private]

5.333.3.11 `MHAWindow::base_t spec2wave_t::postwindow` [private]

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

- `spec2wave.cpp`

5.334 `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`

5.334.1 Constructor & Destructor Documentation

5.334.1.1 `spec_fader_t::spec_fader_t (`
 `unsigned int ch,`
 `mha_real_t fr,`
 `MHAParser::vfloat_t & ng,`
 `MHAParser::float_t & t)`

5.334.1.2 `spec_fader_t::~~spec_fader_t ()` `[inline]`

5.334.2 Member Data Documentation

5.334.2.1 `unsigned int spec_fader_t::nch`

5.334.2.2 `mha_real_t* spec_fader_t::gains`

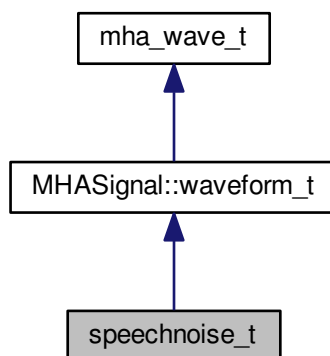
5.334.2.3 `unsigned int spec_fader_t::fr`

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

- **fader_spec.cpp**

5.335 speechnoise_t Class Reference

Inheritance diagram for `speechnoise_t`:



Public Types

Public Member Functions

- **speechnoise_t** (float duration, float srates, unsigned int **channels**, **speechnoise_t::noise_type_t** noise_type=**speechnoise_t::mha**)
- **speechnoise_t** (unsigned int length_samples, float srates, 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 srates)

Additional Inherited Members

5.335.1 Member Enumeration Documentation

5.335.1.1 enum **speechnoise_t::noise_type_t**

Enumerator

mha
olnoise
LTASS_combined
LTASS_female
LTASS_male
white
pink
brown
TEN_SPL
TEN_SPL_250_8k
TEN_SPL_50_16k
sin125
sin250
sin500
sin1k
sin2k
sin4k
sin8k

5.335.2 Constructor & Destructor Documentation

5.335.2.1 `speechnoise_t::speechnoise_t (`
 float *duration*,
 float *srate*,
 unsigned int *channels*,
 speechnoise_t::noise_type_t *noise_type* = speechnoise_t::mha)

5.335.2.2 `speechnoise_t::speechnoise_t (`
 unsigned int *length_samples*,
 float *srate*,
 unsigned int *channels*,
 speechnoise_t::noise_type_t *noise_type* = speechnoise_t::mha)

5.335.3 Member Function Documentation

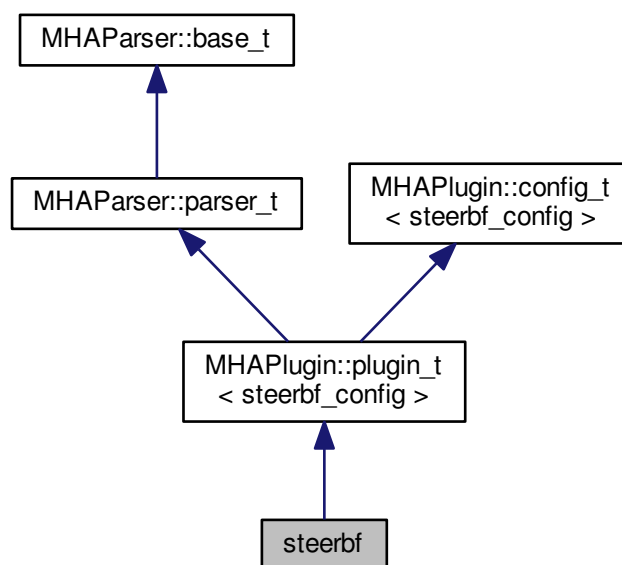
5.335.3.1 `void speechnoise_t::creator (`
 speechnoise_t::noise_type_t *noise_type*,
 float *srate*) [private]

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

- **speechnoise.h**
- **speechnoise.cpp**

5.336 steerbf Class Reference

Inheritance diagram for steerbf:



Public Member Functions

- **steerbf** (**algo_comm_t** &**ac**, const std::string &**chain_name**, const std::string &**algo_name**)
Constructs our plugin.
- **~steerbf** ()
- **mha_spec_t** * **process** (**mha_spec_t** *)
Defers to configuration class.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Public Attributes

- **MHAParser::string_t** **bf_src**
- **parser_int_dyn** **angle_ind**
- **MHAParser::string_t** **angle_src**

Private Member Functions

- void **update_cfg** ()

Private Attributes

- **MHAEvents::patchbay_t** < **steerbf** > **patchbay**

Additional Inherited Members

5.336.1 Constructor & Destructor Documentation

5.336.1.1 **steerbf::steerbf** (
 algo_comm_t & *ac*,
 const std::string & *chain_name*,
 const std::string & *algo_name*)

Constructs our plugin.

5.336.1.2 **steerbf::~~steerbf** ()

5.336.2 Member Function Documentation

5.336.2.1 **mha_spec_t** * **steerbf::process** (
 mha_spec_t * *signal*)

Defers to configuration class.

5.336.2.2 void **steerbf::prepare** (
 mhaconfig_t & *signal_info*) [virtual]

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

| | |
|--------------------|---|
| <i>signal_info</i> | 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. 661).

5.336.2.3 void steerbf::release (
void) [inline],[virtual]

Reimplemented from **MHAPLugin::plugin_t< steerbf_config >** (p. 661).

5.336.2.4 void steerbf::update_cfg () [private]

5.336.3 Member Data Documentation

5.336.3.1 MHAParser::string_t steerbf::bf_src

5.336.3.2 parser_int_dyn steerbf::angle_ind

5.336.3.3 MHAParser::string_t steerbf::angle_src

5.336.3.4 MHAEvents::patchbay_t<steerbf> steerbf::patchbay [private]

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

- **steerbf.h**
- **steerbf.cpp**

5.337 steerbf_config Class Reference

Public Member Functions

- **steerbf_config** (algo_comm_t &ac, const mhaconfig_t in_cfg, steerbf *steerbf)
- **~steerbf_config** ()
- **mha_spec_t * process** (mha_spec_t *)

Private Attributes

- unsigned int **nchan**
- unsigned int **nfreq**
- **MHASignal::spectrum_t** outSpec
- **mha_spec_t** bf_vec
- unsigned int **nangle**
- **steerbf * _steerbf**
- **algo_comm_t & ac**
- std::string **bf_src_copy**

5.337.1 Constructor & Destructor Documentation

5.337.1.1 `steerbf_config::steerbf_config (`
 `algo_comm_t & ac,`
 `const mhaconfig_t in_cfg,`
 `steerbf * steerbf)`

5.337.1.2 `steerbf_config::~~steerbf_config ()`

5.337.2 Member Function Documentation

5.337.2.1 `mha_spec_t * steerbf_config::process (`
 `mha_spec_t * inSpec)`

5.337.3 Member Data Documentation

5.337.3.1 `unsigned int steerbf_config::nchan` [private]

5.337.3.2 `unsigned int steerbf_config::nfreq` [private]

5.337.3.3 `MHASignal::spectrum_t steerbf_config::outSpec` [private]

5.337.3.4 `mha_spec_t steerbf_config::bf_vec` [private]

5.337.3.5 `unsigned int steerbf_config::nangle` [private]

5.337.3.6 `steerbf* steerbf_config::_steerbf` [private]

5.337.3.7 `algo_comm_t& steerbf_config::ac` [private]

5.337.3.8 `std::string steerbf_config::bf_src_copy` [private]

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

- **steerbf.h**
- **steerbf.cpp**

5.338 timo_AC Class Reference

Public Member Functions

- **timo_AC** (`algo_comm_t &ac`, `unsigned int fftlen`, `unsigned int nfreq`, `unsigned int nchan`)
- `void copy ()`
- `void insert ()`

Public Attributes

- MHA_AC::waveform_t gamma_post_AC
- MHA_AC::waveform_t xi_ml_AC
- MHA_AC::spectrum_t lambda_ml_AC
- MHA_AC::spectrum_t lambda_ml_ceps_AC
- MHA_AC::waveform_t lambda_ml_smooth_AC
- MHA_AC::waveform_t max_q_AC
- MHA_AC::waveform_t max_val_AC
- MHA_AC::waveform_t pitch_set_first_AC
- MHA_AC::waveform_t pitch_set_last_AC
- MHA_AC::waveform_t alpha_hat_AC
- MHA_AC::waveform_t alpha_frame_AC
- MHA_AC::spectrum_t lambda_ceps_AC
- MHA_AC::spectrum_t log_lambda_spec_AC
- MHA_AC::waveform_t lambda_spec_AC
- MHA_AC::waveform_t xi_est_AC
- MHA_AC::waveform_t gain_wiener_AC
- MHA_AC::waveform_t winF0_AC
- MHA_AC::waveform_t SPP

5.338.1 Constructor & Destructor Documentation

5.338.1.1 timo_AC::timo_AC (
 algo_comm_t & ac,
 unsigned int *fftl*en,
 unsigned int *nfreq*,
 unsigned int *nchan*) [inline]

5.338.2 Member Function Documentation

5.338.2.1 void timo_AC::copy ()

5.338.2.2 void timo_AC::insert ()

5.338.3 Member Data Documentation

5.338.3.1 MHA_AC::waveform_t timo_AC::gamma_post_AC

5.338.3.2 MHA_AC::waveform_t timo_AC::xi_ml_AC

5.338.3.3 MHA_AC::spectrum_t timo_AC::lambda_ml_AC

5.338.3.4 MHA_AC::spectrum_t timo_AC::lambda_ml_ceps_AC

- 5.338.3.5 MHA_AC::waveform_t timo_AC::lambda_ml_smooth_AC
- 5.338.3.6 MHA_AC::waveform_t timo_AC::max_q_AC
- 5.338.3.7 MHA_AC::waveform_t timo_AC::max_val_AC
- 5.338.3.8 MHA_AC::waveform_t timo_AC::pitch_set_first_AC
- 5.338.3.9 MHA_AC::waveform_t timo_AC::pitch_set_last_AC
- 5.338.3.10 MHA_AC::waveform_t timo_AC::alpha_hat_AC
- 5.338.3.11 MHA_AC::waveform_t timo_AC::alpha_frame_AC
- 5.338.3.12 MHA_AC::spectrum_t timo_AC::lambda_ceps_AC
- 5.338.3.13 MHA_AC::spectrum_t timo_AC::log_lambda_spec_AC
- 5.338.3.14 MHA_AC::waveform_t timo_AC::lambda_spec_AC
- 5.338.3.15 MHA_AC::waveform_t timo_AC::xi_est_AC
- 5.338.3.16 MHA_AC::waveform_t timo_AC::gain_wiener_AC
- 5.338.3.17 MHA_AC::waveform_t timo_AC::winF0_AC
- 5.338.3.18 MHA_AC::waveform_t timo_AC::SPP

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

- **timoconfig.h**
- **timoconfig.cpp**

5.339 timo_params Class Reference

Public Member Functions

- **timo_params** (const **mhaconfig_t** &_in_cfg, float _xi_min_db, float _f0_low, float _f0_high, float _delta_pitch, float _lambda_thresh, float _alpha_pitch, float _beta_const, float _kappa_const, float _prior_q, float _xi_opt_db, float _gain_min_db, std::vector< float > &_winF0, std::vector< float > &_alpha_const_vals, std::vector< float > &_alpha_const_limits_hz, std::string &_noisePow_name)

Public Attributes

- const **mhaconfig_t** **in_cfg**
- float **xi_min_db**
- float **f0_low**
- float **f0_high**
- float **delta_pitch**
- float **lambda_thresh**
- float **alpha_pitch**
- float **beta_const**
- float **kappa_const**
- float **prior_q**
- float **xi_opt_db**
- float **gain_min_db**
- std::vector< float > **winF0**
- std::vector< float > **alpha_const_vals**
- std::vector< float > **alpha_const_limits_hz**
- std::string **noisePow_name**

5.339.1 Constructor & Destructor Documentation

```

5.339.1.1 timo_params::timo_params (
    const mhaconfig_t & _in_cfg,
    float _xi_min_db,
    float _f0_low,
    float _f0_high,
    float _delta_pitch,
    float _lambda_thresh,
    float _alpha_pitch,
    float _beta_const,
    float _kappa_const,
    float _prior_q,
    float _xi_opt_db,
    float _gain_min_db,
    std::vector< float > & _winF0,
    std::vector< float > & _alpha_const_vals,
    std::vector< float > & _alpha_const_limits_hz,
    std::string & _noisePow_name ) [inline]

```

5.339.2 Member Data Documentation

5.339.2.1 const mhaconfig_t timo_params::in_cfg

5.339.2.2 float timo_params::xi_min_db

5.339.2.3 float timo_params::f0_low

- 5.339.2.4 float timo_params::f0_high
- 5.339.2.5 float timo_params::delta_pitch
- 5.339.2.6 float timo_params::lambda_thresh
- 5.339.2.7 float timo_params::alpha_pitch
- 5.339.2.8 float timo_params::beta_const
- 5.339.2.9 float timo_params::kappa_const
- 5.339.2.10 float timo_params::prior_q
- 5.339.2.11 float timo_params::xi_opt_db
- 5.339.2.12 float timo_params::gain_min_db
- 5.339.2.13 std::vector<float> timo_params::winF0
- 5.339.2.14 std::vector<float> timo_params::alpha_const_vals
- 5.339.2.15 std::vector<float> timo_params::alpha_const_limits_hz
- 5.339.2.16 std::string timo_params::noisePow_name

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

- **timoconfig.h**

5.340 timoConfig Class Reference

Public Member Functions

- **timoConfig** (algo_comm_t &ac, timo_params ¶ms)
- **~timoConfig** ()
- **mha_spec_t * process** (mha_spec_t *)

Private Member Functions

- void **copy_AC** (timo_AC &tAC)

Private Attributes

- **algo_comm_t** ac
- **timo_params** params
- unsigned int **fftl**len
- **mha_fft_t** mha_fft
- unsigned int **nfreq**
- unsigned int **nchan**
- **timo_AC** tAC
- float **ola_powspec_scale**
- float **q_low**
- float **q_high**
- **MHASignal::waveform_t** winF0
- float **xi_min**
- float **gain_min**
- **MHASignal::waveform_t** alpha_const
- **MHASignal::waveform_t** alpha_prev
- **MHASignal::waveform_t** noisePow
- **MHASignal::waveform_t** powSpec
- **MHASignal::waveform_t** gamma_post
- **MHASignal::waveform_t** xi_ml
- **MHASignal::spectrum_t** lambda_ml_full
- **MHASignal::spectrum_t** lambda_ml_ceps
- **MHASignal::waveform_t** lambda_ml_smooth
- **MHASignal::waveform_t** alpha_hat
- **MHASignal::waveform_t** alpha_frame
- **MHASignal::spectrum_t** lambda_ceps
- **MHASignal::waveform_t** lambda_ceps_prev
- **MHASignal::spectrum_t** log_lambda_spec
- **MHASignal::waveform_t** lambda_spec
- **MHASignal::waveform_t** xi_est
- **MHASignal::waveform_t** gain_wiener
- **MHASignal::spectrum_t** spec_out
- double * **max_val**
- int * **max_q**
- int * **pitch_set_first**
- int * **pitch_set_last**
- float **priorFact**
- float **xiOpt**
- float **logGLRFact**
- float **GLRexp**
- **MHASignal::waveform_t** GLR

5.340.1 Constructor & Destructor Documentation

5.340.1.1 `timoConfig::timoConfig (`
 `algo_comm_t & ac,`
 `timo_params & params)`

5.340.1.2 `timoConfig::~~timoConfig ()`

5.340.2 Member Function Documentation

5.340.2.1 `mha_spec_t * timoConfig::process (`
 `mha_spec_t * noisyFrame)`

5.340.2.2 `void timoConfig::copy_AC (`
 `timo_AC & tAC) [private]`

5.340.3 Member Data Documentation

5.340.3.1 `algo_comm_t timoConfig::ac [private]`

5.340.3.2 `timo_params timoConfig::params [private]`

5.340.3.3 `unsigned int timoConfig::fftlens [private]`

5.340.3.4 `mha_fft_t timoConfig::mha_fft [private]`

5.340.3.5 `unsigned int timoConfig::nfreq [private]`

5.340.3.6 `unsigned int timoConfig::nchan [private]`

5.340.3.7 `timo_AC timoConfig::tAC [private]`

5.340.3.8 `float timoConfig::ola_powspec_scale [private]`

5.340.3.9 `float timoConfig::q_low [private]`

5.340.3.10 `float timoConfig::q_high [private]`

5.340.3.11 `MHASignal::waveform_t timoConfig::winF0 [private]`

5.340.3.12 `float timoConfig::xi_min [private]`

5.340.3.13 `float timoConfig::gain_min [private]`

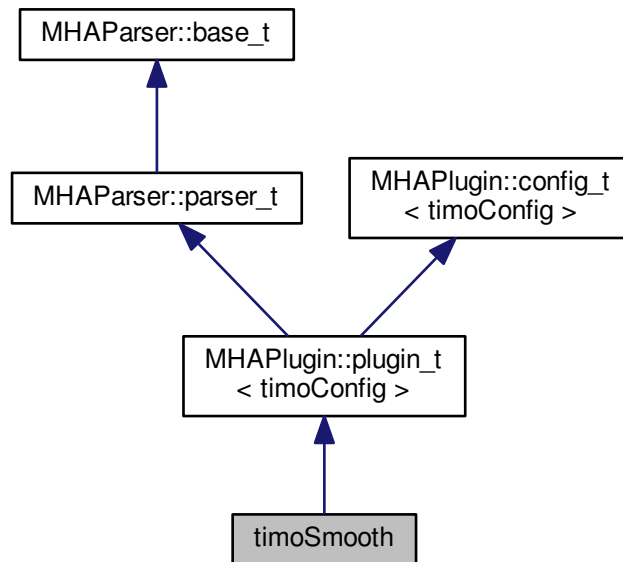
- 5.340.3.14 **MHASignal::waveform_t timoConfig::alpha_const** [private]
- 5.340.3.15 **MHASignal::waveform_t timoConfig::alpha_prev** [private]
- 5.340.3.16 **MHASignal::waveform_t timoConfig::noisePow** [private]
- 5.340.3.17 **MHASignal::waveform_t timoConfig::powSpec** [private]
- 5.340.3.18 **MHASignal::waveform_t timoConfig::gamma_post** [private]
- 5.340.3.19 **MHASignal::waveform_t timoConfig::xi_ml** [private]
- 5.340.3.20 **MHASignal::spectrum_t timoConfig::lambda_ml_full** [private]
- 5.340.3.21 **MHASignal::spectrum_t timoConfig::lambda_ml_ceps** [private]
- 5.340.3.22 **MHASignal::waveform_t timoConfig::lambda_ml_smooth** [private]
- 5.340.3.23 **MHASignal::waveform_t timoConfig::alpha_hat** [private]
- 5.340.3.24 **MHASignal::waveform_t timoConfig::alpha_frame** [private]
- 5.340.3.25 **MHASignal::spectrum_t timoConfig::lambda_ceps** [private]
- 5.340.3.26 **MHASignal::waveform_t timoConfig::lambda_ceps_prev** [private]
- 5.340.3.27 **MHASignal::spectrum_t timoConfig::log_lambda_spec** [private]
- 5.340.3.28 **MHASignal::waveform_t timoConfig::lambda_spec** [private]
- 5.340.3.29 **MHASignal::waveform_t timoConfig::xi_est** [private]
- 5.340.3.30 **MHASignal::waveform_t timoConfig::gain_wiener** [private]
- 5.340.3.31 **MHASignal::spectrum_t timoConfig::spec_out** [private]
- 5.340.3.32 **double* timoConfig::max_val** [private]
- 5.340.3.33 **int* timoConfig::max_q** [private]
- 5.340.3.34 **int* timoConfig::pitch_set_first** [private]
- 5.340.3.35 **int* timoConfig::pitch_set_last** [private]
- 5.340.3.36 **float timoConfig::priorFact** [private]
- 5.340.3.37 **float timoConfig::xiOpt** [private]
- 5.340.3.38 **float timoConfig::logGLRFact** [private]
- 5.340.3.39 **float timoConfig::GLRexp** [private]
- 5.340.3.40 **MHASignal::waveform_t timoConfig::GLR** [private]

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

- **timoconfig.h**
- **timoconfig.cpp**

5.341 timoSmooth Class Reference

Inheritance diagram for timoSmooth:



Public Member Functions

- **timoSmooth** (**algo_comm_t** &ac, const std::string &chain_name, const std::string &algo_name)
Constructs the beamforming plugin.
- **~timoSmooth** ()
- **mha_spec_t * process** (**mha_spec_t ***)
This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.
- void **prepare** (**mhaconfig_t** &)
Plugin preparation.
- void **release** (void)

Private Member Functions

- void **update_cfg** ()
- void **on_model_param_valuechanged** ()

Private Attributes

- **MHAParser::float_t xi_min_db**
- **MHAParser::float_t f0_low**
- **MHAParser::float_t f0_high**
- **MHAParser::float_t delta_pitch**
- **MHAParser::float_t lambda_thresh**
- **MHAParser::float_t alpha_pitch**
- **MHAParser::float_t beta_const**
- **MHAParser::float_t kappa_const**
- **MHAParser::float_t gain_min_db**
- **MHAParser::vfloat_t win_f0**
- **MHAParser::vfloat_t alpha_const_vals**
- **MHAParser::vfloat_t alpha_const_limits_hz**
- **MHAParser::string_t noisePow_name**
- **MHAParser::parser_t spp**
- **MHAParser::float_t prior_q**
- **MHAParser::float_t xi_opt_db**
- **MHAEvents::patchbay_t< timoSmooth > patchbay**
- **bool prepared**

Additional Inherited Members

5.341.1 Constructor & Destructor Documentation

5.341.1.1 **timoSmooth::timoSmooth (**
 algo_comm_t & ac,
 const std::string & chain_name,
 const std::string & algo_name)

Constructs the beamforming plugin.

5.341.1.2 **timoSmooth::~~timoSmooth ()**

5.341.2 Member Function Documentation

5.341.2.1 **mha_spec_t * timoSmooth::process (**
 mha_spec_t * signal)

This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.

Parameters

| | |
|---------------|--|
| <i>signal</i> | Pointer to the input signal structure. |
|---------------|--|

Returns

Returns a pointer to the input signal structure, with a the signal modified by this plugin.

5.341.2.2 `void timoSmooth::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the spectral domain and contains at least one channel

Parameters

| | |
|--------------------|--|
| <i>signal_info</i> | Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate. |
|--------------------|--|

Implements **MHAPLugin::plugin_t< timoConfig >** (p. 661).

5.341.2.3 `void timoSmooth::release (`
`void) [inline], [virtual]`

Reimplemented from **MHAPLugin::plugin_t< timoConfig >** (p. 661).

5.341.2.4 `void timoSmooth::update_cfg (`
`void) [private]`

5.341.2.5 `void timoSmooth::on_model_param_valuechanged () [private]`

5.341.3 Member Data Documentation

5.341.3.1 `MHAParser::float_t timoSmooth::xi_min_db [private]`

5.341.3.2 `MHAParser::float_t timoSmooth::f0_low [private]`

5.341.3.3 `MHAParser::float_t timoSmooth::f0_high [private]`

5.341.3.4 `MHAParser::float_t timoSmooth::delta_pitch [private]`

5.341.3.5 `MHAParser::float_t timoSmooth::lambda_thresh [private]`

5.341.3.6 `MHAParser::float_t timoSmooth::alpha_pitch [private]`

5.341.3.7 `MHAParser::float_t timoSmooth::beta_const [private]`

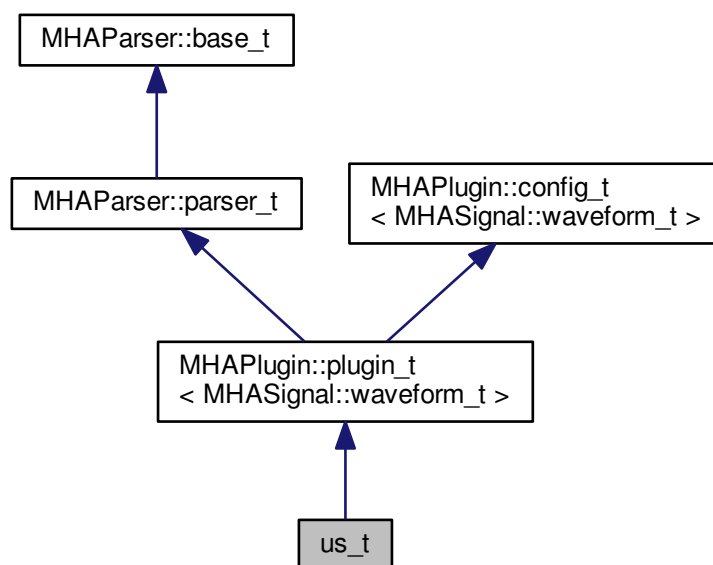
- 5.341.3.8 **MHAParser::float_t timoSmooth::kappa_const** [private]
- 5.341.3.9 **MHAParser::float_t timoSmooth::gain_min_db** [private]
- 5.341.3.10 **MHAParser::vfloat_t timoSmooth::win_f0** [private]
- 5.341.3.11 **MHAParser::vfloat_t timoSmooth::alpha_const_vals** [private]
- 5.341.3.12 **MHAParser::vfloat_t timoSmooth::alpha_const_limits_hz** [private]
- 5.341.3.13 **MHAParser::string_t timoSmooth::noisePow_name** [private]
- 5.341.3.14 **MHAParser::parser_t timoSmooth::spp** [private]
- 5.341.3.15 **MHAParser::float_t timoSmooth::prior_q** [private]
- 5.341.3.16 **MHAParser::float_t timoSmooth::xi_opt_db** [private]
- 5.341.3.17 **MHAEvents::patchbay_t<timoSmooth> timoSmooth::patchbay** [private]
- 5.341.3.18 **bool timoSmooth::prepared** [private]

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

- **timosmooth.h**
- **timoSmooth.cpp**

5.342 us_t Class Reference

Inheritance diagram for us_t:



Public Member Functions

- **us_t** (**algo_comm_t**, std::string, std::string)
- **mha_wave_t * process** (**mha_wave_t ***)
- void **prepare** (**mhaconfig_t** &)
- void **release** ()

Private Attributes

- **MHAParser::int_t ratio**
- **MHAFilter::iir_filter_t antialias**

Additional Inherited Members

5.342.1 Constructor & Destructor Documentation

5.342.1.1 **us_t::us_t** (
 algo_comm_t iac,
 std::string ,
 std::string)

5.342.2 Member Function Documentation

5.342.2.1 **mha_wave_t * us_t::process** (
 mha_wave_t * s)

5.342.2.2 void **us_t::prepare** (
 mhaconfig_t & cf) [virtual]

Implements **MHAPLugin::plugin_t< MHASignal::waveform_t >** (p. 661).

5.342.2.3 void **us_t::release** (
 void) [virtual]

Reimplemented from **MHAPLugin::plugin_t< MHASignal::waveform_t >** (p. 661).

5.342.3 Member Data Documentation

5.342.3.1 **MHAParser::int_t us_t::ratio** [private]

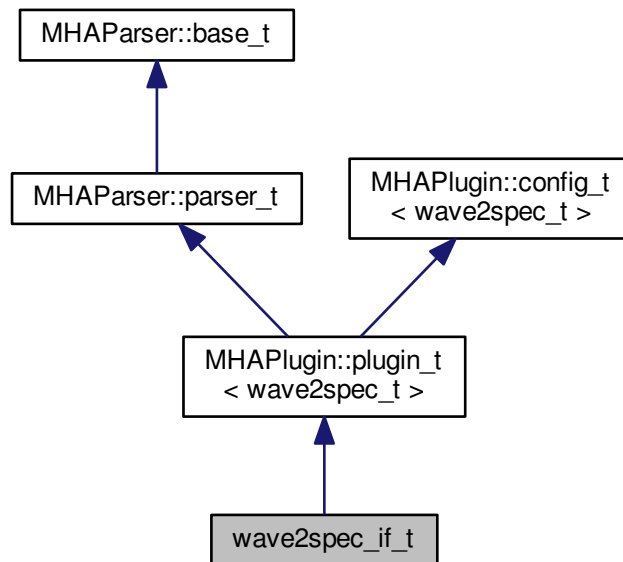
5.342.3.2 **MHAFilter::iir_filter_t us_t::antialias** [private]

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

- **upsample.cpp**

5.343 wave2spec_if_t Class Reference

Inheritance diagram for wave2spec_if_t:



Public Member Functions

- **wave2spec_if_t** (const **algo_comm_t** &, const std::string &, const std::string &)
- void **prepare** (**mhaconfig_t** &)
- void **process** (**mha_wave_t** *, **mha_spec_t** **)
- void **process** (**mha_wave_t** *, **mha_wave_t** **)

Private Member Functions

- void **update** ()

Private Attributes

- **MHAEvents::patchbay_t** < **wave2spec_if_t** > **patchbay**
- **MHAParser::int_t** **nfft**
- **MHAParser::int_t** **nwnd**
- **MHAParser::float_t** **wndpos**
- **windowselector_t** **window_config**
- **MHAParser::bool_t** **return_wave**
- std::string **algo**

Additional Inherited Members

5.343.1 Constructor & Destructor Documentation

5.343.1.1 `wave2spec_if_t::wave2spec_if_t (`
 `const algo_comm_t & iac,`
 `const std::string &`
 `const std::string & ialg)`

5.343.2 Member Function Documentation

5.343.2.1 `void wave2spec_if_t::prepare (`
 `mhaconfig_t & t) [virtual]`

Implements **MHAPugin::plugin_t< wave2spec_t >** (p. [661](#)).

5.343.2.2 `void wave2spec_if_t::process (`
 `mha_wave_t * wave_in,`
 `mha_spec_t ** sout)`

5.343.2.3 `void wave2spec_if_t::process (`
 `mha_wave_t * wave_in,`
 `mha_wave_t ** sout)`

5.343.2.4 `void wave2spec_if_t::update () [private]`

5.343.3 Member Data Documentation

5.343.3.1 `MHAEvents::patchbay_t<wave2spec_if_t> wave2spec_if_t::patchbay [private]`

5.343.3.2 `MHAParser::int_t wave2spec_if_t::nfft [private]`

5.343.3.3 `MHAParser::int_t wave2spec_if_t::nwnd [private]`

5.343.3.4 `MHAParser::float_t wave2spec_if_t::wndpos [private]`

5.343.3.5 `windowselector_t wave2spec_if_t::window_config [private]`

5.343.3.6 `MHAParser::bool_t wave2spec_if_t::return_wave [private]`

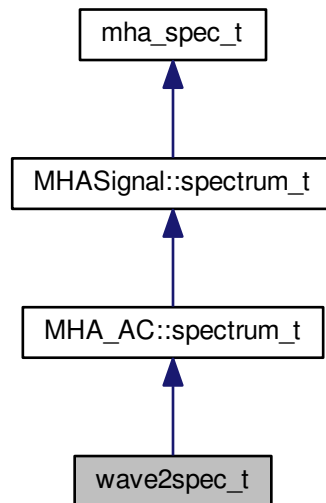
5.343.3.7 `std::string wave2spec_if_t::algo [private]`

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

- **wave2spec.cpp**

5.344 wave2spec_t Class Reference

Inheritance diagram for wave2spec_t:



Public Member Functions

- **wave2spec_t** (unsigned int nfft, unsigned int nwnd_, unsigned int nwndshift_, unsigned int nch, **mha_real_t** wndpos, const **MHAWindow::base_t** &window, **algo_comm_t** ac, std::string algo)
- **mha_spec_t** * **process** (**mha_wave_t** *)
- ~**wave2spec_t** ()

Private Member Functions

- void **calc_pre_wnd** (**MHASignal::waveform_t** &, const **MHASignal::waveform_t** &)

Private Attributes

- unsigned int **nwnd**
- unsigned int **nwndshift**
- **mha_fft_t** **ft**
FFT class.
- unsigned int **npad1**
length of zero padding before window
- unsigned int **npad2**

length of zero padding after window

- **MHAWindow::base_t window**
- **MHASignal::waveform_t calc_in**
- **MHASignal::waveform_t in_buf**
- **MHASignal::spectrum_t spec_in**

non-interleaved, complex, fftlen

Additional Inherited Members

5.344.1 Constructor & Destructor Documentation

5.344.1.1 **wave2spec_t::wave2spec_t (**
 unsigned int *nfft*,
 unsigned int *nwnd*,
 unsigned int *nwndshift*,
 unsigned int *nch*,
 mha_real_t *wndpos*,
 const MHAWindow::base_t & *window*,
 algo_comm_t *ac*,
 std::string *algo*)

5.344.1.2 **wave2spec_t::~~wave2spec_t ()**

5.344.2 Member Function Documentation

5.344.2.1 **mha_spec_t * wave2spec_t::process (**
 mha_wave_t * *wave_in*)

5.344.2.2 **void wave2spec_t::calc_pre_wnd (**
 MHASignal::waveform_t & *dest*,
 const MHASignal::waveform_t & *src*) [private]

5.344.3 Member Data Documentation

5.344.3.1 **unsigned int wave2spec_t::nwnd** [private]

5.344.3.2 **unsigned int wave2spec_t::nwndshift** [private]

5.344.3.3 **mha_fft_t wave2spec_t::ft** [private]

FFT class.

5.344.3.4 **unsigned int wave2spec_t::npad1** [private]

length of zero padding before window

5.344.3.5 unsigned int wave2spec_t::npad2 [private]

length of zero padding after window

5.344.3.6 MHASignal::base_t wave2spec_t::window [private]

5.344.3.7 MHASignal::waveform_t wave2spec_t::calc_in [private]

5.344.3.8 MHASignal::waveform_t wave2spec_t::in_buf [private]

5.344.3.9 MHASignal::spectrum_t wave2spec_t::spec_in [private]

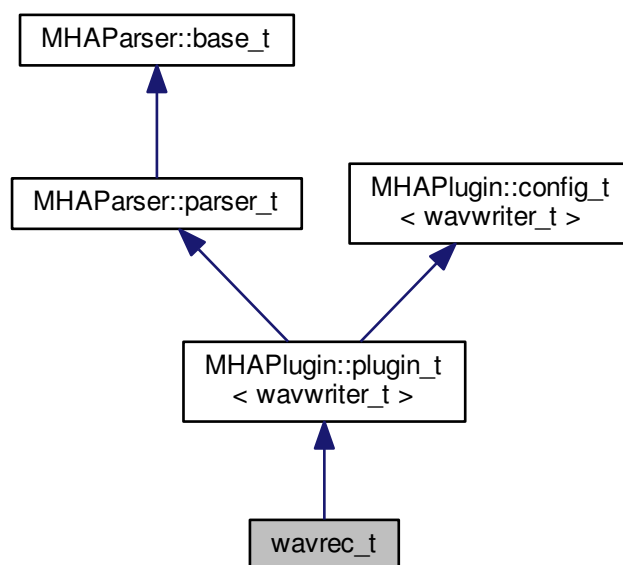
non-interleaved, complex, fftlen

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

- wave2spec.cpp

5.345 wavrec_t Class Reference

Inheritance diagram for wavrec_t:



Public Member Functions

- **mha_wave_t * process** (mha_wave_t *)
- void **prepare** (mhaconfig_t &cf)
- void **release** ()
- **wavrec_t** (const **algo_comm_t** &iac, const std::string &, const std::string &)

Private Member Functions

- void **start_new_session** ()

Private Attributes

- **MHAParser::bool_t** record
- **MHAParser::int_t** fifolen
- **MHAParser::int_t** minwrite
- **MHAParser::string_t** prefix
- **MHAParser::bool_t** use_date
- **MHAEvents::patchbay_t**< wavrec_t > patchbay

Additional Inherited Members

5.345.1 Constructor & Destructor Documentation

5.345.1.1 wavrec_t::wavrec_t (
 const **algo_comm_t** & iac,
 const std::string & ,
 const std::string & *algo_name*)

5.345.2 Member Function Documentation

5.345.2.1 mha_wave_t * wavrec_t::process (
 mha_wave_t * s)

5.345.2.2 void wavrec_t::prepare (
 mhaconfig_t & cf) [virtual]

Implements **MHAPlugin::plugin_t**< **wavwriter_t** > (p. 661).

5.345.2.3 void wavrec_t::release (
 void) [virtual]

Reimplemented from **MHAPlugin::plugin_t**< **wavwriter_t** > (p. 661).

5.345.2.4 void wavrec_t::start_new_session () [private]

5.345.3 Member Data Documentation

5.345.3.1 MHAParser::bool_t wavrec_t::record [private]

5.345.3.2 MHAParser::int_t wavrec_t::fifolen [private]

5.345.3.3 MHAParser::int_t wavrec_t::minwrite [private]

5.345.3.4 MHAParser::string_t wavrec_t::prefix [private]

5.345.3.5 MHAParser::bool_t wavrec_t::use_date [private]

5.345.3.6 MHAEvents::patchbay_t<wavrec_t> wavrec_t::patchbay [private]

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

- **wavrec.cpp**

5.346 wavwriter_t Class Reference

Public Member Functions

- **wavwriter_t** (bool active, const **mhaconfig_t** &cf, unsigned int fifosize, unsigned int minwrite, const std::string &prefix, bool use_date)
- **~wavwriter_t** ()
- void **process** (**mha_wave_t** *)

Private Member Functions

- void **write_thread** ()

Static Private Member Functions

- static void * **write_thread** (void *this_)

Private Attributes

- bool **close_session**
- bool **act_**
- **mhaconfig_t** **cf_**
- SNDFILE * **sf**
- **mha_fifo_t**< **mha_real_t** > **fifo**
- unsigned int **minw_**
- pthread_t **writethread**
- float * **data**

5.346.1 Constructor & Destructor Documentation

5.346.1.1 `wavwriter_t::wavwriter_t (`
 bool active,
 const **mhaconfig_t** & *cf,*
 unsigned int *fifosize,*
 unsigned int *minwrite,*
 const std::string & *prefix,*
 bool *use_date*)

5.346.1.2 `wavwriter_t::~~wavwriter_t ()`

5.346.2 Member Function Documentation

5.346.2.1 `void wavwriter_t::process (`
 mha_wave_t * *s*)

5.346.2.2 `static void* wavwriter_t::write_thread (`
 void * *this_*) [inline], [static], [private]

5.346.2.3 `void wavwriter_t::write_thread ()` [private]

5.346.3 Member Data Documentation

5.346.3.1 `bool wavwriter_t::close_session` [private]

5.346.3.2 `bool wavwriter_t::act_` [private]

5.346.3.3 `mhaconfig_t wavwriter_t::cf_` [private]

5.346.3.4 `SNDFILE* wavwriter_t::sf` [private]

5.346.3.5 `mha_fifo_t<mha_real_t> wavwriter_t::fifo` [private]

5.346.3.6 `unsigned int wavwriter_t::minw_` [private]

5.346.3.7 `pthread_t wavwriter_t::writethread` [private]

5.346.3.8 `float* wavwriter_t::data` [private]

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

- **wavrec.cpp**

5.347 windowselector_t Class Reference

A combination of mha parser variables to describe an overlapadd analysis window.

Public Member Functions

- **windowselector_t** (const std::string &default_type)
constructor creates the mha parser variables that describe an overlapadd analysis window.
- **~windowselector_t** ()
destructor frees window data that were allocated
- const **MHAWindow::base_t** & **get_window_data** (unsigned length)
re-computes the window if required.
- void **insert_items** (MHAParser::parser_t *p)
insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.

Public Attributes

- **MHAEvents::emitter_t** **updated**
A collector event that fires when any of the window parameters managed here is written to.

Private Member Functions

- void **invalidate_window_data** ()
invalidates any allocated window samples.
- void **update_parser** ()
invoked when a parser parameter changes.

Private Attributes

- **MHAWindow::base_t** * **wnd**
*Storage for the window data returned by **get_window_data()** (p. 876)*
- **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

5.347.1 Detailed Description

A combination of mha parser variables to describe an overlapadd analysis window.

Provides a method to get the window samples as an instance of **MHAWindow::base_t** (p. 766) when needed.

5.347.2 Constructor & Destructor Documentation

5.347.2.1 windowselector_t::windowselector_t (const std::string & *default_type*)

constructor creates the mha parser variables that describe an overlapadd analysis window.

Parameters

| | |
|---------------------|--|
| <i>default_type</i> | name of the default analysis window type. Must be one of: "rect", "bartlett", "hanning", "hamming", "blackman" |
|---------------------|--|

5.347.2.2 windowselector_t::~~windowselector_t ()

destructor frees window data that were allocated

5.347.3 Member Function Documentation

5.347.3.1 const MHAWindow::base_t & windowselector_t::get_window_data (unsigned *length*)

re-computes the window if required.

Parameters

| | |
|---------------|--|
| <i>length</i> | the desired window length in samples return the window's samples as a constref to MHAWindow::base_t (p. 766) instance. The referenced instance lives until the window parameters are changed, or this windowselector_t (p. 875) instance is destroyed. |
|---------------|--|

5.347.3.2 void windowselector_t::insert_items (MHAParser::parser_t * *p*)

insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.

Parameters

| | |
|----------|--|
| <i>p</i> | The configuration parser where to insert the window parameters. E.g. the plugin wave2spec's interface class. |
|----------|--|

5.347.3.3 void windowselector_t::invalidate_window_data () [private]

invalidates any allocated window samples.

5.347.3.4 void windowselector_t::update_parser () [private]

invoked when a parser parameter changes.

Calls **invalidate_window_data()** (p. 877) and emits the updated event.

5.347.4 Member Data Documentation

5.347.4.1 MHAEvents::emitter_t windowselector_t::updated

A collector event that fires when any of the window parameters managed here is written to.

5.347.4.2 MHAWindow::base_t* windowselector_t::wnd [private]

Storage for the window data returned by **get_window_data()** (p. 876)

5.347.4.3 MHAParser::kw_t windowselector_t::wndtype [private]

parser variable for window type

5.347.4.4 MHAParser::float_t windowselector_t::wndexp [private]

parser variable for window exponent

5.347.4.5 MHAParser::vfloat_t windowselector_t::userwnd [private]

parser variable for user window samples to use

5.347.4.6 MHAEvents::patchbay_t<windowselector_t> windowselector_t::patchbay [private]

patchbay to watch for changes for the parser variables

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

- **windowselector.h**
- **windowselector.cpp**

6 File Documentation

6.1 ac2wave.cpp File Reference

Classes

- class **ac2wave_t**
- class **ac2wave_if_t**

6.2 ac_monitor_type.cpp File Reference

6.3 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.

6.4 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)

6.4.1 Macro Definition Documentation

6.4.1.1 **#define PATCH_VAR(**
 var) patchbay.connect(&var.valuechanged, this, &acConcat_wave::update↔
 _cfg)

6.4.1.2 **#define INSERT_PATCH(**
 var) insert_member(var); PATCH_VAR(var)

6.5 acConcat_wave.h File Reference

Classes

- class **acConcat_wave_config**
- class **acConcat_wave**

6.6 acmon.cpp File Reference

Classes

- class **acmon::acmon_t**

Namespaces

- **acmon**
Namespace for displaying ac variables as parser monitors.

6.7 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)

6.7.1 Macro Definition Documentation

6.7.1.1 #define **PATCH_VAR**(
 var) patchbay.connect(&var.valuechanged, this, &**acPooling_wave::update**↵
 _cfg)

6.7.1.2 #define **INSERT_PATCH**(
 var) **insert_member**(var); **PATCH_VAR**(var)

6.8 acPooling_wave.h File Reference

Classes

- class **acPooling_wave_config**
- class **acPooling_wave**

6.9 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"**

6.9.1 Macro Definition Documentation

6.9.1.1 #define ACSAVE_FMT_TXT 0

6.9.1.2 #define ACSAVE_SFMT_TXT "txt"

6.9.1.3 #define ACSAVE_FMT_MAT4 1

6.9.1.4 #define ACSAVE_SFMT_MAT4 "mat4"

6.9.1.5 #define ACSAVE_FMT_M 2

6.9.1.6 #define ACSAVE_SFMT_M "m"

6.10 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)**

6.10.1 Macro Definition Documentation

6.10.1.1 #define PATCH_VAR(var) patchbay.connect(&var.valuechanged, this, &acSteer::update_cfg)

6.10.1.2 #define INSERT_PATCH(var) insert_member(var); PATCH_VAR(var)

6.11 acSteer.h File Reference

Classes

- class **acSteer_config**
- class **acSteer**

6.12 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)`

6.12.1 Macro Definition Documentation

6.12.1.1 `#define PATCH_VAR(
var) patchbay.connect(&var.valuechanged, this, &acTransform_wave
::update_cfg)`

6.12.1.2 `#define INSERT_PATCH(
var) insert_member(var); PATCH_VAR(var)`

6.13 acTransform_wave.h File Reference

Classes

- class `acTransform_wave_config`
- class `acTransform_wave`

6.14 adm.cpp File Reference

Classes

- class `adm_rtconfig_t`
- class `adm_if_t`

Functions

- `MHSignal::waveform_t * adm_fir_lp` (unsigned int fs, unsigned f_pass, unsigned int f_stop, unsigned int order)
- `MHSignal::waveform_t * adm_fir_decomb` (unsigned int fs, float dist_m, unsigned int order)

6.14.1 Function Documentation

6.14.1.1 **MHASignal::waveform_t*** **adm_fir_lp** (
 unsigned int *fs*,
 unsigned *f_pass*,
 unsigned int *f_stop*,
 unsigned int *order*)

6.14.1.2 **MHASignal::waveform_t*** **adm_fir_decomb** (
 unsigned int *fs*,
 float *dist_m*,
 unsigned int *order*)

6.15 adm.hh File Reference

Classes

- class **ADM::Linearphase_FIR**< **F** >
An efficient linear-phase fir filter implementation.
- class **ADM::Delay**< **F** >
A delay-line class which can also do subsample-delays for a limited frequency range below fs/4.
- class **ADM::ADM**< **F** >
Adaptive differential microphone, working for speech frequency range.

Namespaces

- **ADM**

Functions

- static double **ADM::subsampledelay_coeff** (double *samples*, double *f_design*, double *fs*=1.0)
compute IIR coefficient for subsample delay

Variables

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

6.16 altplugins.cpp File Reference

Classes

- class **mhaplug_cfg_t**
- class **altplugins_t**

Macros

- **#define MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.16.1 Macro Definition Documentation

6.16.1.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

6.17 analysemhaplugin.cpp File Reference

Functions

- **std::string strdom** (**mha_domain_t d**)
- **void print_ac** (**MHAKernel::algo_comm_class_t &ac**, **std::string txt**)
- **int main** (**int argc**, **char **argv**)

6.17.1 Function Documentation

6.17.1.1 **std::string strdom** (**mha_domain_t d**)

6.17.1.2 **void print_ac** (**MHAKernel::algo_comm_class_t &ac**, **std::string txt**)

6.17.1.3 **int main** (**int argc**, **char ** argv**)

6.18 analysispath.cpp File Reference

Classes

- class **analysepath_t**
- class **plug_t**
- class **analysispath_if_t**

Functions

- static void * **thread_start** (void *instance)

6.18.1 Function Documentation

6.18.1.1 static void* thread_start (
void * *instance*) [static]

6.19 auditory_profile.cpp File Reference

6.20 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)

6.21 browsemhaplugins.cpp File Reference

Macros

- **#define DEBUG(x)** std::cerr << __FILE__ << ":" << __LINE__ << " " << #x<<"="<<x
<< std::endl

Functions

- int **main** (int argc, char **argv)

6.21.1 Macro Definition Documentation

```
6.21.1.1 #define DEBUG(
                x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x <<
                std::endl
```

6.21.2 Function Documentation

```
6.21.2.1 int main (
                int argc,
                char ** argv )
```

6.22 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)

6.23 combinechannels.cpp File Reference

Classes

- class **combc_t**
- class **combc_if_t**

6.24 complex_filter.cpp File Reference

6.25 complex_filter.h File Reference

Classes

- class **MHAFilter::complex_bandpass_t**
Complex bandpass filter.
- class **MHAFilter::gammaflt_t**
Class for gammatone filter.
- class **MHAFilter::thirdoctave_analyzer_t**

Namespaces

- **MHAFilter**

Namespace for IIR and FIR filter classes.

6.26 cpuload.cpp File Reference

Classes

- class **cpuload_t**

6.27 db.cpp File Reference

Classes

- class **db_t**
- class **db_if_t**

6.28 dc.cpp File Reference

Classes

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

Namespaces

- **dc**

Macros

- **#define DUPVEC(x)** v.x.data = **MHASignal::dupvec_chk**(v.x.data,s)

Functions

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

6.28.1 Macro Definition Documentation

6.28.1.1 #define DUPVEC(
x) v.x.data = MHASignal::dupvec_chk(v.x.data,s)

6.29 dc_afterburn.cpp File Reference

Namespaces

- **DynComp**
dynamic compression related classes and functions

Functions

- float **mylogf** (float x)

6.29.1 Function Documentation

6.29.1.1 float mylogf (
float x)

6.30 dc_afterburn.h File Reference

Classes

- class **DynComp::dc_afterburn_vars_t**
*Variables for **dc_afterburn_t** (p. 257) 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

6.31 dc_simple.cpp File Reference

Classes

- class **dc_simple::dc_vars_t**
- class **dc_simple::dc_vars_validator_t**
- class **dc_simple::level_smoother_t**
- class **dc_simple::dc_t**
- class **dc_simple::dc_t::line_t**
- class **dc_simple::dc_if_t**

Namespaces

- **dc_simple**

Typedefs

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

Functions

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

6.32 delay.cpp File Reference

Classes

- class **delay::interface_t**

Namespaces

- **delay**

6.33 delaysum.cpp File Reference

Classes

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

Namespaces

- **delaysum**
This namespace contains the delaysum plugin.

6.34 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)**

6.34.1 Macro Definition Documentation

6.34.1.1 #define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, &**doasvm_classification::update_cfg**)

6.34.1.2 #define INSERT_PATCH(
 var) insert_member(var); **PATCH_VAR(var)**

6.35 doasvm_classification.h File Reference

Classes

- class **doasvm_classification_config**
- class **doasvm_classification**

6.36 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)**

6.36.1 Macro Definition Documentation

6.36.1.1 `#define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, &doasvm_feature_↵
 extraction::update_cfg)`

6.36.1.2 `#define INSERT_PATCH(
 var) insert_member(var); PATCH_VAR(var)`

6.37 doasvm_feature_extraction.h File Reference

Classes

- class **doasvm_feature_extraction_config**
- class **doasvm_feature_extraction**

6.38 doc_appendix.h File Reference

6.39 doc_examples.h File Reference

6.40 doc_frameworks.h File Reference

6.41 doc_general.h File Reference

6.42 doc_kernel.h File Reference

6.43 doc_matlab.h File Reference

6.44 doc_mhamain.h File Reference

6.45 doc_parser.h File Reference

6.46 doc_pluginif.cpp File Reference

6.47 doc_plugins.h File Reference

6.48 doc_system.h File Reference

6.49 doc_toolbox.h File Reference

6.50 downsample.cpp File Reference

Classes

- class **ds_t**

6.51 droptect.cpp File Reference

Classes

- class **droptect_t**
Detect dropouts in a signal with a constant spectrum.

6.52 example1.cpp File Reference

Classes

- class **example1_t**
This C++ class implements the simplest example plugin for the step-by-step tutorial.

6.53 example2.cpp File Reference

Classes

- class **example2_t**
This C++ class implements the second example plugin for the step-by-step tutorial.

6.54 example3.cpp File Reference

Classes

- class **example3_t**
A Plugin class using the openMHA Event mechanism.

6.55 example4.cpp File Reference

Classes

- class **example4_t**
A Plugin class using the spectral signal.

6.56 example5.cpp File Reference

Classes

- class **example5_t**
- class **plugin_interface_t**

Macros

- `#define __declspec(p)`

6.56.1 Macro Definition Documentation

6.56.1.1 `#define __declspec(p)`

6.57 example6.cpp File Reference

Classes

- class `cfg_t`
- class `example6_t`

Macros

- `#define __declspec(p)`

6.57.1 Macro Definition Documentation

6.57.1.1 `#define __declspec(p)`

6.58 fader_spec.cpp File Reference

Classes

- class `spec_fader_t`
- class `fader_if_t`

6.59 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`

6.59.1 Macro Definition Documentation

- 6.59.1.1 `#define DEBUG(
x) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl`

6.60 fftfilterbank.cpp File Reference

Classes

- class `fftfilterbank::fftfb_plug_t`
- class `fftfilterbank::fftfb_interface_t`

Namespaces

- `fftfilterbank`

6.61 fshift_hilbert.cpp File Reference

Classes

- class `hilbert_shifter_t`
- class `frequency_translator_t`

6.62 gain.cpp File Reference

Classes

- class `gain::scaler_t`
- class `gain::gain_if_t`

Namespaces

- **gain**

6.63 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)

6.63.1 Function Documentation

6.63.1.1 `std::vector<mha_real_t> convert_f2logf (`
`const std::vector< mha_real_t > & vF)`

6.63.1.2 `bool isempty (`
`const std::vector< std::vector< mha_real_t > > & arg)`

6.64 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.

6.65 generatemhaplugindoc.cpp File Reference

Classes

- class **latex_doc_t**

Functions

- std::string **conv2latex** (std::string s, bool iscolored=false)
- void **create_latex_doc** (std::map< std::string, std::string > &doc, const std::string &plugname, const std::string &plugin_macro)
- int **main** (int argc, char **argv)

6.65.1 Function Documentation

6.65.1.1 std::string conv2latex (
 std::string s,
 bool *iscolored* = false)

6.65.1.2 void create_latex_doc (
 std::map< std::string, std::string > & doc,
 const std::string & *plugname*,
 const std::string & *plugin_macro*)

6.65.1.3 int main (
 int *argc*,
 char ** *argv*)

6.66 hann.cpp File Reference

Macros

- #define **PI** 3.14159265358979323846

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

6.66.1 Macro Definition Documentation

6.66.1.1 #define PI 3.14159265358979323846

6.66.2 Function Documentation

6.66.2.1 float* hannf (const unsigned int *N*)

6.66.2.2 double* hann (const unsigned int *N*)

6.67 hann.h File Reference

Functions

- float * **hannf** (const unsigned int *N*)
- double * **hann** (const unsigned int *N*)

6.67.1 Function Documentation

6.67.1.1 float* hannf (const unsigned int *N*)

6.67.1.2 double* hann (const unsigned int *N*)

6.68 identity.cpp File Reference

Classes

- class **identity_t**

6.69 ifftshift.cpp File Reference

Functions

- void **ifftshift** (**mha_wave_t** *spec)

6.69.1 Function Documentation

6.69.1.1 void ifftshift (
 mha_wave_t * spec)

6.70 ifftshift.h File Reference

Functions

- void **ifftshift** (mha_wave_t *spec)

6.70.1 Function Documentation

6.70.1.1 void ifftshift (
 mha_wave_t * spec)

6.71 iirfilter.cpp File Reference

Classes

- class **iirfilter_t**

6.72 ipc.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, &ipc::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)

6.72.1 Macro Definition Documentation

6.72.1.1 `#define PATCH_VAR(
var) patchbay.connect(&var.valuechanged, this, &lpc::update_cfg)`

6.72.1.2 `#define INSERT_PATCH(
var) insert_member(var); PATCH_VAR(var)`

6.72.2 Function Documentation

6.72.2.1 `void Levinson2 (
unsigned int P,
const std::vector< mha_real_t > & R,
std::vector< mha_real_t > & A)`

6.73 lpc.h File Reference

Classes

- class **lpc_config**
- class **lpc**

6.74 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)`

6.74.1 Macro Definition Documentation

6.74.1.1 `#define PATCH_VAR(
var) patchbay.connect(&var.valuechanged, this, &lpc_bl_predictor↔
::update_cfg)`

6.74.1.2 `#define INSERT_PATCH(
var) insert_member(var); PATCH_VAR(var)`

6.75 lpc_bl_predictor.h File Reference

Classes

- class **lpc_bl_predictor_config**
- class **lpc_bl_predictor**

Macros

- `#define EPSILON 1e-10`

6.75.1 Macro Definition Documentation

6.75.1.1 `#define EPSILON 1e-10`

6.76 `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)`

6.76.1 Macro Definition Documentation

6.76.1.1 `#define PATCH_VAR(var) patchbay.connect(&var.valuechanged, this, &lpc_burglattice::update_cfg)`

6.76.1.2 `#define INSERT_PATCH(var) insert_member(var); PATCH_VAR(var)`

6.77 `lpc_burg-lattice.h` File Reference

Classes

- class `lpc_burglattice_config`
- class `lpc_burglattice`

Macros

- `#define EPSILON 1e-10`

6.77.1 Macro Definition Documentation

6.77.1.1 `#define EPSILON 1e-10`

6.78 `matrixmixer.cpp` File Reference

Classes

- class `matrixmixer::cfg_t`
- class `matrixmixer::matmix_t`

Namespaces

- **matrixmixer**

6.79 mha.cpp File Reference

Functions

- int **mhamain** (int argc, char *argv[])
- int **main** (int argc, char *argv[])

6.79.1 Function Documentation

6.79.1.1 int mhamain (
 int *argc*,
 char * *argv*[])

6.79.1.2 int main (
 int *argc*,
 char * *argv*[])

6.80 mha.h 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_direction_t**
Channel source direction structure.
- struct **mha_channel_info_t**
Channel information structure.
- struct **mha_wave_t**
Waveform signal structure.
- struct **mha_spec_t**
Spectrum signal structure.
- struct **mha_audio_descriptor_t**
*Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 444)).*
- struct **mha_audio_t**
*An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 436) and **mha_spec_t** (p. 406)).*
- struct **mhaconfig_t**
MHA prepare configuration structure.
- struct **comm_var_t**
Algorithm communication variable structure.
- struct **algo_comm_t**
A reference handle for algorithm communication variables.

Macros

- **#define MHA_CALLBACK_TEST(x)**
Test macro to compare function type definition and declaration.
- **#define MHA_CALLBACK_TEST_PREFIX(prefix, x)**
- **#define MHA_XSTRF(x) MHA_STRF(x)**
- **#define MHA_STRF(x) #x**
- **#define MHA_VERSION_MAJOR 4**
Major version number of MHA.
- **#define MHA_VERSION_MINOR 5**
Minor version number of MHA.
- **#define MHA_VERSION_RELEASE 6**
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_VERS↵
ION_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_RELE↵
ASE)**
Version string of MHA kernel (major.minor.release)
- **#define MHA_WAVEFORM 0**
- **#define MHA_SPECTRUM 1**
- **#define MHA_DOMAIN_MAX 2**
- **#define MHA_DOMAIN_UNKNOWN MHA_DOMAIN_MAX**
- **#define MHA_AC_UNKNOWN 0**
- **#define MHA_AC_CHAR 1**
- **#define MHA_AC_INT 2**
- **#define MHA_AC_MHAREAL 3**
- **#define MHA_AC_FLOAT 4**
- **#define MHA_AC_DOUBLE 5**
- **#define MHA_AC_MHACOMPLEX 6**
- **#define MHA_AC_VEC_FLOAT 51**
- **#define MHA_AC_USER 1000**

Typedefs

- typedef unsigned int **mha_domain_t**
- typedef float **mha_real_t**
openMHA type for real numbers
- typedef void * **mha_fft_t**
Handle for an FFT object.
- typedef struct **algo_comm_t** **algo_comm_t**
- typedef unsigned int(* **MHAGetVersion_t**) (void)
- typedef int(* **MHAInit_t**) (**algo_comm_t** algo_comm, const char *chain, const char *algo, void **h)
- typedef int(* **MHAPrepare_t**) (void *h, **mhaconfig_t** *cfg)
- typedef int(* **MHARelease_t**) (void *h)
- typedef void(* **MHADestroy_t**) (void *h)
- typedef int(* **MHASET_t**) (void *h, const char *cmd, char *retval, unsigned int len)
- typedef const char *(**MHAStrError_t**) (void *h, int err)
- typedef int(* **MHAProc_wave2wave_t**) (void *h, **mha_wave_t** *sIn, **mha_wave_t** **sOut)
- typedef int(* **MHAProc_wave2spec_t**) (void *h, **mha_wave_t** *sIn, **mha_spec_t** **sOut)
- typedef int(* **MHAProc_spec2wave_t**) (void *h, **mha_spec_t** *sIn, **mha_wave_t** **sOut)
- typedef int(* **MHAProc_spec2spec_t**) (void *h, **mha_spec_t** *sIn, **mha_spec_t** **sOut)
- typedef const char *(**MHAPluginDocumentation_t**) (void)
- typedef const char *(**MHAPluginCategory_t**) (void)

6.80.1 Detailed Description

common types for MHA kernel, MHA framework applications and external plugins

6.80.2 Macro Definition Documentation

6.80.2.1 #define MHA_CALLBACK_TEST(x)

Test macro to compare function type definition and declaration.

6.80.2.2 #define MHA_CALLBACK_TEST_PREFIX(prefix, x)

6.80.2.3 #define MHA_XSTRF(x) MHA_STRF(x)

6.80.2.4 #define MHA_STRF(x) #x

6.80.2.5 #define MHA_VERSION_MAJOR 4

Major version number of MHA.

6.80.2.6 #define MHA_VERSION_MINOR 5

Minor version number of MHA.

6.80.2.7 #define MHA_VERSION_RELEASE 6

Release number of MHA.

6.80.2.8 #define MHA_VERSION_BUILD 0

Build number of MHA (currently unused)

**6.80.2.9 #define MHA_STRUCT_SIZEMATCH (unsigned int)((sizeof(mha_real_t)==4)+2*(sizeof(mha_↵
_complex_t)==8)+4*(sizeof(mha_wave_t)==8+2*sizeof(void*)))+8*(sizeof(mha_spec_↵
t)==8+2*sizeof(void*)))+16*(sizeof(mhaconfig_t)==24))**

Test number for structure sizes.

**6.80.2.10 #define MHA_VERSION (unsigned int)((MHA_STRUCT_SIZEMATCH |
(MHA_VERSION_RELEASE << 8) | (MHA_VERSION_MINOR << 16) |
(MHA_VERSION_MAJOR << 24)))**

Full version number of MHA kernel.

**6.80.2.11 #define MHA_VERSION_STRING MHA_XSTRF(MHA_VERSION_MAJOR) "."
MHA_XSTRF(MHA_VERSION_MINOR)**

Version string of MHA kernel (major.minor)

**6.80.2.12 #define MHA_RELEASE_VERSION_STRING MHA_XSTRF(MHA_VE↵
RSION_MAJOR) "." MHA_XSTRF(MHA_VERSION_MINOR) "."
MHA_XSTRF(MHA_VERSION_RELEASE)**

Version string of MHA kernel (major.minor.release)

6.80.2.13 `#define MHA_WAVEFORM 0`

6.80.2.14 `#define MHA_SPECTRUM 1`

6.80.2.15 `#define MHA_DOMAIN_MAX 2`

6.80.2.16 `#define MHA_DOMAIN_UNKNOWN MHA_DOMAIN_MAX`

6.80.2.17 `#define MHA_AC_UNKNOWN 0`

6.80.2.18 `#define MHA_AC_CHAR 1`

6.80.2.19 `#define MHA_AC_INT 2`

6.80.2.20 `#define MHA_AC_MHAREAL 3`

6.80.2.21 `#define MHA_AC_FLOAT 4`

6.80.2.22 `#define MHA_AC_DOUBLE 5`

6.80.2.23 `#define MHA_AC_MHACOMPLEX 6`

6.80.2.24 `#define MHA_AC_VEC_FLOAT 51`

6.80.2.25 `#define MHA_AC_USER 1000`

6.80.3 Typedef Documentation

6.80.3.1 `typedef unsigned int mha_domain_t`

6.80.3.2 `typedef struct algo_comm_t algo_comm_t`

6.80.3.3 `typedef unsigned int(* MHAGetVersion_t) (void)`

6.80.3.4 `typedef int(* MHAInit_t) (algo_comm_t algo_comm, const char *chain, const char *algo, void **h)`

6.80.3.5 `typedef int(* MHAPrepare_t) (void *h, mhaconfig_t *cfg)`

6.80.3.6 `typedef int(* MHARelease_t) (void *h)`

6.80.3.7 `typedef void(* MHADestroy_t) (void *h)`

- 6.80.3.8 `typedef int(* MHASet_t) (void *h, const char *cmd, char *retval, unsigned int len)`
- 6.80.3.9 `typedef const char*(* MHAStrError_t) (void *h, int err)`
- 6.80.3.10 `typedef int(* MHAProc_wave2wave_t) (void *h, mha_wave_t *sIn, mha_wave_t **sOut)`
- 6.80.3.11 `typedef int(* MHAProc_wave2spec_t) (void *h, mha_wave_t *sIn, mha_spec_t **sOut)`
- 6.80.3.12 `typedef int(* MHAProc_spec2wave_t) (void *h, mha_spec_t *sIn, mha_wave_t **sOut)`
- 6.80.3.13 `typedef int(* MHAProc_spec2spec_t) (void *h, mha_spec_t *sIn, mha_spec_t **sOut)`
- 6.80.3.14 `typedef const char*(* MHAPuginDocumentation_t) (void)`
- 6.80.3.15 `typedef const char*(* MHAPuginCategory_t) (void)`

6.81 mha_algo_comm.cpp File Reference

Macros

- `#define AC_SUCCESS 0`
- `#define AC_INVALID_HANDLE -1`
- `#define AC_INVALID_NAME -2`
- `#define AC_STRING_TRUNCATED -3`
- `#define AC_INVALID_OUTPTR -4`
- `#define AC_TYPE_MISMATCH -5`
- `#define AC_DIM_MISMATCH -6`

Variables

- `algo_comm_t algo_comm_default`

6.81.1 Macro Definition Documentation

- 6.81.1.1 `#define AC_SUCCESS 0`
- 6.81.1.2 `#define AC_INVALID_HANDLE -1`
- 6.81.1.3 `#define AC_INVALID_NAME -2`
- 6.81.1.4 `#define AC_STRING_TRUNCATED -3`
- 6.81.1.5 `#define AC_INVALID_OUTPTR -4`
- 6.81.1.6 `#define AC_TYPE_MISMATCH -5`
- 6.81.1.7 `#define AC_DIM_MISMATCH -6`

6.81.2 Variable Documentation

- 6.81.2.1 `algo_comm_t algo_comm_default`

6.82 mha_algo_comm.h File Reference

Header file for Algorithm Communication.

Classes

- class **MHA_AC::spectrum_t**
*Insert a **MHASignal::spectrum_t** (p. 731) class into the AC space.*
- class **MHA_AC::waveform_t**
*Insert a **MHASignal::waveform_t** (p. 743) class into the AC space.*
- class **MHA_AC::int_t**
Insert a integer variable into the AC space.
- class **MHA_AC::float_t**
Insert a float point variable into the AC space.
- class **MHA_AC::double_t**
Insert a double precision floating point variable into the AC space.
- class **MHA_AC::stat_t**
- class **MHA_AC::ac2matrix_helper_t**
- class **MHA_AC::ac2matrix_t**
Copy AC variable to a matrix.
- class **MHA_AC::acspace2matrix_t**
Copy all or a subset of all numeric AC variables into an array of matrixes.

Namespaces

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

Functions

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

6.82.1 Detailed Description

Header file for Algorithm Communication.

6.83 mha_algo_comm.hh File Reference

Classes

- class **MHAKernel::comm_var_map_t**
- class **MHAKernel::algo_comm_class_t**

Namespaces

- **MHAKernel**

Macros

- **#define ALGO_COMM_ID_STR** "MFVK3jL5rmeus1XtggEI971aXCR/GU7RRehKz4kQtrg="

Functions

- **algo_comm_class_t * MHAKernel::algo_comm_safe_cast** (void *)

Variables

- **algo_comm_t algo_comm_default**

6.83.1 Macro Definition Documentation

6.83.1.1 **#define ALGO_COMM_ID_STR** "MFVK3jL5rmeus1XtggEI971aXCR/GU7RRehKz4kQtrg="

6.83.2 Variable Documentation

6.83.2.1 **algo_comm_t algo_comm_default**

6.84 mha_defs.h File Reference

Preprocessor definitions common to all MHA components.

Macros

- `#define __MHA_FUN__ __FUNC__`
- `#define CHECK_EXPR(x) {if(!(x)){throw MHA_Error(__FILE__, __LINE__, "The expression \"\" #x \"\" is invalid.");}}`
- `#define CHECK_VAR(x) {if(!(x)){throw MHA_Error(__FILE__, __LINE__, "The variable \"\" #x \"\" is not defined.");}}`
- `#define __declspec(p)`
- `#define M_PI 3.14159265358979323846`
Define pi if it is not defined yet.
- `#define MIN(a, b) (((a)<(b))?(a):(b))`
Macro for minimum function.
- `#define MAX(a, b) (((a)>(b))?(a):(b))`
Macro for maximum function.
- `#define MHA_EAR_LEFT 0`
- `#define MHA_EAR_RIGHT 1`
- `#define MHA_EAR_MAX 2`

6.84.1 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.

6.84.2 Macro Definition Documentation

6.84.2.1 `#define __MHA_FUN__ __FUNC__`

6.84.2.2 `#define CHECK_EXPR(x) {if(!(x)){throw MHA_Error(__FILE__, __LINE__, "The expression \"\" #x \"\" is invalid.");}}`

6.84.2.3 `#define CHECK_VAR(x) {if(!(x)){throw MHA_Error(__FILE__, __LINE__, "The variable \"\" #x \"\" is not defined.");}}`

6.84.2.4 `#define __declspec(p)`

6.84.2.5 `#define M_PI 3.14159265358979323846`

Define pi if it is not defined yet.

6.84.2.6 `#define MIN(
 a,
 b) (((a)<(b))?a:(b))`

Macro for minimum function.

6.84.2.7 `#define MAX(
 a,
 b) (((a)>(b))?a:(b))`

Macro for maximum function.

6.84.2.8 `#define MHA_EAR_LEFT 0`

6.84.2.9 `#define MHA_EAR_RIGHT 1`

6.84.2.10 `#define MHA_EAR_MAX 2`

6.85 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]`

6.85.1 Macro Definition Documentation

6.85.1.1 #define STRLEN 0x1000

6.85.2 Function Documentation

6.85.2.1 const char* mha_strerror (int *mhaerrno*)

6.85.2.2 void mha_set_user_error (const char * *str*)

6.85.3 Variable Documentation

6.85.3.1 char next_except_str[STRLEN] = ""

6.85.3.2 const char* cstr_strerror[MHA_ERR_USER]

6.86 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)

6.86.1 Macro Definition Documentation

6.86.1.1 `#define MHA_ERR_SUCCESS 0`

6.86.1.2 `#define MHA_ERR_UNKNOWN 1`

6.86.1.3 `#define MHA_ERR_INVALID_HANDLE 2`

6.86.1.4 `#define MHA_ERR_NULL 3`

6.86.1.5 `#define MHA_ERR_VARRANGE 4`

6.86.1.6 `#define MHA_ERR_VARFMT 5`

6.86.1.7 `#define MHA_ERR_USER 10000`

6.86.2 Function Documentation

6.86.2.1 `const char* mha_strerror (`
`int mhaerrno)`

6.86.2.2 `void mha_set_user_error (`
`const char * str)`

6.87 mha_error.cpp File Reference

Implementation of openMHA error handling.

Namespaces

- **mha_error_helpers**

Functions

- unsigned **mha_error_helpers::digits** (unsigned n)
Compute number of decimal digits required to represent an unsigned integer.
- unsigned **mha_error_helpers::snprintf_required_length** (const char *formatstring,...)
snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.
- void **mha_debug** (const char *fmt,...)
Print an info message (stderr on Linux, OutputDebugString in Windows).

6.87.1 Detailed Description

Implementation of openMHA error handling.

This file forms a separate library.

6.88 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. 387).*
- #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. 387) with the values.*

Functions

- void **mha_debug** (const char *fmt,...)
Print an info message (stderr on Linux, OutputDebugString in Windows).
- unsigned **mha_error_helpers::digits** (unsigned n)
Compute number of decimal digits required to represent an unsigned integer.
- unsigned **mha_error_helpers::snprintf_required_length** (const char *formatstring,...)
snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.

6.88.1 Macro Definition Documentation

6.88.1.1 `#define Getmsg(
 e) ((e).get_msg())`

6.89 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.

6.90 mha_events.cpp File Reference

6.91 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.

6.92 mha_fftfb.cpp File Reference

Classes

- class **MHAOviFilter::barkscale::hz2bark_t**
- class **MHAOviFilter::barkscale::bark2hz_t**

Namespaces

- **MHAOvIFilter**
Namespace for overlapping FFT based filter bank classes and functions.
- **MHAOvIFilter::barkscale**
- **MHAOvIFilter::FreqScaleFun**
Transform functions from linear scale in Hz to new frequency scales.
- **MHAOvIFilter::ShapeFun**
Shape functions for overlapping filters.

Macros

- `#define BARKSCALE_ENTRIES 50`

Functions

- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2hz (mha_real_t x)**
Dummy scale transformation Hz to Hz.
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2khz (mha_real_t x)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2octave (mha_real_t x)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2third_octave (mha_real_t x)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2bark (mha_real_t x)**
Transformation to bark scale.
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2bark_analytic (mha_real_t)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2erb (mha_real_t)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2erb_glasberg1990 (mha_real_t)**
- **mha_real_t MHAOvIFilter::FreqScaleFun::hz2log (mha_real_t x)**
Third octave frequency scale.
- **mha_real_t MHAOvIFilter::FreqScaleFun::inv_scale (mha_real_t, mha_real_t (*) (mha_real_t))**
- **mha_real_t MHAOvIFilter::ShapeFun::rect (mha_real_t x)**
Filter shape function for rectangular filters.
- **mha_real_t MHAOvIFilter::ShapeFun::linear (mha_real_t x)**
Filter shape function for sawtooth filters.
- **mha_real_t MHAOvIFilter::ShapeFun::hann (mha_real_t x)**
Filter shape function for hanning shaped filters.
- **mha_real_t MHAOvIFilter::ShapeFun::expflt (mha_real_t)**
- **mha_real_t MHAOvIFilter::ShapeFun::gauss (mha_real_t)**
- **mha_real_t filtershapefun (mha_real_t f, MHAOvIFilter::band_descriptor_t b, mha_real_t plateau)**

Variables

- **mha_real_t MHAOvIFilter::barkscale::vfreq [BARKSCALE_ENTRIES]**
- **mha_real_t MHAOvIFilter::barkscale::vbark [BARKSCALE_ENTRIES]**

6.92.1 Macro Definition Documentation

6.92.1.1 #define BARKSCALE_ENTRIES 50

6.92.2 Function Documentation

6.92.2.1 mha_real_t filtershapefun (
 mha_real_t f,
 MHAOvIFilter::band_descriptor_t b,
 mha_real_t plateau)

6.93 mha_fftfb.hh File Reference

Classes

- class **MHAOvIFilter::band_descriptor_t**
- class **MHAOvIFilter::scale_var_t**
- class **MHAOvIFilter::fscale_t**
- class **MHAOvIFilter::fscale_bw_t**
- class **MHAOvIFilter::fftfb_vars_t**
Set of configuration variables for FFT-based overlapping filters.
- class **MHAOvIFilter::fspacing_t**
Class for frequency spacing, used by filterbank shape generator class.
- class **MHAOvIFilter::fftfb_t**
FFT based overlapping filter bank.
- class **MHAOvIFilter::overlap_save_filterbank_t**
*A time-domain minimal phase filter bank with frequency shapes from **MHAOvIFilter::fftfb_t** (p. 550).*
- class **MHAOvIFilter::overlap_save_filterbank_t::vars_t**
- class **MHAOvIFilter::overlap_save_filterbank_analytic_t**
- class **MHAOvIFilter::fftfb_ac_info_t**

Namespaces

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

Typedefs

- typedef **mha_real_t**(**MHAOvIFilter::scale_fun_t**) (mha_real_t)

6.94 mha_fifo.cpp File Reference

6.95 mha_fifo.h File Reference

Classes

- class **mha_fifo_t**< T >
A FIFO class for blocksize adaptation Synchronization: None.
- class **mha_drifter_fifo_t**< T >
A FIFO class for blocksize adaptation without Synchronization.
- class **mha_fifo_thread_platform_t**
Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.
- class **mha_fifo_posix_threads_t**
- class **mha_fifo_thread_guard_t**
Simple Mutex Guard Class.
- class **mha_fifo_lw_t**< T >
This FIFO uses locks to synchronize access.
- class **mha_dbdbuf_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. 403).*
- class **mha_rt_fifo_t**< T >
Template class for thread safe, half real time safe fifo without explicit locks.

Macros

- #define **mha_fifo_thread_platform_implementation_t** **mha_fifo_posix_threads_t**

6.95.1 Macro Definition Documentation

6.95.1.1 #define mha_fifo_thread_platform_implementation_t mha_fifo_posix_threads_t

6.96 mha_filter.cpp File Reference

Functions

- std::vector< **mha_real_t** > **diff_coeffs** ()

6.96.1 Function Documentation

6.96.1.1 `std::vector<mha_real_t> diff_coeffs ()`

6.97 mha_filter.hh File Reference

Header file for IIR filter classes.

Classes

- class **MHAFilter::filter_t**
Generic IIR filter class.
- class **MHAFilter::diff_t**
Differentiator class (non-normalized)
- class **MHAFilter::o1_ar_filter_t**
First order attack-release lowpass filter.
- class **MHAFilter::o1flt_lowpass_t**
First order low pass filter.
- class **MHAFilter::o1flt_maxtrack_t**
First order maximum tracker.
- class **MHAFilter::o1flt_mintrack_t**
First order minimum tracker.
- class **MHAFilter::iir_filter_state_t**
- class **MHAFilter::iir_filter_t**
IIR filter class wrapper for integration into parser structure.
- class **MHAFilter::adapt_filter_state_t**
- class **MHAFilter::adapt_filter_param_t**
- class **MHAFilter::adapt_filter_t**
Adaptive filter.
- class **MHAFilter::fftfilter_t**
FFT based FIR filter implementation.
- class **MHAFilter::fftfilterbank_t**
FFT based FIR filterbank implementation.
- struct **MHAFilter::transfer_function_t**
a structure containing a source channel number, a target channel number, and an impulse response.
- struct **MHAFilter::transfer_matrix_t**
A sparse matrix of transfer function partitionss.
- class **MHAFilter::partitioned_convolution_t**
A filter class for partitioned convolution.
- struct **MHAFilter::partitioned_convolution_t::index_t**
Bookkeeping class.
- class **MHAFilter::smoothspec_t**
Smooth spectral gains, create a windowed impulse response.

- class **MHAFilter::resampling_filter_t**
Hann shaped low pass filter for resampling.
- class **MHAFilter::polyphase_resampling_t**
A class that performs polyphase resampling.
- class **MHAFilter::blockprocessing_polyphase_resampling_t**
A class that does polyphase resampling and takes into account block processing.
- class **MHAFilter::iir_ord1_real_t**
First order recursive filter.

Namespaces

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

Functions

- void **MHAFilter::make_friendly_number** (mha_real_t &x)
- void **MHAFilter::make_friendly_number** (mha_complex_t &x)
- void **MHAFilter::make_friendly_number** (double &x)
- void **MHAFilter::o1_lp_coeffs** (const mha_real_t tau, const mha_real_t fs, mha_real_t &c1, mha_real_t &c2)
Set first order filter coefficients from time constant and sampling rate.
- void **MHAFilter::butter_stop_ord1** (double *A, double *B, double f1, double f2, double fs)
Setup a first order butterworth band stop filter.
- **MHASignal::waveform_t** * **MHAFilter::spec2fir** (const mha_spec_t *spec, const unsigned int fftlen, const **MHAWindow::base_t** &window, const bool minphase)
Create a windowed impulse response/FIR filter coefficients from a spectrum.
- unsigned **MHAFilter::gcd** (unsigned a, unsigned b)
greatest common divisor
- double **MHAFilter::sinc** (double x)
sin(x)/x function, coping with x=0.
- std::pair< unsigned, unsigned > **MHAFilter::resampling_factors** (float source_sampling_rate, float target_sampling_rate, float factor=1.0f)
Computes rational resampling factor from two sampling rates.

6.97.1 Detailed Description

Header file for IIR filter classes.

6.98 mha_generic_chain.cpp File Reference

Functions

- void **mhaconfig_compare** (mhaconfig_t req, mhaconfig_t avail, const char *cpref)

6.98.1 Function Documentation

6.98.1.1 void mhaconfig_compare (
 mhaconfig_t req,
 mhaconfig_t avail,
 const char * cpref)

6.99 mha_generic_chain.h File Reference

Classes

- class **mhachain::plugs_t**
- class **mhachain::chain_base_t**

Namespaces

- **mhachain**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.99.1 Macro Definition Documentation

6.99.1.1 #define MHAPLUGIN_OVERLOAD_OUTDOMAIN

6.100 mha_io_ifc.h File Reference

Typedefs

- typedef int(* **IOProcessEvent_t**) (void *handle, **mha_wave_t** *sIn, **mha_wave_t** **S↵
 Out)
Event handler for signal stream.
- typedef void(* **IStoppedEvent_t**) (void *handle, int proc_err, int io_err)
Event handler for stop event.
- typedef void(* **IStartedEvent_t**) (void *handle)
Event handler for start event.
- typedef int(* **IOInit_t**) (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void
 *proc_handle, **IStartedEvent_t** start_event, void *start_handle, **IStoppedEvent_t**↵
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(* **IOWStop_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)

6.100.1 Typedef Documentation

6.100.1.1 `typedef int(* IOProcessEvent_t)(void *handle, mha_wave_t *sIn, mha_wave_t **sOut)`

Event handler for signal stream.

This event handler needs to be realtime compatible. All signal path processing will be performed in this callback.

6.100.1.2 `typedef void(* IOStoppedEvent_t)(void *handle, int proc_err, int io_err)`

Event handler for stop event.

This event handler needs to be realtime compatible. The function must return immediatly.

6.100.1.3 `typedef void(* IOStartedEvent_t)(void *handle)`

Event handler for start event.

This event handler needs to be realtime compatible. The function must return immediatly.

6.100.1.4 `typedef int(* IOInit_t)(int fragsize, float samplerate, IOProcessEvent_t proc_event, void *proc_handle, IOStartedEvent_t start_event, void *start_handle, IOStoppedEvent_t stop_event, void *stop_handle, void **handle)`

6.100.1.5 `typedef int(* IOPrepare_t)(void *handle, int num_inchannels, int num_outchannels)`

6.100.1.6 `typedef int(* IOStart_t)(void *handle)`

6.100.1.7 `typedef int(* IOStop_t)(void *handle)`

6.100.1.8 `typedef int(* IORelease_t)(void *handle)`

6.100.1.9 `typedef int(* IOSetVar_t)(void *handle, const char *cmd, char *retval, unsigned int len)`

6.100.1.10 `typedef const char*(* IOStrError_t)(void *handle, int err)`

6.100.1.11 `typedef void(* IODestroy_t)(void *handle)`

6.101 mha_multisrc.cpp File Reference

Namespaces

- **MHAMultiSrc**

Collection of classes for selecting audio chunks from multiple sources.

6.102 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.

6.103 mha_os.cpp File Reference

Functions

- std::string **mha_getenv** (std::string envvar)
- std::list< std::string > **mha_library_paths** ()
- std::list< std::string > **list_dir** (const std::string &path, const std::string &pattern)

6.103.1 Function Documentation

6.103.1.1 std::string mha_getenv (
std::string envvar)

6.103.1.2 std::list<std::string> mha_library_paths ()

6.103.1.3 std::list<std::string> list_dir (
const std::string & path,
const std::string & pattern)

6.104 mha_os.h File Reference

Classes

- class **dynamiclib_t**

Macros

- `#define mha_loadlib(x) dlopen(x,RTLD_NOW)`
- `#define mha_freelib(x) dlclose(x)`
- `#define mha_freelib_success(x) (x == 0)`
- `#define mha_getlibfun(h, x) x ## _cb = (x ## _t)dlsym(h,#x)`
- `#define mha_getlibfun_checked(h, x) x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw MHA_Error(__FILE__,__LINE__,"Function " #x " is undefined.")`
- `#define mha_loadlib_error(x) dLError()`
- `#define mha_lib_extension ".so"`
- `#define mha_msleep(millisecons) usleep((millisecons)*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 (std::string envvar)`
- `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)`

6.104.1 Macro Definition Documentation

6.104.1.1 `#define mha_loadlib(
 x) dlopen(x,RTLD_NOW)`

6.104.1.2 `#define mha_freelib(
 x) dlclose(x)`

6.104.1.3 `#define mha_freelib_success(
 x) (x == 0)`

- 6.104.1.4 `#define mha_getlibfun(
 h,
 x) x ## _cb = (x ## _t)dlsym(h,#x)`
- 6.104.1.5 `#define mha_getlibfun_checked(
 h,
 x) x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw
 MHA_Error(__FILE__, __LINE__, "Function " #x " is undefined.")`
- 6.104.1.6 `#define mha_loadlib_error(
 x) dlerror()`
- 6.104.1.7 `#define mha_lib_extension ".so"`
- 6.104.1.8 `#define mha_msleep(
 milliseconds) usleep((milliseconds)*1000)`
- 6.104.1.9 `#define FMTsz "%zu"`

printf modifier to print integers of type size_t

- 6.104.1.10 `#define MHA_RESOLVE(
 h,
 t) t ## _cb = (t ## _t)(h->resolve(#t))`
- 6.104.1.11 `#define MHA_RESOLVE_CHECKED(
 h,
 t) t ## _cb = (t ## _t)(h->resolve_checked(#t))`

6.104.2 Typedef Documentation

- 6.104.2.1 `typedef void* mha_libhandle_t`

6.104.3 Function Documentation

- 6.104.3.1 `std::string mha_getenv (
 std::string envvar)`
- 6.104.3.2 `std::list<std::string> mha_library_paths ()`
- 6.104.3.3 `std::list<std::string> list_dir (
 const std::string & path,
 const std::string & pattern)`

6.104.3.4 void mha_hton (
 float * *data*,
 unsigned int *len*) [inline]

6.104.3.5 void mha_ntoh (
 float * *data*,
 unsigned int *len*) [inline]

6.104.3.6 void mha_hton (
 uint32_t * *data*,
 unsigned int *len*) [inline]

6.104.3.7 void mha_ntoh (
 uint32_t * *data*,
 unsigned int *len*) [inline]

6.104.3.8 void mha_hton (
 int32_t * *data*,
 unsigned int *len*) [inline]

6.104.3.9 void mha_ntoh (
 int32_t * *data*,
 unsigned int *len*) [inline]

6.105 mha_parser.cpp File Reference

Namespaces

- **MHAParser**
Name space for the openMHA-Parser configuration language.
- **MHAParser::StrCnv**
String converter namespace.

Macros

- #define **MHAPLATFORM** "undefined-linux"

Functions

- int **MHAParser::get_precision** ()
- int **MHAParser::StrCnv::num_brackets** (const std::string &s)
Return number of brackets at beginning and end of string.
- int **MHAParser::StrCnv::bracket_balance** (const std::string &s)
- static std::ostream & **write_float** (std::ostream &o, const float &f)
- static std::string **parse_1_float** (const std::string &s, **mha_real_t** &v)
This internal function parses a floating point number from the beginning of a string.
- static std::string **parse_1_complex** (const std::string &s, **mha_complex_t** &v)
This internal function parses a complex number from the beginning of a string.

6.105.1 Macro Definition Documentation

6.105.1.1 `#define MHAPLATFORM "undefined-linux"`

6.105.2 Function Documentation

6.105.2.1 `static std::ostream& write_float (`
`std::ostream & o,`
`const float & f) [inline],[static]`

6.105.2.2 `static std::string parse_1_float (`
`const std::string & s,`
`mha_real_t & v) [static]`

This internal function parses a floating point number from the beginning of a string.

Parameters

| | |
|----------------|---|
| <code>s</code> | The string to parse |
| <code>v</code> | The float variable to fill with a value |

Returns

The rest of the string.

6.105.2.3 `static std::string parse_1_complex (`
`const std::string & s,`
`mha_complex_t & v) [static]`

This internal function parses a complex number from the beginning of a string.

Parameters

| | |
|----------------|---|
| <code>s</code> | The string to parse |
| <code>v</code> | The complex variable to fill with a value |

Returns

The rest of the string.

6.106 mha_parser.hh File Reference

Header file for the MHA-Parser script language.

Classes

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

- Monitor with string value.*
- class **MHAParser::string_mon_t**
Monitor with string value.
- class **MHAParser::vstring_mon_t**
Vector of monitors with string value.
- class **MHAParser::vint_mon_t**
Vector of ints monitor.
- class **MHAParser::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::string > &`)
Convert to string.
- `std::string MHAParser::StrCnv::val2str` (`const std::vector< std::vector< float > > &`)
Convert to string.
- `std::string MHAParser::StrCnv::val2str` (`const std::vector< std::vector< mha_complex_t > > &`)
Convert to string.

6.106.1 Detailed Description

Header file for the MHA-Parser script language.

6.106.2 Macro Definition Documentation

6.106.2.1 `#define DEFAULT_RETSIZE 0x100000`

6.106.2.2 `#define insert_member(x) insert_item(#x,&x)`

Macro to insert a member variable into a parser.

Parameters

| | |
|----------------|---|
| <code>x</code> | Member variable to be inserted. Name of member variable will be used as configuration name. |
|----------------|---|

See also `MHAParser::parser_t::insert_item()` (p. [622](#)).

6.107 mha_plugin.hh File Reference

Header file for MHA C++ plugin class templates.

Classes

- class **MHAPLugin::cfg_chain_t**< runtime_cfg_t >
- class **MHAPLugin::config_t**< runtime_cfg_t >
Template class for thread safe configuration.
- class **MHAPLugin::plugin_t**< runtime_cfg_t >
The template class for C++ openMHA plugins.

Namespaces

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

Macros

- #define **__declspec**(p)
- #define **WINAPI**
- #define **HINSTANCE** int
- #define **GITCOMMITHASH** "independent-plugin-build"
- #define **MHAPLUGIN_PROC_CALLBACK_PREFIX**(prefix, classname, indom, outdom)
- #define **MHAPLUGIN_INIT_CALLBACKS_PREFIX**(prefix, classname)
- #define **MHAPLUGIN_CALLBACKS_PREFIX**(prefix, classname, indom, outdom)
C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION_PREFIX**(prefix, cat, doc)
- #define **MHAPLUGIN_PROC_CALLBACK**(plugname, classname, indom, outdom) **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.

Functions

- **__attribute__** ((unused)) static const char *mha_git_commit_hash
store git commit hash in every binary plgin to support reproducible research

6.107.1 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`).

6.107.2 Macro Definition Documentation

6.107.2.1 `#define __declspec(
 p)`

6.107.2.2 `#define WINAPI`

6.107.2.3 `#define HINSTANCE int`

6.107.2.4 `#define GITCOMMITHASH "independent-plugin-build"`

6.107.2.5 `#define MHAPLUGIN_PROC_CALLBACK_PREFIX(
 prefix,
 classname,
 indom,
 outdom)`

6.107.2.6 `#define MHAPLUGIN_INIT_CALLBACKS_PREFIX(
 prefix,
 classname)`

6.107.2.7 `#define MHAPLUGIN_DOCUMENTATION_PREFIX(
 prefix,
 cat,
 doc)`

6.107.2.8 `#define MHAPLUGIN_PROC_CALLBACK(
 plugname,
 classname,
 indom,
 outdom) MHAPLUGIN_PROC_CALLBACK_PREFIX(MHA_STATIC_ ##
 plugname ## _,classname,indom,outdom)`

6.107.2.9 `#define MHAPLUGIN_INIT_CALLBACKS(
 plugname,
 classname) MHAPLUGIN_INIT_CALLBACKS_PREFIX(MHA_STATIC_
 ## plugname ## _,classname)`

6.107.3 Function Documentation

6.107.3.1 `__attribute__(
 (unused)) const`

store git commit hash in every binary plgin to support reproducible research

6.108 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)

6.108.1 Function Documentation

6.108.1.1 void mha_tic (
 mha_tictoc_t * t)

6.108.1.2 void mha_platform_tic (
 mha_platform_tictoc_t * t)

6.108.1.3 float mha_toc (
 mha_tictoc_t * t)

6.108.1.4 float mha_platform_toc (
 mha_platform_tictoc_t * t)

6.109 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)

6.109.1 Typedef Documentation

6.109.1.1 typedef mha_tictoc_t mha_platform_tictoc_t

6.109.2 Function Documentation

6.109.2.1 void mha_platform_tic (
mha_platform_tictoc_t * t)6.109.2.2 float mha_platform_toc (
mha_platform_tictoc_t * t)

6.110 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** ()

6.110.1 Typedef Documentation

6.110.1.1 typedef VALUE(* **rb_f_t**) (...)

6.110.2 Function Documentation

6.110.2.1 static void mha_free (
void * *mha*) [static]6.110.2.2 static VALUE mha_alloc (
VALUE *klass*) [static]6.110.2.3 static VALUE mha_exit_request (
VALUE *self*) [static]6.110.2.4 static VALUE mha_parse (
VALUE *self*,
VALUE *request*) [static]

6.110.2.5 void Init_mha_ruby ()

6.111 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. 436) 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. 436) 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. 406) 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.

6.111.1 Macro Definition Documentation

6.111.1.1 `#define MHA_ID_UINT_VECTOR "MHASignal::uint_vector_t"`

6.111.1.2 `#define MHA_ID_MATRIX "MHASignal::matrix_t"`

6.111.1.3 `#define ASSERT_EQUAL_DIM(
a,
b)`

6.111.1.4 `#define ASSERT_EQUAL_DIM_PTR(
a,
b)`

6.111.2 Function Documentation

6.111.2.1 `void set_minabs (
mha_spec_t & self,
const mha_real_t & m)`

6.111.2.2 `mha_spec_t& safe_div (
mha_spec_t & self,
const mha_spec_t & v,
mha_real_t eps)`

In-Place division with lower limit on divisor.

6.111.2.3 `static mha_real_t intensity (
const mha_spec_t & s,
unsigned int channel,
unsigned int fftlen,
mha_real_t * sqfreq_response = 0) [static]`

6.112 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. 406))*
- class **MHASignal::waveform_t**
*signal processing class for waveform data (based on **mha_wave_t** (p. 436))*
- 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. 436).
- **mha_real_t MHASignal::lin2db** (mha_real_t x)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t MHASignal::db2lin** (mha_real_t x)
Conversion from dB scale to linear (no SPL reference)
- **mha_real_t MHASignal::pa2dbspl** (mha_real_t x)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t MHASignal::pa22dbspl** (mha_real_t x, mha_real_t eps=1e-20f)
Conversion from squared Pascal scale to dB SPL.
- **mha_real_t MHASignal::dbspl2pa** (mha_real_t x)
Conversion from dB SPL to linear Pascal scale.
- **mha_real_t MHASignal::smp2sec** (mha_real_t n, mha_real_t srate)
conversion from samples to seconds
- **mha_real_t MHASignal::sec2smp** (mha_real_t sec, mha_real_t srate)
conversion from seconds to samples
- **mha_real_t MHASignal::bin2freq** (mha_real_t bin, unsigned fftlen, mha_real_t srate)
conversion from fft bin index to frequency
- **mha_real_t MHASignal::freq2bin** (mha_real_t freq, unsigned fftlen, mha_real_t srate)
conversion from frequency to fft bin index
- **mha_real_t MHASignal::smp2rad** (mha_real_t samples, unsigned bin, unsigned fftlen)
conversion from delay in samples to phase shift
- **mha_real_t MHASignal::rad2smp** (mha_real_t phase_shift, unsigned bin, unsigned fftlen)
conversion from phase shift to delay in samples
- **template<class elem_type >**
std::vector< elem_type > MHASignal::dupvec (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size.
- **template<class elem_type >**
std::vector< elem_type > MHASignal::dupvec_chk (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size, check for dimension.
- **bool equal_dim** (const mha_wave_t &a, const mha_wave_t &b)
Test for equal dimension of waveform structures.
- **bool equal_dim** (const mha_wave_t &a, const mhaconfig_t &b)
Test for match of waveform dimension with mhaconfig structure.
- **bool equal_dim** (const mha_spec_t &a, const mha_spec_t &b)
Test for equal dimension of spectrum structures.
- **bool equal_dim** (const mha_spec_t &a, const mhaconfig_t &b)
Test for match of spectrum dimension with mhaconfig structure.
- **bool equal_dim** (const mha_wave_t &a, const mha_spec_t &b)
Test for equal dimension of waveform/spectrum structures.
- **bool equal_dim** (const mha_spec_t &a, const mha_wave_t &b)

Test for equal dimension of waveform/spectrum structures.

- void **integrate** (**mha_wave_t** &s)
Numeric integration of a signal vector (real values)
- void **integrate** (**mha_spec_t** &s)
Numeric integration of a signal vector (complex values)
- unsigned int **mha_min_1** (unsigned int a)
- unsigned int **size** (const **mha_wave_t** &s)
Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** &s)
Return size of a spectrum structure.
- unsigned int **size** (const **mha_wave_t** *s)
Return size of a waveform structure.
- unsigned int **size** (const **mha_spec_t** *s)
Return size of a spectrum structure.
- void **clear** (**mha_wave_t** &s)
Set all values of waveform to zero.
- void **clear** (**mha_wave_t** *s)
Set all values of waveform to zero.
- void **clear** (**mha_spec_t** &s)
Set all values of spectrum to zero.
- void **clear** (**mha_spec_t** *s)
Set all values of spectrum to zero.
- void **assign** (**mha_wave_t** self, **mha_real_t** val)
Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_wave_t** self, const **mha_wave_t** &val)
Set all values of waveform 'self' to 'val'.
- void **assign** (**mha_spec_t** self, const **mha_spec_t** &val)
Set all values of spectrum 'self' to 'val'.
- void **timeshift** (**mha_wave_t** &self, int shift)
Time shift of waveform chunk.
- **mha_wave_t** **range** (**mha_wave_t** s, unsigned int k0, unsigned int len)
Return a time interval from a waveform chunk.
- **mha_spec_t** **channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)
Return a channel interval from a spectrum.
- **mha_real_t** & **value** (**mha_wave_t** *s, unsigned int fr, unsigned int ch)
Access an element of a waveform structure.
- const **mha_real_t** & **value** (const **mha_wave_t** *s, unsigned int fr, unsigned int ch)
Constant access to an element of a waveform structure.
- **mha_real_t** & **value** (**mha_wave_t** *s, unsigned int k)
- **mha_complex_t** & **value** (**mha_spec_t** *s, unsigned int k)
- **mha_complex_t** & **value** (**mha_spec_t** *s, unsigned int fr, unsigned int ch)
Access to an element of a spectrum.
- const **mha_complex_t** & **value** (const **mha_spec_t** *s, unsigned int fr, unsigned int ch)
Constant access to an element of a spectrum.
- **mha_real_t** & **value** (**mha_wave_t** &s, unsigned int fr, unsigned int ch)

Access to an element of a waveform structure.

- **const mha_real_t & value** (const **mha_wave_t** &s, unsigned int fr, unsigned int ch)

Constant access to an element of a waveform structure.

- **mha_complex_t & value** (**mha_spec_t** &s, unsigned int fr, unsigned int ch)

Access to an element of a spectrum.

- **const mha_complex_t & value** (const **mha_spec_t** &s, unsigned int fr, unsigned int ch)

Constant access to an element of a spectrum.

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

*Converts a **mha_wave_t** (p. 436) 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. 436) 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. 406) structure into a **std::vector< std::vector<mha_complex_t> >** (outer vector represents channels).*

- **mha_wave_t & operator+=** (**mha_wave_t** &, const **mha_real_t** &)

Addition operator.

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

Addition operator.

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

Subtraction operator.

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

Subtraction operator.

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

Element-wise multiplication operator.

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

Element-wise multiplication operator.

- **mha_spec_t & operator*=** (**mha_spec_t** &, const **mha_real_t** &)

Element-wise multiplication operator.

- **mha_spec_t & operator*=** (**mha_spec_t** &, const **mha_wave_t** &)

Element-wise multiplication operator.

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

Element-wise multiplication operator.

- **mha_spec_t & operator/=** (**mha_spec_t** &, const **mha_spec_t** &)

Element-wise division operator.

- **mha_wave_t & operator/=** (**mha_wave_t** &, const **mha_wave_t** &)

Element-wise division operator.

- **mha_spec_t & operator+=** (**mha_spec_t** &, const **mha_spec_t** &)

Addition operator.

- **mha_spec_t & operator+=** (**mha_spec_t** &, const **mha_real_t** &)

Addition operator.

- **void set_minabs** (**mha_spec_t** &, const **mha_real_t** &)

- **mha_spec_t & safe_div** (**mha_spec_t** &self, const **mha_spec_t** &v, **mha_real_t** eps)

In-Place division with lower limit on divisor.

- **mha_wave_t & operator^=** (mha_wave_t &self, const mha_real_t &arg)
Exponent operator.
- **void MHASignal::copy_channel** (mha_spec_t &self, const mha_spec_t &src, unsigned sch, unsigned dch)
Copy one channel of a source signal.
- **void MHASignal::copy_channel** (mha_wave_t &self, const mha_wave_t &src, unsigned src_channel, unsigned dest_channel)
Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel** (const mha_spec_t &s, unsigned int channel, unsigned int fftlen)
Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity** (const mha_spec_t &s, unsigned int channel, unsigned int fftlen, mha_real_t sqfreq_response[])
Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs** (const mha_spec_t &s, unsigned int channel)
Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel** (const mha_wave_t &s, unsigned int channel)
Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs** (const mha_wave_t &s, unsigned int channel)
Find maximal absolute value.
- **mha_real_t MHASignal::maxabs** (const mha_wave_t &s)
Find maximal absolute value.
- **mha_real_t MHASignal::max** (const mha_wave_t &s)
Find maximal value.
- **mha_real_t MHASignal::min** (const mha_wave_t &s)
Find minimal value.
- **mha_real_t MHASignal::sumsq_channel** (const mha_wave_t &s, unsigned int channel)
Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsq_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 signal 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. 374) variable.
- **mha_complex_t mha_complex** (mha_real_t real, mha_real_t imag=0)
Create a new mha_complex_t (p. 374) 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. 374) 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. 374).
- **mha_complex_t & expi** (mha_complex_t &self, mha_real_t angle)
*replaces the value of the given mha_complex_t (p. 374) 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. 374) 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. 374) with its conjugate.*
- **void conjugate** (**mha_spec_t** &self)
*Replace (!) the value of this **mha_spec_t** (p. 406) with its conjugate.*
- **mha_complex_t _conjugate** (const **mha_complex_t** &self)
Compute the conjugate 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. 374)*
- **std::istream & operator>>** (std::istream &i, **mha_complex_t** &c)
*preliminary istream operator for **mha_complex_t** (p. 374) 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)
Transform waveform segment into spectrum.
- void **mha_fft_spec2wave_scale** (mha_fft_t h, const mha_spec_t *in, mha_wave_t *out)
Transform 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.

6.112.1 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.

6.112.2 Macro Definition Documentation

6.112.2.1 `#define M_PI 3.14159265358979323846`

6.112.2.2 `#define mha_round(
x) (int)((float)x+0.5)`

6.112.3 Function Documentation

6.112.3.1 `unsigned int mha_min_1 (
unsigned int a) [inline]`

6.112.3.2 `mha_real_t& value (
mha_wave_t * s,
unsigned int k) [inline]`

6.112.3.3 `mha_complex_t& value (
mha_spec_t * s,
unsigned int k) [inline]`

6.112.3.4 `void set_minabs (
mha_spec_t & ,
const mha_real_t &)`

6.112.3.5 `mha_spec_t& safe_div (
mha_spec_t & self,
const mha_spec_t & v,
mha_real_t eps)`

In-Place division with lower limit on divisor.

6.112.3.6 `std::ostream& operator<< (
std::ostream & o,
const mha_complex_t & c) [inline]`

ostream operator for **mha_complex_t** (p. [374](#))

6.112.3.7 `std::istream& operator>> (`
 `std::istream & i,`
 `mha_complex_t & c) [inline]`

preliminary istream operator for **mha_complex_t** (p. 374) without error checking

6.113 mha_signal_fft.h File Reference

Classes

- class **MHASignal::fft_t**

Namespaces

- **MHASignal**

Namespace for audio signal handling and processing classes.

6.114 mha_tablelookup.cpp File Reference

6.115 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.

6.115.1 Detailed Description

Header file for table lookup classes.

6.116 mha_tcp.cpp File Reference

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)

6.116.1 Macro Definition Documentation

6.116.1.1 `#define INVALID_SOCKET (-1)`

6.116.1.2 `#define SOCKET_ERROR (-1)`

6.116.1.3 `#define closesocket(
 fd) (close((fd)))`

6.116.1.4 `#define ASYNC_CONNECT_STARTED EINPROGRESS`

6.116.2 Typedef Documentation

6.116.2.1 `typedef int SOCKET`

6.116.3 Function Documentation

6.116.3.1 `static sockaddr_in host_port_to_sock_addr (
 const std::string & host,
 unsigned short port) [static]`

6.116.3.2 `static SOCKET tcp_connect_to (
 const std::string & host,
 unsigned short port) [static]`

6.116.3.3 `static SOCKET tcp_connect_to_with_timeout (
 const std::string & host,
 unsigned short port,
 Timeout_Watcher & timeout_watcher) [static]`

6.116.3.4 `static void* thread_start_func (
 void * thread) [static]`

6.117 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. 411) 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.

6.117.1 Macro Definition Documentation

6.117.1.1 `#define Sleep(
 x) usleep((x)*1000);`

6.118 mha_toolbox.h File Reference

6.119 mha_windowparser.cpp File Reference

Variables

- `float(* wnd_funs [])(float)`

6.119.1 Variable Documentation

6.119.1.1 `float(* wnd_funs[])(float)`

6.120 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.

6.121 mhachain.cpp File Reference

Classes

- class **mhachain::mhachain_t**

Namespaces

- **mhachain**

6.122 mhafw_lib.cpp File Reference

6.123 mhafw_lib.h File Reference

Classes

- class **io_lib_t**
Class for loading MHA sound IO module.
- class **fw_vars_t**
- class **fw_t**

6.124 MHAIOFile.cpp File Reference

Classes

- class **io_file_t**
File IO.

Macros

- **#define DEBUG(x)** std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " << x << std::endl
- **#define ERR_SUCCESS** 0
- **#define ERR_IHANDLE** -1
- **#define ERR_USER** -1000
- **#define MAX_USER_ERR** 0x500
- **#define IOInit** MHA_STATIC_MHAIOFile_IOInit
- **#define IOPrepare** MHA_STATIC_MHAIOFile_IOPrepare
- **#define IOStart** MHA_STATIC_MHAIOFile_IOStart
- **#define IOSTop** MHA_STATIC_MHAIOFile_IOStop
- **#define IORelease** MHA_STATIC_MHAIOFile_IORelease
- **#define IOSetVar** MHA_STATIC_MHAIOFile_IOSetVar
- **#define IOStrError** MHA_STATIC_MHAIOFile_IOStrError
- **#define IODestroy** MHA_STATIC_MHAIOFile_IODestroy
- **#define dummy_interface_test** MHA_STATIC_MHAIOFile_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↵
handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↵
event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *command, char *retval, unsigned int maxretlen)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [MAX_USER_ERR]

6.124.1 Macro Definition Documentation

6.124.1.1 `#define DEBUG(
 x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " << x <<
 std::endl`

6.124.1.2 `#define ERR_SUCCESS 0`

6.124.1.3 `#define ERR_IHANDLE -1`

6.124.1.4 `#define ERR_USER -1000`

6.124.1.5 `#define MAX_USER_ERR 0x500`

6.124.1.6 `#define IOInit MHA_STATIC_MHAIOFile_IOInit`

6.124.1.7 `#define IOPrepare MHA_STATIC_MHAIOFile_IOPrepare`

6.124.1.8 `#define IOStart MHA_STATIC_MHAIOFile_IOStart`

6.124.1.9 `#define IOSTop MHA_STATIC_MHAIOFile_IOStop`

6.124.1.10 `#define IORelease MHA_STATIC_MHAIOFile_IORelease`

6.124.1.11 `#define IOSetVar MHA_STATIC_MHAIOFile_IOSetVar`

6.124.1.12 `#define IOStrError MHA_STATIC_MHAIOFile_IOStrError`

6.124.1.13 `#define IODestroy MHA_STATIC_MHAIOFile_IDestroy`

6.124.1.14 `#define dummy_interface_test MHA_STATIC_MHAIOFile_dummy_interface_test`

6.124.2 Function Documentation

6.124.2.1 `int IOInit (
 int fragsize,
 float samplerate,
 IOProcessEvent_t proc_event,
 void * proc_handle,
 IOStartedEvent_t start_event,
 void * start_handle,
 IOStoppedEvent_t stop_event,
 void * stop_handle,
 void ** handle)`

- 6.124.2.2 int IOPrepare (
 - void * *handle*,
 - int *nch_in*,
 - int *nch_out*)
- 6.124.2.3 int IOStart (
 - void * *handle*)
- 6.124.2.4 int IOStop (
 - void * *handle*)
- 6.124.2.5 int IORelease (
 - void * *handle*)
- 6.124.2.6 int IOSetVar (
 - void * *handle*,
 - const char * *command*,
 - char * *retval*,
 - unsigned int *maxretlen*)
- 6.124.2.7 const char* IOStrError (
 - void * *handle*,
 - int *err*)
- 6.124.2.8 void IODestroy (
 - void * *handle*)
- 6.124.2.9 void dummy_interface_test (
 - void)

6.124.3 Variable Documentation

- 6.124.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.125 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_IDestroy
- **#define dummy_interface_test** MHA_STATIC_MHAIOJack_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↵
handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↵
event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *command, char *retval, unsigned int maxretlen)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [MAX_USER_ERR] = ""

6.125.1 Macro Definition Documentation

6.125.1.1 **#define ERR_SUCCESS** 0

6.125.1.2 **#define ERR_IHANDLE** -1

6.125.1.3 **#define ERR_USER** -1000

6.125.1.4 **#define MAX_USER_ERR** 0x500

- 6.125.1.5 `#define IOInit MHA_STATIC_MHAIOJack_IOInit`
- 6.125.1.6 `#define IOPrepare MHA_STATIC_MHAIOJack_IOPrepare`
- 6.125.1.7 `#define IOStart MHA_STATIC_MHAIOJack_IOStart`
- 6.125.1.8 `#define IOSTop MHA_STATIC_MHAIOJack_IOStop`
- 6.125.1.9 `#define IORelease MHA_STATIC_MHAIOJack_IORelease`
- 6.125.1.10 `#define IOSetVar MHA_STATIC_MHAIOJack_IOSetVar`
- 6.125.1.11 `#define IOStrError MHA_STATIC_MHAIOJack_IOStrError`
- 6.125.1.12 `#define IODestroy MHA_STATIC_MHAIOJack_IDestroy`
- 6.125.1.13 `#define dummy_interface_test MHA_STATIC_MHAIOJack_dummy_interface_test`
- 6.125.2 Function Documentation
 - 6.125.2.1 `int IOInit (`
 - `int fragsize,`
 - `float samplerate,`
 - `IOProcessEvent_t proc_event,`
 - `void * proc_handle,`
 - `IOStartedEvent_t start_event,`
 - `void * start_handle,`
 - `IOStoppedEvent_t stop_event,`
 - `void * stop_handle,`
 - `void ** handle)`
 - 6.125.2.2 `int IOPrepare (`
 - `void * handle,`
 - `int nch_in,`
 - `int nch_out)`
 - 6.125.2.3 `int IOStart (`
 - `void * handle)`
 - 6.125.2.4 `int IOStop (`
 - `void * handle)`
 - 6.125.2.5 `int IORelease (`
 - `void * handle)`

6.125.2.6 `int IOSetVar (`
 `void * handle,`
 `const char * command,`
 `char * retval,`
 `unsigned int maxretlen)`

6.125.2.7 `const char* IOStrError (`
 `void * handle,`
 `int err)`

6.125.2.8 `void IODestroy (`
 `void * handle)`

6.125.2.9 `void dummy_interface_test (`
 `void)`

6.125.3 Variable Documentation

6.125.3.1 `char user_err_msg[MAX_USER_ERR] = ""` `[static]`

6.126 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 *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [MAX_USER_ERR]

6.126.1 Macro Definition Documentation

6.126.1.1 **#define ERR_SUCCESS 0**

6.126.1.2 **#define ERR_IHANDLE -1**

6.126.1.3 **#define ERR_USER -1000**

6.126.1.4 **#define MAX_USER_ERR 0x500**

6.126.1.5 **#define IOInit MHA_STATIC_MHAIOParser_IOInit**

6.126.1.6 **#define IOPrepare MHA_STATIC_MHAIOParser_IOPrepare**

6.126.1.7 **#define IOStart MHA_STATIC_MHAIOParser_IOStart**

6.126.1.8 **#define IOStop MHA_STATIC_MHAIOParser_IOStop**

6.126.1.9 **#define IORelease MHA_STATIC_MHAIOParser_IORelease**

6.126.1.10 **#define IOSetVar MHA_STATIC_MHAIOParser_IOSetVar**

6.126.1.11 **#define IOStrError MHA_STATIC_MHAIOParser_IOStrError**

6.126.1.12 **#define IODestroy MHA_STATIC_MHAIOParser_IDestroy**

6.126.1.13 `#define dummy_interface_test MHA_STATIC_MHAIOParser_dummy_interface_test`

6.126.2 Function Documentation

6.126.2.1 `int IOInit (`
 `int fragsize,`
 `float ,`
 `IOProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle,`
 `void ** handle)`

6.126.2.2 `int IOPrepare (`
 `void * handle,`
 `int nch_in,`
 `int nch_out)`

6.126.2.3 `int IOStart (`
 `void * handle)`

6.126.2.4 `int IOStop (`
 `void * handle)`

6.126.2.5 `int IORelease (`
 `void * handle)`

6.126.2.6 `int IOSetVar (`
 `void * handle,`
 `const char * command,`
 `char * retval,`
 `unsigned int maxretlen)`

6.126.2.7 `const char* IOStrError (`
 `void * handle,`
 `int err)`

6.126.2.8 `void IODestroy (`
 `void * handle)`

6.126.2.9 `void dummy_interface_test (`
 `void)`

6.126.3 Variable Documentation

6.126.3.1 `char user_err_msg[MAX_USER_ERR] [static]`

6.127 MHAIOPortAudio.cpp File Reference

Classes

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

Namespaces

- **MHAIOPortAudio**

Macros

- `#define ERR_SUCCESS 0`
- `#define ERR_IHANDLE -1`
- `#define ERR_USER -1000`
- `#define MAX_USER_ERR 0x500`
- `#define IOInit MHA_STATIC_MHAIOPortAudio_IOInit`
- `#define IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare`
- `#define IOStart MHA_STATIC_MHAIOPortAudio_IOStart`
- `#define IOSTop MHA_STATIC_MHAIOPortAudio_IOStop`
- `#define IORelease MHA_STATIC_MHAIOPortAudio_IORelease`
- `#define IOSetVar MHA_STATIC_MHAIOPortAudio_IOSetVar`
- `#define IOStrError MHA_STATIC_MHAIOPortAudio_IOStrError`
- `#define IODestroy MHA_STATIC_MHAIOPortAudio_IDestroy`
- `#define dummy_interface_test MHA_STATIC_MHAIOPortAudio_dummy_interface_↵
test`

Functions

- static std::string **MHAIOPortAudio::parserFriendlyName** (const std::string &in)
- int **portaudio_callback** (const void *input, void *output, unsigned long frameCount, const PaStreamCallbackTimeInfo *timeInfo, PaStreamCallbackFlags statusFlags, void *user_↵
Data)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↵
handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↵
event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *command, char *retval, unsigned int maxretlen)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [MAX_USER_ERR] = ""
- PaStreamCallback **portaudio_callback**

6.127.1 Macro Definition Documentation

6.127.1.1 **#define** ERR_SUCCESS 0

6.127.1.2 **#define** ERR_IHANDLE -1

6.127.1.3 **#define** ERR_USER -1000

6.127.1.4 **#define** MAX_USER_ERR 0x500

6.127.1.5 **#define** IOInit MHA_STATIC_MHAIOPortAudio_IOInit

6.127.1.6 **#define** IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare

6.127.1.7 **#define** IOStart MHA_STATIC_MHAIOPortAudio_IOStart

6.127.1.8 **#define** IOSTop MHA_STATIC_MHAIOPortAudio_IOStop

6.127.1.9 **#define** IORelease MHA_STATIC_MHAIOPortAudio_IORelease

6.127.1.10 **#define** IOSetVar MHA_STATIC_MHAIOPortAudio_IOSetVar

6.127.1.11 **#define** IOStrError MHA_STATIC_MHAIOPortAudio_IOStrError

6.127.1.12 **#define** IODestroy MHA_STATIC_MHAIOPortAudio_IODestroy

6.127.1.13 **#define** dummy_interface_test MHA_STATIC_MHAIOPortAudio_dummy_interface_test

6.127.2 Function Documentation

6.127.2.1 **int** portaudio_callback (
 const void * *input*,
 void * *output*,
 unsigned long *frameCount*,
 const PaStreamCallbackTimeInfo * *timeInfo*,
 PaStreamCallbackFlags *statusFlags*,
 void * *userData*)

6.127.2.2 int IOInit (
 int *fragsize*,
 float *samplerate*,
 IOProcessEvent_t *proc_event*,
 void * *proc_handle*,
 IOStartedEvent_t *start_event*,
 void * *start_handle*,
 IOStoppedEvent_t *stop_event*,
 void * *stop_handle*,
 void ** *handle*)

6.127.2.3 int IOPrepare (
 void * *handle*,
 int *nch_in*,
 int *nch_out*)

6.127.2.4 int IOStart (
 void * *handle*)

6.127.2.5 int IOStop (
 void * *handle*)

6.127.2.6 int IORelease (
 void * *handle*)

6.127.2.7 int IOSetVar (
 void * *handle*,
 const char * *command*,
 char * *retval*,
 unsigned int *maxretlen*)

6.127.2.8 const char* IOStrError (
 void * *handle*,
 int *err*)

6.127.2.9 void IODestroy (
 void * *handle*)

6.127.2.10 void dummy_interface_test (
 void)

6.127.3 Variable Documentation

6.127.3.1 char user_err_msg[MAX_USER_ERR] = "" [static]

6.127.3.2 PaStreamCallback portaudio_callback

6.128 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_MHAOTCP_IOInit
- **#define IOPrepare** MHA_STATIC_MHAOTCP_IOPrepare
- **#define IOStart** MHA_STATIC_MHAOTCP_IOStart
- **#define IOSTop** MHA_STATIC_MHAOTCP_IOStop
- **#define IORelease** MHA_STATIC_MHAOTCP_IORelease
- **#define IOSetVar** MHA_STATIC_MHAOTCP_IOSetVar
- **#define IOStrError** MHA_STATIC_MHAOTCP_IOStrError
- **#define IODestroy** MHA_STATIC_MHAOTCP_IODestroy
- **#define dummy_interface_test** MHA_STATIC_MHAOTCP_dummy_interface_test

Functions

- static int **copy_error** (**MHA_Error** &e)
- static void * **thread_startup_function** (void *parameter)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_↵
handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_↵
event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int num_inchannels, int num_outchannels)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *cmd, char *retval, unsigned int len)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)
- void **dummy_interface_test** (void)

Variables

- static char **user_err_msg** [MAX_USER_ERR]

6.128.1 Macro Definition Documentation

6.128.1.1 #define ERR_SUCCESS 0

6.128.1.2 #define ERR_IHANDLE -1

6.128.1.3 #define ERR_USER -1000

6.128.1.4 #define MAX_USER_ERR 0x2000

6.128.1.5 #define MHA_ErrorMsg2(
 x,
 y) MHA_Error(__FILE__, __LINE__, (x), (y))

6.128.1.6 #define MHA_ErrorMsg3(
 x,
 y,
 z) MHA_Error(__FILE__, __LINE__, (x), (y), (z))

6.128.1.7 #define MIN_TCP_PORT 0

6.128.1.8 #define MIN_TCP_PORT_STR "0"

6.128.1.9 #define MAX_TCP_PORT 65535

6.128.1.10 #define MAX_TCP_PORT_STR "65535"

6.128.1.11 #define IOInit MHA_STATIC_MHAIOTCP_IOInit

6.128.1.12 #define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare

6.128.1.13 #define IOStart MHA_STATIC_MHAIOTCP_IOStart

6.128.1.14 #define IOStop MHA_STATIC_MHAIOTCP_IOStop

6.128.1.15 #define IORelease MHA_STATIC_MHAIOTCP_IORelease

6.128.1.16 #define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar

6.128.1.17 #define IOStrError MHA_STATIC_MHAIOTCP_IOStrError

6.128.1.18 `#define IODestroy MHA_STATIC_MHAIOTCP_IODestroy`

6.128.1.19 `#define dummy_interface_test MHA_STATIC_MHAIOTCP_dummy_interface_test`

6.128.2 Function Documentation

6.128.2.1 `static int copy_error (`
 `MHA_Error & e) [static]`

6.128.2.2 `static void* thread_startup_function (`
 `void * parameter) [static]`

6.128.2.3 `int IOInit (`
 `int fragsize,`
 `float samplerate,`
 `IOProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle,`
 `void ** handle)`

6.128.2.4 `int IOPrepare (`
 `void * handle,`
 `int num_inchannels,`
 `int num_outchannels)`

6.128.2.5 `int IOStart (`
 `void * handle)`

6.128.2.6 `int IOStop (`
 `void * handle)`

6.128.2.7 `int IORelease (`
 `void * handle)`

6.128.2.8 `int IOSetVar (`
 `void * handle,`
 `const char * cmd,`
 `char * retval,`
 `unsigned int len)`

6.128.2.9 `const char* IOStrError (`
 `void * handle,`
 `int err)`

6.128.2.10 void IODestroy (
void * *handle*)

6.128.2.11 void dummy_interface_test (
void)

6.128.3 Variable Documentation

6.128.3.1 char user_err_msg[MAX_USER_ERR] [static]

6.129 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

6.129.1 Function Documentation

6.129.1.1 static void jack_error_handler (
const char * *msg*) [static]

6.129.1.2 static int dummy_jack_proc_cb (
jack_nframes_t,
void *) [static]

6.129.1.3 void make_friendly_number (
jack_default_audio_sample_t & x) [inline]

6.129.2 Variable Documentation

6.129.2.1 char last_jack_err_msg[MAX_USER_ERR] = ""

6.129.2.2 int last_jack_err = 0

6.130 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]`

6.130.1 Macro Definition Documentation

6.130.1.1 `#define MHAJACK_FW_STARTED 1`

6.130.1.2 `#define MHAJACK_STOPPED 2`

6.130.1.3 `#define MHAJACK_STARTING 8`

6.130.1.4 `#define IO_ERROR_JACK 11`

6.130.1.5 `#define IO_ERROR_MHAJACKLIB 12`

6.130.1.6 `#define MAX_USER_ERR 0x500`

6.130.2 Variable Documentation

6.130.2.1 `char last_jack_err_msg[MAX_USER_ERR]`

6.131 mhamain.cpp File Reference

Classes

- class **mhaserver_t**
MHA Framework listening on TCP port for commands.

Macros

- `#define MAX_LINE_LENGTH 0x100000`
- `#define HELP_TEXT`
- `#define GREETING_TEXT`

Functions

- void **create_lock** (unsigned int p, std::string s)
- void **remove_lock** (unsigned int p)
- int **mhamain** (int argc, char *argv[])

6.131.1 Macro Definition Documentation

6.131.1.1 `#define MAX_LINE_LENGTH 0x100000`6.131.1.2 `#define HELP_TEXT`6.131.1.3 `#define GREETING_TEXT`

6.131.2 Function Documentation

6.131.2.1 `void create_lock (`
 `unsigned int p,`
 `std::string s)`

6.131.2.2 `void remove_lock (`
 `unsigned int p)`

6.131.2.3 `int mhamain (`
 `int argc,`
 `char * argv[])`

6.132 mhappluginloader.cpp File Reference

6.133 mhappluginloader.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::mhappluginloader_t**
- class **MHAParser::mhappluginloader_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. 444) structures, and report differences as an error.*

6.134 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)`

6.134.1 Function Documentation

6.134.1.1 `void write_wave (`
 `const mha_wave_t & sig,`
 `const char * fname,`
 `const float & srate,`
 `const int & format)`

6.134.1.2 `unsigned int validator_channels (`
 `std::vector< int > channel_map,`
 `unsigned int channels)`

6.134.1.3 `unsigned int validator_length (`
 `unsigned int maxlen,`
 `unsigned int frames,`
 `unsigned int startpos)`

6.135 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)

6.135.1 Function Documentation

6.135.1.1 void write_wave (
 const **mha_wave_t** & sig,
 const char * fname,
 const float & srate = 44100,
 const int & format = SF_FORMAT_WAV|SF_FORMAT_FLOAT|SF_ENDIAN↵
 _FILE)

6.136 multibandcompressor.cpp File Reference

Classes

- class **multibandcompressor::plugin_signals_t**
- class **multibandcompressor::fftfb_plug_t**
- class **multibandcompressor::interface_t**

Namespaces

- **multibandcompressor**

6.137 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)

6.137.1 Macro Definition Documentation

6.137.1.1 **#define** NORMALIZATION_TYPES "[none default sum]"

6.137.1.2 **#define** NORM_NONE 0

6.137.1.3 **#define** NORM_DEFAULT 1

6.137.1.4 **#define** NORM_SUM 2

6.137.1.5 **#define** ESTIMATION_TYPES "[previous current]"

6.137.1.6 **#define** ESTIM_PREV 0

6.137.1.7 **#define** ESTIM_CUR 1

6.137.2 Function Documentation

6.137.2.1 void **make_friendly_number_by_limiting** (
 mha_real_t & x) [inline]

6.138 noise.cpp File Reference

Classes

- class **cfg_t**
- class **noise_t**

6.139 noisePowProposedScale.cpp File Reference

Classes

- class **noisePowProposedScale::noisePowProposed**
- class **noisePowProposedScale::interface_t**

Namespaces

- **noisePowProposedScale**

Macros

- `#define POWSPEC_FACTOR 0.0025`

6.139.1 Macro Definition Documentation

6.139.1.1 `#define POWSPEC_FACTOR 0.0025`

6.140 overlapadd.cpp File Reference

Classes

- class `overlapadd::overlapadd_t`
- class `overlapadd::overlapadd_if_t`

Namespaces

- `overlapadd`

6.141 pluginbrowser.cpp File Reference

6.142 pluginbrowser.h File Reference

Classes

- class `plugindescription_t`
- class `pluginloader_t`
- class `pluginbrowser_t`

6.143 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)`

6.143.1 Macro Definition Documentation

6.143.1.1 `#define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this,
 &prediction_error::update_cfg)`

6.143.1.2 `#define INSERT_PATCH(
 var) insert_member(var); PATCH_VAR(var)`

6.143.2 Function Documentation

6.143.2.1 `void make_friendly_number_by_limiting (
 mha_real_t & x) [inline]`

6.144 prediction_error.h File Reference

Classes

- class **prediction_error_config**
- class **prediction_error**

6.145 resampling.cpp File Reference

Classes

- class **MHAPugin_Resampling::resampling_t**
- class **MHAPugin_Resampling::resampling_if_t**

Namespaces

- **MHAPugin_Resampling**

6.146 rmslevel.cpp File Reference

Classes

- class **mon_t**
- class **rmslevel_t**
- class **rmslevel_if_t**

6.147 route.cpp File Reference

Classes

- class **route::process_t**
- class **route::interface_t**

Namespaces

- **route**

6.148 save_spec.cpp File Reference

Classes

- class **save_spec_t**

6.149 save_wave.cpp File Reference

Classes

- class **save_wave_t**

6.150 shadowfilter_begin.cpp File Reference

Classes

- class **shadowfilter_begin::cfg_t**
- class **shadowfilter_begin::shadowfilter_begin_t**

Namespaces

- **shadowfilter_begin**

6.151 shadowfilter_end.cpp File Reference

Classes

- class **shadowfilter_end::cfg_t**
- class **shadowfilter_end::shadowfilter_end_t**

Namespaces

- **shadowfilter_end**

6.152 sine.cpp File Reference

Classes

- struct **sine_cfg_t**
- class **sine_t**

6.153 smoothgains_bridge.cpp File Reference

Classes

- class **smoothgains_bridge::smoothspec_wrap_t**
- class **smoothgains_bridge::overlapadd_if_t**

Namespaces

- **smoothgains_bridge**

6.154 softclip.cpp File Reference

Classes

- class **cfg_t**
- class **softclip_t**

6.155 spec2wave.cpp File Reference

Classes

- class **hanning_ramps_t**
- class **spec2wave_t**
- class **spec2wave_if_t**

Functions

- unsigned int **max** (unsigned int a, unsigned int b)
- unsigned int **min** (unsigned int a, unsigned int b)

6.155.1 Function Documentation

6.155.1.1 unsigned int max (
 unsigned int *a*,
 unsigned int *b*) [inline]

6.155.1.2 unsigned int min (
 unsigned int *a*,
 unsigned int *b*) [inline]

6.156 speechnoise.cpp File Reference

Macros

- #define **NUM_ENTR_MHAORIG** 76
- #define **NUM_ENTR_LTASS** 25
- #define **NUM_ENTR_OLNOISE** 49

Functions

- float **fhz2bandno** (float *x*)
- float **erb_hz_f_hz** (float *f_hz*)
- float **hz2hz** (float *x*)
 Dummy scale transformation Hz to Hz.
- float **bandw_correction** (float *f*, float *ldb*)

Variables

- float **vMHAOrigSpec** [**NUM_ENTR_MHAORIG**] = {-1.473, 0, -4.939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13, -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.14, -27.55, -25.79, -25.89, -26.11, -27.↵
 48, -30.37, -33.13, -36.23, -36.64, -36.35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85, -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.↵
 53, -38.71, -38.7, -38.92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.↵
 43, -77.43}
- float **vMHAOrigFreq** [**NUM_ENTR_MHAORIG**] = {172.266,344.532,516.797,689.↵
 063,861.329,1033.59,1205.86,1378.13,1550.39,1722.66,1894.92,2067.19,2239.↵
 46,2411.72,2583.99,2756.25,2928.52,3100.78,3273.05,3445.32,3617.58,3789.85,3962.↵
 11,4134.38,4306.64,4478.91,4651.18,4823.44,4995.71,5167.97,5340.24,5512.51,5684.↵
 77,5857.04,6029.3,6201.57,6373.83,6546.1,6718.37,6890.63,7062.9,7235.16,7407.↵
 43,7579.69,7751.96,7924.23,8096.49,8268.76,8441.02,8613.29,8785.56,8957.82,9130.↵
 09,9302.35,9474.62,9646.88,9819.15,9991.42,10163.7,10335.9,10508.2,10680.↵
 5,10852.7,11025,11197.3,11369.5,11541.8,11714.1,11886.3,12058.6,12230.9,12403.↵
 1,12575.4,12747.7,12919.9,13092.2}

- float **vLTASS_freq** [NUM_ENTR_LTASS] = {63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000}
- float **vLTASS_combined_lev** [NUM_ENTR_LTASS] = {38.6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7}
- float **vLTASS_female_lev** [NUM_ENTR_LTASS] = {37.0,36.0,37.5,40.1,53.4,62.2,60.↵
9,58.1,61.7,61.7,60.4,58,54.3,52.3,51.7,48.8,47.3,46.7,45.3,44.6,45.2,44.9,45.0,42.↵
8,41.1}
- float **vLTASS_male_lev** [NUM_ENTR_LTASS] = {38.6,43.5,54.4,57.7,56.8,58.2,59.↵
7,60.0,62.4,62.6,60.6,55.7,53.1,53.7,52.3,48.7,48.9,47.0,46.0,44.4,43.3,42.4,41.9,39.↵
8,40.4}
- float **vOlnoiseFreq** [NUM_ENTR_OLNOISE] = {62.5,70.1539,78.7451,88.3884,99.↵
2126,111.362,125,140.308,157.49,176.777,198.425,222.725,250,280.616,314.98,353.↵
553,396.85,445.449,500,561.231,629.961,707.107,793.701,890.899,1000,1122.↵
46,1259.92,1414.21,1587.4,1781.8,2000,2244.92,2519.84,2828.43,3174.8,3563.↵
59,4000,4489.85,5039.68,5656.85,6349.6,7127.19,8000,8979.7,10079.4,11313.↵
7,12699.2,14254.4,16000}
- float **vOlnoiseLev** [NUM_ENTR_OLNOISE] = {45.9042,38.044,48.9444,61.3697,67.↵
6953,69.7451,71.6201,71.2431,65.2754,63.2547,70.2264,72.1434,73.4433,73.2659,69.↵
8424,71.0132,70.9577,70.3492,68.691,64.8436,64.0435,64.2879,60.5889,60.6596,60.↵
3727,61.2003,61.8477,61.1478,61.2312,58.6584,57.2892,56.8299,56.0191,53.3018,56.↵
0525,54.3592,50.8823,55.992,54.6768,47.2616,46.9914,45.209,50.413,47.5848,43.↵
3215,43.754,38.5773,-0.39427,5.74224}

6.156.1 Macro Definition Documentation

6.156.1.1 **#define NUM_ENTR_MHAORIG 76**

6.156.1.2 **#define NUM_ENTR_LTASS 25**

6.156.1.3 **#define NUM_ENTR_OLNOISE 49**

6.156.2 Function Documentation

6.156.2.1 **float fhz2bandno (**
 float x)

6.156.2.2 **float erb_hz_f_hz (**
 float f_hz)

6.156.2.3 **float hz2hz (**
 float x)

Dummy scale transformation Hz to Hz.

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

Parameters

| | |
|----------|-----------------------|
| <i>x</i> | Input frequency in Hz |
|----------|-----------------------|

Returns

Frequency in Hz

6.156.2.4 float bandw_correction (
float *f*,
float *ldb*)

6.156.3 Variable Documentation

- 6.156.3.1 float vMHAOrigSpec[NUM_ENTR_MHAORIG] = {-1.473, 0, -4.939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13, -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.14, -27.55, -25.79, -25.89, -26.11, -27.48, -30.37, -33.13, -36.23, -36.64, -36.35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85, -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.53, -38.71, -38.7, -38.92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.43, -77.43}
- 6.156.3.2 float vMHAOrigFreq[NUM_ENTR_MHAORIG] = {172.266,344.532,516.797,689.063,861.↵
329,1033.59,1205.86,1378.13,1550.39,1722.66,1894.92,2067.19,2239.46,2411.72,2583.99,2756.↵
25,2928.52,3100.78,3273.05,3445.32,3617.58,3789.85,3962.11,4134.38,4306.64,4478.91,4651.↵
18,4823.44,4995.71,5167.97,5340.24,5512.51,5684.77,5857.04,6029.3,6201.57,6373.83,6546.↵
1,6718.37,6890.63,7062.9,7235.16,7407.43,7579.69,7751.96,7924.23,8096.49,8268.76,8441.↵
02,8613.29,8785.56,8957.82,9130.09,9302.35,9474.62,9646.88,9819.15,9991.42,10163.7,10335.↵
9,10508.2,10680.5,10852.7,11025,11197.3,11369.5,11541.8,11714.1,11886.3,12058.6,12230.↵
9,12403.1,12575.4,12747.7,12919.9,13092.2}
- 6.156.3.3 float vLTASS_freq[NUM_ENTR_LTASS] = {63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000}
- 6.156.3.4 float vLTASS_combined_lev[NUM_ENTR_LTASS] = {38.6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7}
- 6.156.3.5 float vLTASS_female_lev[NUM_ENTR_LTASS] = {37.0,36.0,37.5,40.1,53.4,62.2,60.9,58.↵
1,61.7,61.7,60.4,58,54.3,52.3,51.7,48.8,47.3,46.7,45.3,44.6,45.2,44.9,45.0,42.8,41.1}
- 6.156.3.6 float vLTASS_male_lev[NUM_ENTR_LTASS] = {38.6,43.5,54.4,57.7,56.8,58.2,59.7,60.0,62.↵
4,62.6,60.6,55.7,53.1,53.7,52.3,48.7,48.9,47.0,46.0,44.4,43.3,42.4,41.9,39.8,40.4}
- 6.156.3.7 float vOlnoiseFreq[NUM_ENTR_OLNOISE] = {62.5,70.1539,78.7451,88.3884,99.2126,111.↵
362,125,140.308,157.49,176.777,198.425,222.725,250,280.616,314.98,353.553,396.85,445.↵
449,500,561.231,629.961,707.107,793.701,890.899,1000,1122.46,1259.92,1414.21,1587.4,1781.↵
8,2000,2244.92,2519.84,2828.43,3174.8,3563.59,4000,4489.85,5039.68,5656.85,6349.6,7127.↵
19,8000,8979.7,10079.4,11313.7,12699.2,14254.4,16000}

6.156.3.8 float vOlnoiseLev[NUM_ENTR_OLNOISE] = {45.9042,38.044,48.9444,61.3697,67.6953,69.↵
 7451,71.6201,71.2431,65.2754,63.2547,70.2264,72.1434,73.4433,73.2659,69.8424,71.0132,70.↵
 9577,70.3492,68.691,64.8436,64.0435,64.2879,60.5889,60.6596,60.3727,61.2003,61.8477,61.↵
 1478,61.2312,58.6584,57.2892,56.8299,56.0191,53.3018,56.0525,54.3592,50.8823,55.992,54.↵
 6768,47.2616,46.9914,45.209,50.413,47.5848,43.3215,43.754,38.5773,-0.39427,5.74224}

6.157 speechnoise.h File Reference

Classes

- class **speechnoise_t**

6.158 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. 682) instance manages the plugin that performs processing on the reduced set of channels.*
- class **MHAPLugin_Split::split_t**
Implements split plugin.

Namespaces

- **MHAPLugin_Split**

Macros

- **#define MHAPLUGIN_OVERLOAD_OUTDOMAIN**
This define modifies the definition of MHAPLUGIN_CALLBACKS and friends.
- **#define posixthreads 1**
- **#define default_thread_platform_string "posix"**
- **#define default_thread_platform_type posix_threads_t**

Enumerations

6.158.1 Detailed Description

Source code for the split plugin. The split plugin splits the audio signal by channel. The splitted paths execute in parallel.

6.158.2 Macro Definition Documentation

6.158.2.1 `#define MHAPLUGIN_OVERLOAD_OUTDOMAIN`

This define modifies the definition of MHAPLUGIN_CALLBACKS and friends.

The output signal is transferred through a second parameter to the process method, enabling all four domain transformations in a single plugin.

6.158.2.2 `#define posixthreads 1`

6.158.2.3 `#define default_thread_platform_string "posix"`

6.158.2.4 `#define default_thread_platform_type posix_threads_t`

6.159 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)`

6.159.1 Macro Definition Documentation

6.159.1.1 `#define PATCH_VAR(var) patchbay.connect(&var.valuechanged, this, &steerbf::update_cfg)`

6.159.1.2 `#define INSERT_PATCH(var) insert_member(var); PATCH_VAR(var)`

6.160 steerbf.h File Reference

Classes

- class `parser_int_dyn`
- class `steerbf_config`
- class `steerbf`

6.161 testalsadevice.c File Reference

Functions

- int **main** (int argc, char **argv)

6.161.1 Function Documentation

6.161.1.1 int main (
 int *argc*,
 char ** *argv*)

6.162 timoconfig.cpp File Reference

Macros

- #define **LPSCALE** (5.2429e+007)
- #define **POWSPEC_FACTOR** 0.0025
- #define **OVERLAP_FACTOR** 2
- #define **EPSILON** (1e-10)
- #define **CHANLOOP** for (unsigned int c=0; c<nchan; ++c)

6.162.1 Macro Definition Documentation

6.162.1.1 #define LPSCALE (5.2429e+007)

6.162.1.2 #define POWSPEC_FACTOR 0.0025

6.162.1.3 #define OVERLAP_FACTOR 2

6.162.1.4 #define EPSILON (1e-10)

6.162.1.5 #define CHANLOOP for (unsigned int c=0; c<nchan; ++c)

6.163 timoconfig.h File Reference

Classes

- class **timo_AC**
- class **timo_params**
- class **timoConfig**

6.164 timoSmooth.cpp File Reference

Macros

- `#define INSERT_VAR(var) insert_item(#var, &var)`
- `#define PATCH_VAR(var) patchbay.connect(&var.valuechanged, this, &timoSmooth::on_model_param_valuechanged)`
- `#define INSERT_PATCH(var) INSERT_VAR(var); PATCH_VAR(var)`

6.164.1 Macro Definition Documentation

6.164.1.1 `#define INSERT_VAR(
var) insert_item(#var, &var)`

6.164.1.2 `#define PATCH_VAR(
var) patchbay.connect(&var.valuechanged, this,
&timoSmooth::on_model_param_valuechanged)`

6.164.1.3 `#define INSERT_PATCH(
var) INSERT_VAR(var); PATCH_VAR(var)`

6.165 timosmooth.h File Reference

Classes

- class **timoSmooth**

6.166 transducers.cpp File Reference

Classes

- class **softclipper_variables_t**
- class **softclipper_t**
- class **calibrator_variables_t**
- class **calibrator_runtime_layer_t**
- class **calibrator_t**
- class **bbcalib_interface_t**

Typedefs

- typedef **MHAPLugin::config_t**< **MHASignal::async_rmslevel_t** > **rmslevelmeter**
- typedef **MHAPLugin::plugin_t**< **calibrator_runtime_layer_t** > **rtcalibrator**

Functions

- **speechnoise_t::noise_type_t kw_index2type** (unsigned int *idx*)
- **std::vector< int > vint_0123n1** (unsigned int *n*)

6.166.1 Typedef Documentation

6.166.1.1 **typedef MHAPLugin::config_t<MHASignal::async_rmslevel_t> rmslevelmeter**

6.166.1.2 **typedef MHAPLugin::plugin_t<calibrator_runtime_layer_t> rtcalibrator**

6.166.2 Function Documentation

6.166.2.1 **speechnoise_t::noise_type_t kw_index2type** (
unsigned int *idx*)

6.166.2.2 **std::vector<int> vint_0123n1** (
unsigned int *n*)

6.167 upsample.cpp File Reference

Classes

- class **us_t**

6.168 wave2spec.cpp File Reference

Classes

- class **wave2spec_t**
- class **wave2spec_if_t**

Macros

- **#define MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.168.1 Macro Definition Documentation

6.168.1.1 **#define MHAPLUGIN_OVERLOAD_OUTDOMAIN**

6.169 wavrec.cpp File Reference

Classes

- class **wavwriter_t**
- class **wavrec_t**

Macros

- `#define DEBUG(x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x << std::endl`

6.169.1 Macro Definition Documentation

6.169.1.1 `#define DEBUG(
x) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x
<< std::endl`

6.170 windowselector.cpp File Reference

6.171 windowselector.h File Reference

Classes

- class **windowselector_t**
A combination of mha parser variables to describe an overalapadd analysis window.

Index

- `__MHA_FUN__`
 - `mha_defs.h`, 908
- `__attribute__`
 - `mha_plugin.hh`, 931
- `__declspec`
 - `example5.cpp`, 892
 - `example6.cpp`, 892
 - `mha_defs.h`, 908
 - `mha_plugin.hh`, 931
- `_cf`
 - `DynComp::dc_afterburn_t`, 259
- `_channels`
 - `DynComp::dc_afterburn_t`, 259
- `_conjugate`
 - Complex arithmetics in the openMHA, 67
- `_linphase_asym`
 - `MHAFilter::smoothspec_t`, 508
- `_reciprocal`
 - Complex arithmetics in the openMHA, 68
- `_srate`
 - `DynComp::dc_afterburn_t`, 259
- `_steerbf`
 - `steerbf_config`, 854
- `~Async_Notify`
 - `MHA_TCP::Async_Notify`, 409
- `~Connection`
 - `MHA_TCP::Connection`, 413
- `~Delay`
 - `ADM::Delay`, 161
- `~Event_Watcher`
 - `MHA_TCP::Event_Watcher`, 419
- `~Linearphase_FIR`
 - `ADM::Linearphase_FIR`, 163
- `~MHA_Error`
 - `MHA_Error`, 388
- `~Server`
 - `MHA_TCP::Server`, 422
- `~Thread`
 - `MHA_TCP::Thread`, 428
- `~Timeout_Watcher`
 - `MHA_TCP::Timeout_Watcher`, 432
- `~Wakeup_Event`
 - `MHA_TCP::Wakeup_Event`, 434
- `~acConcat_wave`
 - `acConcat_wave`, 129
- `~acConcat_wave_config`
 - `acConcat_wave_config`, 132
- `~acPooling_wave`
 - `acPooling_wave`, 139
- `~acPooling_wave_config`
 - `acPooling_wave_config`, 141
- `~acSteer`
 - `acSteer`, 151
- `~acSteer_config`
 - `acSteer_config`, 153
- `~acTransform_wave`
 - `acTransform_wave`, 155
- `~acTransform_wave_config`
 - `acTransform_wave_config`, 157
- `~acmon_t`
 - `acmon::acmon_t`, 136
- `~acspace2matrix_t`
 - `MHA_AC::acspace2matrix_t`, 360
- `~adm_rtconfig_t`
 - `adm_rtconfig_t`, 169
- `~algo_comm_class_t`
 - `MHAKernel::algo_comm_class_t`, 539
- `~analysepath_t`
 - `analysepath_t`, 180
- `~analysispath_if_t`
 - `analysispath_if_t`, 183
- `~bark2hz_t`
 - `MHAOvfFilter::barkscale::bark2hz_t`, 548
- `~base_t`
 - `MHAParser::base_t`, 571
- `~bbcalib_interface_t`
 - `bbcalib_interface_t`, 192
- `~blockprocessing_polyphase_resampling_t`
 - `MHAFilter::blockprocessing_polyphase_resampling_t`, 458
- `~c_ifc_parser_t`
 - `MHAParser::c_ifc_parser_t`, 583
- `~cfg_chain_t`
 - `MHAPlugin::cfg_chain_t`, 654
- `~cfg_t`
 - `acsave::cfg_t`, 146
- `~config_t`
 - `MHAPlugin::config_t`, 656
- `~connector_base_t`
 - `MHAEvents::connector_base_t`, 446
- `~connector_t`
 - `MHAEvents::connector_t`, 449
- `~db_if_t`
 - `db_if_t`, 213
- `~delay_spec_t`
 - `MHASignal::delay_spec_t`, 697

- ~delay_t
 - MHASignal::delay_t, 698
- ~delay_wave_t
 - MHASignal::delay_wave_t, 700
- ~doasvm_classification
 - doasvm_classification, 243
- ~doasvm_classification_config
 - doasvm_classification_config, 245
- ~doasvm_feature_extraction
 - doasvm_feature_extraction, 247
- ~doasvm_feature_extraction_config
 - doasvm_feature_extraction_config, 249
- ~domain_handler_t
 - MHAPLugin_Split::domain_handler_t, 669
- ~double_t
 - MHA_AC::double_t, 362
- ~doublebuffer_t
 - MHASignal::doublebuffer_t, 702
- ~dynamiclib_t
 - dynamiclib_t, 255
- ~emitter_t
 - MHAEvents::emitter_t, 451
- ~fft_t
 - MHASignal::fft_t, 704
- ~fftfb_t
 - MHAOvFilter::fftfb_t, 551
- ~fftfiler_t
 - MHAFilter::fftfiler_t, 465
- ~fftfilerbank_t
 - MHAFilter::fftfilerbank_t, 469
- ~filter_t
 - MHAFilter::filter_t, 473
- ~float_t
 - MHA_AC::float_t, 363
- ~fourway_processor_t
 - PluginLoader::fourway_processor_t, 803
- ~fw_t
 - fw_t, 296
- ~gaintable_t
 - DynComp::gaintable_t, 264
- ~gammaflt_t
 - MHAFilter::gammaflt_t, 476
- ~hanning_ramps_t
 - hanning_ramps_t, 304
- ~hilbert_shifter_t
 - hilbert_shifter_t, 306
- ~hilbert_t
 - MHASignal::hilbert_t, 708
- ~hz2bark_t
 - MHAOvFilter::barkscale::hz2bark_t, 549
- ~int_t
 - MHA_AC::int_t, 364
- ~io_file_t
 - io_file_t, 311
- ~io_lib_t
 - io_lib_t, 315
- ~io_parser_t
 - io_parser_t, 318
- ~io_portaudio_t
 - MHAIOPortAudio::io_portaudio_t, 521
- ~io_tcp_fwcb_t
 - io_tcp_fwcb_t, 321
- ~io_tcp_parser_t
 - io_tcp_parser_t, 325
- ~io_tcp_sound_t
 - io_tcp_sound_t, 330
- ~io_tcp_t
 - io_tcp_t, 335
- ~linear_table_t
 - MHATableLookup::linear_table_t, 757
- ~lpc
 - lpc, 339
- ~lpc_bl_predictor
 - lpc_bl_predictor, 342
- ~lpc_bl_predictor_config
 - lpc_bl_predictor_config, 344
- ~lpc_burglattice
 - lpc_burglattice, 346
- ~lpc_burglattice_config
 - lpc_burglattice_config, 348
- ~lpc_config
 - lpc_config, 350
- ~matrix_t
 - MHASignal::matrix_t, 716
- ~mha_dblbuf_t
 - mha_dblbuf_t, 377
- ~mha_fifo_lw_t
 - mha_fifo_lw_t, 390
- ~mha_fifo_posix_threads_t
 - mha_fifo_posix_threads_t, 393
- ~mha_fifo_t
 - mha_fifo_t, 396
- ~mha_fifo_thread_guard_t
 - mha_fifo_thread_guard_t, 399
- ~mha_fifo_thread_platform_t
 - mha_fifo_thread_platform_t, 400
- ~mha_rt_fifo_element_t
 - mha_rt_fifo_element_t, 403
- ~mha_rt_fifo_t
 - mha_rt_fifo_t, 404
- ~mhaplug_cfg_t
 - mhaplug_cfg_t, 653

- ~mhappluginloader_t
 - MHAParser::mhappluginloader_t, [617](#)
 - PluginLoader::mhappluginloader_t, [807](#)
- ~mhaserver_t
 - mhaserver_t, [693](#)
- ~overlapadd_if_t
 - overlapadd::overlapadd_if_t, [791](#)
 - smoothgains_bridge::overlapadd_if_t, [838](#)
- ~overlapadd_t
 - overlapadd::overlapadd_t, [793](#)
- ~parser_t
 - MHAParser::parser_t, [622](#)
- ~partitioned_convolution_t
 - MHAFilter::partitioned_convolution_t, [496](#)
- ~patchbay_t
 - MHAEvents::patchbay_t, [452](#)
- ~plug_t
 - plug_t, [796](#)
- ~plugin_t
 - MHAPLugin::plugin_t, [661](#)
- ~pluginloader_t
 - pluginloader_t, [811](#)
- ~plugs_t
 - mhachain::plugs_t, [442](#)
- ~port_t
 - MHAJack::port_t, [536](#)
- ~posix_threads_t
 - MHAPLugin_Split::posix_threads_t, [676](#)
- ~prediction_error
 - prediction_error, [812](#)
- ~prediction_error_config
 - prediction_error_config, [815](#)
- ~rt_nlms_t
 - rt_nlms_t, [824](#)
- ~save_var_t
 - acsave::save_var_t, [148](#)
- ~sf_t
 - MHASndFile::sf_t, [754](#)
- ~sine_t
 - sine_t, [836](#)
- ~smoothspec_t
 - MHAFilter::smoothspec_t, [507](#)
- ~spec2wave_t
 - spec2wave_t, [847](#)
- ~spec_fader_t
 - spec_fader_t, [849](#)
- ~spectrum_t
 - MHA_AC::spectrum_t, [366](#)
 - MHASignal::spectrum_t, [733](#)
- ~split_t
 - MHAPLugin_Split::split_t, [680](#)
- ~splitted_part_t
 - MHAPLugin_Split::splitted_part_t, [684](#)
- ~steerbf
 - steerbf, [852](#)
- ~steerbf_config
 - steerbf_config, [854](#)
- ~table_t
 - MHATableLookup::table_t, [760](#)
- ~thread_platform_t
 - MHAPLugin_Split::thread_platform_t, [688](#)
- ~timoConfig
 - timoConfig, [860](#)
- ~timoSmooth
 - timoSmooth, [863](#)
- ~uint_vector_t
 - MHASignal::uint_vector_t, [741](#)
- ~uni_processor_t
 - MHAPLugin_Split::uni_processor_t, [690](#)
- ~wave2spec_t
 - wave2spec_t, [870](#)
- ~waveform_t
 - MHA_AC::waveform_t, [369](#)
 - MHASignal::waveform_t, [745](#)
- ~wavwriter_t
 - wavwriter_t, [874](#)
- ~windowselector_t
 - windowselector_t, [876](#)
- A
 - lpc_config, [351](#)
 - MHAFilter::filter_t, [474](#)
 - MHAFilter::gammaflt_t, [477](#)
 - MHAFilter::iir_filter_t, [482](#)
- a
 - MHAParser::base_t::replace_t, [577](#)
- A_
 - MHAFilter::complex_bandpass_t, [462](#)
 - MHAFilter::iir_ord1_real_t, [485](#)
- AC_DIM_MISMATCH
 - mha_algo_comm.cpp, [905](#)
- AC_INVALID_HANDLE
 - mha_algo_comm.cpp, [905](#)
- AC_INVALID_NAME
 - mha_algo_comm.cpp, [905](#)
- AC_INVALID_OUTPTR
 - mha_algo_comm.cpp, [905](#)
- AC_STRING_TRUNCATED
 - mha_algo_comm.cpp, [905](#)
- AC_SUCCESS
 - mha_algo_comm.cpp, [905](#)
- AC_TYPE_MISMATCH

- mha_algo_comm.cpp, 905
- ACSAVE_FMT_MAT4
 - acsave.cpp, 880
- ACSAVE_FMT_TXT
 - acsave.cpp, 880
- ACSAVE_FMT_M
 - acsave.cpp, 880
- ACSAVE_SFMT_MAT4
 - acsave.cpp, 880
- ACSAVE_SFMT_TXT
 - acsave.cpp, 880
- ACSAVE_SFMT_M
 - acsave.cpp, 880
- ADM::ADM< F >, 158
- ADM::ADM
 - ADM, 158
 - beta, 159
 - m_beta, 160
 - m_decomb, 160
 - m_delay_back, 160
 - m_delay_front, 159
 - m_lp_bf, 160
 - m_lp_result, 160
 - m_mu_beta, 160
 - m_powerfilter_coeff, 160
 - m_powerfilter_norm, 160
 - m_powerfilter_state, 160
 - process, 159
- ADM::Delay
 - ~Delay, 161
 - Delay, 161
 - m_coeff, 162
 - m_fullsamples, 162
 - m_norm, 162
 - m_now_in, 162
 - m_state, 162
 - process, 161
- ADM::Delay< F >, 160
- ADM::Linearphase_FIR< F >, 162
- ADM::Linearphase_FIR
 - ~Linearphase_FIR, 163
 - Linearphase_FIR, 163
 - m_alphas, 164
 - m_now, 164
 - m_order, 164
 - m_output, 164
 - process, 164
- ADM, 77
 - ADM::ADM, 158
 - C, 78
 - DELAY_FREQ, 78
 - PI, 78
 - START_BETA, 78
 - subsampledelay_coeff, 78
- ALGO_COMM_ID_STR
 - mha_algo_comm.hh, 907
- ASSERT_EQUAL_DIM_PTR
 - mha_signal.cpp, 936
- ASSERT_EQUAL_DIM
 - mha_signal.cpp, 936
- ASYNC_CONNECT_STARTED
 - mha_tcp.cpp, 948
- abandonned
 - mha_rt_fifo_element_t, 403
- abs
 - Complex arithmetics in the openMHA, 66
- abs2
 - Complex arithmetics in the openMHA, 66
- ac
 - ac2wave_t, 127
 - acConcat_wave_config, 132
 - acPooling_wave_config, 141
 - acTransform_wave_config, 157
 - acmon::acmon_t, 137
 - acsave::cfg_t, 147
 - acsave::save_var_t, 149
 - doasvm_classification_config, 245
 - fw_t, 298
 - latex_doc_t, 338
 - lpc_bl_predictor_config, 344
 - lpc_burglattice_config, 348
 - MHA_AC::ac2matrix_helper_t, 356
 - MHA_AC::double_t, 362
 - MHA_AC::float_t, 363
 - MHA_AC::int_t, 364
 - MHA_AC::spectrum_t, 367
 - MHA_AC::waveform_t, 370
 - MHAKernel::algo_comm_class_t, 540
 - MHAMultiSrc::base_t, 542
 - MHAPlugin::plugin_t, 662
 - mhachain::plugs_t, 442
 - PluginLoader::mhapluginloader_t, 809
 - prediction_error_config, 815
 - rt_nlms_t, 824
 - shadowfilter_end::cfg_t, 832
 - steerbf_config, 854
 - timoConfig, 860
- AC variable, 4
- ac2matrix_helper_t
 - MHA_AC::ac2matrix_helper_t, 355
- ac2matrix_t
 - MHA_AC::ac2matrix_t, 357

- ac2wave.cpp, 878
- ac2wave_if_t, 125
 - ac2wave_if_t, 126
 - delay_ac, 126
 - delay_in, 126
 - gain_ac, 126
 - gain_in, 126
 - name, 126
 - patchbay, 126
 - prepare, 126
 - prepared, 126
 - process, 126
 - release, 126
 - update, 126
 - zeros, 126
- ac2wave_t, 127
 - ac, 127
 - ac2wave_t, 127
 - channels, 127
 - delay_ac, 128
 - delay_in, 128
 - frames, 127
 - gain_ac, 128
 - gain_in, 128
 - name, 128
 - process, 127
 - w, 127
- ac_
 - combc_t, 208
 - MHAParser::mhappluginloader_t, 618
- ac_fifo
 - analysepath_t, 181
- ac_monitor_t
 - acmon::ac_monitor_t, 133
- ac_monitor_type.cpp, 878
- ac_monitor_type.hh, 878
- acConcat_wave, 128
 - ~acConcat_wave, 129
 - acConcat_wave, 129
 - name_conAC, 131
 - num_AC, 131
 - patchbay, 131
 - prefix_names_AC, 131
 - prepare, 129
 - process, 129
 - release, 131
 - samples_AC, 131
 - update_cfg, 131
- acConcat_wave.cpp, 878
 - INSERT_PATCH, 878
 - PATCH_VAR, 878
- acConcat_wave.h, 878
- acConcat_wave_config, 131
 - ~acConcat_wave_config, 132
 - ac, 132
 - acConcat_wave_config, 132
 - numSamples_AC, 132
 - process, 132
 - strNames_AC, 132
 - vGCC_con, 132
 - vGCC, 132
- acPooling_wave, 138
 - ~acPooling_wave, 139
 - acPooling_wave, 139
 - alpha, 140
 - like_ratio_name, 140
 - lower_threshold, 140
 - max_pool_ind_name, 140
 - neighbourhood, 140
 - numsamples, 140
 - p_name, 140
 - patchbay, 140
 - pool_name, 140
 - pooling_type, 140
 - pooling_wndlen, 140
 - prepare, 139
 - process, 139
 - release, 140
 - update_cfg, 140
 - upper_threshold, 140
- acPooling_wave.cpp, 879
 - INSERT_PATCH, 879
 - PATCH_VAR, 879
- acPooling_wave.h, 879
- acPooling_wave_config, 141
 - ~acPooling_wave_config, 141
 - ac, 141
 - acPooling_wave_config, 141
 - alpha, 142
 - c, 142
 - insert, 141
 - like_ratio, 142
 - low_thresh, 142
 - neigh, 142
 - p, 142
 - p_max, 142
 - pool, 142
 - pooling_ind, 142
 - pooling_option, 142
 - pooling_size, 142
 - process, 141
 - raw_p_name, 142

- up_thresh, 142
- acSteer, 150
 - ~acSteer, 151
 - acSteer, 151
 - acSteerName1, 152
 - acSteerName2, 152
 - nrefmic, 152
 - nsteerchan, 152
 - patchbay, 152
 - prepare, 151
 - process, 151
 - release, 151
 - steerFile, 152
 - update_cfg, 152
- acSteer.cpp, 880
 - INSERT_PATCH, 880
 - PATCH_VAR, 880
- acSteer.h, 880
- acSteer_config, 152
 - ~acSteer_config, 153
 - acSteer_config, 153
 - insert, 153
 - nangle, 153
 - nchan, 153
 - nfreq, 153
 - nrefmic, 153
 - nsteerchan, 153
 - specSteer1, 153
 - specSteer2, 153
- acSteerName1
 - acSteer, 152
- acSteerName2
 - acSteer, 152
- acTransform_wave, 154
 - ~acTransform_wave, 155
 - acTransform_wave, 155
 - ang_name, 156
 - numsamples, 156
 - patchbay, 156
 - prepare, 155
 - process, 155
 - raw_p_max_name, 156
 - raw_p_name, 156
 - release, 155
 - rotated_p_max_name, 156
 - rotated_p_name, 156
 - to_from, 156
 - update_cfg, 156
- acTransform_wave.cpp, 881
 - INSERT_PATCH, 881
 - PATCH_VAR, 881
- acTransform_wave.h, 881
- acTransform_wave_config, 156
 - ~acTransform_wave_config, 157
 - ac, 157
 - acTransform_wave_config, 157
 - ang_name, 157
 - offset, 157
 - process, 157
 - raw_p_max_name, 157
 - raw_p_name, 157
 - resolution, 157
 - rotated_i, 157
 - rotated_p, 157
 - to_from, 157
- accept
 - MHA_TCP::Server, 422
- accept_event
 - MHA_TCP::Server, 423
- accept_loop
 - io_tcp_t, 335
- acceptor_started
 - mhaserver_t, 693
- ack_fail
 - mhaserver_t, 694
- ack_ok
 - mhaserver_t, 694
- acmon, 77
- acmon.cpp, 879
- acmon::ac_monitor_t, 132
 - ac_monitor_t, 133
 - dimstr, 134
 - getvar, 134
 - mon, 134
 - mon_complex, 134
 - mon_mat, 134
 - mon_mat_complex, 134
 - name, 134
 - p_parser, 135
 - use_mat, 135
- acmon::acmon_t, 135
 - ~acmon_t, 136
 - ac, 137
 - acmon_t, 136
 - algo, 137
 - b_cont, 137
 - b_snapshot, 137
 - chain, 137
 - dimensions, 137
 - dispmode, 137
 - patchbay, 137
 - prepare, 136

- process, 137
- recmode, 137
- release, 136
- save_vars, 137
- update_recmode, 137
- varlist, 137
- vars, 137
- acmon_t
 - acmon::acmon_t, 136
- acsave, 77
- acsave.cpp, 879
 - ACSAVE_FMT_MAT4, 880
 - ACSAVE_FMT_TXT, 880
 - ACSAVE_FMT_M, 880
 - ACSAVE_SFMT_MAT4, 880
 - ACSAVE_SFMT_TXT, 880
 - ACSAVE_SFMT_M, 880
- acsave::acsave_t, 143
 - acsave_t, 144
 - algo, 145
 - b_flushed, 145
 - b_prepared, 145
 - bflush, 145
 - chain, 145
 - event_start_recording, 145
 - event_stop_and_flush, 145
 - fileformat, 145
 - fname, 145
 - patchbay, 145
 - prepare, 144
 - process, 144, 145
 - reclen, 145
 - release, 144
 - variables, 145
 - varlist, 145
 - varlist_t, 144
- acsave::cfg_t, 146
 - ~cfg_t, 146
 - ac, 147
 - cfg_t, 146
 - flush_data, 146
 - max_frames, 147
 - nvars, 147
 - rec_frames, 147
 - store_frame, 146
 - varlist, 147
- acsave::mat4head_t, 147
 - cols, 147
 - imag, 147
 - namelen, 147
 - rows, 147
 - t, 147
- acsave::save_var_t, 148
 - ~save_var_t, 148
 - ac, 149
 - b_complex, 149
 - data, 149
 - framecnt, 149
 - maxframe, 149
 - name, 149
 - ndim, 149
 - nframes, 149
 - save_m, 149
 - save_mat4, 148
 - save_txt, 148
 - save_var_t, 148
 - store_frame, 148
- acsave_t
 - acsave::acsave_t, 144
- acspace2matrix_t
 - MHA_AC::acspace2matrix_t, 359
- acspace_template
 - analysispath_if_t, 183
- act_
 - wavwriter_t, 874
- actgains
 - fader_if_t, 282
- activate_query
 - MHAParser::base_t, 575
- acvar
 - MHA_AC::ac2matrix_helper_t, 356
- adapt_filter_param_t
 - MHAFilter::adapt_filter_param_t, 454
- adapt_filter_state_t
 - MHAFilter::adapt_filter_state_t, 455
- adapt_filter_t
 - MHAFilter::adapt_filter_t, 456
- add
 - MHASignal::loop_wavefragment_t, 710
- add_entry
 - MHAParser::keyword_list_t, 602
 - MHATableLookup::linear_table_t, 758
 - MHATableLookup::xy_table_t, 763
- add_fun
 - MHAOvFilter::scale_var_t, 567
- add_parent_on_insert
 - MHAParser::base_t, 575
- add_plug
 - altplugs_t, 179
- add_plugin
 - pluginbrowser_t, 799
- add_plugins

- pluginbrowser_t, 799
- add_replace_pair
 - MHAParser::base_t, 575
- added_via_plugs
 - altplugs_t, 179
- adm
 - adm_rtconfig_t, 170
- adm.cpp, 881
 - adm_fir_decomb, 882
 - adm_fir_lp, 882
- adm.hh, 882
- adm_fir_decomb
 - adm.cpp, 882
- adm_fir_lp
 - adm.cpp, 882
- adm_if_t, 165
 - adm_if_t, 166
 - beta, 167
 - bypass, 167
 - coeff_decomb, 167
 - coeff_lp, 167
 - decomb_order, 167
 - distances, 167
 - front_channels, 167
 - input_channels, 167
 - is_prepared, 167
 - lp_order, 167
 - mu_beta, 167
 - out, 167
 - patchbay, 167
 - prepare, 166
 - process, 166
 - rear_channels, 167
 - release, 166
 - srate, 167
 - tau_beta, 167
 - update, 166
- adm_rtconfig_t, 167
 - ~adm_rtconfig_t, 169
 - adm, 170
 - adm_rtconfig_t, 168
 - adm_t, 168
 - adms, 170
 - check_index, 169
 - decomb_coeffs, 170
 - front_channel, 170
 - front_channels, 170
 - lp_coeffs, 170
 - num_adms, 169
 - rear_channel, 170
 - rear_channels, 170
- adm_t
 - adm_rtconfig_t, 168
- adms
 - adm_rtconfig_t, 170
- algo
 - acmon::acmon_t, 137
 - acsave::acsave_t, 145
 - analysispath_if_t, 183
 - coherence::cohflt_if_t, 201
 - db_if_t, 214
 - dc::dc_if_t, 217
 - fftfilterbank::fftfb_interface_t, 289
 - MHAPPlugin_Resampling::resampling_if_t, 665
 - multibandcompressor::interface_t, 779
 - nlms_t, 783
 - overlapadd::overlapadd_if_t, 792
 - route::interface_t, 821
 - smoothgains_bridge::overlapadd_if_t, 839
 - wave2spec_if_t, 868
- algo_comm_class_t
 - MHAKernel::algo_comm_class_t, 539
- algo_comm_default
 - mha_algo_comm.cpp, 905
 - mha_algo_comm.hh, 907
- algo_comm_id_string
 - MHAKernel::algo_comm_class_t, 540
- algo_comm_id_string_len
 - MHAKernel::algo_comm_class_t, 540
- algo_comm_safe_cast
 - MHAKernel, 96
- algo_comm_t, 171
 - get_entries, 175
 - get_error, 175
 - get_var, 174
 - get_var_float, 174
 - get_var_int, 174
 - handle, 171
 - insert_var, 171
 - insert_var_float, 172
 - insert_var_int, 172
 - is_var, 173
 - mha.h, 904
 - remove_ref, 173
 - remove_var, 172
- algo_name
 - lpc, 340
- algos
 - MHAPPlugin_Split::split_t, 681
 - mhachain::chain_base_t, 439

- mhachain::plugs_t, 442
- all_dump
 - MHAParser, 107
- all_ids
 - MHAParser, 107
- alloc_plugs
 - mhachain::plugs_t, 442
- almost
 - Complex arithmetics in the openMHA, 68
- alpha
 - acPooling_wave, 140
 - acPooling_wave_config, 142
 - cfg_t, 199
 - coherence::cohflt_t, 203
 - coherence::vars_t, 205
- alpha_const
 - timoConfig, 860
- alpha_const_limits_hz
 - timo_params, 858
 - timoSmooth, 865
- alpha_const_vals
 - timo_params, 858
 - timoSmooth, 865
- alpha_frame
 - timoConfig, 861
- alpha_frame_AC
 - timo_AC, 856
- alpha_hat
 - timoConfig, 861
- alpha_hat_AC
 - timo_AC, 856
- alpha_pitch
 - timo_params, 858
 - timoSmooth, 864
- alpha_prev
 - timoConfig, 861
- alphaPH1mean
 - noisePowProposedScale::interface_t, 787
- alphaPH1mean_
 - noisePowProposedScale::noisePow↔Proposed, 789
- alphaPSD_
 - noisePowProposedScale::noisePow↔Proposed, 789
- alphaPSD
 - noisePowProposedScale::interface_t, 787
- altplugs.cpp, 883
 - MHAPLUGIN_OVERLOAD_OUTDOM↔AIN, 883
- altplugs_t, 176
 - add_plug, 179
 - added_via_plugs, 179
 - altplugs_t, 177
 - cf_in, 179
 - cfout, 179
 - current, 179
 - delete_plug, 179
 - event_add_plug, 178
 - event_delete_plug, 178
 - event_select_plug, 178
 - event_set_plugs, 178
 - fallback_spec, 179
 - fallback_wave, 179
 - nondefault_labels, 179
 - parse, 178
 - parser_plugs, 179
 - patchbay, 179
 - plugs, 179
 - prepare, 177
 - prepared, 179
 - proc_ramp, 178
 - process, 178
 - ramp_counter, 179
 - ramp_len, 179
 - ramplen, 179
 - release, 177
 - select_plug, 179
 - selected_plug, 179
 - update_ramplen, 178
 - update_selector_list, 178
 - use_own_ac, 179
- amplitude
 - sine_cfg_t, 835
- analysemhaplugin.cpp, 883
 - main, 883
 - print_ac, 883
 - strdom, 883
- analysepath_t, 180
 - ~analysepath_t, 180
 - ac_fifo, 181
 - analysepath_t, 180
 - attr, 181
 - cond_to_process, 181
 - flag_terminate_inner_thread, 181
 - has_inner_error, 181
 - inner_ac_copy, 181
 - inner_error, 181
 - inner_input, 181
 - inner_out_domain, 181
 - inner_process_wave2spec, 181
 - inner_process_wave2wave, 181
 - input_to_process, 181

- libdata, 181
- outer_ac, 181
- outer_ac_copy, 181
- priority, 181
- ProcessMutex, 181
- rt_process, 181
- scheduler, 181
- svc, 181
- thread, 181
- wave_fifo, 181
- analysispath.cpp, 883
- thread_start, 884
- analysispath_if_t, 182
- ~analysispath_if_t, 183
- acspace_template, 183
- algo, 183
- analysispath_if_t, 183
- chain, 183
- fifolen, 183
- fragsize, 183
- libname, 183
- loadlib, 183
- patchbay, 183
- plug, 183
- prepare, 183
- priority, 183
- process, 183
- release, 183
- vars, 183
- analytic
 - hilbert_shifter_t, 306
- ang_name
 - acTransform_wave, 156
 - acTransform_wave_config, 157
- angle
 - Complex arithmetics in the openMHA, 63
- angle_ind
 - steerbf, 853
- angle_src
 - steerbf, 853
- angles
 - doasvm_classification, 244
- announce_port
 - mhaserver_t, 694
- antialias
 - ds_t, 254
 - us_t, 866
- apply_gains
 - MHAOvFilter::fftfb_t, 551
 - multibandcompressor::plugin_signals_t, 780
- aquire_mutex
 - mha_fifo_posix_threads_t, 393
 - mha_fifo_thread_platform_t, 401
- arg
 - MHA_TCP::Thread, 429
- assign
 - MHASignal::waveform_t, 749
 - Vector and matrix processing toolbox, 48, 49
- assign_channel
 - MHASignal::waveform_t, 749
- assign_frame
 - MHASignal::waveform_t, 749
- Async_Notify
 - MHA_TCP::Async_Notify, 409
- async_poll_msg
 - fw_t, 298
- async_read
 - fw_t, 298
- async_rmslevel_t
 - MHASignal::async_rmslevel_t, 695
- attack
 - cfg_t, 199
 - dc::dc_t, 219
 - dc_simple::level_smoother_t, 235
 - softclip_t, 842
 - softclipper_t, 843
- attr
 - analysepath_t, 181
 - MHAPlugin_Split::posix_threads_t, 677
- auditory_profile.cpp, 884
- auditory_profile.h, 884
- AuditoryProfile, 78
- AuditoryProfile::fmap_t, 184
- get_frequencies, 184
- get_values, 184
- isempty, 184
- AuditoryProfile::parser_t, 185
- get_current_profile, 186
- L, 186
- parser_t, 186
- R, 186
- AuditoryProfile::parser_t::ear_t, 186
- ear_t, 187
- get_ear, 187
- HTL, 187
- UCL, 187
- AuditoryProfile::parser_t::fmap_t, 187
- f, 188
- fmap_t, 188
- get_fmap, 188

- name_, 188
- patchbay, 188
- validate, 188
- value, 188
- AuditoryProfile::profile_t, 189
 - get_ear, 189
 - L, 190
 - R, 190
- AuditoryProfile::profile_t::ear_t, 190
 - convert_empty2normal, 190
 - HTL, 190
 - UCL, 190
- average
 - coherence::vars_t, 205
- avg_ipd
 - coherence::cohflt_t, 203
- azimuth
 - mha_direction_t, 380
- B
 - MHAFilter::filter_t, 474
 - MHAFilter::iir_filter_t, 482
- b
 - doasvm_classification, 244
 - MHAParser::base_t::replace_t, 577
- B_
 - MHAFilter::complex_bandpass_t, 462
 - MHAFilter::iir_ord1_real_t, 485
- b_check_version
 - PluginLoader::mhapluginloader_t, 810
- b_complex
 - acsave::save_var_t, 149
- b_cont
 - acmon::acmon_t, 137
- b_est
 - lpc_bl_predictor_config, 344
- b_exit_request
 - fw_t, 299
- b_flushed
 - acsave::acsave_t, 145
- b_fw_started
 - io_parser_t, 319
- b_is_input
 - calibrator_runtime_layer_t, 194
 - calibrator_t, 196
- b_is_prepared
 - PluginLoader::mhapluginloader_t, 810
- b_loop
 - MHASignal::loop_wavefragment_t, 712
- b_ltg
 - coherence::cohflt_t, 204
- b_prepared
 - acsave::acsave_t, 145
 - io_file_t, 313
 - io_parser_t, 319
 - MHAJack::client_t, 534
 - mhachain::chain_base_t, 439
 - mhachain::plugs_t, 442
- b_ready
 - MHAJack::client_avg_t, 525
- b_snapshot
 - acmon::acmon_t, 137
- b_starting
 - io_parser_t, 319
- b_stopped
 - io_parser_t, 319
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 527
- b_use_clipping
 - calibrator_runtime_layer_t, 194
- b_use_fir
 - calibrator_runtime_layer_t, 194
- b_use_profiling
 - mhachain::plugs_t, 443
- BARKSCALE_ENTRIES
 - mha_fftfb.cpp, 915
- blInvert
 - coherence::cohflt_t, 204
- backward
 - lpc_bl_predictor_config, 344
 - lpc_burglattice_config, 349
 - MHASignal::fft_t, 705
- backward_scale
 - MHASignal::fft_t, 705
- band_weights
 - dc::dc_vars_t, 222
- bands
 - dc::wideband_inhib_vars_t, 226
 - MHAOvFilter::fspacing_t, 560
- bandw_correction
 - speechnoise.cpp, 979
- bark2hz_t
 - MHAOvFilter::barkscale::bark2hz_t, 548
- bartlett
 - MHAWindow, 122
- bartlett_t
 - MHAWindow::bartlett_t, 765
- base_t
 - MHAMultiSrc::base_t, 542
 - MHAParser::base_t, 571
 - MHAWindow::base_t, 766, 767
- basename
 - save_spec_t, 827

- save_wave_t, 828
- shadowfilter_begin::shadowfilter_begin←
_t, 831
- shadowfilter_end::shadowfilter_end←
_t, 834
- bbcalib_interface_t, 191
 - ~bbcalib_interface_t, 192
 - bbcalib_interface_t, 192
 - calib_in, 192
 - calib_out, 192
 - plugloader, 192
 - prepare, 192
 - process, 192
 - release, 192
- bbgain
 - gain::gain_if_t, 302
- beta
 - ADM::ADM, 159
 - adm_if_t, 167
- beta_const
 - timo_params, 858
 - timoSmooth, 864
- bf_src
 - steerbf, 853
- bf_src_copy
 - steerbf_config, 854
- bf_vec
 - steerbf_config, 854
- bflush
 - acsave::acsave_t, 145
- bin1
 - MHAOvFilter::fftfb_t, 552
- bin2
 - MHAOvFilter::fftfb_t, 552
- bin2freq
 - Vector and matrix processing toolbox, 44
- blackman
 - MHAWindow, 122
- blackman_t
 - MHAWindow::blackman_t, 768
- blockprocessing_polyphase_resampling_t
 - MHAFilter::blockprocessing_polyphase←
_resampling_t, 458
- blocks
 - droptect_t, 252
- bookkeeping
 - MHAFilter::partitioned_convolution_t, 497
 - MHAParser::mhapluginloader_t, 618
- bool_mon_t
 - MHAParser::bool_mon_t, 578
- bool_t
 - MHAParser::bool_t, 581
- bprofiling
 - mhachain::chain_base_t, 439
- bracket_balance
 - MHAParser::StrCnv, 109
- brown
 - speechnoise_t, 850
- browsemhaplugins.cpp, 884
 - DEBUG, 885
 - main, 885
- buf
 - mha_fifo_t, 398
 - mha_spec_t, 407
 - mha_wave_t, 436
- buf_c_in
 - MHASignal::hilbert_fftw_t, 707
- buf_c_out
 - MHASignal::hilbert_fftw_t, 707
- buf_in
 - MHASignal::fft_t, 706
- buf_out
 - MHASignal::fft_t, 706
- buf_r_in
 - MHASignal::hilbert_fftw_t, 707
- buf_r_out
 - MHASignal::hilbert_fftw_t, 707
- buf_uses_placement_new
 - mha_fifo_t, 398
- buffer
 - MHASignal::delay_spec_t, 697
 - MHASignal::delay_t, 699
 - MHASignal::delay_wave_t, 700
- buffered_incoming_bytes
 - MHA_TCP::Connection, 417
- buffered_outgoing_bytes
 - MHA_TCP::Connection, 417
- burn
 - DynComp::dc_afterburn_rt_t, 256
 - DynComp::dc_afterburn_t, 258
 - multibandcompressor::interface_t, 779
- butter_stop_ord1
 - MHAFilter, 91
- bw
 - MHAOvFilter::fscale_bw_t, 556
- bw_
 - MHAFilter::gammaflt_t, 478
- bw_generator
 - MHAFilter::thirdoctave_analyzer_t, 509
- bw_hz
 - MHAOvFilter::fscale_bw_t, 556
- bw_name

- dc::dc_vars_t, 222
- bwv
 - MHAOvFilter::fftfb_ac_info_t, 550
 - multibandcompressor::fftfb_plug_t, 777
- bypass
 - adm_if_t, 167
 - db_if_t, 214
 - dc::dc_t, 219
 - dc::dc_vars_t, 222
 - dc_simple::dc_vars_t, 233
 - DynComp::dc_afterburn_vars_t, 261
- C
 - ADM, 78
- c
 - acPooling_wave_config, 142
 - doasvm_classification_config, 245
 - io_tcp_sound_t::float_union, 333
 - nlms_t, 783
 - prediction_error, 813
- c1_a
 - MHAFilter::o1_ar_filter_t, 488
- c1_r
 - MHAFilter::o1_ar_filter_t, 488
- c2_a
 - MHAFilter::o1_ar_filter_t, 488
- c2_r
 - MHAFilter::o1_ar_filter_t, 488
- c_ifc_parser_t
 - MHAParser::c_ifc_parser_t, 583
- c_min
 - coherence::cohflt_t, 203
- c_parse_cmd
 - MHAParser::c_ifc_parser_t, 584
- c_parse_cmd_t
 - MHAParser, 106
- c_parse_err
 - MHAParser::c_ifc_parser_t, 584
- c_parse_err_t
 - MHAParser, 106
- c_scale
 - coherence::cohflt_t, 203
- CHANLOOP
 - timoconfig.cpp, 982
- CHECK_EXPR
 - mha_defs.h, 908
- CHECK_VAR
 - mha_defs.h, 908
- cLTASS
 - MHAOvFilter::fftfb_ac_info_t, 550
 - MHAOvFilter::fftfb_vars_t, 555
- calc_in
 - wave2spec_t, 871
- calc_out
 - overlapadd::overlapadd_t, 794
 - spec2wave_t, 848
- calc_pre_wnd
 - wave2spec_t, 870
- calib_in
 - bbcalib_interface_t, 192
- calib_out
 - bbcalib_interface_t, 192
- calibrator_runtime_layer_t, 192
 - b_is_input, 194
 - b_use_clipping, 194
 - b_use_fir, 194
 - calibrator_runtime_layer_t, 193
 - fir, 193
 - firfir2fftlens, 193
 - firfirlens, 193
 - gain, 193
 - pmode, 194
 - process, 193
 - quant, 193
 - softclip, 193
 - speechnoise, 194
- calibrator_t, 194
 - b_is_input, 196
 - calibrator_t, 195
 - patchbay, 196
 - prepare, 195
 - prepared, 196
 - process, 195
 - read_levels, 196
 - release, 195
 - update, 196
 - update_tau_level, 196
 - vars, 196
- calibrator_variables_t, 196
 - calibrator_variables_t, 197
 - config_parser, 197
 - do_clipping, 197
 - fir, 197
 - fragsize, 197
 - nbits, 197
 - num_channels, 197
 - peaklevel, 197
 - rmslevel, 197
 - softclip, 197
 - spnoise_channels, 197
 - spnoise_level, 197
 - spnoise_mode, 197
 - spnoise_parser, 197

- srate, [197](#)
- tau_level, [197](#)
- can_read
 - MHAFilter::blockprocessing_polyphase↔
_resampling_t, [459](#)
- can_read_bytes
 - MHA_TCP::Connection, [415](#)
- can_read_line
 - MHA_TCP::Connection, [415](#)
- can_sysread
 - MHA_TCP::Connection, [413](#)
- can_syswrite
 - MHA_TCP::Connection, [413](#)
- can_update
 - fader_wave::level_adapt_t, [286](#)
- catch_condition
 - MHAPLugin_Split::posix_threads_t, [677](#)
- catch_thread
 - MHAPLugin_Split::dummy_threads_t, [673](#)
 - MHAPLugin_Split::posix_threads_t, [676](#)
 - MHAPLugin_Split::thread_platform_t, [689](#)
- categories
 - plugindescription_t, [800](#)
- cdata
 - MHASignal::matrix_t, [721](#)
 - mha_audio_t, [372](#)
- center_frequencies
 - dc::dc_vars_t, [222](#)
 - dc_simple::dc_if_t, [228](#)
- cf
 - MHAFilter::thirddoctave_analyzer_t, [509](#)
 - MHAOvFilter::band_descriptor_t, [547](#)
 - MHAOvFilter::fftfb_vars_t, [555](#)
 - mha_audio_descriptor_t, [371](#)
- cf2bands
 - MHAOvFilter::fspacing_t, [560](#)
- cf_
 - MHAFilter::gammaflt_t, [478](#)
 - wavwriter_t, [874](#)
- cf_generator
 - MHAFilter::thirddoctave_analyzer_t, [509](#)
- cf_h
 - MHAOvFilter::band_descriptor_t, [547](#)
- cf_in
 - overlapadd::overlapadd_if_t, [792](#)
 - smoothgains_bridge::overlapadd_if_↔
t, [839](#)
- cf_in_
 - MHAParser::mhapluginloader_t, [618](#)
- cf_input
 - PluginLoader::mhapluginloader_t, [810](#)
- cf_l
 - MHAOvFilter::band_descriptor_t, [547](#)
- cf_name
 - dc::dc_vars_t, [222](#)
- cf_out
 - overlapadd::overlapadd_if_t, [792](#)
 - smoothgains_bridge::overlapadd_if_↔
t, [839](#)
- cf_out_
 - MHAParser::mhapluginloader_t, [618](#)
- cf_output
 - PluginLoader::mhapluginloader_t, [810](#)
- cfac
 - route::interface_t, [821](#)
- cfg
 - MHAPLugin::config_t, [659](#)
- cfg_
 - MHAFilter::thirddoctave_analyzer_t, [509](#)
- cfg_chain
 - MHAPLugin::config_t, [659](#)
- cfg_chain_current
 - MHAPLugin::config_t, [659](#)
- cfg_chain_t
 - MHAPLugin::cfg_chain_t, [654](#)
- cfg_dump
 - MHAParser, [106](#)
- cfg_dump_short
 - MHAParser, [107](#)
- cfg_t, [198](#)
 - acsave::cfg_t, [146](#)
 - alpha, [199](#)
 - attack, [199](#)
 - cfg_t, [198](#)
 - channel, [199](#)
 - decay, [199](#)
 - frozen_noise_, [199](#)
 - gain_spec_, [199](#)
 - gain_wave_, [199](#)
 - matrixmixer::cfg_t, [352](#)
 - pos, [199](#)
 - process, [199](#)
 - replace_, [199](#)
 - shadowfilter_begin::cfg_t, [829](#)
 - shadowfilter_end::cfg_t, [832](#)
 - start_lin, [199](#)
 - use_frozen_, [199](#)
- cfin
 - altplugs_t, [179](#)
 - fw_t, [299](#)
 - mhachain::chain_base_t, [439](#)
 - route::interface_t, [821](#)

- cfout
 - altplugs_t, 179
 - fw_t, 299
 - mhachain::chain_base_t, 439
 - route::interface_t, 821
- cfv
 - MHAOvFilter::fftfb_ac_info_t, 549
 - multibandcompressor::fftfb_plug_t, 777
- cg
 - coherence::cohflt_t, 203
- ch
 - MHASignal::doublebuffer_t, 703
- chain
 - acmon::acmon_t, 137
 - acsave::acsave_t, 145
 - analysispath_if_t, 183
 - db_if_t, 214
 - MHAPLugin_Resampling::resampling_if←_t, 665
 - mhachain::chain_base_t, 439
 - mhachain::plugs_t, 443
- chain_base_t
 - mhachain::chain_base_t, 438
- chains
 - MHAPLugin_Split::split_t, 682
- channel
 - cfg_t, 199
 - example5_t, 279
 - MHAMultiSrc::channel_t, 543
- channel_gain_name
 - combc_if_t, 207
- channel_gains_
 - combc_t, 208
- channel_info
 - mha_spec_t, 407
 - mha_wave_t, 437
- channel_no
 - example6_t, 280
- channelconfig_out_
 - MHAOvFilter::overlap_save_filterbank_t, 564
- channels
 - ac2wave_t, 127
 - dc::wideband_inhib_vars_t, 226
 - MHAFilter::fftfilter_t, 466
 - MHAFilter::filter_t, 475
 - MHAParser::mhaconfig_mon_t, 615
 - MHAPLugin_Split::split_t, 681
 - MHASignal::delay_t, 699
 - mhaconfig_t, 445
 - prediction_error_config, 816
 - rt_nlms_t, 824
 - sine_cfg_t, 835
 - sine_t, 837
 - Vector and matrix processing toolbox, 41
- channels_t
 - MHAMultiSrc::channels_t, 543
- chdir
 - mha_audio_descriptor_t, 371
- check_index
 - adm_rtconfig_t, 169
- check_low
 - MHAParser::range_var_t, 627
- check_range
 - MHAParser::range_var_t, 628
- check_sound_data_type
 - io_tcp_sound_t, 331
- check_up
 - MHAParser::range_var_t, 628
- chname
 - dc::dc_vars_t, 222
- chunkbytes_in
 - io_tcp_sound_t, 331
- ci
 - matrixmixer::matmix_t, 354
- cleanup_plugs
 - mhachain::plugs_t, 442
- cleanup_unused_cfg
 - MHAPLugin::config_t, 658
- clear
 - MHATableLookup::linear_table_t, 758
 - MHATableLookup::table_t, 760
 - MHATableLookup::xy_table_t, 763
 - mha_fifo_t, 398
 - Vector and matrix processing toolbox, 48
- clear_chains
 - MHAPLugin_Split::split_t, 680
- clear_plugins
 - pluginbrowser_t, 799
- Client
 - MHA_TCP::Client, 410
- client_avg_t
 - MHAJack::client_avg_t, 524
- client_noncont_t
 - MHAJack::client_noncont_t, 527
- client_t
 - MHAJack::client_t, 530
- clientid
 - dc::dc_vars_t, 222
 - dc_simple::dc_if_t, 228
- clientname
 - MHAIOJack::io_jack_t, 516

- clipmeter
 - softclipper_t, 843
- clipped
 - softclipper_variables_t, 845
- close_session
 - wavwriter_t, 874
- closed
 - MHA_TCP::Connection, 417
- closesocket
 - mha_tcp.cpp, 948
- cmd_prepare
 - MHAIOPortAudio::io_portaudio_t, 521
- cmd_release
 - MHAIOPortAudio::io_portaudio_t, 522
- cmd_start
 - MHAIOPortAudio::io_portaudio_t, 521
- cmd_stop
 - MHAIOPortAudio::io_portaudio_t, 521
- co
 - matrixmixer::matmix_t, 354
- coeff_decomb
 - adm_if_t, 167
- coeff_lp
 - adm_if_t, 167
- coh_c
 - coherence::cohflt_t, 204
- coh_rlp
 - coherence::cohflt_t, 204
- coherence, 79
 - getcipd, 79
- coherence.cpp, 885
- coherence::cohflt_if_t, 200
 - algo, 201
 - cohflt_if_t, 201
 - patchbay, 201
 - prepare, 201
 - process, 201
 - release, 201
 - update, 201
 - vars, 201
- coherence::cohflt_t, 202
 - alpha, 203
 - avg_ipd, 203
 - b_ltg, 204
 - blinvert, 204
 - c_min, 203
 - c_scale, 203
 - cg, 203
 - coh_c, 204
 - coh_rlp, 204
 - cohflt_t, 203
 - g, 203
 - gain, 204
 - gain_delay, 204
 - insert, 203
 - limit, 203
 - lp1i, 203
 - lp1ltg, 204
 - lp1r, 203
 - nbands, 203
 - process, 203
 - s_out, 204
 - staticgain, 204
- coherence::vars_t, 204
 - alpha, 205
 - average, 205
 - delay, 205
 - invert, 205
 - limit, 205
 - ltgcomp, 205
 - ltgtau, 205
 - mapping, 205
 - staticgain, 205
 - tau, 205
 - tau_unit, 205
 - vars_t, 205
- cohflt_if_t
 - coherence::cohflt_if_t, 201
- cohflt_t
 - coherence::cohflt_t, 203
- collect_result
 - MHAPlugin_Split::split_t, 680
 - MHAPlugin_Split::splitted_part_t, 686
- colored_intensity
 - Vector and matrix processing toolbox, 56
- cols
 - acsave::mat4head_t, 147
- combc_if_t, 206
 - channel_gain_name, 207
 - combc_if_t, 207
 - element_gain_name, 207
 - interleaved, 207
 - outchannels, 207
 - prepare, 207
 - process, 207
- combc_t, 207
 - ac, 208
 - channel_gains_, 208
 - combc_t, 208
 - element_gain_name_, 208
 - interleaved_, 208
 - nbands, 208

- process, 208
- s_out, 208
- w_out, 208
- combinechannels.cpp, 885
- comm_var_t, 209
 - data, 210
 - data_type, 209
 - num_entries, 209
 - stride, 210
- commentate
 - MHAParser, 106
- commit
 - DynComp::dc_afterburn_vars_t, 261
- commit_pending
 - DynComp::dc_afterburn_t, 259
- commit_t
 - MHAParser::commit_t, 586
- Communication between algorithms, 27
 - get_var_float, 29
 - get_var_int, 29
 - get_var_spectrum, 28
 - get_var_vfloat, 30
 - get_var_waveform, 29
- comp_each_iter
 - lpc, 340
 - lpc_config, 350
- comp_iter
 - lpc_config, 351
- Complex arithmetics in the openMHA, 60
 - _conjugate, 67
 - _reciprocal, 68
 - abs, 66
 - abs2, 66
 - almost, 68
 - angle, 63
 - conjugate, 67
 - expi, 63, 65
 - mha_complex, 62
 - normalize, 68
 - operator!=, 67
 - operator<, 68
 - operator*, 65, 66
 - operator*=, 65
 - operator+, 64
 - operator+=", 64
 - operator-, 64, 65, 67
 - operator-=, 64
 - operator/, 66, 67
 - operator/=", 66
 - operator==, 67
 - reciprocal, 67
 - safe_div, 66
 - set, 62, 63
 - stdcomplex, 63
- complex_bandpass_t
 - MHAFilter::complex_bandpass_t, 461
- complex_filter.cpp, 885
- complex_filter.h, 885
- complex_mon_t
 - MHAParser::complex_mon_t, 587
- complex_ofs
 - MHASignal::matrix_t, 721
- complex_t
 - MHAParser::complex_t, 589
- compression
 - dc_simple::dc_t, 230
- compute_something
 - cpuload_t, 211
- compute_something_else
 - cpuload_t, 212
- Concept of Variables and Data Exchange in the openMHA, 4
- cond_to_process
 - analysepath_t, 181
- config_file_splitter_t
 - PluginLoader::config_file_splitter_t, 801
- config_parser
 - calibrator_variables_t, 197
- config_t
 - MHAPlugin::config_t, 656
- configfile
 - PluginLoader::config_file_splitter_t, 802
- configname
 - PluginLoader::config_file_splitter_t, 802
- configuration, 4
- configuration variable, 4
- conflux
 - DynComp::dc_afterburn_rt_t, 257
 - DynComp::dc_afterburn_vars_t, 261
- conjugate
 - Complex arithmetics in the openMHA, 67
 - Vector and matrix processing toolbox, 59
- connect
 - MHAEvents::emitter_t, 451
 - MHAEvents::patchbay_t, 452, 453
- connect_input
 - MHAJack::client_t, 531
- connect_output
 - MHAJack::client_t, 531
- connect_to
 - MHAJack::port_t, 537
- connected

- io_tcp_parser_t, 328
- Connection
 - MHA_TCP::Connection, 413
- connection_loop
 - io_tcp_t, 335
- connections
 - MHAEvents::emitter_t, 451
- connections_in
 - MHAIOJack::io_jack_t, 516
- connections_out
 - MHAIOJack::io_jack_t, 516
- connector
 - MHAFilter::adapt_filter_t, 457
 - MHAFilter::iir_filter_t, 482
 - MHAParser::mhapluginloader_t, 618
- connector_base_t
 - MHAEvents::connector_base_t, 446
- connector_t
 - MHAEvents::connector_t, 449
- cons
 - MHAEvents::patchbay_t, 453
- consecutive_dropouts
 - droptect_t, 252
- contained_frames
 - MHASignal::ringbuffer_t, 726
- conv2latex
 - generatemhaplugindoc.cpp, 895
- convert_empty2normal
 - AuditoryProfile::profile_t::ear_t, 190
- convert_f2logf
 - gaintable.cpp, 894
- copy
 - MHASignal::spectrum_t, 734
 - MHASignal::waveform_t, 750
 - timo_AC, 855
- copy_AC
 - timoConfig, 860
- copy_channel
 - MHASignal::spectrum_t, 735
 - MHASignal::waveform_t, 750
 - Vector and matrix processing toolbox, 55
- copy_error
 - MHAIOTCP.cpp, 965
- copy_from_at
 - MHASignal::waveform_t, 750
- copy_output_spec
 - MHAPLugin_Split::split_t, 680
- copy_output_wave
 - MHAPLugin_Split::split_t, 680
- copy_permuted
 - MHASignal, 120
- corr_out
 - lpc_config, 351
- cpuload.cpp, 886
- cpuload_t, 210
 - compute_something, 211
 - compute_something_else, 212
 - cpuload_t, 211
 - factor, 212
 - phase, 212
 - prepare, 211
 - process, 211
 - result, 212
 - table, 212
 - use_sine, 212
- create_latex_doc
 - generatemhaplugindoc.cpp, 895
- create_lock
 - mhamain.cpp, 969
- creator
 - speechnoise_t, 851
- creator_A
 - MHAFilter::complex_bandpass_t, 461
- creator_B
 - MHAFilter::complex_bandpass_t, 461
- cstr_strerror
 - mha_errno.c, 910
- current
 - altplugins_t, 179
 - dc::wideband_inhib_vars_t, 225
 - mha_rt_fifo_t, 406
- current_input_signal_buffer_half_index
 - MHAFilter::partitioned_convolution_t, 497
- current_output_partition_index
 - MHAFilter::partitioned_convolution_t, 497
- current_powspec
 - droptect_t, 252
- current_thread_priority
 - MHAPLugin_Split::posix_threads_t, 676
- current_thread_scheduler
 - MHAPLugin_Split::posix_threads_t, 676
- DEBUG
 - browsemhaplugins.cpp, 885
 - fader_wave.cpp, 893
 - MHAIOFile.cpp, 953
 - wavrec.cpp, 985
- DEFAULT_RETSIZE
 - mha_parser.hh, 929
- DELAY_FREQ
 - ADM, 78
- DUPVEC
 - dc.cpp, 887

- data
 - acsave::save_var_t, 149
 - comm_var_t, 210
 - DynComp::gaintable_t, 266
 - MHA_AC::acspace2matrix_t, 361
 - MHA_AC::double_t, 362
 - MHA_AC::float_t, 363
 - MHA_AC::int_t, 364
 - MHAParser::bool_mon_t, 579
 - MHAParser::bool_t, 581
 - MHAParser::complex_mon_t, 588
 - MHAParser::complex_t, 590
 - MHAParser::float_mon_t, 593
 - MHAParser::float_t, 596
 - MHAParser::int_mon_t, 598
 - MHAParser::int_t, 600
 - MHAParser::kw_t, 606
 - MHAParser::mcomplex_mon_t, 607
 - MHAParser::mcomplex_t, 609
 - MHAParser::mfloat_mon_t, 611
 - MHAParser::mfloat_t, 614
 - MHAParser::string_mon_t, 630
 - MHAParser::string_t, 632
 - MHAParser::vcomplex_mon_t, 635
 - MHAParser::vcomplex_t, 637
 - MHAParser::vfloat_mon_t, 639
 - MHAParser::vfloat_t, 642
 - MHAParser::vint_mon_t, 643
 - MHAParser::vint_t, 646
 - MHAParser::vstring_mon_t, 647
 - MHAParser::vstring_t, 649
 - MHAPlugin::cfg_chain_t, 654
 - MHASignal::uint_vector_t, 742
 - mha_rt_fifo_element_t, 403
 - wavwriter_t, 874
- data_is_initialized
 - MHAParser::base_t, 576
- data_type
 - comm_var_t, 209
- db.cpp, 886
- db2lin
 - Vector and matrix processing toolbox, 42
- db_if_t, 212
 - ~db_if_t, 213
 - algo, 214
 - bypass, 214
 - chain, 214
 - db_if_t, 213
 - fragsize, 214
 - patchbay, 214
 - plugloader, 214
 - prepare, 213
 - process, 213
 - release, 213
- db_t, 214
 - db_t, 215
 - inner_process, 215
 - plugloader, 215
- dbspl2pa
 - Vector and matrix processing toolbox, 43
- DC
 - dc_simple, 81
- dc, 79
 - get_audiochannels, 80
- dc.cpp, 886
 - DUPVEC, 887
- dc::dc_if_t, 216
 - algo, 217
 - dc_if_t, 217
 - patchbay, 217
 - prepare, 217
 - process, 217
 - update, 217
 - update_monitors, 217
 - wbinhib, 217
- dc::dc_t, 218
 - attack, 219
 - bypass, 219
 - dc_t, 219
 - decay, 219
 - explicit_insert, 219
 - get_level_in_db, 219
 - get_level_in_db_adjusted, 219
 - get_nbands, 219
 - gt, 219
 - inhib_gain, 220
 - k_nyquist, 220
 - level_in_db, 220
 - level_in_db_adjusted, 220
 - max_level_difference, 220
 - naudiochannels, 220
 - nbands, 220
 - powersum, 219
 - process, 219
 - rmslevel, 219
- dc::dc_vars_t, 220
 - band_weights, 222
 - bw_name, 222
 - bypass, 222
 - center_frequencies, 222
 - cf_name, 222
 - chname, 222

- clientid, [222](#)
- dc_vars_t, [221](#)
- edge_frequencies, [222](#)
- ef_name, [222](#)
- filterbank, [222](#)
- filtered_level, [222](#)
- gainrule, [222](#)
- gtdata, [221](#)
- gtmin, [221](#)
- gtstep, [221](#)
- input_level, [222](#)
- max_level_difference, [222](#)
- modified, [222](#)
- powersum, [221](#)
- preset, [222](#)
- tauattack, [221](#)
- taudecay, [222](#)
- taurmslevel, [221](#)
- use_wbinhib, [222](#)
- dc::dc_vars_validator_t, [223](#)
 - dc_vars_validator_t, [223](#)
- dc::wb_inhib_cfg_t, [223](#)
 - dl_diff, [224](#)
 - dl_map_max, [224](#)
 - dl_map_min, [224](#)
 - g_scale, [224](#)
 - l_min, [224](#)
 - wb_inhib_cfg_t, [224](#)
 - weights, [224](#)
- dc::wideband_inhib_vars_t, [224](#)
 - bands, [226](#)
 - channels, [226](#)
 - current, [225](#)
 - dl_map_max, [225](#)
 - dl_map_min, [225](#)
 - g_scale, [225](#)
 - l_min, [225](#)
 - patchbay, [226](#)
 - setchannels, [225](#)
 - update, [225](#)
 - weights, [225](#)
 - wideband_inhib_vars_t, [225](#)
- dc_afterburn.cpp, [887](#)
 - mylogf, [887](#)
- dc_afterburn.h, [887](#)
- dc_afterburn_rt_t
 - DynComp::dc_afterburn_rt_t, [256](#)
- dc_afterburn_t
 - DynComp::dc_afterburn_t, [258](#)
- dc_afterburn_vars_t
 - DynComp::dc_afterburn_vars_t, [261](#)
- dc_if_t
 - dc::dc_if_t, [217](#)
 - dc_simple::dc_if_t, [227](#)
- dc_simple, [80](#)
 - DC, [81](#)
 - force_resize, [81](#)
 - LEVEL, [81](#)
 - not_zero, [81](#)
 - test_fail, [81](#)
- dc_simple.cpp, [888](#)
- dc_simple::dc_if_t, [226](#)
 - center_frequencies, [228](#)
 - clientid, [228](#)
 - dc_if_t, [227](#)
 - edge_frequencies, [228](#)
 - filterbank, [228](#)
 - gainrule, [228](#)
 - has_been_modified, [228](#)
 - modified, [228](#)
 - mon_g, [228](#)
 - mon_l, [228](#)
 - patchbay, [228](#)
 - prepare, [227](#)
 - prepared, [228](#)
 - preset, [228](#)
 - process, [227](#), [228](#)
 - read_modified, [228](#)
 - release, [227](#)
 - update_dc, [228](#)
 - update_gain_mon, [228](#)
 - update_level, [228](#)
 - update_level_mon, [228](#)
- dc_simple::dc_t, [229](#)
 - compression, [230](#)
 - dc_t, [230](#)
 - expansion, [230](#)
 - expansion_threshold, [230](#)
 - limiter, [230](#)
 - limiter_threshold, [230](#)
 - maxgain, [230](#)
 - mon_g, [230](#)
 - mon_l, [230](#)
 - nbands, [230](#)
 - process, [230](#)
- dc_simple::dc_t::line_t, [231](#)
 - line_t, [231](#)
 - m, [231](#)
 - operator(), [231](#)
 - y0, [231](#)
- dc_simple::dc_vars_t, [232](#)
 - bypass, [233](#)

- dc_vars_t, 232
- expansion_slope, 233
- expansion_threshold, 233
- g50, 232
- g80, 232
- limiter_threshold, 233
- maxgain, 232
- tauattack, 233
- taudecay, 233
- dc_simple::dc_vars_validator_t, 233
 - dc_vars_validator_t, 234
- dc_simple::level_smoother_t, 234
 - attack, 235
 - decay, 235
 - fftlens, 235
 - level_smoother_t, 235
 - level_spec, 235
 - level_wave, 235
 - nbands, 235
 - process, 235
- dc_t
 - dc::dc_t, 219
 - dc_simple::dc_t, 230
- dc_vars_t
 - dc::dc_vars_t, 221
 - dc_simple::dc_vars_t, 232
- dc_vars_validator_t
 - dc::dc_vars_validator_t, 223
 - dc_simple::dc_vars_validator_t, 234
- deallocate_domains
 - MHAPLugin_Split::domain_handler_t, 670
- debug
 - io_tcp_parser_t, 328
- debug_file
 - io_tcp_parser_t, 329
- debug_filename
 - io_tcp_parser_t, 329
- decay
 - cfg_t, 199
 - dc::dc_t, 219
 - dc_simple::level_smoother_t, 235
 - softclip_t, 842
 - softclipper_t, 843
- decomb_coeffs
 - adm_rtconfig_t, 170
- decomb_order
 - adm_if_t, 167
- decrease_condition
 - mha_fifo_posix_threads_t, 394
- decrement
 - mha_fifo_posix_threads_t, 393
- mha_fifo_thread_platform_t, 401
- default_thread_platform_string
 - split.cpp, 981
- default_thread_platform_type
 - split.cpp, 981
- defaultHighInputLatency
 - MHAIOPortAudio::device_info_t, 519
- defaultHighOutputLatency
 - MHAIOPortAudio::device_info_t, 519
- defaultLowInputLatency
 - MHAIOPortAudio::device_info_t, 519
- defaultLowOutputLatency
 - MHAIOPortAudio::device_info_t, 519
- defaultSampleRate
 - MHAIOPortAudio::device_info_t, 519
- Delay
 - ADM::Delay, 161
- delay, 81
 - coherence::vars_t, 205
 - delaysum::delaysum_if_t, 239
 - MHAFilter::gammaflt_t, 477
 - MHAFilter::partitioned_convolution_t←
::index_t, 499
 - MHAPLugin_Split::split_t, 682
 - MHASignal::delay_spec_t, 697
 - MHASignal::delay_wave_t, 700
 - mha_dblbuf_t, 379
- delay.cpp, 888
- delay::interface_t, 236
 - delays, 237
 - interface_t, 237
 - patchbay, 237
 - prepare, 237
 - process, 237
 - update, 237
- delay_ac
 - ac2wave_if_t, 126
 - ac2wave_t, 128
- delay_d
 - prediction_error, 814
- delay_in
 - ac2wave_if_t, 126
 - ac2wave_t, 128
- delay_spec_t
 - MHASignal::delay_spec_t, 697
- delay_t
 - MHASignal::delay_t, 698
- delay_w
 - prediction_error, 814
- delay_wave_t
 - MHASignal::delay_wave_t, 700

- delays
 - delay::interface_t, [237](#)
 - MHASignal::delay_t, [699](#)
- delays_in
 - MHAIOJack::io_jack_t, [516](#)
- delays_out
 - MHAIOJack::io_jack_t, [517](#)
- delaysum, [81](#)
- delaysum.cpp, [889](#)
- delaysum::delaysum_if_t, [237](#)
 - delay, [239](#)
 - delaysum_if_t, [239](#)
 - patchbay, [240](#)
 - prepare, [239](#)
 - process, [239](#)
 - release, [239](#)
 - update_cfg, [239](#)
 - weights, [239](#)
- delaysum::delaysum_t, [240](#)
 - delaysum_t, [241](#)
 - out, [241](#)
 - process, [241](#)
 - weights, [241](#)
- delaysum_if_t
 - delaysum::delaysum_if_t, [239](#)
- delaysum_t
 - delaysum::delaysum_t, [241](#)
- delete_plug
 - altplugs_t, [179](#)
- delta_pitch
 - timo_params, [858](#)
 - timoSmooth, [864](#)
- descriptor
 - mha_audio_t, [372](#)
- desired_fill_count
 - mha_drifter_fifo_t, [385](#)
- device_index
 - MHAIOPortAudio::io_portaudio_t, [522](#)
- device_index_updated
 - MHAIOPortAudio::io_portaudio_t, [521](#)
- device_info
 - MHAIOPortAudio::io_portaudio_t, [522](#)
- device_info_t
 - MHAIOPortAudio::device_info_t, [519](#)
- device_name
 - MHAIOPortAudio::io_portaudio_t, [522](#)
- device_name_updated
 - MHAIOPortAudio::io_portaudio_t, [521](#)
- df
 - frequency_translator_t, [293](#)
- diff_coeffs
 - mha_filter.cpp, [917](#)
- diff_t
 - MHAFilter::diff_t, [463](#)
- digits
 - mha_error_helpers, [85](#)
- dimension
 - MHASignal::matrix_t, [717](#)
- dimensions
 - acmon::acmon_t, [137](#)
- dimstr
 - acmon::ac_monitor_t, [134](#)
- dir
 - mha_channel_info_t, [373](#)
- dir_t
 - MHAJack::port_t, [535](#)
- dir_type
 - MHAJack::port_t, [537](#)
- discard
 - MHASignal::ringbuffer_t, [727](#)
- disconnect
 - MHAEvents::emitter_t, [451](#)
- dispmode
 - acmon::acmon_t, [137](#)
- distance
 - mha_direction_t, [381](#)
- distances
 - adm_if_t, [167](#)
- dl_diff
 - dc::wb_inhib_cfg_t, [224](#)
- dl_map_max
 - dc::wb_inhib_cfg_t, [224](#)
 - dc::wideband_inhib_vars_t, [225](#)
- dl_map_min
 - dc::wb_inhib_cfg_t, [224](#)
 - dc::wideband_inhib_vars_t, [225](#)
- dm
 - lpc_burglattice_config, [349](#)
- do_clipping
 - calibrator_variables_t, [197](#)
- doagcc
 - doasvm_feature_extraction_config, [249](#)
- doasvm
 - doasvm_classification_config, [245](#)
- doasvm_classification, [242](#)
 - ~doasvm_classification, [243](#)
- angles, [244](#)
- b, [244](#)
- doasvm_classification, [243](#)
- max_p_ind_name, [244](#)
- p_name, [244](#)
- patchbay, [244](#)

- prepare, [243](#)
- process, [243](#)
- release, [243](#)
- update_cfg, [244](#)
- vGCC_name, [244](#)
- w, [244](#)
- x, [244](#)
- y, [244](#)
- doasvm_classification.cpp, [889](#)
 - INSERT_PATCH, [889](#)
 - PATCH_VAR, [889](#)
- doasvm_classification.h, [889](#)
- doasvm_classification_config, [244](#)
 - ~doasvm_classification_config, [245](#)
 - ac, [245](#)
 - c, [245](#)
 - doasvm, [245](#)
 - doasvm_classification_config, [245](#)
 - p, [245](#)
 - p_max, [245](#)
 - process, [245](#)
- doasvm_feature_extraction, [246](#)
 - ~doasvm_feature_extraction, [247](#)
 - doasvm_feature_extraction, [247](#)
 - fftlens, [248](#)
 - max_lag, [248](#)
 - nupsample, [248](#)
 - patchbay, [248](#)
 - prepare, [247](#)
 - process, [247](#)
 - release, [247](#)
 - update_cfg, [248](#)
 - vGCC_name, [248](#)
- doasvm_feature_extraction.cpp, [889](#)
 - INSERT_PATCH, [890](#)
 - PATCH_VAR, [890](#)
- doasvm_feature_extraction.h, [890](#)
- doasvm_feature_extraction_config, [248](#)
 - ~doasvm_feature_extraction_config, [249](#)
 - doagcc, [249](#)
 - doasvm_feature_extraction_config, [249](#)
 - fft, [249](#)
 - fftlens, [249](#)
 - G, [249](#)
 - G_length, [249](#)
 - GCC_end, [249](#)
 - GCC_start, [249](#)
 - hifftwin, [249](#)
 - hifftwin_sum, [249](#)
 - hwin, [249](#)
 - ifft, [249](#)
 - in_spec, [249](#)
 - proc_wave, [249](#)
 - process, [249](#)
 - vGCC_ac, [249](#)
 - vGCC, [249](#)
 - wndlen, [249](#)
- doc_appendix.h, [890](#)
- doc_examples.h, [890](#)
- doc_frameworks.h, [890](#)
- doc_general.h, [890](#)
- doc_kernel.h, [890](#)
- doc_matlab.h, [890](#)
- doc_mhamain.h, [890](#)
- doc_parser.h, [890](#)
- doc_pluginif.cpp, [890](#)
- doc_plugins.h, [890](#)
- doc_system.h, [890](#)
- doc_toolbox.h, [890](#)
- documentation
 - pluginindexdescription_t, [800](#)
- domain
 - MHAParser::mhaconfig_mon_t, [615](#)
 - MHAPugin_Split::splitted_part_t, [686](#)
 - mhaconfig_t, [445](#)
- domain_handler_t
 - MHAPugin_Split::domain_handler_t, [669](#)
- double_t
 - MHA_AC::double_t, [362](#)
- doublebuffer_t
 - MHASignal::doublebuffer_t, [701](#)
- down
 - MHASignal::schroeder_t, [730](#)
- downsample.cpp, [890](#)
- downsampling_factor
 - MHAFilter::polyphase_resampling_t, [503](#)
- downscale
 - MHASignal::quantizer_t, [724](#)
- dphi
 - hilbert_shifter_t, [306](#)
- drain
 - DynComp::dc_afterburn_vars_t, [261](#)
- drain_inv
 - DynComp::dc_afterburn_rt_t, [257](#)
- dropouts
 - droptect_t, [252](#)
- droptect.cpp, [891](#)
- droptect_t, [250](#)
 - blocks, [252](#)
 - consecutive_dropouts, [252](#)
 - current_powspec, [252](#)
 - dropouts, [252](#)

- droptect_t, [251](#)
- filter_activated, [252](#)
- filtered_powspec, [252](#)
- filtered_powspec_mon, [252](#)
- level_mon, [252](#)
- period, [252](#)
- prepare, [251](#)
- process, [251](#)
- release, [251](#)
- reset, [252](#)
- tau, [252](#)
- threshold, [252](#)
- ds_t, [253](#)
 - antialias, [254](#)
 - ds_t, [253](#)
 - prepare, [254](#)
 - process, [254](#)
 - ratio, [254](#)
 - release, [254](#)
- dt
 - mha_audio_descriptor_t, [371](#)
- mtime
 - MHA_TCP, [88](#)
- dummy_interface_test
 - MHAIOFile.cpp, [953](#), [954](#)
 - MHAIOJack.cpp, [956](#), [957](#)
 - MHAIOParser.cpp, [958](#), [959](#)
 - MHAIOPortAudio.cpp, [961](#), [962](#)
 - MHAIoTCP.cpp, [965](#), [966](#)
- dummy_jack_proc_cb
 - mhajack.cpp, [966](#)
- dummy_threads_t
 - MHAPLugin_Split::dummy_threads_t, [673](#)
- dump_mha
 - fw_t, [298](#)
- dup
 - MHAFilter::thirdoctave_analyzer_t, [509](#)
- dupvec
 - Vector and matrix processing toolbox, [45](#)
- dupvec_chk
 - Vector and matrix processing toolbox, [46](#)
- DynComp, [82](#)
 - interp1, [82](#)
 - interp2, [83](#)
- DynComp::dc_afterburn_rt_t, [255](#)
 - burn, [256](#)
 - conflux, [257](#)
 - dc_afterburn_rt_t, [256](#)
 - drain_inv, [257](#)
 - lp, [257](#)
 - maxgain, [257](#)
 - mpo_inv, [257](#)
- DynComp::dc_afterburn_t, [257](#)
 - _cf, [259](#)
 - _channels, [259](#)
 - _srate, [259](#)
 - burn, [258](#)
 - commit_pending, [259](#)
 - dc_afterburn_t, [258](#)
 - fb_pars_configured, [259](#)
 - patchbay, [259](#)
 - set_fb_pars, [258](#)
 - unset_fb_pars, [258](#)
 - update, [259](#)
 - update_burner, [258](#)
- DynComp::dc_afterburn_vars_t, [259](#)
 - bypass, [261](#)
 - commit, [261](#)
 - conflux, [261](#)
 - dc_afterburn_vars_t, [261](#)
 - drain, [261](#)
 - f, [261](#)
 - maxgain, [261](#)
 - mpo, [261](#)
 - taugain, [261](#)
- DynComp::gaintable_t, [261](#)
 - ~gaintable_t, [264](#)
 - data, [266](#)
 - gaintable_t, [263](#)
 - get_gain, [264](#), [265](#)
 - get_iofun, [265](#)
 - get_vF, [265](#)
 - get_vL, [265](#)
 - nbands, [265](#)
 - nchannels, [265](#)
 - num_channels, [265](#)
 - num_F, [265](#)
 - num_L, [265](#)
 - update, [264](#)
 - vFlog, [266](#)
 - vF, [266](#)
 - vL, [265](#)
- dynamiclib_t, [254](#)
 - ~dynamiclib_t, [255](#)
 - dynamiclib_t, [255](#)
 - fullname, [255](#)
 - getmodulename, [255](#)
 - getname, [255](#)
 - h, [255](#)
 - modulename, [255](#)
 - resolve, [255](#)
 - resolve_checked, [255](#)

- EPSILON
 - lpc_bl_predictor.h, [899](#)
 - lpc_burg-lattice.h, [899](#)
 - timoconfig.cpp, [982](#)
- EPrew
 - prediction_error_config, [817](#)
- ERR_IHANDLE
 - MHAIOFile.cpp, [953](#)
 - MHAIOJack.cpp, [955](#)
 - MHAIOParser.cpp, [958](#)
 - MHAIOPortAudio.cpp, [961](#)
 - MHAIOTCP.cpp, [964](#)
- ERR_SUCCESS
 - MHAIOFile.cpp, [953](#)
 - MHAIOJack.cpp, [955](#)
 - MHAIOParser.cpp, [958](#)
 - MHAIOPortAudio.cpp, [961](#)
 - MHAIOTCP.cpp, [964](#)
- ERR_USER
 - MHAIOFile.cpp, [953](#)
 - MHAIOJack.cpp, [955](#)
 - MHAIOParser.cpp, [958](#)
 - MHAIOPortAudio.cpp, [961](#)
 - MHAIOTCP.cpp, [964](#)
- ESTIM_CUR
 - nlms_wave.cpp, [972](#)
- ESTIM_PREV
 - nlms_wave.cpp, [972](#)
- ESTIMATION_TYPES
 - nlms_wave.cpp, [972](#)
- ear_t
 - AuditoryProfile::parser_t::ear_t, [187](#)
- edge_frequencies
 - dc::dc_vars_t, [222](#)
 - dc_simple::dc_if_t, [228](#)
- ef
 - MHAOvFilter::fftfb_vars_t, [555](#)
- ef2bands
 - MHAOvFilter::fspacing_t, [560](#)
- ef_h
 - MHAOvFilter::band_descriptor_t, [547](#)
- ef_l
 - MHAOvFilter::band_descriptor_t, [547](#)
- ef_name
 - dc::dc_vars_t, [222](#)
- efv
 - MHAOvFilter::fftfb_ac_info_t, [549](#)
 - multibandcompressor::fftfb_plug_t, [777](#)
- element_gain_name
 - combc_if_t, [207](#)
- element_gain_name_
 - combc_t, [208](#)
- elevation
 - mha_direction_t, [380](#)
- emit_event
 - MHAEvents::connector_base_t, [447](#)
 - MHAEvents::connector_t, [449](#)
- emitter
 - MHAEvents::connector_t, [450](#)
- emitter_die
 - MHAEvents::connector_base_t, [447](#)
- emitter_is_alive
 - MHAEvents::connector_base_t, [447](#)
- empty_string
 - MHAParser::keyword_list_t, [603](#)
- end_time
 - MHA_TCP::Timeout_Event, [431](#)
- entries
 - MHAParser::keyword_list_t, [603](#)
 - MHAParser::parser_t, [624](#)
- entry
 - MHAParser::entry_t, [590](#)
- entry_map_t
 - MHAParser, [106](#)
- entry_t
 - MHAParser::entry_t, [590](#)
- envelope_delay
 - MHAFilter::gammaflt_t, [478](#)
- envreplace
 - MHAParser, [107](#)
- eof
 - MHA_TCP::Connection, [414](#)
- epsilon
 - smoothgains_bridge::overlapadd_if_t, [839](#)
- equal_dim
 - Vector and matrix processing toolbox, [46, 47](#)
- equidist2bands
 - MHAOvFilter::fspacing_t, [560](#)
- erb_hz_f_hz
 - speechnoise.cpp, [978](#)
- err_in
 - MHAFilter::adapt_filter_param_t, [454](#)
 - MHAFilter::adapt_filter_t, [457](#)
- error
 - MHA_TCP::Thread, [430](#)
 - mha_fifo_lw_t, [392](#)
- Error handling in the openMHA, [31](#)
 - MHA_ErrorMsg, [31](#)
 - MHA_assert, [32](#)
 - MHA_assert_equal, [32](#)

- mha_debug, 32
- errorlog
 - fw_t, 298
- estimateDebug
 - noisePowProposedScale::noisePow↔
Proposed, 789
- estimtype
 - nlms_t, 783
- event_add_plug
 - altplugs_t, 178
- event_delete_plug
 - altplugs_t, 178
- event_select_plug
 - altplugs_t, 178
- event_set_plugs
 - altplugs_t, 178
- event_start_recording
 - acsave::acsave_t, 145
- event_stop_and_flush
 - acsave::acsave_t, 145
- eventhandler
 - MHAEvents::connector_t, 450
- eventhandler_s
 - MHAEvents::connector_t, 450
- eventhandler_suu
 - MHAEvents::connector_t, 450
- Events
 - MHA_TCP::Event_Watcher, 419
- events
 - MHA_TCP::Event_Watcher, 419
- example1.cpp, 891
- example1_t, 266
 - example1_t, 267
 - prepare, 267
 - process, 268
 - release, 267
- example2.cpp, 891
- example2_t, 268
 - example2_t, 270
 - factor, 271
 - prepare, 270
 - process, 270
 - release, 270
 - scale_ch, 271
- example3.cpp, 891
- example3_t, 271
 - example3_t, 273
 - factor, 274
 - on_prereadaccess, 273
 - on_scale_ch_readaccess, 273
 - on_scale_ch_valuechanged, 273
- on_scale_ch_writeaccess, 273
- patchbay, 274
- prepare, 273
- prepared, 274
- process, 274
- release, 274
- scale_ch, 274
- example4.cpp, 891
- example4_t, 275
 - example4_t, 276
 - factor, 278
 - on_prereadaccess, 277
 - on_scale_ch_readaccess, 277
 - on_scale_ch_valuechanged, 277
 - on_scale_ch_writeaccess, 277
 - patchbay, 278
 - prepare, 277
 - prepared, 278
 - process, 277
 - release, 277
 - scale_ch, 278
- example5.cpp, 891
 - __declspec, 892
- example5_t, 278
 - channel, 279
 - example5_t, 279
 - process, 279
 - scale, 279
- example6.cpp, 892
 - __declspec, 892
- example6_t, 279
 - channel_no, 280
 - example6_t, 280
 - patchbay, 280
 - prepare, 280
 - process, 280
 - rmsdb, 280
 - update_cfg, 280
- exec_fw_command
 - fw_t, 297
- exit_on_stop
 - fw_t, 298
- exit_request
 - fw_t, 296
- expansion
 - dc_simple::dc_t, 230
- expansion_slope
 - dc_simple::dc_vars_t, 233
- expansion_threshold
 - dc_simple::dc_t, 230
 - dc_simple::dc_vars_t, 233

- expflt
 - MHAOvFilter::ShapeFun, 103
- expi
 - Complex arithmetics in the openMHA, 63, 65
- explicit_insert
 - dc::dc_t, 219
- export_to
 - MHASignal::spectrum_t, 735
 - MHASignal::waveform_t, 751
- expression_t, 281
 - MHAParser::expression_t, 591
- extern_connector
 - MHAParser::commit_t, 586
- F
 - prediction_error_config, 816
 - rt_nlms_t, 824
- f
 - AuditoryProfile::parser_t::fmap_t, 188
 - DynComp::dc_afterburn_vars_t, 261
 - io_tcp_sound_t::float_union, 333
 - MHAOvFilter::fftfb_vars_t, 554
 - MHAOvFilter::fscale_t, 558
- f0_high
 - timo_params, 857
 - timoSmooth, 864
- f0_low
 - timo_params, 857
 - timoSmooth, 864
- F_Uflt
 - prediction_error_config, 816
- f_est
 - lpc_bl_predictor_config, 344
- f_hz
 - MHAOvFilter::fscale_t, 558
- FINISHED
 - MHA_TCP::Thread, 428
- FMTsz
 - mha_os.h, 923
- factor
 - cpuload_t, 212
 - example2_t, 271
 - example3_t, 274
 - example4_t, 278
 - plugin_interface_t, 798
- fader_if_t, 281
 - actgains, 282
 - fader_if_t, 282
 - newgains, 282
 - patchbay, 282
 - prepare, 282
 - process, 282
 - tau, 282
 - update_cfg, 282
- fader_spec.cpp, 892
- fader_wave, 83
 - level_adaptor, 83
- fader_wave.cpp, 892
 - DEBUG, 893
- fader_wave::fader_wave_if_t, 283
 - fader_wave_if_t, 284
 - gain, 284
 - patchbay, 284
 - prepare, 284
 - prepared, 284
 - process, 284
 - ramplen, 284
 - release, 284
 - set_level, 284
- fader_wave::level_adapt_t, 285
 - can_update, 286
 - get_level, 286
 - ilen, 286
 - l_new, 286
 - l_old, 286
 - level_adapt_t, 285
 - pos, 286
 - update_frame, 286
 - wnd, 286
- fader_wave_if_t
 - fader_wave::fader_wave_if_t, 284
- fail_on_async_jackerror
 - MHAJack::client_t, 534
- fail_on_nonmonotonic
 - MHAOvFilter::fftfb_vars_t, 555
- fail_on_nonmonotonic_cf
 - MHAOvFilter::fspacing_t, 560
- fail_on_unique_bins
 - MHAOvFilter::fftfb_vars_t, 555
- fail_on_unique_fftbins
 - MHAOvFilter::fspacing_t, 560
- fallback_spec
 - altplugs_t, 179
- fallback_wave
 - altplugs_t, 179
- Fast Fourier Transform functions, 70
 - mha_fft_backward, 74
 - mha_fft_backward_scale, 75
 - mha_fft_forward, 74
 - mha_fft_forward_scale, 75
 - mha_fft_free, 71
 - mha_fft_new, 71

- mha_fft_spec2wave, 73
 - mha_fft_spec2wave_scale, 76
 - mha_fft_t, 71
 - mha_fft_wave2spec, 72
 - mha_fft_wave2spec_scale, 75
- fatallog
 - fw_t, 298
- fb
 - MHAFilter::thirddoctave_analyzer_t, 510
- fb_acinfo
 - fftfilterbank::fftfb_plug_t, 291
- fb_pars_configured
 - DynComp::dc_afterburn_t, 259
- fd
 - MHA_TCP::Connection, 417
 - MHA_TCP::OS_EVENT_TYPE, 420
- fft
 - doasvm_feature_extraction_config, 249
 - MHAFilter::fftfiler_t, 467
 - MHAFilter::fftfilerbank_t, 471
 - MHAFilter::partitioned_convolution_t, 498
 - MHAFilter::smoothspec_t, 508
 - overlapadd::overlapadd_t, 793
- fft_t
 - MHASignal::fft_t, 704
- fftfb_ac_info_t
 - MHAOvFilter::fftfb_ac_info_t, 549
- fftfb_interface_t
 - fftfilerbank::fftfb_interface_t, 287
- fftfb_plug_t
 - fftfilerbank::fftfb_plug_t, 291
 - multibandcompressor::fftfb_plug_t, 777
- fftfb_t
 - MHAOvFilter::fftfb_t, 551
- fftfb_vars_t
 - MHAOvFilter::fftfb_vars_t, 554
- fftfiler_t
 - MHAFilter::fftfiler_t, 464
- fftfilerbank, 83
- fftfilerbank.cpp, 893
- fftfilerbank::fftfb_interface_t, 286
 - algo, 289
 - fftfb_interface_t, 287
 - nbands, 289
 - nchannels, 289
 - patchbay, 289
 - prepare, 287
 - prepared, 289
 - process, 289
 - release, 289
 - return_imag, 289
 - update_cfg, 289
- fftfilerbank::fftfb_plug_t, 290
 - fb_acinfo, 291
 - fftfb_plug_t, 291
 - imag, 291
 - insert, 291
 - process, 291
 - return_imag_, 291
 - s_out, 291
- fftfilerbank_t
 - MHAFilter::fftfilerbank_t, 468
- fftlens
 - dc_simple::level_smoother_t, 235
 - doasvm_feature_extraction, 248
 - doasvm_feature_extraction_config, 249
 - MHAFilter::fftfiler_t, 466
 - MHAFilter::fftfilerbank_t, 470
 - MHAFilter::smoothspec_t, 508
 - MHAOvFilter::fftfb_t, 552
 - MHAOvFilter::overlap_save_filerbank_↔
t::vars_t, 565
 - MHAParser::mhaconfig_mon_t, 615
 - mhaconfig_t, 445
 - rmslevel_t, 819
 - timoConfig, 860
- fftw_plan_fft
 - MHASignal::fft_t, 706
- fftw_plan_ifft
 - MHASignal::fft_t, 706
- fftw_plan_spec2wave
 - MHASignal::fft_t, 706
- fftw_plan_wave2spec
 - MHASignal::fft_t, 706
- fhz2bandno
 - speechnoise.cpp, 978
- fifo
 - wavwriter_t, 874
- fifo_size
 - mha_dblbuf_t, 379
- fifolen
 - analysispath_if_t, 183
 - wavrec_t, 873
- fileformat
 - acsave::acsave_t, 145
- filename_input
 - io_file_t, 312
- filename_output
 - io_file_t, 312
- fill_info
 - MHAIOPortAudio::device_info_t, 519
- filled

- MHASignal::async_rmslevel_t, 696
- filter
 - MHAFilter::adapt_filter_state_t, 455
 - MHAFilter::adapt_filter_t, 456
 - MHAFilter::complex_bandpass_t, 461, 462
 - MHAFilter::fftfiler_t, 465, 466
 - MHAFilter::fftfilerbank_t, 469, 470
 - MHAFilter::filter_t, 473, 474
 - MHAFilter::iir_filter_t, 481
- filter_activated
 - droptect_t, 252
- filter_analytic
 - MHAOvIFilter::overlap_save_filterbank_↔
analytic_t, 562
- filter_partitions
 - MHAFilter::partitioned_convolution_t, 496
- filter_t
 - MHAFilter::filter_t, 472, 473
- filterbank
 - dc::dc_vars_t, 222
 - dc_simple::dc_if_t, 228
- filtered_level
 - dc::dc_vars_t, 222
- filtered_powspec
 - droptect_t, 252
- filtered_powspec_mon
 - droptect_t, 252
- filtershapefun
 - mha_fftfb.cpp, 915
- fir
 - calibrator_runtime_layer_t, 193
 - calibrator_variables_t, 197
- firchannels
 - MHAFilter::fftfilerbank_t, 470
- firfir2fftltn
 - calibrator_runtime_layer_t, 193
- firfirltn
 - calibrator_runtime_layer_t, 193
- flag_terminate_inner_thread
 - analysepath_t, 181
- flags
 - MHAJack::client_t, 534
- float_mon_t
 - MHAParser::float_mon_t, 592
- float_t
 - MHA_AC::float_t, 363
 - MHAParser::float_t, 595
- flush_data
 - acsave::cfg_t, 146
- fmap_t
 - AuditoryProfile::parser_t::fmap_t, 188
- fmax
 - frequency_translator_t, 293
- fmin
 - frequency_translator_t, 293
- fname
 - acsave::acsave_t, 145
- for_each
 - Vector and matrix processing toolbox, 42
- force_remove_item
 - MHAParser::parser_t, 622
- force_resize
 - dc_simple, 81
- forward
 - lpc_bl_predictor_config, 344
 - lpc_burglattice_config, 348
 - MHASignal::fft_t, 704
- forward_scale
 - MHASignal::fft_t, 705
- fr
 - spec_fader_t, 849
- frag_out
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 528
- fragsize
 - analysispath_if_t, 183
 - calibrator_variables_t, 197
 - db_if_t, 214
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_sound_t, 332
 - MHAFilter::fftfiler_t, 466
 - MHAFilter::fftfilerbank_t, 470
 - MHAFilter::partitioned_convolution_t, 496
 - MHAFilter::resampling_filter_t, 505
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 533
 - MHAParser::mhaconfig_mon_t, 615
 - MHAPlugin_Resampling::resampling_if_↔
_t, 665
 - mhaconfig_t, 445
- fragsize_in
 - MHAFilter::blockprocessing_polyphase_↔
_resampling_t, 459
- fragsize_out
 - MHAFilter::blockprocessing_polyphase_↔
_resampling_t, 459
- fragsize_validator
 - MHAFilter::resampling_filter_t, 505
- frame
 - MHA_AC::acspace2matrix_t, 361

- framecnt
 - acsave::save_var_t, [149](#)
- frameno
 - MHA_AC::acspace2matrix_t, [361](#)
 - noisePowProposedScale::noisePow↔
Proposed, [789](#)
- frames
 - ac2wave_t, [127](#)
 - prediction_error_config, [815](#)
 - rt_nlms_t, [824](#)
- frameshift
 - hilbert_shifter_t, [306](#)
- framework_thread_priority
 - MHAPLugin_Split::split_t, [682](#)
- framework_thread_scheduler
 - MHAPLugin_Split::split_t, [681](#)
- freq2bin
 - Vector and matrix processing toolbox, [44](#)
- frequency
 - sine_t, [837](#)
- frequency_response
 - MHAFilter::partitioned_convolution_t, [497](#)
- frequency_translator_t, [292](#)
 - df, [293](#)
 - fmax, [293](#)
 - fmin, [293](#)
 - frequency_translator_t, [293](#)
 - irslens, [293](#)
 - patchbay, [293](#)
 - phasemode, [293](#)
 - prepare, [293](#)
 - process, [293](#)
 - release, [293](#)
 - update, [293](#)
- front_channel
 - adm_rtconfig_t, [170](#)
- front_channels
 - adm_if_t, [167](#)
 - adm_rtconfig_t, [170](#)
- frozen_noise_
 - cfg_t, [199](#)
- frozennoise_length
 - noise_t, [785](#)
- fs
 - MHAFilter::o1_ar_filter_t, [488](#)
- fs_
 - MHAOvFilter::fspacing_t, [560](#)
- fscale
 - MHAOvFilter::fftfb_vars_t, [554](#)
- fscale_bw_t
 - MHAOvFilter::fscale_bw_t, [556](#)
- fscale_t
 - MHAOvFilter::fscale_t, [558](#)
- fshift_hilbert.cpp, [893](#)
- fspacing_t
 - MHAOvFilter::fspacing_t, [560](#)
- ft
 - spec2wave_t, [847](#)
 - wave2spec_t, [870](#)
- ftype
 - MHAOvFilter::fftfb_vars_t, [554](#)
- fu
 - rt_nlms_t, [825](#)
- fu_previous
 - rt_nlms_t, [825](#)
- fufit
 - rt_nlms_t, [825](#)
- fullname
 - dynamiclib_t, [255](#)
 - MHAParser::base_t, [575](#)
 - plugindescription_t, [800](#)
- fullspec
 - hilbert_shifter_t, [306](#)
- fun_t
 - MHAWindow::fun_t, [769](#)
- funcs
 - MHAOvFilter::scale_var_t, [567](#)
- fw_cmd
 - fw_t, [298](#)
- fw_exiting
 - fw_t, [296](#)
- fw_fragsize
 - MHAIOJack::io_jack_t, [516](#)
- fw_running
 - fw_t, [296](#)
- fw_samplerate
 - MHAIOJack::io_jack_t, [516](#)
- fw_sleep
 - fw_t, [298](#)
- fw_sleep_cmd
 - fw_t, [297](#)
- fw_starting
 - fw_t, [296](#)
- fw_stopped
 - fw_t, [296](#)
- fw_stopping
 - fw_t, [296](#)
- fw_t, [294](#)
 - ~fw_t, [296](#)
 - ac, [298](#)
 - async_poll_msg, [298](#)
 - async_read, [298](#)

- b_exit_request, 299
- cfin, 299
- cfout, 299
- dump_mha, 298
- errorlog, 298
- exec_fw_command, 297
- exit_on_stop, 298
- exit_request, 296
- fatallog, 298
- fw_cmd, 298
- fw_exiting, 296
- fw_running, 296
- fw_sleep, 298
- fw_sleep_cmd, 297
- fw_starting, 296
- fw_stopped, 296
- fw_stopping, 296
- fw_t, 296
- fw_unprepared, 296
- fw_until, 298
- fw_until_cmd, 297
- get_input_signal_dimension, 297
- get_parserstate, 298
- inst_name, 298
- io_error, 299
- io_lib, 299
- io_name, 298
- load_io_lib, 297
- load_proc_lib, 297
- nchannels_out, 298
- parserstate, 298
- patchbay, 299
- plugin_paths, 298
- plugins, 298
- prepare, 296
- prepare_vars, 298
- proc_error, 299
- proc_error_string, 299
- proc_lib, 299
- proc_name, 298
- process, 297
- quit, 297
- release, 297
- start, 296
- started, 297
- state, 299
- state_t, 296
- stop, 296
- stopped, 297
- fw_unprepared
 - fw_t, 296
- fw_until
 - fw_t, 298
- fw_until_cmd
 - fw_t, 297
- fw_vars_t, 299
 - fw_vars_t, 300
 - lock_channels, 300
 - lock_srate_fragsize, 300
 - pfragmentsize, 300
 - pinchannels, 300
 - psrate, 300
 - unlock_channels, 300
 - unlock_srate_fragsize, 300
- fwcb
 - io_tcp_t, 336
- G
 - doasvm_feature_extraction_config, 249
- g
 - coherence::cohflt_t, 203
- g50
 - dc_simple::dc_vars_t, 232
- g80
 - dc_simple::dc_vars_t, 232
- G_ERRNO
 - MHA_TCP, 88
- G_length
 - doasvm_feature_extraction_config, 249
- g_scale
 - dc::wb_inhib_cfg_t, 224
 - dc::wideband_inhib_vars_t, 225
- GCC_end
 - doasvm_feature_extraction_config, 249
- GCC_start
 - doasvm_feature_extraction_config, 249
- GITCOMMITHASH
 - mha_plugin.hh, 931
- GLRDebug
 - noisePowProposedScale::noisePow↔
 - Proposed, 789
- GLRexp
 - noisePowProposedScale::noisePow↔
 - Proposed, 789
 - timoConfig, 861
- GLR
 - timoConfig, 861
- GREETING_TEXT
 - mhamain.cpp, 969
- gain, 84
 - calibrator_runtime_layer_t, 193
 - coherence::cohflt_t, 204
 - fader_wave::fader_wave_if_t, 284

- multibandcompressor::plugin_signals_t, 781
- gain.cpp, 893
- gain::gain_if_t, 301
 - bbgain, 302
 - gain_if_t, 302
 - gains, 302
 - patchbay, 302
 - prepare, 302
 - process, 302
 - release, 302
 - update_bbgain, 302
 - update_gain, 302
 - update_minmax, 302
 - vmax, 302
 - vmin, 302
- gain::scaler_t, 303
 - scaler_t, 303
- gain_ac
 - ac2wave_if_t, 126
 - ac2wave_t, 128
- gain_delay
 - coherence::cohflt_t, 204
- gain_if_t
 - gain::gain_if_t, 302
- gain_in
 - ac2wave_if_t, 126
 - ac2wave_t, 128
- gain_min
 - timoConfig, 860
- gain_min_db
 - timo_params, 858
 - timoSmooth, 865
- gain_spec_
 - cfg_t, 199
- gain_wave_
 - cfg_t, 199
- gain_wiener
 - timoConfig, 861
- gain_wiener_AC
 - timo_AC, 856
- gainrule
 - dc::dc_vars_t, 222
 - dc_simple::dc_if_t, 228
- gains
 - gain::gain_if_t, 302
 - prediction_error, 813
 - shadowfilter_end::cfg_t, 832
 - spec_fader_t, 849
- gaintable.cpp, 894
 - convert_f2logf, 894
 - isempty, 894
- gaintable.h, 894
- gaintable_t
 - DynComp::gaintable_t, 263
- gammaflt_t
 - MHAFilter::gammaflt_t, 476
- gamma_post
 - timoConfig, 861
- gamma_post_AC
 - timo_AC, 855
- gauss
 - MHAOvFilter::ShapeFun, 103
- gcd
 - MHAFilter, 92
- generatemhaplugindoc.cpp, 895
 - conv2latex, 895
 - create_latex_doc, 895
 - main, 895
- get_A
 - MHAFilter::gammaflt_t, 477
- get_a
 - MHAParser::base_t::replace_t, 577
- get_ac
 - latex_doc_t, 337
 - plug_t, 797
- get_accept_event
 - MHA_TCP::Server, 422
- get_all_input_ports
 - MHAIOJack::io_jack_t, 516
- get_all_output_ports
 - MHAIOJack::io_jack_t, 516
- get_audiochannels
 - dc, 80
- get_available_space
 - mha_drifter_fifo_t, 384
 - mha_fifo_t, 397
- get_b
 - MHAParser::base_t::replace_t, 577
- get_bw_hz
 - MHAOvFilter::fscale_bw_t, 556
- get_c1
 - MHAFilter::o1flt_lowpass_t, 490
- get_c_handle
 - MHAKernel::algo_comm_class_t, 539
- get_categories
 - latex_doc_t, 337
 - PluginLoader::mhapluginloader_t, 809
- get_cdata
 - MHASignal::matrix_t, 721
- get_cf_fftbins
 - MHAOvFilter::fspacing_t, 560

- get_cf_hz
 - MHAFilter::thirdoctave_analyzer_t, 509
 - MHAOvFilter::fspacing_t, 560
- get_cfin
 - MHAParser::mhapluginloader_t, 618
- get_cfout
 - MHAParser::mhapluginloader_t, 618
- get_channelconfig
 - MHAOvFilter::overlap_save_filterbank_t, 564
- get_comm_var
 - MHASignal::matrix_t, 716
- get_configfile
 - PluginLoader::config_file_splitter_t, 802
- get_configname
 - PluginLoader::config_file_splitter_t, 801
- get_connected
 - io_tcp_parser_t, 326
- get_cpu_load
 - MHAJack::client_t, 532
- get_current_profile
 - AuditoryProfile::parser_t, 186
- get_delay
 - mha_dblbuf_t, 377
- get_delays_in
 - MHAIOJack::io_jack_t, 516
- get_delays_out
 - MHAIOJack::io_jack_t, 516
- get_des_fill_count
 - mha_drifter_fifo_t, 384
- get_documentation
 - PluginLoader::mhapluginloader_t, 809
- get_ear
 - AuditoryProfile::parser_t::ear_t, 187
 - AuditoryProfile::profile_t, 189
- get_ef_hz
 - MHAOvFilter::fspacing_t, 560
- get_entries
 - algo_comm_t, 175
 - MHAKernel::algo_comm_class_t, 540
 - MHAParser::keyword_list_t, 602
- get_error
 - algo_comm_t, 175
 - MHAKernel::algo_comm_class_t, 540
- get_f_hz
 - MHAOvFilter::fscale_t, 558
- get_fbpower
 - MHAOvFilter::fftfb_t, 551
- get_fbpower_db
 - MHAOvFilter::fftfb_t, 551
- get_fd
 - MHA_TCP::Connection, 414
- get_fftlens
 - MHAOvFilter::fftfb_t, 552
- get_fifo_size
 - mha_dblbuf_t, 377
- get_fill_count
 - mha_drifter_fifo_t, 384
 - mha_fifo_t, 397
- get_fmap
 - AuditoryProfile::parser_t::fmap_t, 188
- get_fragsize
 - MHAJack::client_t, 531
- get_frequencies
 - AuditoryProfile::fmap_t, 184
- get_fun
 - MHAOvFilter::scale_var_t, 567
- get_gain
 - DynComp::gaintable_t, 264, 265
- get_handle
 - plug_t, 797
- get_index
 - MHAParser::keyword_list_t, 602
 - MHASignal::matrix_t, 720
- get_inner_error
 - mha_dblbuf_t, 377
- get_inner_size
 - mha_dblbuf_t, 377
- get_input_channels
 - mha_dblbuf_t, 377
- get_input_fifo_fill_count
 - mha_dblbuf_t, 377
- get_input_fifo_space
 - mha_dblbuf_t, 377
- get_input_signal_dimension
 - fw_t, 297
- get_interface
 - MHA_TCP::Server, 422
- get_iofun
 - DynComp::gaintable_t, 265
- get_irs
 - MHAFilter::fftfilterbank_t, 470
- get_last_name
 - MHAParser::mhapluginloader_t, 618
- get_last_output
 - MHAFilter::o1flt_lowpass_t, 490
- get_latex_doc
 - latex_doc_t, 337
- get_len_A
 - MHAFilter::filter_t, 474
- get_len_B
 - MHAFilter::filter_t, 474

- get_length
 - MHASignal::uint_vector_t, [742](#)
- get_level
 - fader_wave::level_adapt_t, [286](#)
- get_level_in_db
 - dc::dc_t, [219](#)
- get_level_in_db_adjusted
 - dc::dc_t, [219](#)
- get_libname
 - PluginLoader::config_file_splitter_t, [802](#)
- get_local_address
 - io_tcp_parser_t, [325](#)
- get_local_port
 - io_tcp_parser_t, [325](#)
- get_longmsg
 - MHA_Error, [388](#)
- get_ltass_gain_db
 - MHAOvFilter::fftfb_t, [552](#)
- get_main_category
 - latex_doc_t, [337](#)
- get_mapping
 - MHASignal::loop_wavefragment_t, [711](#)
- get_max_fill_count
 - mha_fifo_t, [397](#)
- get_min_fill_count
 - mha_drifter_fifo_t, [385](#)
- get_msg
 - MHA_Error, [388](#)
- get_my_input_ports
 - MHAJack::client_t, [532](#)
- get_my_output_ports
 - MHAJack::client_t, [532](#)
- get_name
 - MHAOvFilter::scale_var_t, [567](#)
- get_nbands
 - dc::dc_t, [219](#)
- get_nelements
 - MHASignal::matrix_t, [717](#)
- get_nreals
 - MHASignal::matrix_t, [720](#)
- get_origname
 - PluginLoader::config_file_splitter_t, [802](#)
- get_os_event
 - MHA_TCP::Timeout_Event, [431](#)
 - MHA_TCP::Wakeup_Event, [434](#)
- get_outer_size
 - mha_dblbuf_t, [377](#)
- get_output_channels
 - mha_dblbuf_t, [377](#)
- get_output_fifo_fill_count
 - mha_dblbuf_t, [377](#)
- get_output_fifo_space
 - mha_dblbuf_t, [377](#)
- get_parser_tab
 - latex_doc_t, [337](#)
- get_parser_var
 - latex_doc_t, [337](#)
- get_parserstate
 - fw_t, [298](#)
- get_paths
 - pluginbrowser_t, [799](#)
- get_peer_address
 - MHA_TCP::Connection, [414](#)
- get_peer_port
 - MHA_TCP::Connection, [414](#)
- get_physical_input_ports
 - MHAIOJack::io_jack_t, [516](#)
- get_physical_output_ports
 - MHAIOJack::io_jack_t, [516](#)
- get_plugins
 - pluginbrowser_t, [799](#)
- get_port
 - MHA_TCP::Server, [422](#)
- get_port_capture_latency
 - MHAJack, [95](#)
- get_port_capture_latency_int
 - MHAJack, [95](#)
- get_port_playback_latency
 - MHAJack, [95](#)
- get_port_playback_latency_int
 - MHAJack, [96](#)
- get_ports
 - MHAJack::client_t, [532](#)
- get_precision
 - MHAParser, [106](#)
- get_process_spec
 - plug_t, [797](#)
- get_process_wave
 - plug_t, [797](#)
- get_rdata
 - MHASignal::matrix_t, [721](#)
- get_read_event
 - MHA_TCP::Connection, [414](#)
- get_resynthesis_gain
 - MHAFilter::gammaflt_t, [477](#)
- get_server_port_open
 - io_tcp_parser_t, [326](#)
- get_short_name
 - MHAJack::port_t, [537](#)
- get_signal
 - MHAPLugin_Split::domain_handler_t, [671](#)
- get_size

- MHASignal::waveform_t, 753
- get_srate
 - MHAJack::client_t, 532
- get_type
 - MHAParser::window_t, 652
- get_value
 - MHAParser::keyword_list_t, 602
- get_values
 - AuditoryProfile::fmap_t, 184
- get_var
 - algo_comm_t, 174
 - MHAKernel::algo_comm_class_t, 539
- get_var_float
 - algo_comm_t, 174
 - Communication between algorithms, 29
 - MHAKernel::algo_comm_class_t, 539
- get_var_int
 - algo_comm_t, 174
 - Communication between algorithms, 29
 - MHAKernel::algo_comm_class_t, 539
- get_var_spectrum
 - Communication between algorithms, 28
- get_var_vfloat
 - Communication between algorithms, 30
- get_var_waveform
 - Communication between algorithms, 29
- get_vF
 - DynComp::gaintable_t, 265
- get_vL
 - DynComp::gaintable_t, 265
- get_weights
 - MHAFilter::complex_bandpass_t, 461
 - MHAFilter::gammaflt_t, 477
- get_window
 - MHAParser::window_t, 651, 652
- get_window_data
 - windowselector_t, 876
- get_write_event
 - MHA_TCP::Connection, 414
- get_xlimits
 - MHATableLookup::xy_table_t, 764
- get_xruns
 - MHAJack::client_t, 532
- get_xruns_reset
 - MHAJack::client_t, 532
- getcipd
 - coherence, 79
- getdata
 - MHASignal::uint_vector_t, 742
- getfullname
 - PluginLoader::mhapluginloader_t, 808
- getmodulename
 - dynamiclib_t, 255
- Getmsg
 - mha_error.hh, 913
- getname
 - dynamiclib_t, 255
 - MHA_AC::ac2matrix_t, 357
- getusername
 - MHA_AC::ac2matrix_t, 358
- getvar
 - acmon::ac_monitor_t, 134
 - MHA_AC::ac2matrix_helper_t, 355
- GF
 - MHAFilter::gammaflt_t, 477
- groupdelay_t
 - MHASignal::schroeder_t, 729
- gt
 - dc::dc_t, 219
- gtdata
 - dc::dc_vars_t, 221
- gtmin
 - dc::dc_vars_t, 221
- gtstep
 - dc::dc_vars_t, 221
- h
 - dynamiclib_t, 255
 - MHASignal::hilbert_t, 708
- H_ERRNO
 - MHA_TCP, 88
- HELP_TEXT
 - mhamain.cpp, 969
- HINSTANCE
 - mha_plugin.hh, 931
- HSTRError
 - MHA_TCP, 87
- HTL
 - AuditoryProfile::parser_t::ear_t, 187
 - AuditoryProfile::profile_t::ear_t, 190
- hamming
 - MHAWindow, 122
- hamming_t
 - MHAWindow::hamming_t, 771
- handle
 - algo_comm_t, 171
- hann
 - hann.cpp, 896
 - hann.h, 896
 - MHAOvFilter::ShapeFun, 101
- hann.cpp, 895
 - hann, 896
 - hannf, 896

- PI, 896
- hann.h, 896
 - hann, 896
 - hannf, 896
- hannf
 - hann.cpp, 896
 - hann.h, 896
- hanning
 - MHAWindow, 122
- hanning_ramps_t, 303
 - ~hanning_ramps_t, 304
 - hanning_ramps_t, 304
 - len_a, 304
 - len_b, 304
 - operator(), 304
 - ramp_a, 304
 - ramp_b, 304
- hanning_t
 - MHAWindow::hanning_t, 772
- hardlimit
 - softclipper_t, 843
 - softclipper_variables_t, 845
- has_been_modified
 - dc_simple::dc_if_t, 228
- has_inner_error
 - analysepath_t, 181
- has_key
 - MHAKernel::comm_var_map_t, 541
- has_parser
 - PluginLoader::mhapluginloader_t, 808
- has_process
 - PluginLoader::mhapluginloader_t, 808
- header
 - io_tcp_sound_t, 331
- help
 - MHAParser::base_t, 576
- hifftwin
 - doasvm_feature_extraction_config, 249
- hifftwin_sum
 - doasvm_feature_extraction_config, 249
- high_side_flat
 - MHAOvFilter::band_descriptor_t, 547
- hilbert
 - MHASignal::hilbert_fftw_t, 707
- hilbert_fftw_t
 - MHASignal::hilbert_fftw_t, 707
- hilbert_shifter_t, 305
 - ~hilbert_shifter_t, 306
 - analytic, 306
 - dphi, 306
 - frameshift, 306
 - fullspec, 306
 - hilbert_shifter_t, 306
 - kmax, 306
 - kmin, 306
 - mhafft, 306
 - mixw_ref, 306
 - mixw_shift, 306
 - phi, 306
 - plan_spec2analytic, 306
 - process, 306
 - shifted, 306
- hilbert_t
 - MHASignal::hilbert_t, 708
- host_port_to_sock_addr
 - mha_tcp.cpp, 948
- hostApi
 - MHAIOPortAudio::device_info_t, 519
- Hs
 - MHAFilter::fftfilterbank_t, 470
- hton
 - io_tcp_sound_t, 332
- hw
 - MHAFilter::fftfilterbank_t, 470
- hwin
 - doasvm_feature_extraction_config, 249
- hz2bark
 - MHAOvFilter::FreqScaleFun, 99
- hz2bark_analytic
 - MHAOvFilter::FreqScaleFun, 100
- hz2bark_t
 - MHAOvFilter::barkscale::hz2bark_t, 549
- hz2erb
 - MHAOvFilter::FreqScaleFun, 100
- hz2erb_glasberg1990
 - MHAOvFilter::FreqScaleFun, 100
- hz2hz
 - MHAOvFilter::FreqScaleFun, 99
 - speechnoise.cpp, 978
- hz2khz
 - MHAOvFilter::FreqScaleFun, 99
- hz2log
 - MHAOvFilter::FreqScaleFun, 100
- hz2octave
 - MHAOvFilter::FreqScaleFun, 99
- hz2third_octave
 - MHAOvFilter::FreqScaleFun, 99
- hz2unit
 - MHAOvFilter::scale_var_t, 567
- i
 - io_tcp_sound_t::float_union, 333
- INSERT_PATCH

- acConcat_wave.cpp, 878
- acPooling_wave.cpp, 879
- acSteer.cpp, 880
- acTransform_wave.cpp, 881
- doasvm_classification.cpp, 889
- doasvm_feature_extraction.cpp, 890
- lpc.cpp, 898
- lpc_bl_predictor.cpp, 898
- lpc_burg-lattice.cpp, 899
- prediction_error.cpp, 974
- steerbf.cpp, 981
- timoSmooth.cpp, 983
- INSERT_VAR
 - timoSmooth.cpp, 983
- INVALID_SOCKET
 - mha_tcp.cpp, 948
- INVALID_THREAD_PRIORITY
 - MHAPlugin_Split, 113
- IO_ERROR_JACK
 - mhajack.h, 968
- IO_ERROR_MHAJACKLIB
 - mhajack.h, 968
- IODestroy
 - MHAIOFile.cpp, 953, 954
 - MHAIOJack.cpp, 956, 957
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IODestroy_cb
 - io_lib_t, 316
- IODestroy_t
 - mha_io_ifc.h, 920
- IOInit
 - MHAIOFile.cpp, 953
 - MHAIOJack.cpp, 955, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961
 - MHAITCP.cpp, 964, 965
- IOInit_cb
 - io_lib_t, 316
- IOInit_t
 - mha_io_ifc.h, 920
- IOPrepare
 - MHAIOFile.cpp, 953
 - MHAIOJack.cpp, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IOPrepare_cb
 - io_lib_t, 316
- IOPrepare_t
 - mha_io_ifc.h, 920
- mha_io_ifc.h, 920
- IOProcessEvent_t
 - mha_io_ifc.h, 920
- IORelease
 - MHAIOFile.cpp, 953, 954
 - MHAIOJack.cpp, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IORelease_cb
 - io_lib_t, 316
- IORelease_t
 - mha_io_ifc.h, 920
- IOSetVar
 - MHAIOFile.cpp, 953, 954
 - MHAIOJack.cpp, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IOSetVar_cb
 - io_lib_t, 316
- IOSetVar_t
 - mha_io_ifc.h, 920
- IOStart
 - MHAIOFile.cpp, 953, 954
 - MHAIOJack.cpp, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IOStart_cb
 - io_lib_t, 316
- IOStart_t
 - mha_io_ifc.h, 920
- IOStartedEvent_t
 - mha_io_ifc.h, 920
- IOStop
 - MHAIOFile.cpp, 953, 954
 - MHAIOJack.cpp, 956
 - MHAIOParser.cpp, 958, 959
 - MHAIOPortAudio.cpp, 961, 962
 - MHAITCP.cpp, 964, 965
- IOStop_cb
 - io_lib_t, 316
- IOStop_t
 - mha_io_ifc.h, 920
- IOStoppedEvent
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 527
- IOStoppedEvent_t
 - mha_io_ifc.h, 920
- IOStrError

- MHAIOFile.cpp, 953, 954
- MHAIOJack.cpp, 956, 957
- MHAIOParser.cpp, 958, 959
- MHAIOPortAudio.cpp, 961, 962
- MHAIOTCP.cpp, 964, 965
- IOStrError_cb
 - io_lib_t, 316
- IOStrError_t
 - mha_io_ifc.h, 920
- id
 - mha_channel_info_t, 373
- id_str
 - MHAParser::base_t, 576
- id_string
 - MHAParser::parser_t, 624
- identity
 - MHASignal::schroeder_t, 731
- identity.cpp, 896
- identity_t, 307
 - identity_t, 307
 - prepare, 308
 - process, 307, 308
 - release, 308
- idstr
 - mha_channel_info_t, 373
- iface
 - MHA_TCP::Server, 423
- ifft
 - doasvm_feature_extraction_config, 249
- ifftshift
 - ifftshift.cpp, 897
 - ifftshift.h, 897
- ifftshift.cpp, 896
- ifftshift, 897
- ifftshift.h, 897
- ifftshift, 897
- ignore
 - MHA_TCP::Event_Watcher, 419
- ignored_by
 - MHA_TCP::Wakeup_Event, 434
- iir_filter_state_t
 - MHAFilter::iir_filter_state_t, 479
- iir_filter_t
 - MHAFilter::iir_filter_t, 480
- iir_ord1_real_t
 - MHAFilter::iir_ord1_real_t, 483
- iirfilter.cpp, 897
- iirfilter_t, 308
 - iirfilter_t, 309
 - prepare_, 309
 - process, 309
- release_, 309
- ilen
 - fader_wave::level_adapt_t, 286
- im
 - mha_complex_t, 374
- imag
 - acsave::mat4head_t, 147
 - fftfilterbank::fftfb_plug_t, 291
 - MHASignal::matrix_t, 718–720
- imagfb
 - MHAOvFilter::overlap_save_filterbank_↔
analytic_t, 562
- impulse_response
 - MHAFilter::polyphase_resampling_t, 503
 - MHAFilter::transfer_function_t, 512
- in_buf
 - wave2spec_t, 871
- in_cfg
 - timo_params, 857
- in_spec
 - doasvm_feature_extraction_config, 249
 - shadowfilter_end::cfg_t, 832
- in_spec_copy
 - shadowfilter_begin::cfg_t, 829
- inbuf
 - MHA_TCP::Connection, 417
- inch
 - MHAJack::client_t, 534
- increase_condition
 - mha_fifo_posix_threads_t, 394
- increment
 - mha_fifo_posix_threads_t, 393
 - mha_fifo_thread_platform_t, 401
- index
 - MHAParser::keyword_list_t, 603
- index_t
 - MHAFilter::partitioned_convolution_t↔
::index_t, 498, 499
- inhib_gain
 - dc::dc_t, 220
- Init_mha_ruby
 - mha_ruby.cpp, 933
- init_peer_data
 - MHA_TCP::Connection, 413
- initialize
 - MHA_TCP::Server, 422
- inner2outer_resampling
 - MHAPlugin_Resampling::resampling_t,
667
- inner_ac_copy
 - analysepath_t, 181

- inner_error
 - analysepath_t, [181](#)
 - mha_dblbuf_t, [380](#)
- inner_fragsize
 - MHAPLugin_Resampling::resampling_t, [667](#)
- inner_in
 - MHASignal::doublebuffer_t, [703](#)
- inner_input
 - analysepath_t, [181](#)
- inner_out
 - MHASignal::doublebuffer_t, [703](#)
- inner_out_domain
 - analysepath_t, [181](#)
- inner_process
 - db_t, [215](#)
 - MHASignal::doublebuffer_t, [702](#)
- inner_process_wave2spec
 - analysepath_t, [181](#)
- inner_process_wave2wave
 - analysepath_t, [181](#)
- inner_signal
 - MHAPLugin_Resampling::resampling_t, [667](#)
- inner_size
 - mha_dblbuf_t, [379](#)
- inner_srate
 - MHAPLugin_Resampling::resampling_t, [667](#)
- input
 - io_parser_t, [319](#)
 - MHAJack::port_t, [535](#)
 - MHASignal::loop_wavefragment_t, [710](#)
 - mha_dblbuf_t, [378](#)
- input_cfg
 - MHAPLugin::plugin_t, [662](#)
- input_cfg_
 - MHAPLugin::plugin_t, [663](#)
- input_channels
 - adm_if_t, [167](#)
 - mha_dblbuf_t, [379](#)
- input_domain
 - PluginLoader::mhapluginloader_t, [808](#)
- input_fifo
 - mha_dblbuf_t, [379](#)
- input_level
 - dc::dc_vars_t, [222](#)
- input_portnames
 - MHAJack::client_t, [534](#)
- input_signal_spec
 - MHAFilter::partitioned_convolution_t, [497](#)
- input_signal_wave
 - MHAFilter::partitioned_convolution_t, [496](#)
- input_to_process
 - analysepath_t, [181](#)
- inputPow
 - noisePowProposedScale::noisePow↔Proposed, [789](#)
- inputSpec
 - noisePowProposedScale::noisePow↔Proposed, [789](#)
- inputchannels
 - MHAFilter::fftfilterbank_t, [470](#)
- insert
 - acPooling_wave_config, [141](#)
 - acSteer_config, [153](#)
 - coherence::cohflt_t, [203](#)
 - fftfilterbank::fftfb_plug_t, [291](#)
 - lpc_config, [350](#)
 - MHA_AC::ac2matrix_t, [358](#)
 - MHA_AC::acspace2matrix_t, [361](#)
 - MHA_AC::spectrum_t, [366](#)
 - MHA_AC::stat_t, [368](#)
 - MHA_AC::waveform_t, [370](#)
 - MHAOvIFilter::fftfb_ac_info_t, [549](#)
 - multibandcompressor::fftfb_plug_t, [777](#)
 - noisePowProposedScale::noisePow↔Proposed, [788](#)
 - prediction_error_config, [815](#)
 - rmslevel_t, [819](#)
 - rt_nlms_t, [824](#)
 - timo_AC, [855](#)
- insert_item
 - MHAParser::parser_t, [622](#)
- insert_items
 - windowselector_t, [876](#)
- insert_member
 - mha_parser.hh, [929](#)
- insert_var
 - algo_comm_t, [171](#)
 - MHAKernel::algo_comm_class_t, [539](#)
- insert_var_float
 - algo_comm_t, [172](#)
 - MHAKernel::algo_comm_class_t, [539](#)
- insert_var_int
 - algo_comm_t, [172](#)
 - MHAKernel::algo_comm_class_t, [539](#)
- inspect
 - MHAFilter::complex_bandpass_t, [462](#)
 - MHAFilter::gammaflt_t, [477](#)
 - MHASignal::delay_t, [699](#)
- inst_name

- fw_t, 298
- int_mon_t
 - MHAParser::int_mon_t, 597
- int_t
 - MHA_AC::int_t, 364
 - MHAParser::int_t, 599
- integrate
 - Vector and matrix processing toolbox, 47
- intensity
 - mha_signal.cpp, 936
- interface_t
 - delay::interface_t, 237
 - multibandcompressor::interface_t, 779
 - noisePowProposedScale::interface_t, 787
 - route::interface_t, 821
- interleaved
 - combc_if_t, 207
- interleaved_
 - combc_t, 208
- intern_level
 - MHASignal::loop_wavefragment_t, 712
- internal_fir
 - MHAFilter::smoothspec_t, 508
- internal_start
 - MHAJack::client_t, 533
- internal_stop
 - MHAJack::client_t, 533
- interp
 - MHATableLookup::linear_table_t, 757
 - MHATableLookup::table_t, 760
 - MHATableLookup::xy_table_t, 762
- interp1
 - DynComp, 82
- interp2
 - DynComp, 83
- inv_scale
 - MHAOvfFilter::FreqScaleFun, 100
- invalidate_window_data
 - windowselector_t, 877
- invert
 - coherence::vars_t, 205
- inwave
 - lpc_config, 351
- io
 - MHAJack, 95
 - MHAJack::client_avg_t, 524
 - MHAJack::client_noncont_t, 527
- io_err
 - io_tcp_fwcb_t, 323
- io_error
 - fw_t, 299
- io_file_t, 309
 - ~io_file_t, 311
 - b_prepared, 313
 - filename_input, 312
 - filename_output, 312
 - fragsize, 312
 - io_file_t, 311
 - length, 312
 - nchannels_file_in, 312
 - nchannels_in, 312
 - nchannels_out, 312
 - output_sample_format, 312
 - prepare, 311
 - proc_event, 312
 - proc_handle, 312
 - release, 311
 - s_file_in, 313
 - s_in, 312
 - s_out, 313
 - samplerate, 312
 - setlock, 311
 - sf_in, 313
 - sf_out, 313
 - sfinf_in, 313
 - sfinf_out, 313
 - start, 311
 - start_event, 312
 - start_handle, 312
 - startsample, 312
 - stop, 311
 - stop_event, 312
 - stop_handle, 312
 - stopped, 311
 - strict_channel_match, 312
 - strict_srate_match, 312
 - total_read, 313
- io_jack_t
 - MHAIOJack::io_jack_t, 515
- io_lib
 - fw_t, 299
- io_lib_t, 313
 - ~io_lib_t, 315
 - IODestroy_cb, 316
 - IOInit_cb, 316
 - IOPrepare_cb, 316
 - IORelease_cb, 316
 - IOSetVar_cb, 316
 - IOStart_cb, 316
 - IOStop_cb, 316
 - IOStrError_cb, 316
 - io_lib_t, 315

- lib_data, 316
- lib_err, 316
- lib_handle, 316
- lib_str_error, 316
- prepare, 315
- release, 316
- start, 315
- stop, 315
- test_error, 316
- io_name
 - fw_t, 298
- io_parser_t, 317
 - ~io_parser_t, 318
 - b_fw_started, 319
 - b_prepared, 319
 - b_starting, 319
 - b_stopped, 319
 - fragsize, 319
 - input, 319
 - io_parser_t, 318
 - nchannels_in, 319
 - nchannels_out, 319
 - output, 319
 - patchbay, 319
 - prepare, 318
 - proc_event, 319
 - proc_handle, 319
 - process_frame, 319
 - release, 319
 - s_in, 319
 - s_out, 319
 - start, 318
 - start_event, 319
 - start_handle, 319
 - started, 319
 - stop, 319
 - stop_event, 319
 - stop_handle, 319
 - stopped, 319
- io_portaudio_t
 - MHAIOPortAudio::io_portaudio_t, 521
- io_tcp_fwcb_t, 320
 - ~io_tcp_fwcb_t, 321
- io_tcp_fwcb_t, 321
 - proc_err, 323
 - proc_event, 322
 - proc_handle, 322
 - process, 321
 - set_errnos, 321
 - start, 321
 - start_event, 322
 - start_handle, 323
 - stop, 322
 - stop_event, 322
 - stop_handle, 323
- io_tcp_parser_t, 323
 - ~io_tcp_parser_t, 325
 - connected, 328
 - debug, 328
 - debug_file, 329
 - debug_filename, 329
 - get_connected, 326
 - get_local_address, 325
 - get_local_port, 325
 - get_server_port_open, 326
 - io_tcp_parser_t, 325
 - local_address, 328
 - local_port, 328
 - peer_address, 328
 - peer_port, 329
 - server_port_open, 328
 - set_connected, 327
 - set_local_port, 325
 - set_new_peer, 327
 - set_server_port_open, 326
- io_tcp_sound_t, 329
 - ~io_tcp_sound_t, 330
 - check_sound_data_type, 331
 - chunkbytes_in, 331
 - fragsize, 332
 - header, 331
 - hton, 332
 - io_tcp_sound_t, 330
 - ntoh, 331
 - num_inchannels, 332
 - num_outchannels, 333
 - prepare, 331
 - release, 331
 - s_in, 333
 - samplerate, 332
- io_tcp_sound_t::float_union, 333
 - c, 333
 - f, 333
 - i, 333
- io_tcp_t, 334
 - ~io_tcp_t, 335
 - accept_loop, 335
 - connection_loop, 335
 - fwcb, 336
 - io_tcp_t, 335
 - notify_release, 336

- notify_start, 336
- notify_stop, 336
- parse, 336
- parser, 336
- prepare, 335
- release, 335
- server, 336
- sound, 336
- start, 335
- stop, 335
- thread, 336
- job
 - MHAJack::port_t, 537
- irslen
 - frequency_translator_t, 293
- irslen_inner2outer
 - MHAPLugin_Resampling::resampling_if↔_t, 665
- irslen_outer2inner
 - MHAPLugin_Resampling::resampling_if↔_t, 665
- irswnd
 - MHAOvFilter::overlap_save_filterbank↔_t::vars_t, 565
 - smoothgains_bridge::overlapadd_if↔_t, 839
- is_complex
 - MHA_AC::ac2matrix_helper_t, 356
 - mha_audio_descriptor_t, 371
- is_playback_active
 - MHASignal::loop_wavefragment_t, 712
- is_prepared
 - adm_if_t, 167
 - MHAPLugin::plugin_t, 662
 - PluginLoader::mhapluginloader_t, 809
- is_prepared_
 - MHAPLugin::plugin_t, 663
- is_same_size
 - MHASignal::matrix_t, 717
- is_var
 - algo_comm_t, 173
 - MHAKernel::algo_comm_class_t, 539
- iscomplex
 - MHASignal::matrix_t, 717
- isempty
 - AuditoryProfile::fmap_t, 184
 - gaintable.cpp, 894
 - MHAFilter::transfer_function_t, 512
- isval
 - MHAParser::kw_t, 605
- iter
 - prediction_error_config, 816
- iterator
 - MHA_TCP::Event_Watcher, 419
- jack_error_handler
 - mhajack.cpp, 966
- jack_proc_cb
 - MHAJack::client_t, 533
- jack_xrun_cb
 - MHAJack::client_t, 533
- jc
 - MHAJack::client_t, 534
 - MHAJack::port_t, 537
- jstate_prev
 - MHAJack::client_t, 534
- k_inner
 - MHASignal::doublebuffer_t, 703
- k_nyquist
 - dc::dc_t, 220
- k_outer
 - MHASignal::doublebuffer_t, 703
- kappa
 - lpc_burglattice_config, 349
- kappa_block
 - lpc_burglattice_config, 349
- kappa_const
 - timo_params, 858
 - timoSmooth, 864
- keyword_list_t
 - MHAParser::keyword_list_t, 601
- kick_condition
 - MHAPLugin_Split::posix_threads_t, 676
- kick_thread
 - MHAPLugin_Split::dummy_threads_t, 673
 - MHAPLugin_Split::posix_threads_t, 676
 - MHAPLugin_Split::thread_platform_t, 689
- kicked
 - MHAPLugin_Split::posix_threads_t, 677
- km
 - lpc_bl_predictor_config, 345
- kmax
 - hilbert_shifter_t, 306
- kmin
 - hilbert_shifter_t, 306
- kth_smallest
 - MHASignal, 116
- kw_index2type
 - transducers.cpp, 984
- kw_t
 - MHAParser::kw_t, 604, 605
- L

- AuditoryProfile::parser_t, 186
- AuditoryProfile::profile_t, 190
- l_min
 - dc::wb_inhib_cfg_t, 224
 - dc::wideband_inhib_vars_t, 225
- l_new
 - fader_wave::level_adapt_t, 286
- l_old
 - fader_wave::level_adapt_t, 286
- LEVEL
 - dc_simple, 81
- LPSCALE
 - timoconfig.cpp, 982
- LTASS_combined
 - speechnoise_t, 850
- LTASS_female
 - speechnoise_t, 850
- LTASS_male
 - speechnoise_t, 850
- lambda
 - lpc_burglattice, 347
 - lpc_burglattice_config, 349
- lambda_ceps
 - timoConfig, 861
- lambda_ceps_AC
 - timo_AC, 856
- lambda_ceps_prev
 - timoConfig, 861
- lambda_ml_AC
 - timo_AC, 855
- lambda_ml_ceps
 - timoConfig, 861
- lambda_ml_ceps_AC
 - timo_AC, 855
- lambda_ml_full
 - timoConfig, 861
- lambda_ml_smooth
 - timoConfig, 861
- lambda_ml_smooth_AC
 - timo_AC, 855
- lambda_smoothing_power
 - nlms_t, 783
- lambda_spec
 - timoConfig, 861
- lambda_spec_AC
 - timo_AC, 856
- lambda_thresh
 - timo_params, 858
 - timoSmooth, 864
- last_complex_bin
 - MHASignal::subsample_delay_t, 739
- last_config
 - MHAPLugin::config_t, 658
- last_errormsg
 - MHAParser::parser_t, 625
- last_jack_err
 - mhajack.cpp, 966
- last_jack_err_msg
 - mhajack.cpp, 966
 - mhajack.h, 968
- last_name
 - MHAParser::mhapluginloader_t, 618
- latex_doc_t, 336
 - ac, 338
 - get_ac, 337
 - get_categories, 337
 - get_latex_doc, 337
 - get_main_category, 337
 - get_parser_tab, 337
 - get_parser_var, 337
 - latex_doc_t, 337
 - latex_plugname, 338
 - loader, 338
 - parsername, 337
 - plugin_macro, 338
 - plugname, 338
 - strdom, 337
- latex_plugname
 - latex_doc_t, 338
- len
 - MHA_AC::acspace2matrix_t, 361
 - MHAFilter::filter_t, 475
 - MHATableLookup::linear_table_t, 759
- len_A
 - MHAFilter::filter_t, 475
- len_a
 - hanning_ramps_t, 304
- len_B
 - MHAFilter::filter_t, 475
- len_b
 - hanning_ramps_t, 304
- length
 - io_file_t, 312
 - MHASignal::uint_vector_t, 742
- lev
 - noise_t, 785
 - sine_t, 837
- level
 - rmslevel_t, 819
- level_adapt_t
 - fader_wave::level_adapt_t, 285
- level_adaptor

- fader_wave, [83](#)
- level_db
 - rmslevel_t, [819](#)
- level_in_db
 - dc::dc_t, [220](#)
- level_in_db_adjusted
 - dc::dc_t, [220](#)
- level_mode_t
 - MHASignal::loop_wavefragment_t, [710](#)
- level_mon
 - droptect_t, [252](#)
- level_smoother_t
 - dc_simple::level_smoother_t, [235](#)
- level_spec
 - dc_simple::level_smoother_t, [235](#)
- level_wave
 - dc_simple::level_smoother_t, [235](#)
- Levinson2
 - lpc.cpp, [898](#)
- lib_data
 - io_lib_t, [316](#)
 - PluginLoader::mhapluginloader_t, [809](#)
- lib_err
 - io_lib_t, [316](#)
 - PluginLoader::mhapluginloader_t, [809](#)
- lib_handle
 - io_lib_t, [316](#)
 - PluginLoader::mhapluginloader_t, [809](#)
- lib_str_error
 - io_lib_t, [316](#)
- libdata
 - analysepath_t, [181](#)
 - MHAParser::c_ifc_parser_t, [584](#)
- liberr
 - MHAParser::c_ifc_parser_t, [584](#)
- libname
 - analysispath_if_t, [183](#)
 - PluginLoader::config_file_splitter_t, [802](#)
- library_paths
 - pluginbrowser_t, [800](#)
- like_ratio
 - acPooling_wave_config, [142](#)
- like_ratio_name
 - acPooling_wave, [140](#)
- limit
 - coherence::cohflt_t, [203](#)
 - coherence::vars_t, [205](#)
 - MHASignal, [116](#)
 - MHASignal::quantizer_t, [724](#)
 - MHASignal::waveform_t, [751](#)
- limiter
 - dc_simple::dc_t, [230](#)
- limiter_threshold
 - dc_simple::dc_t, [230](#)
 - dc_simple::dc_vars_t, [233](#)
- lin2db
 - Vector and matrix processing toolbox, [42](#)
- line_t
 - dc_simple::dc_t::line_t, [231](#)
- linear
 - MHAOvFilter::ShapeFun, [101](#)
 - softclipper_t, [843](#)
 - softclipper_variables_t, [845](#)
- linear_table_t
 - MHATableLookup::linear_table_t, [757](#)
- Linearphase_FIR
 - ADM::Linearphase_FIR, [163](#)
- list_dir
 - mha_os.cpp, [921](#)
 - mha_os.h, [923](#)
- load_io_lib
 - fw_t, [297](#)
- load_plug
 - MHAParser::mhapluginloader_t, [618](#)
- load_proc_lib
 - fw_t, [297](#)
- loader
 - latex_doc_t, [338](#)
- loadlib
 - analysispath_if_t, [183](#)
- local_address
 - io_tcp_parser_t, [328](#)
- local_get_entries
 - MHAKernel::algo_comm_class_t, [540](#)
- local_get_var
 - MHAKernel::algo_comm_class_t, [540](#)
- local_insert_var
 - MHAKernel::algo_comm_class_t, [540](#)
- local_is_var
 - MHAKernel::algo_comm_class_t, [540](#)
- local_port
 - io_tcp_parser_t, [328](#)
- local_remove_ref
 - MHAKernel::algo_comm_class_t, [540](#)
- local_remove_var
 - MHAKernel::algo_comm_class_t, [540](#)
- locate_end
 - MHASignal::loop_wavefragment_t, [712](#)
- lock_channels
 - fw_vars_t, [300](#)
- lock_srate_frgsize
 - fw_vars_t, [300](#)

locked
 MHAParser::variable_t, 633
log_down
 MHASignal::schroeder_t, 731
log_lambda_spec
 timoConfig, 861
log_lambda_spec_AC
 timo_AC, 856
log_up
 MHASignal::schroeder_t, 731
logGLRFact
 noisePowProposedScale::noisePow↔
 Proposed, 789
 timoConfig, 861
logfile
 mhaserver_t, 694
logstring
 mhaserver_t, 693
longmsg
 MHA_Error, 389
lookup
 MHATableLookup::linear_table_t, 757
 MHATableLookup::table_t, 760
 MHATableLookup::xy_table_t, 762
loop_wavefragment_t
 MHASignal::loop_wavefragment_t, 710
low_incl
 MHAParser::range_var_t, 627
low_limit
 MHAParser::range_var_t, 627
low_side_flat
 MHAOvIFilter::band_descriptor_t, 547
low_thresh
 acPooling_wave_config, 142
lower_threshold
 acPooling_wave, 140
lp
 DynComp::dc_afterburn_rt_t, 257
lp1i
 coherence::cohflt_t, 203
lp1ltg
 coherence::cohflt_t, 204
lp1r
 coherence::cohflt_t, 203
lp_coeffs
 adm_rtconfig_t, 170
lp_order
 adm_if_t, 167
lpc, 338
 ~lpc, 339
 algo_name, 340
 comp_each_iter, 340
 lpc, 339
 lpc_buffer_size, 340
 lpc_order, 340
 norm, 340
 patchbay, 340
 prepare, 339
 process, 339
 release, 340
 shift, 340
 update_cfg, 340
lpc.cpp, 897
 INSERT_PATCH, 898
 Levinson2, 898
 PATCH_VAR, 898
lpc.h, 898
lpc_bl_predictor, 341
 ~lpc_bl_predictor, 342
 lpc_bl_predictor, 342
 lpc_order, 343
 name_b, 343
 name_f, 343
 name_kappa, 343
 name_lpc_b, 343
 name_lpc_f, 343
 patchbay, 343
 prepare, 342
 process, 342
 release, 342
 update_cfg, 343
lpc_bl_predictor.cpp, 898
 INSERT_PATCH, 898
 PATCH_VAR, 898
lpc_bl_predictor.h, 898
 EPSILON, 899
lpc_bl_predictor_config, 343
 ~lpc_bl_predictor_config, 344
 ac, 344
 b_est, 344
 backward, 344
 f_est, 344
 forward, 344
 km, 345
 lpc_bl_predictor_config, 344
 lpc_order, 344
 name_b, 345
 name_f, 344
 name_km, 344
 process, 344
 s_b, 345
 s_f, 345

- lpc_buffer_size
 - lpc, 340
 - lpc_config, 351
- lpc_burg-lattice.cpp, 899
 - INSERT_PATCH, 899
 - PATCH_VAR, 899
- lpc_burg-lattice.h, 899
 - EPSILON, 899
- lpc_burglattice, 345
 - ~lpc_burglattice, 346
 - lambda, 347
 - lpc_burglattice, 346
 - lpc_order, 347
 - name_b, 347
 - name_f, 347
 - name_kappa, 347
 - patchbay, 347
 - prepare, 347
 - process, 347
 - release, 347
 - update_cfg, 347
- lpc_burglattice_config, 348
 - ~lpc_burglattice_config, 348
 - ac, 348
 - backward, 349
 - dm, 349
 - forward, 348
 - kappa, 349
 - kappa_block, 349
 - lambda, 349
 - lpc_burglattice_config, 348
 - lpc_order, 349
 - name_b, 349
 - name_f, 349
 - nm, 349
 - process, 348
 - s_b, 349
 - s_f, 349
- lpc_config, 349
 - ~lpc_config, 350
 - A, 351
 - comp_each_iter, 350
 - comp_iter, 351
 - corr_out, 351
 - insert, 350
 - inwave, 351
 - lpc_buffer_size, 351
 - lpc_config, 350
 - lpc_out, 351
 - N, 351
 - norm, 350
 - order, 350
 - process, 350
 - R, 351
 - sample, 351
 - shift, 350
- lpc_order
 - lpc, 340
 - lpc_bl_predictor, 343
 - lpc_bl_predictor_config, 344
 - lpc_burglattice, 347
 - lpc_burglattice_config, 349
 - prediction_error, 814
- lpc_out
 - lpc_config, 351
- ltgcomp
 - coherence::vars_t, 205
- ltgtau
 - coherence::vars_t, 205
- lval
 - MHAParser::expression_t, 591
- m
 - dc_simple::dc_t::line_t, 231
 - matrixmixer::cfg_t, 352
- M_PI
 - mha_defs.h, 908
 - mha_signal.hh, 945
- m_alphas
 - ADM::Linearphase_FIR, 164
- m_beta
 - ADM::ADM, 160
- m_coeff
 - ADM::Delay, 162
- m_decomb
 - ADM::ADM, 160
- m_delay_back
 - ADM::ADM, 160
- m_delay_front
 - ADM::ADM, 159
- m_fullsamples
 - ADM::Delay, 162
- m_lp_bf
 - ADM::ADM, 160
- m_lp_result
 - ADM::ADM, 160
- m_mu_beta
 - ADM::ADM, 160
- m_norm
 - ADM::Delay, 162
- m_now
 - ADM::Linearphase_FIR, 164
- m_now_in

- ADM::Delay, 162
- m_order
 - ADM::Linearphase_FIR, 164
- m_output
 - ADM::Linearphase_FIR, 164
- m_powerfilter_coeff
 - ADM::ADM, 160
- m_powerfilter_norm
 - ADM::ADM, 160
- m_powerfilter_state
 - ADM::ADM, 160
- m_state
 - ADM::Delay, 162
- MAX_LINE_LENGTH
 - mhamain.cpp, 969
- MAX_TCP_PORT_STR
 - MHA_IOTCP.cpp, 964
- MAX_TCP_PORT
 - MHA_IOTCP.cpp, 964
- MAX_USER_ERR
 - MHA_IOFile.cpp, 953
 - MHA_IOJack.cpp, 955
 - MHA_IOParser.cpp, 958
 - MHA_IOPortAudio.cpp, 961
 - MHA_IOTCP.cpp, 964
 - mhajack.h, 968
- MAX
 - mha_defs.h, 909
- MHA_AC::ac2matrix_helper_t, 355
 - ac, 356
 - ac2matrix_helper_t, 355
 - acvar, 356
 - getvar, 355
 - is_complex, 356
 - name, 356
 - size, 356
 - username, 356
- MHA_AC::ac2matrix_t, 356
 - ac2matrix_t, 357
 - getname, 357
 - getusername, 358
 - insert, 358
 - update, 357
- MHA_AC::acspace2matrix_t, 358
 - ~acspace2matrix_t, 360
 - acspace2matrix_t, 359
 - data, 361
 - frame, 361
 - framenos, 361
 - insert, 361
 - len, 361
 - operator=, 360
 - operator[], 360
 - size, 361
 - update, 361
- MHA_AC::double_t, 361
 - ~double_t, 362
 - ac, 362
 - data, 362
 - double_t, 362
- MHA_AC::float_t, 363
 - ~float_t, 363
 - ac, 363
 - data, 363
 - float_t, 363
- MHA_AC::int_t, 364
 - ~int_t, 364
 - ac, 364
 - data, 364
 - int_t, 364
- MHA_AC::spectrum_t, 365
 - ~spectrum_t, 366
 - ac, 367
 - insert, 366
 - name, 367
 - spectrum_t, 366
- MHA_AC::stat_t, 367
 - insert, 368
 - mean, 368
 - stat_t, 368
 - std, 368
 - update, 368
- MHA_AC::waveform_t, 368
 - ~waveform_t, 369
 - ac, 370
 - insert, 370
 - name, 370
 - waveform_t, 369
- MHA_AC_CHAR
 - mha.h, 904
- MHA_AC_DOUBLE
 - mha.h, 904
- MHA_AC_FLOAT
 - mha.h, 904
- MHA_AC_INT
 - mha.h, 904
- MHA_AC_MHACOMPLEX
 - mha.h, 904
- MHA_AC_MHAREAL
 - mha.h, 904
- MHA_AC_UNKNOWN
 - mha.h, 904

- MHA_AC_USER
 - mha.h, [904](#)
- MHA_AC_VEC_FLOAT
 - mha.h, [904](#)
- MHA_AC, [84](#)
- MHA_CALLBACK_TEST_PREFIX
 - mha.h, [902](#)
- MHA_CALLBACK_TEST
 - mha.h, [902](#)
- MHA_DOMAIN_MAX
 - mha.h, [904](#)
- MHA_DOMAIN_UNKNOWN
 - mha.h, [904](#)
- MHA_EAR_LEFT
 - mha_defs.h, [909](#)
- MHA_EAR_MAX
 - mha_defs.h, [909](#)
- MHA_EAR_RIGHT
 - mha_defs.h, [909](#)
- MHA_ERR_INVALID_HANDLE
 - mha_errno.h, [911](#)
- MHA_ERR_NULL
 - mha_errno.h, [911](#)
- MHA_ERR_SUCCESS
 - mha_errno.h, [911](#)
- MHA_ERR_UNKNOWN
 - mha_errno.h, [911](#)
- MHA_ERR_USER
 - mha_errno.h, [911](#)
- MHA_ERR_VARFMT
 - mha_errno.h, [911](#)
- MHA_ERR_VARRANGE
 - mha_errno.h, [911](#)
- MHA_Error, [387](#)
 - ~MHA_Error, [388](#)
 - get_longmsg, [388](#)
 - get_msg, [388](#)
 - longmsg, [389](#)
 - MHA_Error, [388](#)
 - msg, [389](#)
 - operator=, [388](#)
 - what, [388](#)
- MHA_ErrorMsg
 - Error handling in the openMHA, [31](#)
- MHA_ErrorMsg2
 - MHA_IOTCP.cpp, [964](#)
- MHA_ErrorMsg3
 - MHA_IOTCP.cpp, [964](#)
- MHA_ID_MATRIX
 - mha_signal.cpp, [936](#)
- MHA_ID_UINT_VECTOR
 - mha_signal.cpp, [936](#)
- MHA_RELEASE_VERSION_STRING
 - mha.h, [903](#)
- MHA_RESOLVE_CHECKED
 - mha_os.h, [923](#)
- MHA_RESOLVE
 - mha_os.h, [923](#)
- MHA_SPECTRUM
 - mha.h, [904](#)
- MHA_STRUCT_SIZEMATCH
 - mha.h, [903](#)
- MHA_STRF
 - mha.h, [902](#)
- MHA_TCP::Async_Notify, [408](#)
 - ~Async_Notify, [409](#)
 - Async_Notify, [409](#)
 - pipe, [409](#)
 - reset, [409](#)
 - set, [409](#)
- MHA_TCP::Client, [409](#)
 - Client, [410](#)
- MHA_TCP::Connection, [411](#)
 - ~Connection, [413](#)
 - buffered_incoming_bytes, [417](#)
 - buffered_outgoing_bytes, [417](#)
 - can_read_bytes, [415](#)
 - can_read_line, [415](#)
 - can_sysread, [413](#)
 - can_syswrite, [413](#)
 - closed, [417](#)
 - Connection, [413](#)
 - eof, [414](#)
 - fd, [417](#)
 - get_fd, [414](#)
 - get_peer_address, [414](#)
 - get_peer_port, [414](#)
 - get_read_event, [414](#)
 - get_write_event, [414](#)
 - inbuf, [417](#)
 - init_peer_data, [413](#)
 - needs_write, [416](#)
 - outbuf, [417](#)
 - peer_addr, [417](#)
 - read_bytes, [416](#)
 - read_event, [417](#)
 - read_line, [415](#)
 - sysread, [413](#)
 - syswrite, [414](#)
 - try_write, [416](#)
 - write, [416](#)
 - write_event, [417](#)

- MHA_TCP::Event_Watcher, 418
 - ~Event_Watcher, 419
 - Events, 419
 - events, 419
 - ignore, 419
 - iterator, 419
 - observe, 419
 - wait, 419
- MHA_TCP::OS_EVENT_TYPE, 420
 - fd, 420
 - mode, 420
 - R, 420
 - T, 420
 - timeout, 420
 - W, 420
 - X, 420
- MHA_TCP::Server, 421
 - ~Server, 422
 - accept, 422
 - accept_event, 423
 - get_accept_event, 422
 - get_interface, 422
 - get_port, 422
 - iface, 423
 - initialize, 422
 - port, 423
 - Server, 421, 422
 - serversocket, 423
 - sock_addr, 423
 - try_accept, 423
- MHA_TCP::Sockaccept_Event, 424
 - Sockaccept_Event, 424
- MHA_TCP::Sockread_Event, 424
 - Sockread_Event, 425
- MHA_TCP::Sockwrite_Event, 426
 - Sockwrite_Event, 426
- MHA_TCP::Thread, 426
 - ~Thread, 428
 - arg, 429
 - error, 430
 - FINISHED, 428
 - PREPARED, 428
 - RUNNING, 428
 - return_value, 429
 - run, 429
 - state, 429
 - thr_f, 428
 - Thread, 428
 - thread_arg, 430
 - thread_attr, 429
 - thread_finish_event, 429
 - thread_func, 429
 - thread_handle, 429
- MHA_TCP::Timeout_Event, 430
 - end_time, 431
 - get_os_event, 431
 - Timeout_Event, 431
- MHA_TCP::Timeout_Watcher, 431
 - ~Timeout_Watcher, 432
 - timeout, 432
 - Timeout_Watcher, 432
- MHA_TCP::Wakeup_Event, 433
 - ~Wakeup_Event, 434
 - get_os_event, 434
 - ignored_by, 434
 - observed_by, 434
 - observers, 435
 - os_event, 435
 - os_event_valid, 435
 - reset, 434
 - status, 435
 - Wakeup_Event, 434
- MHA_TCP, 86
 - dtime, 88
 - G_ERRNO, 88
 - H_ERRNO, 88
 - HSTRERROR, 87
 - N_ERRNO, 87
 - SOCKET, 87
 - STRERROR, 87
 - stime, 88
- MHA_VERSION_BUILD
 - mha.h, 903
- MHA_VERSION_MAJOR
 - mha.h, 902
- MHA_VERSION_MINOR
 - mha.h, 902
- MHA_VERSION_RELEASE
 - mha.h, 903
- MHA_VERSION_STRING
 - mha.h, 903
- MHA_VERSION
 - mha.h, 903
- MHA_WAVEFORM
 - mha.h, 903
- MHA_XSTRF
 - mha.h, 902
- MHA_assert
 - Error handling in the openMHA, 32
- MHA_assert_equal
 - Error handling in the openMHA, 32
- MHADestroy_cb

- PluginLoader::mhappluginloader_t, 809
- MHADestroy_t
 - mha.h, 904
- MHAEvents, 88
- MHAEvents::connector_base_t, 446
 - ~connector_base_t, 446
 - connector_base_t, 446
 - emit_event, 447
 - emitter_die, 447
 - emitter_is_alive, 447
- MHAEvents::connector_t
 - ~connector_t, 449
 - connector_t, 449
 - emit_event, 449
 - emitter, 450
 - eventhandler, 450
 - eventhandler_s, 450
 - eventhandler_suu, 450
 - receiver, 450
- MHAEvents::connector_t< receiver_t >, 448
- MHAEvents::emitter_t, 450
 - ~emitter_t, 451
 - connect, 451
 - connections, 451
 - disconnect, 451
 - operator(), 451
- MHAEvents::patchbay_t
 - ~patchbay_t, 452
 - connect, 452, 453
 - cons, 453
- MHAEvents::patchbay_t< receiver_t >, 452
- MHAFilter, 89
 - butter_stop_ord1, 91
 - gcd, 92
 - make_friendly_number, 91
 - o1_lp_coeffs, 91
 - resampling_factors, 92
 - sinc, 92
 - spec2fir, 92
- MHAFilter::adapt_filter_param_t, 453
 - adapt_filter_param_t, 454
 - err_in, 454
 - mu, 454
- MHAFilter::adapt_filter_state_t, 454
 - adapt_filter_state_t, 455
 - filter, 455
 - nchannels, 455
 - ntaps, 455
 - od, 455
 - oy, 455
 - W, 455
 - X, 455
- MHAFilter::adapt_filter_t, 455
 - adapt_filter_t, 456
 - connector, 457
 - err_in, 457
 - filter, 456
 - mu, 457
 - nchannels, 457
 - ntaps, 457
 - set_channelcnt, 456
 - update_mu, 456
 - update_ntaps, 456
- MHAFilter::blockprocessing_polyphase_↔
 - resampling_t, 457
 - ~blockprocessing_polyphase_resampling_↔
 - _t, 458
 - blockprocessing_polyphase_resampling_↔
 - _t, 458
 - can_read, 459
 - fragsize_in, 459
 - fragsize_out, 459
 - num_channels, 459
 - read, 459
 - resampling, 459
 - write, 458
- MHAFilter::complex_bandpass_t, 460
 - A_, 462
 - B_, 462
 - complex_bandpass_t, 461
 - creator_A, 461
 - creator_B, 461
 - filter, 461, 462
 - get_weights, 461
 - inspect, 462
 - set_state, 461
 - set_weights, 461
 - Yn, 462
- MHAFilter::diff_t, 462
 - diff_t, 463
- MHAFilter::fftfilter_t, 463
 - ~fftfilter_t, 465
 - channels, 466
 - fft, 467
 - fftfilter_t, 464
 - fftlens, 466
 - filter, 465, 466
 - fragsize, 466
 - sInput, 467
 - sWeights, 467
 - update_coeffs, 465
 - wIRS_fft, 467

- wInput, 467
- wInput_fft, 466
- wOutput, 467
- wOutput_fft, 467
- MHAFilter::fftfilterbank_t, 467
 - ~fftfilterbank_t, 469
 - fft, 471
 - fftfilterbank_t, 468
 - fftlens, 470
 - filter, 469, 470
 - firchannels, 470
 - fragsize, 470
 - get_irs, 470
 - Hs, 470
 - hw, 470
 - inputchannels, 470
 - outputchannels, 470
 - tail, 471
 - update_coeffs, 469
 - Xs, 470
 - xw, 470
 - Ys, 471
 - yw, 470
 - yw_temp, 471
- MHAFilter::filter_t, 471
 - ~filter_t, 473
 - A, 474
 - B, 474
 - channels, 475
 - filter, 473, 474
 - filter_t, 472, 473
 - get_len_A, 474
 - get_len_B, 474
 - len, 475
 - len_A, 475
 - len_B, 475
 - state, 475
- MHAFilter::gammaflt_t, 475
 - ~gammaflt_t, 476
 - A, 477
 - bw_, 478
 - cf_, 478
 - delay, 477
 - envelope_delay, 478
 - gammaflt_t, 476
 - get_A, 477
 - get_resynthesis_gain, 477
 - get_weights, 477
 - GF, 477
 - inspect, 477
 - operator(), 476, 477
 - phase_correction, 477
 - reset_state, 477
 - resynthesis_gain, 478
 - set_weights, 477
 - srate_, 478
- MHAFilter::iir_filter_state_t, 478
 - iir_filter_state_t, 479
- MHAFilter::iir_filter_t, 479
 - A, 482
 - B, 482
 - connector, 482
 - filter, 481
 - iir_filter_t, 480
 - nchannels, 482
 - resize, 482
 - update_filter, 482
- MHAFilter::iir_ord1_real_t, 482
 - A_, 485
 - B_, 485
 - iir_ord1_real_t, 483
 - operator(), 484
 - set_state, 484
 - Yn, 485
- MHAFilter::o1_ar_filter_t, 485
 - c1_a, 488
 - c1_r, 488
 - c2_a, 488
 - c2_r, 488
 - fs, 488
 - o1_ar_filter_t, 486
 - operator(), 487, 488
 - set_tau_attack, 487
 - set_tau_release, 487
- MHAFilter::o1flt_lowpass_t, 488
 - get_c1, 490
 - get_last_output, 490
 - o1flt_lowpass_t, 490
 - set_tau, 490
- MHAFilter::o1flt_maxtrack_t, 491
 - o1flt_maxtrack_t, 492
 - set_tau, 492
- MHAFilter::o1flt_mintrack_t, 492
 - o1flt_mintrack_t, 494
 - set_tau, 494
- MHAFilter::partitioned_convolution_t, 494
 - ~partitioned_convolution_t, 496
 - bookkeeping, 497
 - current_input_signal_buffer_half_index, 497
 - current_output_partition_index, 497
 - fft, 498

- filter_partitions, 496
- fragsize, 496
- frequency_response, 497
- input_signal_spec, 497
- input_signal_wave, 496
- nchannels_in, 496
- nchannels_out, 496
- output_partitions, 496
- output_signal_spec, 497
- output_signal_wave, 497
- partitioned_convolution_t, 495
- process, 496
- MHAFilter::partitioned_convolution_t::index←
_t, 498
- delay, 499
- index_t, 498, 499
- source_channel_index, 499
- target_channel_index, 499
- MHAFilter::polyphase_resampling_t, 499
- downsampling_factor, 503
- impulse_response, 503
- now_index, 503
- polyphase_resampling_t, 501
- read, 502
- readable_frames, 502
- ringbuffer, 503
- underflow, 503
- upsampling_factor, 503
- write, 502
- MHAFilter::resampling_filter_t, 504
- fragsize, 505
- fragsize_validator, 505
- resampling_filter_t, 504
- MHAFilter::smoothspec_t, 505
- _linphase_asym, 508
- ~smoothspec_t, 507
- fft, 508
- fftlens, 508
- internal_fir, 508
- minphase, 508
- nchannels, 508
- smoothspec, 507
- smoothspec_t, 506
- spec2fir, 507
- tmp_spec, 508
- tmp_wave, 508
- window, 508
- MHAFilter::thirddoctave_analyzer_t, 508
- bw_generator, 509
- cf, 509
- cf_generator, 509
- cfg_, 509
- dup, 509
- fb, 510
- get_cf_hz, 509
- nbands, 509
- nchannels, 509
- out_chunk, 510
- out_chunk_im, 510
- process, 509
- thirddoctave_analyzer_t, 509
- MHAFilter::transfer_function_t, 510
- impulse_response, 512
- isempty, 512
- non_empty_partitions, 511
- partitions, 511
- source_channel_index, 512
- target_channel_index, 512
- transfer_function_t, 511
- MHAFilter::transfer_matrix_t, 513
- non_empty_partitions, 513
- partitions, 513
- MHAGetVersion_cb
- PluginLoader::mhapluginloader_t, 809
- MHAGetVersion_t
- mha.h, 904
- MHAIOFile.cpp, 952
- DEBUG, 953
- dummy_interface_test, 953, 954
- ERR_IHANDLE, 953
- ERR_SUCCESS, 953
- ERR_USER, 953
- IODestroy, 953, 954
- IOInit, 953
- IOPrepare, 953
- IORelease, 953, 954
- IOSetVar, 953, 954
- IOStart, 953, 954
- IOStop, 953, 954
- IOStrError, 953, 954
- MAX_USER_ERR, 953
- user_err_msg, 954
- MHAIOJack, 93
- MHAIOJack.cpp, 954
- dummy_interface_test, 956, 957
- ERR_IHANDLE, 955
- ERR_SUCCESS, 955
- ERR_USER, 955
- IODestroy, 956, 957
- IOInit, 955, 956
- IOPrepare, 956
- IORelease, 956

- IOSetVar, 956
- IOStart, 956
- IOWStop, 956
- IOStrError, 956, 957
- MAX_USER_ERR, 955
- user_err_msg, 957
- MHAIOJack::io_jack_t, 514
 - clientname, 516
 - connections_in, 516
 - connections_out, 516
 - delays_in, 516
 - delays_out, 517
 - fw_fragsize, 516
 - fw_samplerate, 516
 - get_all_input_ports, 516
 - get_all_output_ports, 516
 - get_delays_in, 516
 - get_delays_out, 516
 - get_physical_input_ports, 516
 - get_physical_output_ports, 516
 - io_jack_t, 515
 - patchbay, 517
 - portnames_in, 517
 - portnames_out, 517
 - ports_in_all, 517
 - ports_in_physical, 517
 - ports_out_all, 517
 - ports_out_physical, 517
 - ports_parser, 517
 - prepare, 515
 - read_get_cpu_load, 516
 - read_get_scheduler, 516
 - read_get_xruns, 516
 - reconnect_inports, 516
 - reconnect_outports, 516
 - release, 515
 - servername, 516
 - state_cpuload, 517
 - state_parser, 517
 - state_priority, 517
 - state_scheduler, 517
 - state_xruns, 517
- MHAIOParser.cpp, 957
 - dummy_interface_test, 958, 959
 - ERR_IHANDLE, 958
 - ERR_SUCCESS, 958
 - ERR_USER, 958
 - IODestroy, 958, 959
 - IOInit, 958, 959
 - IOWPrepare, 958, 959
 - IOWRelease, 958, 959
- IOSetVar, 958, 959
- IOStart, 958, 959
- IOWStop, 958, 959
- IOStrError, 958, 959
- MAX_USER_ERR, 958
- user_err_msg, 959
- MHAIOPortAudio, 93
 - parserFriendlyName, 94
- MHAIOPortAudio.cpp, 960
 - dummy_interface_test, 961, 962
 - ERR_IHANDLE, 961
 - ERR_SUCCESS, 961
 - ERR_USER, 961
 - IODestroy, 961, 962
 - IOInit, 961
 - IOWPrepare, 961, 962
 - IOWRelease, 961, 962
 - IOSetVar, 961, 962
 - IOStart, 961, 962
 - IOWStop, 961, 962
 - IOStrError, 961, 962
 - MAX_USER_ERR, 961
 - portaudio_callback, 961, 962
 - user_err_msg, 962
- MHAIOPortAudio::device_info_t, 518
 - defaultHighInputLatency, 519
 - defaultHighOutputLatency, 519
 - defaultLowInputLatency, 519
 - defaultLowOutputLatency, 519
 - defaultSampleRate, 519
 - device_info_t, 519
 - fill_info, 519
 - hostApi, 519
 - maxInputChannels, 519
 - maxOutputChannels, 519
 - name, 519
 - numDevices, 519
 - structVersion, 519
- MHAIOPortAudio::io_portaudio_t, 520
 - ~io_portaudio_t, 521
 - cmd_prepare, 521
 - cmd_release, 522
 - cmd_start, 521
 - cmd_stop, 521
 - device_index, 522
 - device_index_updated, 521
 - device_info, 522
 - device_name, 522
 - device_name_updated, 521
 - fragsize, 522
 - io_portaudio_t, 521

- nchannels_in, 522
- nchannels_out, 522
- patchbay, 522
- portaudio_callback, 522
- portaudio_stream, 522
- proc_event, 522
- proc_handle, 522
- s_in, 522
- s_out, 522
- samplerate, 522
- start_event, 522
- start_handle, 522
- stop_event, 522
- stop_handle, 522
- MHAIoTTCP.cpp, 962
 - copy_error, 965
 - dummy_interface_test, 965, 966
 - ERR_IHANDLE, 964
 - ERR_SUCCESS, 964
 - ERR_USER, 964
 - IODestroy, 964, 965
 - IOInit, 964, 965
 - IOPrepare, 964, 965
 - IORelease, 964, 965
 - IOSetVar, 964, 965
 - IOStart, 964, 965
 - IOStop, 964, 965
 - IOStrError, 964, 965
 - MAX_TCP_PORT_STR, 964
 - MAX_TCP_PORT, 964
 - MAX_USER_ERR, 964
 - MHA_ErrorMsg2, 964
 - MHA_ErrorMsg3, 964
 - MIN_TCP_PORT_STR, 964
 - MIN_TCP_PORT, 964
 - thread_startup_function, 965
 - user_err_msg, 966
- MHAInit_cb
 - PluginLoader::mhapluginloader_t, 809
- MHAInit_t
 - mha.h, 904
- MHAJACK_FW_STARTED
 - mhajack.h, 968
- MHAJACK_STARTING
 - mhajack.h, 968
- MHAJACK_STOPPED
 - mhajack.h, 968
- MHAJack, 94
 - get_port_capture_latency, 95
 - get_port_capture_latency_int, 95
 - get_port_playback_latency, 95
 - get_port_playback_latency_int, 96
 - io, 95
- MHAJack::client_avg_t, 523
 - b_ready, 525
 - b_stopped, 525
 - client_avg_t, 524
 - frag_out, 525
 - IOStoppedEvent, 525
 - io, 524
 - n, 525
 - name, 525
 - nrep, 525
 - pos, 525
 - proc, 525
 - sn_in, 525
 - sn_out, 525
- MHAJack::client_noncont_t, 526
 - b_stopped, 527
 - client_noncont_t, 527
 - frag_out, 528
 - IOStoppedEvent, 527
 - io, 527
 - name, 528
 - pos, 527
 - proc, 527
 - sn_in, 527
 - sn_out, 528
- MHAJack::client_t, 528
 - b_prepared, 534
 - client_t, 530
 - connect_input, 531
 - connect_output, 531
 - fail_on_async_jackerror, 534
 - flags, 534
 - fragsize, 533
 - get_cpu_load, 532
 - get_fragsize, 531
 - get_my_input_ports, 532
 - get_my_output_ports, 532
 - get_ports, 532
 - get_srate, 532
 - get_xruns, 532
 - get_xruns_reset, 532
 - inch, 534
 - input_portnames, 534
 - internal_start, 533
 - internal_stop, 533
 - jack_proc_cb, 533
 - jack_xrun_cb, 533
 - jc, 534
 - jstate_prev, 534

- nchannels_in, 533
- nchannels_out, 533
- num_xruns, 533
- outch, 534
- output_portnames, 534
- prepare, 530, 531
- prepare_impl, 532
- proc_event, 533
- proc_handle, 533
- release, 531
- s_in, 534
- s_out, 534
- samplerate, 533
- set_input_portnames, 532
- set_output_portnames, 532
- set_use_jack_transport, 532
- start, 531
- start_event, 534
- start_handle, 534
- stop, 531
- stop_event, 534
- stop_handle, 534
- stopped, 533
- str_error, 532
- use_jack_transport, 534
- MHAJack::port_t, 534
 - ~port_t, 536
 - connect_to, 537
 - dir_t, 535
 - dir_type, 537
 - get_short_name, 537
 - input, 535
 - iob, 537
 - jc, 537
 - mute, 536
 - output, 535
 - port, 537
 - port_t, 535, 536
 - read, 536
 - write, 536
- MHAKernel, 96
 - algo_comm_safe_cast, 96
- MHAKernel::algo_comm_class_t, 537
 - ~algo_comm_class_t, 539
 - ac, 540
 - algo_comm_class_t, 539
 - algo_comm_id_string, 540
 - algo_comm_id_string_len, 540
 - get_c_handle, 539
 - get_entries, 540
 - get_error, 540
 - get_var, 539
 - get_var_float, 539
 - get_var_int, 539
 - insert_var, 539
 - insert_var_float, 539
 - insert_var_int, 539
 - is_var, 539
 - local_get_entries, 540
 - local_get_var, 540
 - local_insert_var, 540
 - local_is_var, 540
 - local_remove_ref, 540
 - local_remove_var, 540
 - remove_ref, 539
 - remove_var, 539
 - size, 540
 - vars, 540
- MHAKernel::comm_var_map_t, 541
 - has_key, 541
- MHAMultiSrc, 96
- MHAMultiSrc::base_t, 541
 - ac, 542
 - base_t, 542
 - select_source, 542
- MHAMultiSrc::channel_t, 543
 - channel, 543
 - name, 543
- MHAMultiSrc::channels_t, 543
 - channels_t, 543
- MHAMultiSrc::spectrum_t, 544
 - spectrum_t, 544
 - update, 545
- MHAMultiSrc::waveform_t, 545
 - update, 546
 - waveform_t, 546
- MHAOvfFilter, 97
 - scale_fun_t, 98
- MHAOvfFilter::FreqScaleFun, 98
 - hz2bark, 99
 - hz2bark_analytic, 100
 - hz2erb, 100
 - hz2erb_glasberg1990, 100
 - hz2hz, 99
 - hz2khz, 99
 - hz2log, 100
 - hz2octave, 99
 - hz2third_octave, 99
 - inv_scale, 100
- MHAOvfFilter::ShapeFun, 100
 - expflt, 103
 - gauss, 103

- hann, [101](#)
- linear, [101](#)
- rect, [101](#)
- MHAOvIFilter::band_descriptor_t, [546](#)
 - cf, [547](#)
 - cf_h, [547](#)
 - cf_l, [547](#)
 - ef_h, [547](#)
 - ef_l, [547](#)
 - high_side_flat, [547](#)
 - low_side_flat, [547](#)
- MHAOvIFilter::barkscale, [98](#)
 - vbark, [98](#)
 - vfreq, [98](#)
- MHAOvIFilter::barkscale::bark2hz_t, [547](#)
 - ~bark2hz_t, [548](#)
 - bark2hz_t, [548](#)
- MHAOvIFilter::barkscale::hz2bark_t, [548](#)
 - ~hz2bark_t, [549](#)
 - hz2bark_t, [549](#)
- MHAOvIFilter::fftfb_ac_info_t, [549](#)
 - bwv, [550](#)
 - cLTASS, [550](#)
 - cfv, [549](#)
 - efv, [549](#)
 - fftfb_ac_info_t, [549](#)
 - insert, [549](#)
- MHAOvIFilter::fftfb_t, [550](#)
 - ~fftfb_t, [551](#)
 - apply_gains, [551](#)
 - bin1, [552](#)
 - bin2, [552](#)
 - fftfb_t, [551](#)
 - fftl, [552](#)
 - get_fbpower, [551](#)
 - get_fbpower_db, [551](#)
 - get_fftl, [552](#)
 - get_ltass_gain_db, [552](#)
 - samplingrate, [552](#)
 - shape, [552](#)
 - vbin1, [552](#)
 - vbin2, [552](#)
 - w, [552](#)
- MHAOvIFilter::fftfb_vars_t, [553](#)
 - cLTASS, [555](#)
 - cf, [555](#)
 - ef, [555](#)
 - f, [554](#)
 - fail_on_nonmonotonic, [555](#)
 - fail_on_unique_bins, [555](#)
 - fftfb_vars_t, [554](#)
 - fscale, [554](#)
 - ftype, [554](#)
 - normalize, [555](#)
 - ovltype, [554](#)
 - plateau, [554](#)
 - shapes, [555](#)
- MHAOvIFilter::fscale_bw_t, [555](#)
 - bw, [556](#)
 - bw_hz, [556](#)
 - fscale_bw_t, [556](#)
 - get_bw_hz, [556](#)
 - update_hz, [556](#)
 - updater, [556](#)
- MHAOvIFilter::fscale_t, [557](#)
 - f, [558](#)
 - f_hz, [558](#)
 - fscale_t, [558](#)
 - get_f_hz, [558](#)
 - unit, [558](#)
 - update_hz, [558](#)
 - updater, [558](#)
- MHAOvIFilter::fspacing_t, [558](#)
 - bands, [560](#)
 - cf2bands, [560](#)
 - ef2bands, [560](#)
 - equidist2bands, [560](#)
 - fail_on_nonmonotonic_cf, [560](#)
 - fail_on_unique_fftbins, [560](#)
 - fs, [560](#)
 - fspacing_t, [560](#)
 - get_cf_fftb, [560](#)
 - get_cf_hz, [560](#)
 - get_ef_hz, [560](#)
 - nbands, [560](#)
 - nfft, [560](#)
 - symmetry_scale, [560](#)
- MHAOvIFilter::overlap_save_filterbank_↔
 - analytic_t, [561](#)
 - filter_analytic, [562](#)
 - imagfb, [562](#)
 - overlap_save_filterbank_analytic_t, [562](#)
- MHAOvIFilter::overlap_save_filterbank_t, [562](#)
 - channelconfig_out, [564](#)
 - get_channelconfig, [564](#)
 - overlap_save_filterbank_t, [564](#)
- MHAOvIFilter::overlap_save_filterbank_t↔
 - ::vars_t, [564](#)
 - fftl, [565](#)
 - irswnd, [565](#)
 - phasemodel, [565](#)
 - vars_t, [565](#)

- MHAOvIFilter::scale_var_t, 566
 - add_fun, 567
 - funs, 567
 - get_fun, 567
 - get_name, 567
 - hz2unit, 567
 - names, 567
 - scale_var_t, 567
 - unit2hz, 567
- MHAPLATFORM
 - mha_parser.cpp, 925
- MHAPLUGIN_CALLBACKS_PREFIX
 - The openMHA Plugins (programming interface), 8
- MHAPLUGIN_CALLBACKS
 - The openMHA Plugins (programming interface), 8
- MHAPLUGIN_DOCUMENTATION_PREFIX
 - mha_plugin.hh, 931
- MHAPLUGIN_DOCUMENTATION
 - The openMHA Plugins (programming interface), 9
- MHAPLUGIN_INIT_CALLBACKS_PREFIX
 - mha_plugin.hh, 931
- MHAPLUGIN_INIT_CALLBACKS
 - mha_plugin.hh, 931
- MHAPLUGIN_OVERLOAD_OUTDOMAIN
 - altplugs.cpp, 883
 - mha_generic_chain.h, 919
 - split.cpp, 981
 - wave2spec.cpp, 984
- MHAPLUGIN_PROC_CALLBACK_PREFIX
 - mha_plugin.hh, 931
- MHAPLUGIN_PROC_CALLBACK
 - mha_plugin.hh, 931
- MHAParser, 103
 - all_dump, 107
 - all_ids, 107
 - c_parse_cmd_t, 106
 - c_parse_err_t, 106
 - cfg_dump, 106
 - cfg_dump_short, 107
 - commentate, 106
 - entry_map_t, 106
 - envreplace, 107
 - get_precision, 106
 - mon_dump, 107
 - opact_map_t, 106
 - opact_t, 106
 - query_map_t, 106
 - query_t, 106
 - strreplace, 107
 - trim, 106
- MHAParser::StrCnv, 107
 - bracket_balance, 109
 - num_brackets, 109
 - str2val, 109, 110
 - str2val < mha_real_t >, 110
 - val2str, 110, 111
- MHAParser::base_t, 568
 - ~base_t, 571
 - activate_query, 575
 - add_parent_on_insert, 575
 - add_replace_pair, 575
 - base_t, 571
 - data_is_initialized, 576
 - fullname, 575
 - help, 576
 - id_str, 576
 - nested_lock, 576
 - notify, 575
 - op_query, 572
 - op_setval, 572
 - op_subparse, 572
 - operators, 576
 - oplist, 575
 - parent, 576
 - parse, 571, 572
 - prereadaccess, 576
 - queries, 576
 - query_addsubst, 574
 - query_cmds, 574
 - query_dump, 572
 - query_entries, 572
 - query_help, 574
 - query_id, 574
 - query_listids, 574
 - query_perm, 573
 - query_range, 573
 - query_readfile, 573
 - query_savefile, 573
 - query_savefile_compact, 574
 - query_savemons, 574
 - query_subst, 574
 - query_type, 573
 - query_val, 573
 - query_version, 574
 - readaccess, 576
 - repl_list, 576
 - repl_list_t, 571
 - rm_parent_on_remove, 575
 - set_help, 575

- set_node_id, 574
- thefullname, 576
- valuechanged, 575
- writeaccess, 575
- MHAParser::base_t::replace_t, 577
 - a, 577
 - b, 577
 - get_a, 577
 - get_b, 577
 - replace, 577
 - replace_t, 577
- MHAParser::bool_mon_t, 578
 - bool_mon_t, 578
 - data, 579
 - query_type, 579
 - query_val, 579
- MHAParser::bool_t, 579
 - bool_t, 581
 - data, 581
 - op_setval, 581
 - query_type, 581
 - query_val, 581
- MHAParser::c_ifc_parser_t, 582
 - ~c_ifc_parser_t, 583
 - c_ifc_parser_t, 583
 - c_parse_cmd, 584
 - c_parse_err, 584
 - libdata, 584
 - liberr, 584
 - modulename, 584
 - op_query, 583
 - op_setval, 583
 - op_subparse, 583
 - ret_size, 584
 - retv, 584
 - set_parse_cb, 583
 - test_error, 583
- MHAParser::commit_t
 - commit_t, 586
 - extern_connector, 586
- MHAParser::commit_t < receiver_t >, 584
- MHAParser::complex_mon_t, 586
 - complex_mon_t, 587
 - data, 588
 - query_type, 587
 - query_val, 587
- MHAParser::complex_t, 588
 - complex_t, 589
 - data, 590
 - op_setval, 589
 - query_type, 589
 - query_val, 589
- MHAParser::entry_t, 590
 - entry, 590
 - entry_t, 590
 - name, 590
- MHAParser::expression_t, 591
 - expression_t, 591
 - lval, 591
 - op, 591
 - rval, 591
- MHAParser::float_mon_t, 592
 - data, 593
 - float_mon_t, 592
 - query_type, 593
 - query_val, 593
- MHAParser::float_t, 593
 - data, 596
 - float_t, 595
 - op_setval, 595
 - query_type, 595
 - query_val, 595
- MHAParser::int_mon_t, 596
 - data, 598
 - int_mon_t, 597
 - query_type, 597
 - query_val, 597
- MHAParser::int_t, 598
 - data, 600
 - int_t, 599
 - op_setval, 600
 - query_type, 600
 - query_val, 600
- MHAParser::keyword_list_t, 600
 - add_entry, 602
 - empty_string, 603
 - entries, 603
 - get_entries, 602
 - get_index, 602
 - get_value, 602
 - index, 603
 - keyword_list_t, 601
 - set_entries, 602
 - set_index, 602
 - set_value, 602
 - size_t, 601
 - validate, 602
- MHAParser::kw_t, 603
 - data, 606
 - isval, 605
 - kw_t, 604, 605
 - op_setval, 605

- query_range, 605
- query_type, 605
- query_val, 605
- set_range, 605
- validate, 605
- MHAParser::mcomplex_mon_t, 606
 - data, 607
 - mcomplex_mon_t, 607
 - query_type, 607
 - query_val, 607
- MHAParser::mcomplex_t, 608
 - data, 609
 - mcomplex_t, 609
 - op_setval, 609
 - query_type, 609
 - query_val, 609
- MHAParser::mfloat_mon_t, 610
 - data, 611
 - mfloat_mon_t, 610
 - query_type, 611
 - query_val, 611
- MHAParser::mfloat_t, 611
 - data, 614
 - mfloat_t, 613
 - op_setval, 613
 - query_type, 613
 - query_val, 613
- MHAParser::mhaconfig_mon_t, 614
 - channels, 615
 - domain, 615
 - ffflen, 615
 - fragsize, 615
 - mhaconfig_mon_t, 615
 - srate, 616
 - update, 615
 - wndlen, 615
- MHAParser::mhapluginloader_t, 616
 - ~mhapluginloader_t, 617
 - ac_, 618
 - bookkeeping, 618
 - cf_in_, 618
 - cf_out_, 618
 - connector, 618
 - get_cfin, 618
 - get_cfout, 618
 - get_last_name, 618
 - last_name, 618
 - load_plug, 618
 - mhapluginloader_t, 617
 - parent_, 618
 - plug, 618
 - plugname, 618
 - plugname_name_, 618
 - prefix_, 618
 - prepare, 617
 - process, 617, 618
 - release, 617
- MHAParser::monitor_t, 619
 - monitor_t, 619
 - op_query, 619
 - query_dump, 619
 - query_perm, 620
- MHAParser::parser_t, 620
 - ~parser_t, 622
 - entries, 624
 - force_remove_item, 622
 - id_string, 624
 - insert_item, 622
 - last_errormsg, 625
 - op_query, 623
 - op_setval, 623
 - op_subparse, 623
 - parser_t, 622
 - query_dump, 623
 - query_entries, 623
 - query_listids, 624
 - query_readfile, 624
 - query_savefile, 624
 - query_savefile_compact, 624
 - query_savemons, 624
 - query_type, 623
 - query_val, 624
 - remove_item, 622, 623
 - set_id_string, 624
 - srcfile, 624
 - srcline, 625
- MHAParser::range_var_t, 625
 - check_low, 627
 - check_range, 628
 - check_up, 628
 - low_incl, 627
 - low_limit, 627
 - query_range, 626
 - range_var_t, 626
 - set_range, 626
 - up_incl, 627
 - up_limit, 627
 - validate, 627
- MHAParser::string_mon_t, 628
 - data, 630
 - query_type, 629
 - query_val, 629

- string_mon_t, 629
- MHAParser::string_t, 630
 - data, 632
 - op_setval, 631
 - query_type, 631
 - query_val, 631
 - string_t, 631
- MHAParser::variable_t, 632
 - locked, 633
 - op_setval, 633
 - query_perm, 633
 - setlock, 633
 - variable_t, 633
- MHAParser::vcomplex_mon_t, 634
 - data, 635
 - query_type, 635
 - query_val, 635
 - vcomplex_mon_t, 635
- MHAParser::vcomplex_t, 636
 - data, 637
 - op_setval, 637
 - query_type, 637
 - query_val, 637
 - vcomplex_t, 637
- MHAParser::vfloat_mon_t, 638
 - data, 639
 - query_type, 639
 - query_val, 639
 - vfloat_mon_t, 639
- MHAParser::vfloat_t, 640
 - data, 642
 - op_setval, 641
 - query_type, 641
 - query_val, 641
 - vfloat_t, 641
- MHAParser::vint_mon_t, 642
 - data, 643
 - query_type, 643
 - query_val, 643
 - vint_mon_t, 643
- MHAParser::vint_t, 644
 - data, 646
 - op_setval, 645
 - query_type, 645
 - query_val, 645
 - vint_t, 645
- MHAParser::vstring_mon_t, 646
 - data, 647
 - query_type, 647
 - query_val, 647
 - vstring_mon_t, 647
- MHAParser::vstring_t, 648
 - data, 649
 - op_setval, 649
 - query_type, 649
 - query_val, 649
 - vstring_t, 649
- MHAParser::window_t, 650
 - get_type, 652
 - get_window, 651, 652
 - user, 652
 - window_t, 651
 - wnd_bartlett, 651
 - wnd_blackman, 651
 - wnd_hamming, 651
 - wnd_hann, 651
 - wnd_rect, 651
 - wnd_user, 651
 - wtype, 652
 - wtype_t, 651
- MHAPLugin, 112
- MHAPLugin::cfg_chain_t
 - ~cfg_chain_t, 654
 - cfg_chain_t, 654
 - data, 654
 - next, 654
 - not_in_use, 654
- MHAPLugin::cfg_chain_t< runtime_cfg_t >, 653
- MHAPLugin::config_t
 - ~config_t, 656
 - cfg, 659
 - cfg_chain, 659
 - cfg_chain_current, 659
 - cleanup_unused_cfg, 658
 - config_t, 656
 - last_config, 658
 - poll_config, 656
 - push_config, 658
 - remove_all_cfg, 658
- MHAPLugin::config_t< runtime_cfg_t >, 654
- MHAPLugin::plugin_t
 - ~plugin_t, 661
 - ac, 662
 - input_cfg, 662
 - input_cfg_, 663
 - is_prepared, 662
 - is_prepared_, 663
 - mhaconfig_in, 663
 - mhaconfig_out, 663
 - output_cfg, 662
 - output_cfg_, 663

- plugin_t, 661
- prepare, 661
- prepare_, 662
- release, 661
- release_, 662
- tftype, 662
- MHAPLugin::plugin_t< runtime_cfg_t >, 659
- MHAPLugin_Resampling, 112
- MHAPLugin_Resampling::resampling_if_←
t, 664
- algo, 665
- chain, 665
- fragsize, 665
- irslen_inner2outer, 665
- irslen_outer2inner, 665
- nyquist_ratio, 665
- plugloader, 665
- prepare, 665
- process, 665
- release, 665
- resampling_if_t, 665
- srate, 665
- MHAPLugin_Resampling::resampling_t, 666
- inner2outer_resampling, 667
- inner_fragsize, 667
- inner_signal, 667
- inner_srate, 667
- nchannels_in, 667
- nchannels_out, 667
- outer2inner_resampling, 667
- outer_fragsize, 666
- outer_srate, 667
- output_signal, 667
- plugloader, 667
- process, 666
- resampling_t, 666
- MHAPLugin_Split, 112
- INVALID_THREAD_PRIORITY, 113
- MHAPLugin_Split::domain_handler_t, 667
- ~domain_handler_t, 669
- deallocate_domains, 670
- domain_handler_t, 669
- get_signal, 671
- operator=, 669
- process, 671
- processor, 672
- put_signal, 670
- set_input_domain, 669
- set_output_domain, 669
- spec_in, 672
- spec_out, 672
- wave_in, 672
- wave_out, 672
- MHAPLugin_Split::dummy_threads_t, 672
- catch_thread, 673
- dummy_threads_t, 673
- kick_thread, 673
- MHAPLugin_Split::posix_threads_t, 674
- ~posix_threads_t, 676
- attr, 677
- catch_condition, 677
- catch_thread, 676
- current_thread_priority, 676
- current_thread_scheduler, 676
- kick_condition, 676
- kick_thread, 676
- kicked, 677
- main, 676
- mutex, 676
- posix_threads_t, 675
- priority, 677
- processing_done, 677
- scheduler, 677
- termination_request, 677
- thread, 677
- thread_start, 676
- MHAPLugin_Split::split_t, 678
- ~split_t, 680
- algos, 681
- chains, 682
- channels, 681
- clear_chains, 680
- collect_result, 680
- copy_output_spec, 680
- copy_output_wave, 680
- delay, 682
- framework_thread_priority, 682
- framework_thread_scheduler, 681
- patchbay, 681
- prepare_, 680
- process, 680
- release_, 680
- signal_out, 681
- spec_out, 682
- split_t, 680
- thread_platform, 681
- trigger_processing, 680
- update, 680
- wave_out, 682
- worker_thread_priority, 681
- worker_thread_scheduler, 681
- MHAPLugin_Split::splitted_part_t, 682

- ~splitted_part_t, 684
- collect_result, 686
- domain, 686
- operator=, 685
- parse, 685
- plug, 686
- prepare, 685
- release, 685
- splitted_part_t, 684
- thread, 687
- trigger_processing, 686
- MHAPPlugin_Split::thread_platform_t, 687
 - ~thread_platform_t, 688
 - catch_thread, 689
 - kick_thread, 689
 - operator=, 689
 - processor, 689
 - thread_platform_t, 688
- MHAPPlugin_Split::uni_processor_t, 690
 - ~uni_processor_t, 690
 - process, 691
- MHAPPluginCategory_t
 - mha.h, 905
- MHAPPluginDocumentation_t
 - mha.h, 905
- MHAPPrepare_cb
 - PluginLoader::mhapluginloader_t, 809
- MHAPPrepare_t
 - mha.h, 904
- MHAProc_spec2spec_cb
 - PluginLoader::mhapluginloader_t, 809
- MHAProc_spec2spec_t
 - mha.h, 905
- MHAProc_spec2wave_cb
 - PluginLoader::mhapluginloader_t, 810
- MHAProc_spec2wave_t
 - mha.h, 905
- MHAProc_wave2spec_cb
 - PluginLoader::mhapluginloader_t, 809
- MHAProc_wave2spec_t
 - mha.h, 905
- MHAProc_wave2wave_cb
 - PluginLoader::mhapluginloader_t, 809
- MHAProc_wave2wave_t
 - mha.h, 905
- MHARelease_cb
 - PluginLoader::mhapluginloader_t, 809
- MHARelease_t
 - mha.h, 904
- MHASet_cb
 - PluginLoader::mhapluginloader_t, 810
- MHASet_t
 - mha.h, 904
- MHASignal, 113
 - copy_permuted, 120
 - kth_smallest, 116
 - limit, 116
 - mean, 118
 - median, 117
 - quantile, 118
 - saveas_mat4, 119
 - scale, 116
 - signal_counter, 120
- MHASignal::async_rmslevel_t, 694
 - async_rmslevel_t, 695
 - filled, 696
 - peaklevel, 696
 - pos, 696
 - process, 696
 - rmslevel, 695
- MHASignal::delay_spec_t, 696
 - ~delay_spec_t, 697
 - buffer, 697
 - delay, 697
 - delay_spec_t, 697
 - pos, 697
 - process, 697
- MHASignal::delay_t, 697
 - ~delay_t, 698
 - buffer, 699
 - channels, 699
 - delay_t, 698
 - delays, 699
 - inspect, 699
 - pos, 699
 - process, 698
- MHASignal::delay_wave_t, 699
 - ~delay_wave_t, 700
 - buffer, 700
 - delay, 700
 - delay_wave_t, 700
 - pos, 700
 - process, 700
- MHASignal::doublebuffer_t, 700
 - ~doublebuffer_t, 702
 - ch, 703
 - doublebuffer_t, 701
 - inner_in, 703
 - inner_out, 703
 - inner_process, 702
 - k_inner, 703
 - k_outer, 703

- min, 702
- outer_out, 703
- outer_process, 702
- this_outer_out, 703
- MHASignal::fft_t, 703
 - ~fft_t, 704
 - backward, 705
 - backward_scale, 705
 - buf_in, 706
 - buf_out, 706
 - fft_t, 704
 - fftw_plan_fft, 706
 - fftw_plan_ifft, 706
 - fftw_plan_spec2wave, 706
 - fftw_plan_wave2spec, 706
 - forward, 704
 - forward_scale, 705
 - n_im, 706
 - n_re, 706
 - nfft, 706
 - scale, 706
 - sort_fftw2spec, 705
 - sort_spec2fftw, 705
 - spec2wave, 704
 - spec2wave_scale, 705
 - wave2spec, 704
 - wave2spec_scale, 705
- MHASignal::hilbert_fftw_t, 706
 - buf_c_in, 707
 - buf_c_out, 707
 - buf_r_in, 707
 - buf_r_out, 707
 - hilbert, 707
 - hilbert_fftw_t, 707
 - n, 707
 - p1, 707
 - p2, 707
 - sc, 707
- MHASignal::hilbert_t, 707
 - ~hilbert_t, 708
 - h, 708
 - hilbert_t, 708
 - operator(), 708
- MHASignal::loop_wavefragment_t, 709
 - add, 710
 - b_loop, 712
 - get_mapping, 711
 - input, 710
 - intern_level, 712
 - is_playback_active, 712
 - level_mode_t, 710
 - locate_end, 712
 - loop_wavefragment_t, 710
 - mute, 710
 - peak, 710
 - playback, 711, 712
 - playback_channels, 712
 - playback_mode_t, 710
 - pos, 712
 - relative, 710
 - replace, 710
 - rewind, 712
 - rms, 710
 - rms_limit40, 710
 - set_level_db, 712
 - set_level_lin, 712
- MHASignal::matrix_t, 713
 - ~matrix_t, 716
 - cdata, 721
 - complex_ofs, 721
 - dimension, 717
 - get_cdata, 721
 - get_comm_var, 716
 - get_index, 720
 - get_nelements, 717
 - get_nreals, 720
 - get_rdata, 721
 - imag, 718–720
 - is_same_size, 717
 - iscomplex, 717
 - matrix_t, 715, 716
 - nelements, 721
 - numbytes, 720
 - operator(), 718–720
 - operator=, 716
 - rdata, 721
 - real, 717–719
 - size, 717
 - write, 720
- MHASignal::minphase_t, 721
 - minphase_t, 722
 - operator(), 722
 - phase, 723
- MHASignal::quantizer_t, 723
 - downscale, 724
 - limit, 724
 - operator(), 724
 - quantizer_t, 723
 - up_limit, 724
 - upscale, 724
- MHASignal::ringbuffer_t, 724
 - contained_frames, 726

- discard, 727
- next_read_frame_index, 727
- next_write_frame_index, 728
- ringbuffer_t, 726
- value, 726
- write, 727
- MHASignal::schroeder_t, 728
 - down, 730
 - groupdelay_t, 729
 - identity, 731
 - log_down, 731
 - log_up, 731
 - schroeder_t, 730
 - sign_t, 730
 - up, 730
- MHASignal::spectrum_t, 731
 - ~spectrum_t, 733
 - copy, 734
 - copy_channel, 735
 - export_to, 735
 - operator(), 734
 - operator[], 734
 - scale, 735
 - scale_channel, 736
 - spectrum_t, 733
 - value, 734
- MHASignal::stat_t, 736
 - mean, 737
 - mean_std, 737
 - n, 737
 - push, 737
 - stat_t, 737
 - sum, 737
 - sum2, 737
- MHASignal::subsample_delay_t, 737
 - last_complex_bin, 739
 - phase_gains, 739
 - process, 739
 - subsample_delay_t, 738
- MHASignal::uint_vector_t, 740
 - ~uint_vector_t, 741
 - data, 742
 - get_length, 742
 - getdata, 742
 - length, 742
 - numbytes, 742
 - operator=, 741
 - operator==, 741
 - operator[], 742
 - uint_vector_t, 741
 - write, 742
- MHASignal::waveform_t, 743
 - ~waveform_t, 745
 - assign, 749
 - assign_channel, 749
 - assign_frame, 749
 - copy, 750
 - copy_channel, 750
 - copy_from_at, 750
 - export_to, 751
 - get_size, 753
 - limit, 751
 - operator(), 746, 747
 - operator=, 746
 - operator[], 746
 - power, 751
 - powspec, 752
 - scale, 752
 - scale_channel, 752
 - scale_frame, 753
 - sum, 747, 748
 - sum_channel, 748
 - sumsq, 748
 - value, 746, 747
 - waveform_t, 745
- MHASndFile, 120
- MHASndFile::sf_t, 753
 - ~sf_t, 754
 - sf, 754
 - sf_t, 754
- MHASndFile::sf_wave_t, 754
 - sf_wave_t, 755
- MHAStrError_cb
 - PluginLoader::mhapluginloader_t, 810
- MHAStrError_t
 - mha.h, 905
- MHATableLookup, 120
- MHATableLookup::linear_table_t, 755
 - ~linear_table_t, 757
 - add_entry, 758
 - clear, 758
 - interp, 757
 - len, 759
 - linear_table_t, 757
 - lookup, 757
 - prepare, 758
 - scalefac, 759
 - set_xmax, 758
 - set_xmin, 757
 - vec_y, 759
 - vy, 759
 - xmax, 759

- xmin, [759](#)
- MHATableLookup::table_t, [759](#)
 - ~table_t, [760](#)
 - clear, [760](#)
 - interp, [760](#)
 - lookup, [760](#)
 - table_t, [760](#)
- MHATableLookup::xy_table_t, [761](#)
 - add_entry, [763](#)
 - clear, [763](#)
 - get_xlimits, [764](#)
 - interp, [762](#)
 - lookup, [762](#)
 - mXY, [764](#)
 - set_xfun, [763](#)
 - set_xyfun, [764](#)
 - set_yfun, [764](#)
 - xfun, [764](#)
 - xy_table_t, [762](#)
 - xyfun, [764](#)
 - yfun, [764](#)
- MHAWindow, [121](#)
 - bartlett, [122](#)
 - blackman, [122](#)
 - hamming, [122](#)
 - hanning, [122](#)
 - rect, [122](#)
- MHAWindow::bartlett_t, [765](#)
 - bartlett_t, [765](#)
- MHAWindow::base_t, [766](#)
 - base_t, [766](#), [767](#)
 - operator(), [767](#)
 - ramp_begin, [767](#)
 - ramp_end, [767](#)
- MHAWindow::blackman_t, [768](#)
 - blackman_t, [768](#)
- MHAWindow::fun_t, [769](#)
 - fun_t, [769](#)
- MHAWindow::hamming_t, [770](#)
 - hamming_t, [771](#)
- MHAWindow::hanning_t, [771](#)
 - hanning_t, [772](#)
- MHAWindow::rect_t, [772](#)
 - rect_t, [773](#)
- MHAWindow::user_t, [773](#)
 - user_t, [774](#)
- MIN_TCP_PORT_STR
 - MHAIOTCP.cpp, [964](#)
- MIN_TCP_PORT
 - MHAIOTCP.cpp, [964](#)
- MIN
 - mha_defs.h, [908](#)
- mXY
 - MHATableLookup::xy_table_t, [764](#)
- main
 - analysemhaplugin.cpp, [883](#)
 - browsemhaplugins.cpp, [885](#)
 - generatemhapolugindoc.cpp, [895](#)
 - MHAPLugin_Split::posix_threads_t, [676](#)
 - mha.cpp, [900](#)
 - testalsadevice.c, [982](#)
- make_friendly_number
 - MHAFilter, [91](#)
 - mhajack.cpp, [966](#)
- make_friendly_number_by_limiting
 - nlms_wave.cpp, [972](#)
 - prediction_error.cpp, [974](#)
- mapping
 - coherence::vars_t, [205](#)
- matmix_t
 - matrixmixer::matmix_t, [354](#)
- matrix_t
 - MHASignal::matrix_t, [715](#), [716](#)
- matrixmixer, [84](#)
- matrixmixer.cpp, [899](#)
- matrixmixer::cfg_t, [351](#)
 - cfg_t, [352](#)
 - m, [352](#)
 - process, [352](#)
 - sout, [352](#)
 - wout, [352](#)
- matrixmixer::matmix_t, [353](#)
 - ci, [354](#)
 - co, [354](#)
 - matmix_t, [354](#)
 - mixer, [354](#)
 - patchbay, [354](#)
 - prepare, [354](#)
 - process, [354](#)
 - update_m, [354](#)
- max
 - spec2wave.cpp, [977](#)
 - Vector and matrix processing toolbox, [58](#)
- max_clipped
 - softclipper_variables_t, [845](#)
- max_fill_count
 - mha_fifo_t, [398](#)
- max_frames
 - acsave::cfg_t, [147](#)
- max_lag
 - doasvm_feature_extraction, [248](#)
- max_level_difference

- dc::dc_t, 220
- dc::dc_vars_t, 222
- max_p_ind_name
 - doasvm_classification, 244
- max_pool_ind_name
 - acPooling_wave, 140
- max_q
 - timoConfig, 861
- max_q_AC
 - timo_AC, 856
- max_val
 - timoConfig, 861
- max_val_AC
 - timo_AC, 856
- maxInputChannels
 - MHAIOPortAudio::device_info_t, 519
- maxOutputChannels
 - MHAIOPortAudio::device_info_t, 519
- maxabs
 - Vector and matrix processing toolbox, 56, 57
- maxframe
 - acsave::save_var_t, 149
- maxgain
 - dc_simple::dc_t, 230
 - dc_simple::dc_vars_t, 232
 - DynComp::dc_afterburn_rt_t, 257
 - DynComp::dc_afterburn_vars_t, 261
- maximum_reader_xruns_in_succession_↔
 - before_stop
 - mha_drifter_fifo_t, 386
- maximum_writer_xruns_in_succession_↔
 - before_stop
 - mha_drifter_fifo_t, 386
- mcomplex_mon_t
 - MHAParser::mcomplex_mon_t, 607
- mcomplex_t
 - MHAParser::mcomplex_t, 609
- mean
 - MHA_AC::stat_t, 368
 - MHASignal, 118
 - MHASignal::stat_t, 737
- mean_std
 - MHASignal::stat_t, 737
- median
 - MHASignal, 117
- mfloat_mon_t
 - MHAParser::mfloat_mon_t, 610
- mfloat_t
 - MHAParser::mfloat_t, 613
- mha
 - speechnoise_t, 850
- mha.cpp, 900
 - main, 900
 - mhamain, 900
- mha.h, 900
 - algo_comm_t, 904
 - MHA_AC_CHAR, 904
 - MHA_AC_DOUBLE, 904
 - MHA_AC_FLOAT, 904
 - MHA_AC_INT, 904
 - MHA_AC_MHACOMPLEX, 904
 - MHA_AC_MHAREAL, 904
 - MHA_AC_UNKNOWN, 904
 - MHA_AC_USER, 904
 - MHA_AC_VEC_FLOAT, 904
 - MHA_CALLBACK_TEST_PREFIX, 902
 - MHA_CALLBACK_TEST, 902
 - MHA_DOMAIN_MAX, 904
 - MHA_DOMAIN_UNKNOWN, 904
 - MHA_RELEASE_VERSION_STRING, 903
 - MHA_SPECTRUM, 904
 - MHA_STRUCT_SIZEMATCH, 903
 - MHA_STRF, 902
 - MHA_VERSION_BUILD, 903
 - MHA_VERSION_MAJOR, 902
 - MHA_VERSION_MINOR, 902
 - MHA_VERSION_RELEASE, 903
 - MHA_VERSION_STRING, 903
 - MHA_VERSION, 903
 - MHA_WAVEFORM, 903
 - MHA_XSTRF, 902
 - MHADestroy_t, 904
 - MHAGetVersion_t, 904
 - MHAINit_t, 904
 - MHAPuginCategory_t, 905
 - MHAPuginDocumentation_t, 905
 - MHAPrepare_t, 904
 - MHAProc_spec2spec_t, 905
 - MHAProc_spec2wave_t, 905
 - MHAProc_wave2spec_t, 905
 - MHAProc_wave2wave_t, 905
 - MHARelease_t, 904
 - MHASET_t, 904
 - MHAStrError_t, 905
 - mha_domain_t, 904
- mha_algo_comm.cpp, 905
 - AC_DIM_MISMATCH, 905
 - AC_INVALID_HANDLE, 905
 - AC_INVALID_NAME, 905
 - AC_INVALID_OUTPTR, 905

- AC_STRING_TRUNCATED, 905
- AC_SUCCESS, 905
- AC_TYPE_MISMATCH, 905
- algo_comm_default, 905
- mha_algo_comm.h, 905
- mha_algo_comm.hh, 907
 - ALGO_COMM_ID_STR, 907
 - algo_comm_default, 907
- mha_alloc
 - mha_ruby.cpp, 933
- mha_audio_descriptor_t, 370
 - cf, 371
 - chdir, 371
 - dt, 371
 - is_complex, 371
 - n_channels, 371
 - n_freqs, 371
 - n_samples, 371
- mha_audio_t, 371
 - cdata, 372
 - descriptor, 372
 - rdata, 372
- mha_channel_info_t, 372
 - dir, 373
 - id, 373
 - idstr, 373
 - peaklevel, 373
 - side, 373
- mha_complex
 - Complex arithmetics in the openMHA, 62
- mha_complex_t, 374
 - im, 374
 - re, 374
- mha_dblbuf_t
 - ~mha_dblbuf_t, 377
 - delay, 379
 - fifo_size, 379
 - get_delay, 377
 - get_fifo_size, 377
 - get_inner_error, 377
 - get_inner_size, 377
 - get_input_channels, 377
 - get_input_fifo_fill_count, 377
 - get_input_fifo_space, 377
 - get_outer_size, 377
 - get_output_channels, 377
 - get_output_fifo_fill_count, 377
 - get_output_fifo_space, 377
 - inner_error, 380
 - inner_size, 379
 - input, 378
 - input_channels, 379
 - input_fifo, 379
 - mha_dblbuf_t, 376
 - outer_error, 380
 - outer_size, 379
 - output, 378
 - output_channels, 379
 - output_fifo, 379
 - process, 378
 - provoke_inner_error, 378
 - provoke_outer_error, 378
 - value_type, 376
- mha_dblbuf_t< FIFO >, 374
- mha_debug
 - Error handling in the openMHA, 32
- mha_defs.h, 907
 - __MHA_FUN__, 908
 - __declspec, 908
 - CHECK_EXPR, 908
 - CHECK_VAR, 908
 - M_PI, 908
 - MAX, 909
 - MHA_EAR_LEFT, 909
 - MHA_EAR_MAX, 909
 - MHA_EAR_RIGHT, 909
 - MIN, 908
- mha_direction_t, 380
 - azimuth, 380
 - distance, 381
 - elevation, 380
- mha_domain_t
 - mha.h, 904
- mha_drifter_fifo_t
 - desired_fill_count, 385
 - get_available_space, 384
 - get_des_fill_count, 384
 - get_fill_count, 384
 - get_min_fill_count, 385
 - maximum_reader_xruns_in_succession↵
 - _before_stop, 386
 - maximum_writer_xruns_in_succession↵
 - _before_stop, 386
 - mha_drifter_fifo_t, 383
 - minimum_fill_count, 385
 - null_data, 386
 - read, 384
 - reader_started, 385
 - reader_xruns_in_succession, 386
 - reader_xruns_since_start, 386
 - reader_xruns_total, 386
 - starting, 385

- startup_zeros, 387
- stop, 385
- write, 383
- writer_started, 385
- writer_xruns_in_succession, 386
- writer_xruns_since_start, 386
- writer_xruns_total, 386
- mha_drifter_fifo_t< T >, 381
- mha_errno.c, 909
 - cstr_strerror, 910
 - mha_set_user_error, 910
 - mha_strerror, 910
 - next_except_str, 910
 - STRLEN, 910
- mha_errno.h, 910
 - MHA_ERR_INVALID_HANDLE, 911
 - MHA_ERR_NULL, 911
 - MHA_ERR_SUCCESS, 911
 - MHA_ERR_UNKNOWN, 911
 - MHA_ERR_USER, 911
 - MHA_ERR_VARFMT, 911
 - MHA_ERR_VARRANGE, 911
 - mha_set_user_error, 911
 - mha_strerror, 911
- mha_error.cpp, 911
- mha_error.hh, 912
 - Getmsg, 913
- mha_error_helpers, 85
 - digits, 85
 - snprintf_required_length, 85
- mha_event_emitter.h, 913
- mha_events.cpp, 913
- mha_events.h, 913
- mha_exit_request
 - mha_ruby.cpp, 933
- mha_fft
 - timoConfig, 860
- mha_fft_backward
 - Fast Fourier Transform functions, 74
- mha_fft_backward_scale
 - Fast Fourier Transform functions, 75
- mha_fft_forward
 - Fast Fourier Transform functions, 74
- mha_fft_forward_scale
 - Fast Fourier Transform functions, 75
- mha_fft_free
 - Fast Fourier Transform functions, 71
- mha_fft_new
 - Fast Fourier Transform functions, 71
- mha_fft_spec2wave
 - Fast Fourier Transform functions, 73
- mha_fft_spec2wave_scale
 - Fast Fourier Transform functions, 76
- mha_fft_t
 - Fast Fourier Transform functions, 71
- mha_fft_wave2spec
 - Fast Fourier Transform functions, 72
- mha_fft_wave2spec_scale
 - Fast Fourier Transform functions, 75
- mha_fftfb.cpp, 913
 - BARKSCALE_ENTRIES, 915
 - filtershapefun, 915
- mha_fftfb.hh, 915
- mha_fifo.cpp, 916
- mha_fifo.h, 916
 - mha_fifo_thread_platform_implementation←_t, 916
- mha_fifo_lw_t
 - ~mha_fifo_lw_t, 390
 - error, 392
 - mha_fifo_lw_t, 390
 - read, 391
 - set_error, 391
 - sync, 391
 - write, 390
- mha_fifo_lw_t< T >, 389
- mha_fifo_posix_threads_t, 392
 - ~mha_fifo_posix_threads_t, 393
 - acquire_mutex, 393
 - decrease_condition, 394
 - decrement, 393
 - increase_condition, 394
 - increment, 393
 - mha_fifo_posix_threads_t, 393
 - mutex, 394
 - release_mutex, 393
 - wait_for_decrease, 393
 - wait_for_increase, 393
- mha_fifo_t
 - ~mha_fifo_t, 396
 - buf, 398
 - buf_uses_placement_new, 398
 - clear, 398
 - get_available_space, 397
 - get_fill_count, 397
 - get_max_fill_count, 397
 - max_fill_count, 398
 - mha_fifo_t, 396
 - operator=, 397
 - read, 397
 - read_ptr, 398
 - value_type, 396

- write, 396
- write_ptr, 398
- mha_fifo_t< T >, 394
- mha_fifo_thread_guard_t, 398
 - ~mha_fifo_thread_guard_t, 399
 - mha_fifo_thread_guard_t, 399
 - sync, 399
- mha_fifo_thread_platform_implementation_t
 - mha_fifo.h, 916
- mha_fifo_thread_platform_t, 399
 - ~mha_fifo_thread_platform_t, 400
 - aquire_mutex, 401
 - decrement, 401
 - increment, 401
 - mha_fifo_thread_platform_t, 400
 - operator=, 402
 - release_mutex, 401
 - wait_for_decrease, 401
 - wait_for_increase, 401
- mha_filter.cpp, 916
 - diff_coeffs, 917
- mha_filter.hh, 917
- mha_free
 - mha_ruby.cpp, 933
- mha_freelib
 - mha_os.h, 922
- mha_freelib_success
 - mha_os.h, 922
- mha_generic_chain.cpp, 918
 - mhaconfig_compare, 919
- mha_generic_chain.h, 919
 - MHAPLUGIN_OVERLOAD_OUTDOM←
AIN, 919
- mha_getenv
 - mha_os.cpp, 921
 - mha_os.h, 923
- mha_getlibfun
 - mha_os.h, 922
- mha_getlibfun_checked
 - mha_os.h, 923
- mha_hton
 - mha_os.h, 923, 924
- mha_io_ifc.h, 919
 - IODestroy_t, 920
 - IOInit_t, 920
 - IOPrepare_t, 920
 - IOProcessEvent_t, 920
 - IORelease_t, 920
 - IOSetVar_t, 920
 - IOStart_t, 920
 - IOStartedEvent_t, 920
 - IOStop_t, 920
 - IOStoppedEvent_t, 920
 - IOStrError_t, 920
- mha_lib_extension
 - mha_os.h, 923
- mha_libhandle_t
 - mha_os.h, 923
- mha_library_paths
 - mha_os.cpp, 921
 - mha_os.h, 923
- mha_loadlib
 - mha_os.h, 922
- mha_loadlib_error
 - mha_os.h, 923
- mha_min_1
 - mha_signal.hh, 945
- mha_msleep
 - mha_os.h, 923
- mha_multisrc.cpp, 920
- mha_multisrc.h, 921
- mha_ntoh
 - mha_os.h, 924
- mha_os.cpp, 921
 - list_dir, 921
 - mha_getenv, 921
 - mha_library_paths, 921
- mha_os.h, 921
 - FMTsz, 923
 - list_dir, 923
 - MHA_RESOLVE_CHECKED, 923
 - MHA_RESOLVE, 923
 - mha_freelib, 922
 - mha_freelib_success, 922
 - mha_getenv, 923
 - mha_getlibfun, 922
 - mha_getlibfun_checked, 923
 - mha_hton, 923, 924
 - mha_lib_extension, 923
 - mha_libhandle_t, 923
 - mha_library_paths, 923
 - mha_loadlib, 922
 - mha_loadlib_error, 923
 - mha_msleep, 923
 - mha_ntoh, 924
- mha_parse
 - mha_ruby.cpp, 933
- mha_parser.cpp, 924
 - MHAPLATFORM, 925
 - parse_1_complex, 925
 - parse_1_float, 925
 - write_float, 925

- mha_parser.hh, 925
 - DEFAULT_RETSIZE, 929
 - insert_member, 929
- mha_platform_tic
 - mha_profiling.c, 932
 - mha_profiling.h, 933
- mha_platform_tictoc_t
 - mha_profiling.h, 933
- mha_platform_toc
 - mha_profiling.c, 932
 - mha_profiling.h, 933
- mha_plugin.hh, 929
 - __attribute__, 931
 - __declspec, 931
 - GITCOMMITHASH, 931
 - HINSTANCE, 931
 - MHAPLUGIN_DOCUMENTATION_PTR, 931
 - MHAPLUGIN_INIT_CALLBACKS_PRE, 931
 - MHAPLUGIN_INIT_CALLBACKS, 931
 - MHAPLUGIN_PROC_CALLBACK_PTR, 931
 - MHAPLUGIN_PROC_CALLBACK, 931
 - WINAPI, 931
- mha_profiling.c, 932
 - mha_platform_tic, 932
 - mha_platform_toc, 932
 - mha_tic, 932
 - mha_toc, 932
- mha_profiling.h, 932
 - mha_platform_tic, 933
 - mha_platform_tictoc_t, 933
 - mha_platform_toc, 933
- mha_real_t
 - Vector and matrix processing toolbox, 41
- mha_round
 - mha_signal.hh, 945
- mha_rt_fifo_element_t
 - ~mha_rt_fifo_element_t, 403
 - abandoned, 403
 - data, 403
 - mha_rt_fifo_element_t, 402
 - next, 403
- mha_rt_fifo_element_t< T >, 402
- mha_rt_fifo_t
 - ~mha_rt_fifo_t, 404
 - current, 406
 - mha_rt_fifo_t, 404
 - poll, 405
 - poll_1, 405
 - push, 405
 - remove_abandoned, 405
 - remove_all, 405
 - root, 406
- mha_rt_fifo_t< T >, 403
- mha_ruby.cpp, 933
 - Init_mha_ruby, 933
 - mha_alloc, 933
 - mha_exit_request, 933
 - mha_free, 933
 - mha_parse, 933
 - rb_f_t, 933
- mha_set_user_error
 - mha_errno.c, 910
 - mha_errno.h, 911
- mha_signal.cpp, 933
 - ASSERT_EQUAL_DIM_PTR, 936
 - ASSERT_EQUAL_DIM, 936
 - intensity, 936
 - MHA_ID_MATRIX, 936
 - MHA_ID_UINT_VECTOR, 936
 - safe_div, 936
 - set_minabs, 936
- mha_signal.hh, 936
 - M_PI, 945
 - mha_min_1, 945
 - mha_round, 945
 - operator<<, 945
 - operator>>, 945
 - safe_div, 945
 - set_minabs, 945
 - value, 945
- mha_signal_fft.h, 946
- mha_spec_t, 406
 - buf, 407
 - channel_info, 407
 - num_channels, 407
 - num_frames, 407
- mha_strerror
 - mha_errno.c, 910
 - mha_errno.h, 911
- mha_tablelookup.cpp, 946
- mha_tablelookup.hh, 946
- mha_tcp.cpp, 947
 - ASYNC_CONNECT_STARTED, 948
 - closesocket, 948
 - host_port_to_sock_addr, 948
 - INVALID_SOCKET, 948
 - SOCKET_ERROR, 948
 - SOCKET, 948
 - tcp_connect_to, 948

- tcp_connect_to_with_timeout, 948
- thread_start_func, 948
- mha_tcp.hh, 948
 - Sleep, 950
- mha_test_struct_size
 - PluginLoader::mhapluginloader_t, 809
- mha_tic
 - mha_profiling.c, 932
- mha_tictoc_t, 435
 - t, 435
 - tv1, 435
 - tv2, 435
 - tz, 435
- mha_toc
 - mha_profiling.c, 932
- mha_toolbox.h, 950
- mha_wave_t, 436
 - buf, 436
 - channel_info, 437
 - num_channels, 436
 - num_frames, 437
- mha_windowparser.cpp, 950
 - wnd_funs, 950
- mha_windowparser.h, 950
- mhachain, 88
- mhachain.cpp, 951
- mhachain::chain_base_t, 437
 - algos, 439
 - b_prepared, 439
 - bprofiling, 439
 - cfin, 439
 - cfout, 439
 - chain, 439
 - chain_base_t, 438
 - old_algos, 439
 - patchbay, 439
 - prepare, 439
 - process, 438, 439
 - release, 439
 - update, 439
- mhachain::mhachain_t, 440
 - mhachain_t, 440
- mhachain::plugs_t, 441
 - ~plugs_t, 442
 - ac, 442
 - algos, 442
 - alloc_plugs, 442
 - b_prepared, 442
 - b_use_profiling, 443
 - chain, 443
 - cleanup_plugs, 442
 - parser, 442
 - plugs_t, 442
 - prepare, 442
 - prepared, 442
 - proc_cnt, 443
 - process, 442
 - prof_algos, 443
 - prof_cfg, 443
 - prof_init, 443
 - prof_load_con, 443
 - prof_prepare, 443
 - prof_process, 443
 - prof_process_load, 443
 - prof_process_tt, 443
 - prof_release, 443
 - prof_tt_con, 443
 - profiling, 443
 - release, 442
 - tictoc, 443
 - update_proc_load, 442
- mhachain_t
 - mhachain::mhachain_t, 440
- mhaconfig_compare
 - mha_generic_chain.cpp, 919
 - PluginLoader, 123
- mhaconfig_in
 - MHAPLugin::plugin_t, 663
- mhaconfig_mon_t
 - MHAParser::mhaconfig_mon_t, 615
- mhaconfig_out
 - MHAPLugin::plugin_t, 663
- mhaconfig_t, 444
 - channels, 445
 - domain, 445
 - fftlens, 445
 - fragsize, 445
 - srate, 445
 - wndlen, 445
- mhafft
 - hilbert_shifter_t, 306
- mhafw_lib.cpp, 951
- mhafw_lib.h, 951
- mhajack.cpp, 966
 - dummy_jack_proc_cb, 966
 - jack_error_handler, 966
 - last_jack_err, 966
 - last_jack_err_msg, 966
 - make_friendly_number, 966
- mhajack.h, 966
 - IO_ERROR_JACK, 968
 - IO_ERROR_MHAJACKLIB, 968

- last_jack_err_msg, 968
- MAX_USER_ERR, 968
- MHAJACK_FW_STARTED, 968
- MHAJACK_STARTING, 968
- MHAJACK_STOPPED, 968
- mhamain
 - mha.cpp, 900
 - mhamain.cpp, 969
- mhamain.cpp, 968
 - create_lock, 969
 - GREETING_TEXT, 969
 - HELP_TEXT, 969
 - MAX_LINE_LENGTH, 969
 - mhamain, 969
 - remove_lock, 969
- mhaplugin_cfg_t, 653
 - ~mhaplugin_cfg_t, 653
 - mhaplugin_cfg_t, 653
- mhapluginloader.cpp, 969
- mhapluginloader.h, 969
- mhapluginloader_t
 - MHAParser::mhapluginloader_t, 617
 - PluginLoader::mhapluginloader_t, 807
- mhaserver_t, 691
 - ~mhaserver_t, 693
 - acceptor_started, 693
 - ack_fail, 694
 - ack_ok, 694
 - announce_port, 694
 - logfile, 694
 - logstring, 693
 - mhaserver_t, 692
 - pid_mon, 694
 - port, 694
 - received_group, 693
 - run, 693
 - set_announce_port, 693
 - tcpserver, 694
- mhasndfile.cpp, 970
 - validator_channels, 970
 - validator_length, 970
 - write_wave, 970
- mhasndfile.h, 970
 - write_wave, 971
- mhastrdomain
 - PluginLoader, 123
- min
 - MHASignal::doublebuffer_t, 702
 - spec2wave.cpp, 977
 - Vector and matrix processing toolbox, 58
- minimum_fill_count
 - mha_drifter_fifo_t, 385
- minphase
 - MHAFilter::smoothspec_t, 508
- minphase_t
 - MHASignal::minphase_t, 722
- minw_
 - wavwriter_t, 874
- minwrite
 - wavrec_t, 873
- mix
 - sine_cfg_t, 835
- mixer
 - matrixmixer::matmix_t, 354
- mixw_ref
 - hilbert_shifter_t, 306
- mixw_shift
 - hilbert_shifter_t, 306
- mode
 - MHA_TCP::OS_EVENT_TYPE, 420
 - noise_t, 785
 - sine_t, 837
 - smoothgains_bridge::overlapadd_if_↔
t, 839
- modified
 - dc::dc_vars_t, 222
 - dc_simple::dc_if_t, 228
- modulename
 - dynamiclib_t, 255
 - MHAParser::c_ifc_parser_t, 584
- mon
 - acmon::ac_monitor_t, 134
- mon_complex
 - acmon::ac_monitor_t, 134
- mon_dump
 - MHAParser, 107
- mon_g
 - dc_simple::dc_if_t, 228
 - dc_simple::dc_t, 230
- mon_l
 - dc_simple::dc_if_t, 228
 - dc_simple::dc_t, 230
- mon_mat
 - acmon::ac_monitor_t, 134
- mon_mat_complex
 - acmon::ac_monitor_t, 134
- mon_t, 775
 - mon_t, 775
 - store, 775
- monitor variable, 4
- monitor_t
 - MHAParser::monitor_t, 619

- mpo
 - DynComp::dc_afterburn_vars_t, 261
- mpo_inv
 - DynComp::dc_afterburn_rt_t, 257
- msg
 - MHA_Error, 389
- mu
 - MHAFilter::adapt_filter_param_t, 454
 - MHAFilter::adapt_filter_t, 457
- mu_beta
 - adm_if_t, 167
- multibandcompressor, 122
- multibandcompressor.cpp, 971
- multibandcompressor::fftfb_plug_t, 776
 - bwv, 777
 - cfv, 777
 - efv, 777
 - fftfb_plug_t, 777
 - insert, 777
- multibandcompressor::interface_t, 778
 - algo, 779
 - burn, 779
 - interface_t, 779
 - num_channels, 779
 - patchbay, 779
 - plug, 780
 - plug_sigs, 780
 - prepare, 779
 - process, 779
 - release, 779
 - update_cfg, 779
- multibandcompressor::plugin_signals_t, 780
 - apply_gains, 780
 - gain, 781
 - plug_level, 781
 - plug_output, 781
 - plugin_signals_t, 780
 - update_levels, 780
- mute
 - MHAJack::port_t, 536
 - MHASignal::loop_wavefragment_t, 710
- mutex
 - MHAPlugin_Split::posix_threads_t, 676
 - mha_fifo_posix_threads_t, 394
- mylogf
 - dc_afterburn.cpp, 887
- N
 - lpc_config, 351
- n
 - MHAJack::client_avg_t, 525
 - MHASignal::hilbert_fftw_t, 707
 - MHASignal::stat_t, 737
- N_ERRNO
 - MHA_TCP, 87
- n_channels
 - mha_audio_descriptor_t, 371
- n_freqs
 - mha_audio_descriptor_t, 371
- n_im
 - MHASignal::fft_t, 706
- n_no_update
 - nlms_t, 783
 - prediction_error, 814
- n_no_update_
 - prediction_error_config, 816
 - rt_nlms_t, 825
- n_pad1
 - overlapadd::overlapadd_t, 794
- n_pad2
 - overlapadd::overlapadd_t, 794
- n_re
 - MHASignal::fft_t, 706
- n_samples
 - mha_audio_descriptor_t, 371
- n_zero
 - overlapadd::overlapadd_t, 794
- NORM_DEFAULT
 - nlms_wave.cpp, 972
- NORM_NONE
 - nlms_wave.cpp, 972
- NORM_SUM
 - nlms_wave.cpp, 972
- NORMALIZATION_TYPES
 - nlms_wave.cpp, 972
- NUM_ENTR_LTASS
 - speechnoise.cpp, 978
- NUM_ENTR_MHAORIG
 - speechnoise.cpp, 978
- NUM_ENTR_OLNOISE
 - speechnoise.cpp, 978
- name
 - ac2wave_if_t, 126
 - ac2wave_t, 128
 - acmon::ac_monitor_t, 134
 - acsave::save_var_t, 149
 - MHA_AC::ac2matrix_helper_t, 356
 - MHA_AC::spectrum_t, 367
 - MHA_AC::waveform_t, 370
 - MHAIOPortAudio::device_info_t, 519
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 528
 - MHAMultiSrc::channel_t, 543

- MHAParser::entry_t, 590
- noisePowProposedScale::interface_t, 787
- plugindescription_t, 800
- rmslevel_if_t, 818
- shadowfilter_end::cfg_t, 832
- name_
 - AuditoryProfile::parser_t::fmap_t, 188
- name_b
 - lpc_bl_predictor, 343
 - lpc_bl_predictor_config, 345
 - lpc_burglattice, 347
 - lpc_burglattice_config, 349
- name_conAC
 - acConcat_wave, 131
- name_d
 - nlms_t, 783
- name_d_
 - prediction_error_config, 816
 - rt_nlms_t, 825
- name_e
 - nlms_t, 783
 - prediction_error, 813
- name_e_
 - rt_nlms_t, 825
- name_f
 - lpc_bl_predictor, 343
 - lpc_bl_predictor_config, 344
 - lpc_burglattice, 347
 - lpc_burglattice_config, 349
 - nlms_t, 783
 - prediction_error, 813
- name_kappa
 - lpc_bl_predictor, 343
 - lpc_burglattice, 347
- name_km
 - lpc_bl_predictor_config, 344
- name_lpc
 - prediction_error, 813
- name_lpc_
 - prediction_error_config, 816
- name_lpc_b
 - lpc_bl_predictor, 343
- name_lpc_f
 - lpc_bl_predictor, 343
- name_u
 - nlms_t, 783
- name_u_
 - rt_nlms_t, 825
- namelen
 - acsave::mat4head_t, 147
- names
 - MHAOvFilter::scale_var_t, 567
- nangle
 - acSteer_config, 153
 - steerbf_config, 854
- naudiochannels
 - dc::dc_t, 220
- nbands
 - coherence::cohflt_t, 203
 - combc_t, 208
 - dc::dc_t, 220
 - dc_simple::dc_t, 230
 - dc_simple::level_smoother_t, 235
 - DynComp::gaintable_t, 265
 - fftfilterbank::fftfb_interface_t, 289
 - MHAFilter::thirdoctave_analyzer_t, 509
 - MHAOvFilter::fspacing_t, 560
- nbits
 - calibrator_variables_t, 197
- nch
 - shadowfilter_begin::cfg_t, 829
 - shadowfilter_begin::shadowfilter_begin←_t, 831
 - spec_fader_t, 849
- nch_out
 - shadowfilter_end::cfg_t, 832
- nchan
 - acSteer_config, 153
 - steerbf_config, 854
 - timoConfig, 860
- nchannels
 - DynComp::gaintable_t, 265
 - fftfilterbank::fftfb_interface_t, 289
 - MHAFilter::adapt_filter_state_t, 455
 - MHAFilter::adapt_filter_t, 457
 - MHAFilter::iir_filter_t, 482
 - MHAFilter::smoothspec_t, 508
 - MHAFilter::thirdoctave_analyzer_t, 509
- nchannels_file_in
 - io_file_t, 312
- nchannels_in
 - io_file_t, 312
 - io_parser_t, 319
 - MHAFilter::partitioned_convolution_t, 496
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 533
 - MHAPLugin_Resampling::resampling_t, 667
- nchannels_out
 - fw_t, 298
 - io_file_t, 312
 - io_parser_t, 319

- MHAFilter::partitioned_convolution_t, 496
- MHAIOPortAudio::io_portaudio_t, 522
- MHAJack::client_t, 533
- MHAPlugin_Resampling::resampling_t, 667
- ndim
 - acsave::save_var_t, 149
- needs_write
 - MHA_TCP::Connection, 416
- neigh
 - acPooling_wave_config, 142
- neighbourhood
 - acPooling_wave, 140
- nelements
 - MHASignal::matrix_t, 721
- nested_lock
 - MHAParser::base_t, 576
- newgains
 - fader_if_t, 282
- next
 - MHAPlugin::cfg_chain_t, 654
 - mha_rt_fifo_element_t, 403
- next_except_str
 - mha_errno.c, 910
- next_read_frame_index
 - MHASignal::ringbuffer_t, 727
- next_write_frame_index
 - MHASignal::ringbuffer_t, 728
- nfft
 - MHASignal::fft_t, 706
 - overlapadd::overlapadd_if_t, 791
 - shadowfilter_end::cfg_t, 832
 - spec2wave_t, 848
 - wave2spec_if_t, 868
- nfft_
 - MHAOvFilter::fspacing_t, 560
- nframes
 - acsave::save_var_t, 149
- nfreq
 - acSteer_config, 153
 - steerbf_config, 854
 - timoConfig, 860
- nlms_t, 781
 - algo, 783
 - c, 783
 - estimtype, 783
 - lambda_smoothing_power, 783
 - n_no_update, 783
 - name_d, 783
 - name_e, 783
 - name_f, 783
 - name_u, 783
 - nlms_t, 782
 - normtype, 783
 - ntaps, 783
 - patchbay, 783
 - prepare, 782
 - process, 783
 - release, 782
 - rho, 783
 - update, 783
- nlms_wave.cpp, 971
 - ESTIM_CUR, 972
 - ESTIM_PREV, 972
 - ESTIMATION_TYPES, 972
 - make_friendly_number_by_limiting, 972
 - NORM_DEFAULT, 972
 - NORM_NONE, 972
 - NORM_SUM, 972
 - NORMALIZATION_TYPES, 972
- nm
 - lpc_burglattice_config, 349
- no_iter
 - prediction_error_config, 816
 - rt_nlms_t, 825
- noise.cpp, 972
- noise_t, 784
 - frozennoise_length, 785
 - lev, 785
 - mode, 785
 - noise_t, 785
 - patchbay, 785
 - prepare, 785
 - process, 785
 - update_cfg, 785
- noise_type_t
 - speechnoise_t, 850
- noisePow
 - noisePowProposedScale::noisePow↔Proposed, 789
 - timoConfig, 861
- noisePow_name
 - timo_params, 858
 - timoSmooth, 865
- noisePowProposed
 - noisePowProposedScale::noisePow↔Proposed, 788
- noisePowProposedScale, 122
- noisePowProposedScale.cpp, 972
 - POWSPEC_FACTOR, 973
- noisePowProposedScale::interface_t, 786
 - alphaPH1mean, 787

- alphaPSD, [787](#)
- interface_t, [787](#)
- name, [787](#)
- patchbay, [787](#)
- prepare, [787](#)
- process, [787](#)
- q, [787](#)
- update_cfg, [787](#)
- xiOptDb, [787](#)
- noisePowProposedScale::noisePowProposed, [788](#)
- alphaPH1mean_, [789](#)
- alphaPSD_, [789](#)
- estimateDebug, [789](#)
- framenos, [789](#)
- GLRDebug, [789](#)
- GLRexp, [789](#)
- inputPow, [789](#)
- inputSpec, [789](#)
- insert, [788](#)
- logGLRFact, [789](#)
- noisePow, [789](#)
- noisePowProposed, [788](#)
- noisyPer, [789](#)
- PH1Debug, [789](#)
- PH1mean, [789](#)
- priorFact, [789](#)
- process, [788](#)
- snrPost1Debug, [789](#)
- xiOpt, [789](#)
- noisyPer
 - noisePowProposedScale::noisePowProposed, [789](#)
- non_empty_partitions
 - MHAFilter::transfer_function_t, [511](#)
 - MHAFilter::transfer_matrix_t, [513](#)
- nondefault_labels
 - altplugs_t, [179](#)
- norm
 - lpc, [340](#)
 - lpc_config, [350](#)
- normalize
 - Complex arithmetics in the openMHA, [68](#)
 - MHAOvFilter::fftfb_vars_t, [555](#)
- normtype
 - nlms_t, [783](#)
- not_in_use
 - MHAPugin::cfg_chain_t, [654](#)
- not_zero
 - dc_simple, [81](#)
- notify
 - MHAParser::base_t, [575](#)
- notify_release
 - io_tcp_t, [336](#)
- notify_start
 - io_tcp_t, [336](#)
- notify_stop
 - io_tcp_t, [336](#)
- now_index
 - MHAFilter::polyphase_resampling_t, [503](#)
- npad1
 - spec2wave_t, [847](#)
 - wave2spec_t, [870](#)
- npad2
 - spec2wave_t, [848](#)
 - wave2spec_t, [870](#)
- nrefmic
 - acSteer, [152](#)
 - acSteer_config, [153](#)
- nrep
 - MHAJack::client_avg_t, [525](#)
- nsteerchan
 - acSteer, [152](#)
 - acSteer_config, [153](#)
- ntaps
 - MHAFilter::adapt_filter_state_t, [455](#)
 - MHAFilter::adapt_filter_t, [457](#)
 - nlms_t, [783](#)
 - prediction_error, [813](#)
 - prediction_error_config, [815](#)
 - rt_nlms_t, [824](#)
- ntoh
 - io_tcp_sound_t, [331](#)
- ntracks
 - shadowfilter_begin::cfg_t, [829](#)
 - shadowfilter_begin::shadowfilter_begin_t, [831](#)
 - shadowfilter_end::cfg_t, [832](#)
- null_data
 - mha_drifter_fifo_t, [386](#)
- num_AC
 - acConcat_wave, [131](#)
- num_adms
 - adm_rtconfig_t, [169](#)
- num_brackets
 - MHAParser::StrCnv, [109](#)
- num_channels
 - calibrator_variables_t, [197](#)
 - DynComp::gaintable_t, [265](#)
 - MHAFilter::blockprocessing_polyphase_resampling_t, [459](#)
 - mha_spec_t, [407](#)

- mha_wave_t, 436
 - multibandcompressor::interface_t, 779
- num_entries
 - comm_var_t, 209
- num_F
 - DynComp::gaintable_t, 265
- num_frames
 - mha_spec_t, 407
 - mha_wave_t, 437
- num_inchannels
 - io_tcp_sound_t, 332
- num_L
 - DynComp::gaintable_t, 265
- num_outchannels
 - io_tcp_sound_t, 333
- num_xruns
 - MHAJack::client_t, 533
- numDevices
 - MHAIOPortAudio::device_info_t, 519
- numSamples_AC
 - acConcat_wave_config, 132
- numbytes
 - MHASignal::matrix_t, 720
 - MHASignal::uint_vector_t, 742
- numsamples
 - acPooling_wave, 140
 - acTransform_wave, 156
- nupsample
 - doasvm_feature_extraction, 248
- nvars
 - acsave::cfg_t, 147
- nwnd
 - overlapadd::overlapadd_if_t, 791
 - wave2spec_if_t, 868
 - wave2spec_t, 870
- nwndshift
 - spec2wave_t, 848
 - wave2spec_t, 870
- nyquist_ratio
 - MHAPlugin_Resampling::resampling_if←
_t, 665
- o1_ar_filter_t
 - MHAFilter::o1_ar_filter_t, 486
- o1_lp_coeffs
 - MHAFilter, 91
- o1flt_lowpass_t
 - MHAFilter::o1flt_lowpass_t, 490
- o1flt_maxtrack_t
 - MHAFilter::o1flt_maxtrack_t, 492
- o1flt_mintrack_t
 - MHAFilter::o1flt_mintrack_t, 494
- OVERLAP_FACTOR
 - timoconfig.cpp, 982
- observe
 - MHA_TCP::Event_Watcher, 419
- observed_by
 - MHA_TCP::Wakeup_Event, 434
- observers
 - MHA_TCP::Wakeup_Event, 435
- od
 - MHAFilter::adapt_filter_state_t, 455
- offset
 - acTransform_wave_config, 157
- ola1
 - overlapadd::overlapadd_t, 793
- ola2
 - overlapadd::overlapadd_t, 793
- ola_powspec_scale
 - timoConfig, 860
- old_algos
 - mhachain::chain_base_t, 439
- olnoise
 - speechnoise_t, 850
- on_model_param_valuechanged
 - timoSmooth, 864
- on_preadaccess
 - example3_t, 273
 - example4_t, 277
- on_scale_ch_readaccess
 - example3_t, 273
 - example4_t, 277
- on_scale_ch_valuechanged
 - example3_t, 273
 - example4_t, 277
- on_scale_ch_writeaccess
 - example3_t, 273
 - example4_t, 277
- op
 - MHAParser::expression_t, 591
- op_query
 - MHAParser::base_t, 572
 - MHAParser::c_ifc_parser_t, 583
 - MHAParser::monitor_t, 619
 - MHAParser::parser_t, 623
- op_setval
 - MHAParser::base_t, 572
 - MHAParser::bool_t, 581
 - MHAParser::c_ifc_parser_t, 583
 - MHAParser::complex_t, 589
 - MHAParser::float_t, 595
 - MHAParser::int_t, 600
 - MHAParser::kw_t, 605

- MHAParser::mcomplex_t, 609
- MHAParser::mfloat_t, 613
- MHAParser::parser_t, 623
- MHAParser::string_t, 631
- MHAParser::variable_t, 633
- MHAParser::vcomplex_t, 637
- MHAParser::vfloat_t, 641
- MHAParser::vint_t, 645
- MHAParser::vstring_t, 649
- op_subparse
 - MHAParser::base_t, 572
 - MHAParser::c_ifc_parser_t, 583
 - MHAParser::parser_t, 623
- opact_map_t
 - MHAParser, 106
- opact_t
 - MHAParser, 106
- operator!=
 - Complex arithmetics in the openMHA, 67
- operator<
 - Complex arithmetics in the openMHA, 68
- operator<<
 - mha_signal.hh, 945
- operator>>
 - mha_signal.hh, 945
- operator*
 - Complex arithmetics in the openMHA, 65, 66
- operator*=
 - Complex arithmetics in the openMHA, 65
 - Vector and matrix processing toolbox, 53, 54
- operator^=
 - Vector and matrix processing toolbox, 55
- operator()
 - dc_simple::dc_t::line_t, 231
 - hanning_ramps_t, 304
 - MHAEvents::emitter_t, 451
 - MHAFilter::gammaflt_t, 476, 477
 - MHAFilter::iir_ord1_real_t, 484
 - MHAFilter::o1_ar_filter_t, 487, 488
 - MHASignal::hilbert_t, 708
 - MHASignal::matrix_t, 718–720
 - MHASignal::minphase_t, 722
 - MHASignal::quantizer_t, 724
 - MHASignal::spectrum_t, 734
 - MHASignal::waveform_t, 746, 747
 - MHAWindow::base_t, 767
- operator+
 - Complex arithmetics in the openMHA, 64
- operator+=
 - Complex arithmetics in the openMHA, 64
 - Vector and matrix processing toolbox, 53, 54
- operator-
 - Complex arithmetics in the openMHA, 64, 65, 67
- operator-=
 - Complex arithmetics in the openMHA, 64
 - Vector and matrix processing toolbox, 53
- operator/
 - Complex arithmetics in the openMHA, 66, 67
- operator/=
 - Complex arithmetics in the openMHA, 66
 - Vector and matrix processing toolbox, 54
- operator=
 - MHA_AC::acspace2matrix_t, 360
 - MHA_Error, 388
 - MHAPlugin_Split::domain_handler_t, 669
 - MHAPlugin_Split::splitted_part_t, 685
 - MHAPlugin_Split::thread_platform_t, 689
 - MHASignal::matrix_t, 716
 - MHASignal::uint_vector_t, 741
 - MHASignal::waveform_t, 746
 - mha_fifo_t, 397
 - mha_fifo_thread_platform_t, 402
- operator==
 - Complex arithmetics in the openMHA, 67
 - MHASignal::uint_vector_t, 741
- operator[]
 - MHA_AC::acspace2matrix_t, 360
 - MHASignal::spectrum_t, 734
 - MHASignal::uint_vector_t, 742
 - MHASignal::waveform_t, 746
- operators
 - MHAParser::base_t, 576
- oplist
 - MHAParser::base_t, 575
- order
 - lpc_config, 350
- origname
 - PluginLoader::config_file_splitter_t, 802
- os_event
 - MHA_TCP::Wakeup_Event, 435
- os_event_valid
 - MHA_TCP::Wakeup_Event, 435
- out
 - adm_if_t, 167
 - delaysum::delaysum_t, 241
- out_buf
 - overlapadd::overlapadd_t, 794

- spec2wave_t, 848
- out_chunk
 - MHAFilter::thirdoctave_analyzer_t, 510
- out_chunk_im
 - MHAFilter::thirdoctave_analyzer_t, 510
- out_spec
 - shadowfilter_begin::cfg_t, 829
 - shadowfilter_end::cfg_t, 832
- outSpec
 - steerbf_config, 854
- outbuf
 - MHA_TCP::Connection, 417
- outch
 - MHAJack::client_t, 534
- outchannels
 - combc_if_t, 207
- outer2inner_resampling
 - MHAPLugin_Resampling::resampling_t, 667
- outer_ac
 - analysepath_t, 181
- outer_ac_copy
 - analysepath_t, 181
- outer_error
 - mha_dblbuf_t, 380
- outer_fragsize
 - MHAPLugin_Resampling::resampling_t, 666
- outer_out
 - MHASignal::doublebuffer_t, 703
- outer_process
 - MHASignal::doublebuffer_t, 702
- outer_size
 - mha_dblbuf_t, 379
- outer_srate
 - MHAPLugin_Resampling::resampling_t, 667
- output
 - io_parser_t, 319
 - MHAJack::port_t, 535
 - mha_dblbuf_t, 378
- output_cfg
 - MHAPLugin::plugin_t, 662
- output_cfg_
 - MHAPLugin::plugin_t, 663
- output_channels
 - mha_dblbuf_t, 379
- output_domain
 - PluginLoader::mhapluginloader_t, 808
- output_fifo
 - mha_dblbuf_t, 379
- output_partitions
 - MHAFilter::partitioned_convolution_t, 496
- output_portnames
 - MHAJack::client_t, 534
- output_sample_format
 - io_file_t, 312
- output_signal
 - MHAPLugin_Resampling::resampling_t, 667
- output_signal_spec
 - MHAFilter::partitioned_convolution_t, 497
- output_signal_wave
 - MHAFilter::partitioned_convolution_t, 497
- outputchannels
 - MHAFilter::fftfilterbank_t, 470
- overlap_save_filterbank_analytic_t
 - MHAOvIFilter::overlap_save_filterbank_↔_analytic_t, 562
- overlap_save_filterbank_t
 - MHAOvIFilter::overlap_save_filterbank_t, 564
- overlapadd, 123
- overlapadd.cpp, 973
- overlapadd::overlapadd_if_t, 790
 - ~overlapadd_if_t, 791
 - algo, 792
 - cf_in, 792
 - cf_out, 792
 - nfft, 791
 - nwnd, 791
 - overlapadd_if_t, 791
 - patchbay, 791
 - plugloader, 792
 - postscale, 792
 - prepare, 791
 - prescale, 792
 - process, 791
 - release, 791
 - update, 791
 - window, 792
 - wndexp, 792
 - wndpos, 792
 - zerowindow, 792
- overlapadd::overlapadd_t, 792
 - ~overlapadd_t, 793
 - calc_out, 794
 - fft, 793
 - n_pad1, 794
 - n_pad2, 794
 - n_zero, 794
 - ola1, 793

- ola2, 793
- out_buf, 794
- overlapadd_t, 793
- postwnd, 793
- prewnd, 793
- spec_in, 794
- wave_in1, 793
- wave_out1, 794
- write_buf, 794
- overlapadd_if_t
 - overlapadd::overlapadd_if_t, 791
 - smoothgains_bridge::overlapadd_if_t, 838
- overlapadd_t
 - overlapadd::overlapadd_t, 793
- ovltype
 - MHAOvFilter::fftfb_vars_t, 554
- oy
 - MHAFilter::adapt_filter_state_t, 455
- p
 - acPooling_wave_config, 142
 - doasvm_classification_config, 245
 - pluginbrowser_t, 800
- p1
 - MHASignal::hilbert_fftw_t, 707
- p2
 - MHASignal::hilbert_fftw_t, 707
- P_Sum
 - rt_nlms_t, 825
- p_max
 - acPooling_wave_config, 142
 - doasvm_classification_config, 245
- p_name
 - acPooling_wave, 140
 - doasvm_classification, 244
- p_parser
 - acmon::ac_monitor_t, 135
- PATCH_VAR
 - acConcat_wave.cpp, 878
 - acPooling_wave.cpp, 879
 - acSteer.cpp, 880
 - acTransform_wave.cpp, 881
 - doasvm_classification.cpp, 889
 - doasvm_feature_extraction.cpp, 890
 - lpc.cpp, 898
 - lpc_bl_predictor.cpp, 898
 - lpc_burg-lattice.cpp, 899
 - prediction_error.cpp, 974
 - steerbf.cpp, 981
 - timoSmooth.cpp, 983
- PH1Debug
 - noisePowProposedScale::noisePow↔
 - Proposed, 789
- PH1mean
 - noisePowProposedScale::noisePow↔
 - Proposed, 789
- POWSPEC_FACTOR
 - noisePowProposedScale.cpp, 973
 - timoconfig.cpp, 982
- PREPARED
 - MHA_TCP::Thread, 428
- pa22dbspl
 - Vector and matrix processing toolbox, 43
- pa2dbspl
 - Vector and matrix processing toolbox, 42
- params
 - timoConfig, 860
- parent
 - MHAParser::base_t, 576
- parent_
 - MHAParser::mhapluginloader_t, 618
- parse
 - altplugs_t, 178
 - io_tcp_t, 336
 - MHAParser::base_t, 571, 572
 - MHAPlugin_Split::splitted_part_t, 685
 - PluginLoader::fourway_processor_t, 805
 - PluginLoader::mhapluginloader_t, 807
- parse_1_complex
 - mha_parser.cpp, 925
- parse_1_float
 - mha_parser.cpp, 925
- parser
 - io_tcp_t, 336
 - mhachain::plugs_t, 442
- parser_int_dyn, 795
 - parser_int_dyn, 795
 - set_max_angle_ind, 796
- parser_plugs
 - altplugs_t, 179
- parser_t
 - AuditoryProfile::parser_t, 186
 - MHAParser::parser_t, 622
- parserFriendlyName
 - MHAIOPortAudio, 94
- parsername
 - latex_doc_t, 337
- parserstate
 - fw_t, 298
- partitioned_convolution_t
 - MHAFilter::partitioned_convolution_t, 495
- partitions

- MHAFilter::transfer_function_t, 511
- MHAFilter::transfer_matrix_t, 513
- patchbay
 - ac2wave_if_t, 126
 - acConcat_wave, 131
 - acPooling_wave, 140
 - acSteer, 152
 - acTransform_wave, 156
 - acmon::acmon_t, 137
 - acsave::acsave_t, 145
 - adm_if_t, 167
 - altplugs_t, 179
 - analysispath_if_t, 183
 - AuditoryProfile::parser_t::fmap_t, 188
 - calibrator_t, 196
 - coherence::cohflt_if_t, 201
 - db_if_t, 214
 - dc::dc_if_t, 217
 - dc::wideband_inhib_vars_t, 226
 - dc_simple::dc_if_t, 228
 - delay::interface_t, 237
 - delaysum::delaysum_if_t, 240
 - doasvm_classification, 244
 - doasvm_feature_extraction, 248
 - DynComp::dc_afterburn_t, 259
 - example3_t, 274
 - example4_t, 278
 - example6_t, 280
 - fader_if_t, 282
 - fader_wave::fader_wave_if_t, 284
 - fftfilterbank::fftfb_interface_t, 289
 - frequency_translator_t, 293
 - fw_t, 299
 - gain::gain_if_t, 302
 - io_parser_t, 319
 - lpc, 340
 - lpc_bl_predictor, 343
 - lpc_burglattice, 347
 - MHAIOJack::io_jack_t, 517
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAPugin_Split::split_t, 681
 - matrixmixer::matmix_t, 354
 - mhachain::chain_base_t, 439
 - multibandcompressor::interface_t, 779
 - nlms_t, 783
 - noise_t, 785
 - noisePowProposedScale::interface_t, 787
 - overlapadd::overlapadd_if_t, 791
 - plugin_interface_t, 798
 - prediction_error, 814
 - route::interface_t, 821
 - sine_t, 837
 - smoothgains_bridge::overlapadd_if_t, 839
 - softclip_t, 842
 - spec2wave_if_t, 846
 - steerbf, 853
 - timoSmooth, 865
 - wave2spec_if_t, 868
 - wavrec_t, 873
 - windowselector_t, 877
- peak
 - MHASignal::loop_wavefragment_t, 710
 - rmslevel_t, 819
- peak_db
 - rmslevel_t, 819
- peaklevel
 - calibrator_variables_t, 197
 - MHASignal::async_rmslevel_t, 696
 - mha_channel_info_t, 373
- peer_addr
 - MHA_TCP::Connection, 417
- peer_address
 - io_tcp_parser_t, 328
- peer_port
 - io_tcp_parser_t, 329
- period
 - droptect_t, 252
- pfragmentsize
 - fw_vars_t, 300
- phase
 - cpuload_t, 212
 - MHASignal::minphase_t, 723
- phase_correction
 - MHAFilter::gammaflt_t, 477
- phase_div_2pi
 - sine_t, 837
- phase_gains
 - MHASignal::subsample_delay_t, 739
- phase_increment_div_2pi
 - sine_cfg_t, 835
- phasemode
 - frequency_translator_t, 293
- phasemodel
 - MHAOvFilter::overlap_save_filterbank_t::vars_t, 565
- phi
 - hilbert_shifter_t, 306
- PI
 - ADM, 78
 - hann.cpp, 896
- pid_mon

- mhaserver_t, 694
- pinchannels
 - fw_vars_t, 300
- pink
 - speechnoise_t, 850
- pipe
 - MHA_TCP::Async_Notify, 409
- pitch_set_first
 - timoConfig, 861
- pitch_set_first_AC
 - timo_AC, 856
- pitch_set_last
 - timoConfig, 861
- pitch_set_last_AC
 - timo_AC, 856
- plan_spec2analytic
 - hilbert_shifter_t, 306
- plateau
 - MHAOvIFilter::fftfb_vars_t, 554
- playback
 - MHASignal::loop_wavefragment_t, 711, 712
- playback_channels
 - MHASignal::loop_wavefragment_t, 712
- playback_mode_t
 - MHASignal::loop_wavefragment_t, 710
- plug
 - analysispath_if_t, 183
 - MHAParser::mhappluginloader_t, 618
 - MHAPlugin_Split::splitted_part_t, 686
 - multibandcompressor::interface_t, 780
- plug_level
 - multibandcompressor::plugin_signals_t, 781
- plug_output
 - multibandcompressor::plugin_signals_t, 781
- plug_sigs
 - multibandcompressor::interface_t, 780
- plug_t, 796
 - ~plug_t, 796
 - get_ac, 797
 - get_handle, 797
 - get_process_spec, 797
 - get_process_wave, 797
 - plug_t, 796
- plugin_categories
 - PluginLoader::mhappluginloader_t, 810
- plugin_documentation
 - PluginLoader::mhappluginloader_t, 810
- plugin_extension
 - pluginbrowser_t, 799
- plugin_interface_t, 797
 - factor, 798
 - patchbay, 798
 - plugin_interface_t, 798
 - prepare, 798
 - process, 798
 - scale_ch, 798
 - update_cfg, 798
- plugin_macro
 - latex_doc_t, 338
- plugin_paths
 - fw_t, 298
- plugin_signals_t
 - multibandcompressor::plugin_signals_t, 780
- plugin_t
 - MHAPlugin::plugin_t, 661
- PluginLoader, 123
 - mhaconfig_compare, 123
 - mhastrdomain, 123
- PluginLoader::config_file_splitter_t, 801
 - config_file_splitter_t, 801
 - configfile, 802
 - configname, 802
 - get_configfile, 802
 - get_configname, 801
 - get_libname, 802
 - get_origname, 802
 - libname, 802
 - origname, 802
- PluginLoader::fourway_processor_t, 802
 - ~fourway_processor_t, 803
 - parse, 805
 - prepare, 805
 - process, 803, 804
 - release, 805
- PluginLoader::mhappluginloader_t, 805
 - ~mhappluginloader_t, 807
 - ac, 809
 - b_check_version, 810
 - b_is_prepared, 810
 - cf_input, 810
 - cf_output, 810
 - get_categories, 809
 - get_documentation, 809
 - getfullname, 808
 - has_parser, 808
 - has_process, 808
 - input_domain, 808
 - is_prepared, 809

- lib_data, 809
- lib_err, 809
- lib_handle, 809
- MHADestroy_cb, 809
- MHAGetVersion_cb, 809
- MHAINit_cb, 809
- MHAPrepair_cb, 809
- MHAProc_spec2spec_cb, 809
- MHAProc_spec2wave_cb, 810
- MHAProc_wave2spec_cb, 809
- MHAProc_wave2wave_cb, 809
- MHARelease_cb, 809
- MHASET_cb, 810
- MHASTrError_cb, 810
- mha_test_struct_size, 809
- mhapluginloader_t, 807
- output_domain, 808
- parse, 807
- plugin_categories, 810
- plugin_documentation, 810
- prepare, 808
- process, 808
- release, 808
- resolve_and_init, 809
- test_error, 809
- test_version, 809
- pluginbrowser.cpp, 973
- pluginbrowser.h, 973
- pluginbrowser_t, 799
 - add_plugin, 799
 - add_plugins, 799
 - clear_plugins, 799
 - get_paths, 799
 - get_plugins, 799
 - library_paths, 800
 - p, 800
 - plugin_extension, 799
 - pluginbrowser_t, 799
 - plugins, 800
 - scan_plugin, 799
 - scan_plugins, 799
- plugindescription_t, 800
 - categories, 800
 - documentation, 800
 - fullname, 800
 - name, 800
 - queries, 800
 - query_cmds, 800
 - spec2spec, 800
 - spec2wave, 800
 - wave2spec, 800
 - wave2wave, 800
- pluginloader_t, 810
 - ~pluginloader_t, 811
 - pluginloader_t, 811
- plugins
 - fw_t, 298
 - pluginbrowser_t, 800
- plugloader
 - bbcalib_interface_t, 192
 - db_if_t, 214
 - db_t, 215
 - MHAPLugin_Resampling::resampling_if←_t, 665
 - MHAPLugin_Resampling::resampling_t, 667
 - overlapadd::overlapadd_if_t, 792
 - smoothgains_bridge::overlapadd_if←_t, 839
- plugname
 - latex_doc_t, 338
 - MHAParser::mhapluginloader_t, 618
- plugname_name_
 - MHAParser::mhapluginloader_t, 618
- plugs
 - altplugs_t, 179
- plugs_t
 - mhachain::plugs_t, 442
- pmode
 - calibrator_runtime_layer_t, 194
- poll
 - mha_rt_fifo_t, 405
- poll_1
 - mha_rt_fifo_t, 405
- poll_config
 - MHAPLugin::config_t, 656
- polyphase_resampling_t
 - MHAFilter::polyphase_resampling_t, 501
- pool
 - acPooling_wave_config, 142
- pool_name
 - acPooling_wave, 140
- pooling_ind
 - acPooling_wave_config, 142
- pooling_option
 - acPooling_wave_config, 142
- pooling_size
 - acPooling_wave_config, 142
- pooling_type
 - acPooling_wave, 140
- pooling_wndlen
 - acPooling_wave, 140

- port
 - MHA_TCP::Server, [423](#)
 - MHAJack::port_t, [537](#)
 - mhaserver_t, [694](#)
- port_t
 - MHAJack::port_t, [535](#), [536](#)
- portaudio_callback
 - MHAIOPortAudio.cpp, [961](#), [962](#)
 - MHAIOPortAudio::io_portaudio_t, [522](#)
- portaudio_stream
 - MHAIOPortAudio::io_portaudio_t, [522](#)
- portnames_in
 - MHAIOJack::io_jack_t, [517](#)
- portnames_out
 - MHAIOJack::io_jack_t, [517](#)
- ports_in_all
 - MHAIOJack::io_jack_t, [517](#)
- ports_in_physical
 - MHAIOJack::io_jack_t, [517](#)
- ports_out_all
 - MHAIOJack::io_jack_t, [517](#)
- ports_out_physical
 - MHAIOJack::io_jack_t, [517](#)
- ports_parser
 - MHAIOJack::io_jack_t, [517](#)
- pos
 - cfg_t, [199](#)
 - fader_wave::level_adapt_t, [286](#)
 - MHAJack::client_avg_t, [525](#)
 - MHAJack::client_noncont_t, [527](#)
 - MHASignal::async_rmslevel_t, [696](#)
 - MHASignal::delay_spec_t, [697](#)
 - MHASignal::delay_t, [699](#)
 - MHASignal::delay_wave_t, [700](#)
 - MHASignal::loop_wavefragment_t, [712](#)
- posix_threads_t
 - MHAPlugin_Split::posix_threads_t, [675](#)
- posixthreads
 - split.cpp, [981](#)
- postscale
 - overlapadd::overlapadd_if_t, [792](#)
- postwindow
 - spec2wave_t, [848](#)
- postwnd
 - overlapadd::overlapadd_t, [793](#)
- powSpec
 - timoConfig, [861](#)
- power
 - MHASignal::waveform_t, [751](#)
- powersum
 - dc::dc_t, [219](#)
 - dc::dc_vars_t, [221](#)
- powspec
 - MHASignal::waveform_t, [752](#)
- pred_err_delay
 - prediction_error, [814](#)
- prediction_error, [811](#)
 - ~prediction_error, [812](#)
 - c, [813](#)
 - delay_d, [814](#)
 - delay_w, [814](#)
 - gains, [813](#)
 - lpc_order, [814](#)
 - n_no_update, [814](#)
 - name_e, [813](#)
 - name_f, [813](#)
 - name_lpc, [813](#)
 - ntaps, [813](#)
 - patchbay, [814](#)
 - pred_err_delay, [814](#)
 - prediction_error, [812](#)
 - prepare, [813](#)
 - process, [813](#)
 - release, [813](#)
 - rho, [813](#)
 - update_cfg, [813](#)
- prediction_error.cpp, [973](#)
 - INSERT_PATCH, [974](#)
 - make_friendly_number_by_limiting, [974](#)
 - PATCH_VAR, [974](#)
- prediction_error.h, [974](#)
- prediction_error_config, [814](#)
 - ~prediction_error_config, [815](#)
 - ac, [815](#)
 - channels, [816](#)
 - EPrew, [817](#)
 - F, [816](#)
 - F_Uflt, [816](#)
 - frames, [815](#)
 - insert, [815](#)
 - iter, [816](#)
 - n_no_update_, [816](#)
 - name_d_, [816](#)
 - name_lpc_, [816](#)
 - no_iter, [816](#)
 - ntaps, [815](#)
 - prediction_error_config, [815](#)
 - process, [815](#)
 - Pu, [816](#)
 - s_E_pred_err_delay, [816](#)
 - s_LPC, [816](#)
 - s_U_delay, [816](#)

- s_U_delayflt, 816
- s_Usmpl, 817
- s_Wflt, 816
- s_Y_delay, 816
- s_Y_delayflt, 816
- s_E, 816
- s_U, 816
- s_W, 816
- smp1, 817
- UPrew, 817
- UPrewW, 817
- UbufferPrew, 816
- v_G, 816
- YPrew, 817
- prefix
 - wavrec_t, 873
- prefix_
 - MHAParser::mhapluginloader_t, 618
- prefix_names_AC
 - acConcat_wave, 131
- prepare
 - ac2wave_if_t, 126
 - acConcat_wave, 129
 - acPooling_wave, 139
 - acSteer, 151
 - acTransform_wave, 155
 - acmon::acmon_t, 136
 - acsave::acsave_t, 144
 - adm_if_t, 166
 - altplugs_t, 177
 - analysispath_if_t, 183
 - bbcalib_interface_t, 192
 - calibrator_t, 195
 - coherence::cohflt_if_t, 201
 - combc_if_t, 207
 - cpuload_t, 211
 - db_if_t, 213
 - dc::dc_if_t, 217
 - dc_simple::dc_if_t, 227
 - delay::interface_t, 237
 - delaysum::delaysum_if_t, 239
 - doasvm_classification, 243
 - doasvm_feature_extraction, 247
 - droptect_t, 251
 - ds_t, 254
 - example1_t, 267
 - example2_t, 270
 - example3_t, 273
 - example4_t, 277
 - example6_t, 280
 - fader_if_t, 282
 - fader_wave::fader_wave_if_t, 284
 - fftfilterbank::fftfb_interface_t, 287
 - frequency_translator_t, 293
 - fw_t, 296
 - gain::gain_if_t, 302
 - identity_t, 308
 - io_file_t, 311
 - io_lib_t, 315
 - io_parser_t, 318
 - io_tcp_sound_t, 331
 - io_tcp_t, 335
 - lpc, 339
 - lpc_bl_predictor, 342
 - lpc_burglattice, 347
 - MHAIOJack::io_jack_t, 515
 - MHAJack::client_t, 530, 531
 - MHAParser::mhapluginloader_t, 617
 - MHAPlugin::plugin_t, 661
 - MHAPlugin_Resampling::resampling_if_↔_t, 665
 - MHAPlugin_Split::splitted_part_t, 685
 - MHATableLookup::linear_table_t, 758
 - matrixmixer::matmix_t, 354
 - mhachain::chain_base_t, 439
 - mhachain::plugs_t, 442
 - multibandcompressor::interface_t, 779
 - nlms_t, 782
 - noise_t, 785
 - noisePowProposedScale::interface_t, 787
 - overlapadd::overlapadd_if_t, 791
 - plugin_interface_t, 798
 - PluginLoader::fourway_processor_t, 805
 - PluginLoader::mhapluginloader_t, 808
 - prediction_error, 813
 - rmslevel_if_t, 818
 - route::interface_t, 821
 - save_spec_t, 827
 - save_wave_t, 828
 - shadowfilter_begin::shadowfilter_begin_↔_t, 831
 - shadowfilter_end::shadowfilter_end_↔_t, 834
 - sine_t, 836
 - smoothgains_bridge::overlapadd_if_↔_t, 838
 - softclip_t, 842
 - spec2wave_if_t, 846
 - steerbf, 852
 - timoSmooth, 864
 - us_t, 866
 - wave2spec_if_t, 868

- wavrec_t, 872
- prepare_
 - iirfilter_t, 309
 - MHAPLugin::plugin_t, 662
 - MHAPLugin_Split::split_t, 680
- prepare_impl
 - MHAJack::client_t, 532
- prepare_vars
 - fw_t, 298
- prepared
 - ac2wave_if_t, 126
 - altplugs_t, 179
 - calibrator_t, 196
 - dc_simple::dc_if_t, 228
 - example3_t, 274
 - example4_t, 278
 - fader_wave::fader_wave_if_t, 284
 - fftfilterbank::fftfb_interface_t, 289
 - mhachain::plugs_t, 442
 - route::interface_t, 821
 - timoSmooth, 865
- prereadaccess
 - MHAParser::base_t, 576
- prescale
 - overlapadd::overlapadd_if_t, 792
- preset
 - dc::dc_vars_t, 222
 - dc_simple::dc_if_t, 228
- prewnd
 - overlapadd::overlapadd_t, 793
- print_ac
 - analysemhaplugin.cpp, 883
- prior_q
 - timo_params, 858
 - timoSmooth, 865
- priorFact
 - noisePowProposedScale::noisePow↔
Proposed, 789
 - timoConfig, 861
- priority
 - analysepath_t, 181
 - analysispath_if_t, 183
 - MHAPLugin_Split::posix_threads_t, 677
- proc
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 527
- proc_1
 - smoothgains_bridge::smoothspec_↔
wrap_t, 840
- proc_2
 - smoothgains_bridge::smoothspec_↔
- wrap_t, 840
- proc_cnt
 - mhachain::plugs_t, 443
- proc_err
 - io_tcp_fwcb_t, 323
- proc_error
 - fw_t, 299
- proc_error_string
 - fw_t, 299
- proc_event
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_fwcb_t, 322
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 533
- proc_handle
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_fwcb_t, 322
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 533
- proc_lib
 - fw_t, 299
- proc_name
 - fw_t, 298
- proc_ramp
 - altplugs_t, 178
- proc_wave
 - doasvm_feature_extraction_config, 249
- process
 - ADM::ADM, 159
 - ADM::Delay, 161
 - ADM::Linearphase_FIR, 164
 - ac2wave_if_t, 126
 - ac2wave_t, 127
 - acConcat_wave, 129
 - acConcat_wave_config, 132
 - acPooling_wave, 139
 - acPooling_wave_config, 141
 - acSteer, 151
 - acTransform_wave, 155
 - acTransform_wave_config, 157
 - acmon::acmon_t, 137
 - acsave::acsave_t, 144, 145
 - adm_if_t, 166
 - altplugs_t, 178
 - analysispath_if_t, 183
 - bbcalib_interface_t, 192
 - calibrator_runtime_layer_t, 193
 - calibrator_t, 195
 - cfg_t, 199

- coherence::cohflt_if_t, 201
- coherence::cohflt_t, 203
- combc_if_t, 207
- combc_t, 208
- cpuload_t, 211
- db_if_t, 213
- dc::dc_if_t, 217
- dc::dc_t, 219
- dc_simple::dc_if_t, 227, 228
- dc_simple::dc_t, 230
- dc_simple::level_smoother_t, 235
- delay::interface_t, 237
- delaysum::delaysum_if_t, 239
- delaysum::delaysum_t, 241
- doasvm_classification, 243
- doasvm_classification_config, 245
- doasvm_feature_extraction, 247
- doasvm_feature_extraction_config, 249
- droptect_t, 251
- ds_t, 254
- example1_t, 268
- example2_t, 270
- example3_t, 274
- example4_t, 277
- example5_t, 279
- example6_t, 280
- fader_if_t, 282
- fader_wave::fader_wave_if_t, 284
- fftfilterbank::fftfb_interface_t, 289
- fftfilterbank::fftfb_plug_t, 291
- frequency_translator_t, 293
- fw_t, 297
- gain::gain_if_t, 302
- hilbert_shifter_t, 306
- identity_t, 307, 308
- iirfilter_t, 309
- io_tcp_fwcb_t, 321
- lpc, 339
- lpc_bl_predictor, 342
- lpc_bl_predictor_config, 344
- lpc_burglattice, 347
- lpc_burglattice_config, 348
- lpc_config, 350
- MHAFilter::partitioned_convolution_t, 496
- MHAFilter::thirdoctave_analyzer_t, 509
- MHAParser::mhapluginloader_t, 617, 618
- MHAPLugin_Resampling::resampling_if_↵
_t, 665
- MHAPLugin_Resampling::resampling_t,
666
- MHAPLugin_Split::domain_handler_t, 671
- MHAPLugin_Split::split_t, 680
- MHAPLugin_Split::uni_processor_t, 691
- MHASignal::async_rmslevel_t, 696
- MHASignal::delay_spec_t, 697
- MHASignal::delay_t, 698
- MHASignal::delay_wave_t, 700
- MHASignal::subsample_delay_t, 739
- matrixmixer::cfg_t, 352
- matrixmixer::matmix_t, 354
- mha_dblbuf_t, 378
- mhachain::chain_base_t, 438, 439
- mhachain::plugs_t, 442
- multibandcompressor::interface_t, 779
- nlms_t, 783
- noise_t, 785
- noisePowProposedScale::interface_t, 787
- noisePowProposedScale::noisePow_↵
Proposed, 788
- overlapadd::overlapadd_if_t, 791
- plugin_interface_t, 798
- PluginLoader::fourway_processor_t, 803,
804
- PluginLoader::mhapluginloader_t, 808
- prediction_error, 813
- prediction_error_config, 815
- rmslevel_if_t, 818
- rmslevel_t, 819
- route::interface_t, 821
- route::process_t, 822
- rt_nlms_t, 824
- save_spec_t, 827
- save_wave_t, 828
- shadowfilter_begin::cfg_t, 829
- shadowfilter_begin::shadowfilter_begin_↵
_t, 831
- shadowfilter_end::cfg_t, 832
- shadowfilter_end::shadowfilter_end_↵
t, 834
- sine_t, 836
- smoothgains_bridge::overlapadd_if_↵
t, 838
- softclip_t, 842
- softclipper_t, 843
- spec2wave_if_t, 846
- spec2wave_t, 847
- steerbf, 852
- steerbf_config, 854
- timoConfig, 860
- timoSmooth, 863
- us_t, 866
- wave2spec_if_t, 868

- wave2spec_t, 870
 - wavrec_t, 872
 - wavwriter_t, 874
- process_frame
 - io_parser_t, 319
- process_t
 - route::process_t, 822
- ProcessMutex
 - analysepath_t, 181
- processing_done
 - MHAPLugin_Split::posix_threads_t, 677
- processor
 - MHAPLugin_Split::domain_handler_t, 672
 - MHAPLugin_Split::thread_platform_t, 689
- prof_algos
 - mhachain::plugs_t, 443
- prof_cfg
 - mhachain::plugs_t, 443
- prof_init
 - mhachain::plugs_t, 443
- prof_load_con
 - mhachain::plugs_t, 443
- prof_prepare
 - mhachain::plugs_t, 443
- prof_process
 - mhachain::plugs_t, 443
- prof_process_load
 - mhachain::plugs_t, 443
- prof_process_tt
 - mhachain::plugs_t, 443
- prof_release
 - mhachain::plugs_t, 443
- prof_tt_con
 - mhachain::plugs_t, 443
- profiling
 - mhachain::plugs_t, 443
- provoke_inner_error
 - mha_dblbuf_t, 378
- provoke_outer_error
 - mha_dblbuf_t, 378
- psrate
 - fw_vars_t, 300
- Pu
 - prediction_error_config, 816
 - rt_nlms_t, 824
- push
 - MHASignal::stat_t, 737
 - mha_rt_fifo_t, 405
- push_config
 - MHAPLugin::config_t, 658
- put_signal
 - MHAPLugin_Split::domain_handler_t, 670
- q
 - noisePowProposedScale::interface_t, 787
- q_high
 - timoConfig, 860
- q_low
 - timoConfig, 860
- quant
 - calibrator_runtime_layer_t, 193
- quantile
 - MHASignal, 118
- quantizer_t
 - MHASignal::quantizer_t, 723
- queries
 - MHAParser::base_t, 576
 - plugindescription_t, 800
- query_addsubst
 - MHAParser::base_t, 574
- query_cmds
 - MHAParser::base_t, 574
 - plugindescription_t, 800
- query_dump
 - MHAParser::base_t, 572
 - MHAParser::monitor_t, 619
 - MHAParser::parser_t, 623
- query_entries
 - MHAParser::base_t, 572
 - MHAParser::parser_t, 623
- query_help
 - MHAParser::base_t, 574
- query_id
 - MHAParser::base_t, 574
- query_listids
 - MHAParser::base_t, 574
 - MHAParser::parser_t, 624
- query_map_t
 - MHAParser, 106
- query_perm
 - MHAParser::base_t, 573
 - MHAParser::monitor_t, 620
 - MHAParser::variable_t, 633
- query_range
 - MHAParser::base_t, 573
 - MHAParser::kw_t, 605
 - MHAParser::range_var_t, 626
- query_readfile
 - MHAParser::base_t, 573
 - MHAParser::parser_t, 624
- query_savefile
 - MHAParser::base_t, 573
 - MHAParser::parser_t, 624

- query_savefile_compact
 - MHAParser::base_t, [574](#)
 - MHAParser::parser_t, [624](#)
- query_savemons
 - MHAParser::base_t, [574](#)
 - MHAParser::parser_t, [624](#)
- query_subst
 - MHAParser::base_t, [574](#)
- query_t
 - MHAParser, [106](#)
- query_type
 - MHAParser::base_t, [573](#)
 - MHAParser::bool_mon_t, [579](#)
 - MHAParser::bool_t, [581](#)
 - MHAParser::complex_mon_t, [587](#)
 - MHAParser::complex_t, [589](#)
 - MHAParser::float_mon_t, [593](#)
 - MHAParser::float_t, [595](#)
 - MHAParser::int_mon_t, [597](#)
 - MHAParser::int_t, [600](#)
 - MHAParser::kw_t, [605](#)
 - MHAParser::mcomplex_mon_t, [607](#)
 - MHAParser::mcomplex_t, [609](#)
 - MHAParser::mfloat_mon_t, [611](#)
 - MHAParser::mfloat_t, [613](#)
 - MHAParser::parser_t, [623](#)
 - MHAParser::string_mon_t, [629](#)
 - MHAParser::string_t, [631](#)
 - MHAParser::vcomplex_mon_t, [635](#)
 - MHAParser::vcomplex_t, [637](#)
 - MHAParser::vfloat_mon_t, [639](#)
 - MHAParser::vfloat_t, [641](#)
 - MHAParser::vint_mon_t, [643](#)
 - MHAParser::vint_t, [645](#)
 - MHAParser::vstring_mon_t, [647](#)
 - MHAParser::vstring_t, [649](#)
- query_val
 - MHAParser::base_t, [573](#)
 - MHAParser::bool_mon_t, [579](#)
 - MHAParser::bool_t, [581](#)
 - MHAParser::complex_mon_t, [587](#)
 - MHAParser::complex_t, [589](#)
 - MHAParser::float_mon_t, [593](#)
 - MHAParser::float_t, [595](#)
 - MHAParser::int_mon_t, [597](#)
 - MHAParser::int_t, [600](#)
 - MHAParser::kw_t, [605](#)
 - MHAParser::mcomplex_mon_t, [607](#)
 - MHAParser::mcomplex_t, [609](#)
 - MHAParser::mfloat_mon_t, [611](#)
 - MHAParser::mfloat_t, [613](#)
 - MHAParser::parser_t, [624](#)
 - MHAParser::string_mon_t, [629](#)
 - MHAParser::string_t, [631](#)
 - MHAParser::vcomplex_mon_t, [635](#)
 - MHAParser::vcomplex_t, [637](#)
 - MHAParser::vfloat_mon_t, [639](#)
 - MHAParser::vfloat_t, [641](#)
 - MHAParser::vint_mon_t, [643](#)
 - MHAParser::vint_t, [645](#)
 - MHAParser::vstring_mon_t, [647](#)
 - MHAParser::vstring_t, [649](#)
- quit
 - fw_t, [297](#)
- R
 - AuditoryProfile::parser_t, [186](#)
 - AuditoryProfile::profile_t, [190](#)
 - lpc_config, [351](#)
 - MHA_TCP::OS_EVENT_TYPE, [420](#)
- RUNNING
 - MHA_TCP::Thread, [428](#)
- rad2smpl
 - Vector and matrix processing toolbox, [45](#)
- ramp_a
 - hanning_ramps_t, [304](#)
- ramp_b
 - hanning_ramps_t, [304](#)
- ramp_begin
 - MHAWindow::base_t, [767](#)
- ramp_counter
 - altplugs_t, [179](#)
- ramp_end
 - MHAWindow::base_t, [767](#)
- ramp_len
 - altplugs_t, [179](#)
- ramplen
 - altplugs_t, [179](#)
 - fader_wave::fader_wave_if_t, [284](#)
 - spec2wave_if_t, [846](#)
- ramps
 - spec2wave_t, [848](#)
- range
 - Vector and matrix processing toolbox, [41](#)
- range_var_t
 - MHAParser::range_var_t, [626](#)
- ratio
 - ds_t, [254](#)
 - us_t, [866](#)
- raw_p_max_name
 - acTransform_wave, [156](#)

- acTransform_wave_config, 157
- raw_p_name
 - acPooling_wave_config, 142
 - acTransform_wave, 156
 - acTransform_wave_config, 157
- rb_f_t
 - mha_ruby.cpp, 933
- rdata
 - MHASignal::matrix_t, 721
 - mha_audio_t, 372
- re
 - mha_complex_t, 374
- read
 - MHAFilter::blockprocessing_polyphase↔_resampling_t, 459
 - MHAFilter::polyphase_resampling_t, 502
 - MHAJack::port_t, 536
 - mha_drifter_fifo_t, 384
 - mha_fifo_lw_t, 391
 - mha_fifo_t, 397
- read_bytes
 - MHA_TCP::Connection, 416
- read_event
 - MHA_TCP::Connection, 417
- read_get_cpu_load
 - MHAIOJack::io_jack_t, 516
- read_get_scheduler
 - MHAIOJack::io_jack_t, 516
- read_get_xruns
 - MHAIOJack::io_jack_t, 516
- read_levels
 - calibrator_t, 196
- read_line
 - MHA_TCP::Connection, 415
- read_modified
 - dc_simple::dc_if_t, 228
- read_ptr
 - mha_fifo_t, 398
- readable_frames
 - MHAFilter::polyphase_resampling_t, 502
- readaccess
 - MHAParser::base_t, 576
- reader_started
 - mha_drifter_fifo_t, 385
- reader_xruns_in_succession
 - mha_drifter_fifo_t, 386
- reader_xruns_since_start
 - mha_drifter_fifo_t, 386
- reader_xruns_total
 - mha_drifter_fifo_t, 386
- real
 - MHASignal::matrix_t, 717–719
- rear_channel
 - adm_rtconfig_t, 170
- rear_channels
 - adm_if_t, 167
 - adm_rtconfig_t, 170
- rec_frames
 - acsave::cfg_t, 147
- received_group
 - mhaserver_t, 693
- receiver
 - MHAEvents::connector_t, 450
- reciprocal
 - Complex arithmetics in the openMHA, 67
- reclen
 - acsave::acsave_t, 145
- recmode
 - acmon::acmon_t, 137
- reconnect_inports
 - MHAIOJack::io_jack_t, 516
- reconnect_outports
 - MHAIOJack::io_jack_t, 516
- record
 - wavrec_t, 873
- rect
 - MHAOvIFilter::ShapeFun, 101
 - MHAWindow, 122
- rect_t
 - MHAWindow::rect_t, 773
- relative
 - MHASignal::loop_wavefragment_t, 710
- release
 - ac2wave_if_t, 126
 - acConcat_wave, 131
 - acPooling_wave, 140
 - acSteer, 151
 - acTransform_wave, 155
 - acmon::acmon_t, 136
 - acsave::acsave_t, 144
 - adm_if_t, 166
 - altplugs_t, 177
 - analysispath_if_t, 183
 - bbcalib_interface_t, 192
 - calibrator_t, 195
 - coherence::cohflt_if_t, 201
 - db_if_t, 213
 - dc_simple::dc_if_t, 227
 - delaysum::delaysum_if_t, 239
 - doasvm_classification, 243
 - doasvm_feature_extraction, 247
 - droptect_t, 251

- ds_t, 254
- example1_t, 267
- example2_t, 270
- example3_t, 274
- example4_t, 277
- fader_wave::fader_wave_if_t, 284
- fftfilterbank::fftfb_interface_t, 289
- frequency_translator_t, 293
- fw_t, 297
- gain::gain_if_t, 302
- identity_t, 308
- io_file_t, 311
- io_lib_t, 316
- io_parser_t, 319
- io_tcp_sound_t, 331
- io_tcp_t, 335
- lpc, 340
- lpc_bl_predictor, 342
- lpc_burglattice, 347
- MHAIOJack::io_jack_t, 515
- MHAJack::client_t, 531
- MHAParser::mhapluginloader_t, 617
- MHAPLugin::plugin_t, 661
- MHAPLugin_Resampling::resampling_if↵_t, 665
- MHAPLugin_Split::splitted_part_t, 685
- mhachain::chain_base_t, 439
- mhachain::plugs_t, 442
- multibandcompressor::interface_t, 779
- nlms_t, 782
- overlapadd::overlapadd_if_t, 791
- PluginLoader::fourway_processor_t, 805
- PluginLoader::mhapluginloader_t, 808
- prediction_error, 813
- route::interface_t, 821
- smoothgains_bridge::overlapadd_if↵_t, 838
- steerbf, 853
- timoSmooth, 864
- us_t, 866
- wavrec_t, 872
- release_
 - iirfilter_t, 309
 - MHAPLugin::plugin_t, 662
 - MHAPLugin_Split::split_t, 680
- release_mutex
 - mha_fifo_posix_threads_t, 393
 - mha_fifo_thread_platform_t, 401
- remove_abandonned
 - mha_rt_fifo_t, 405
- remove_all
 - mha_rt_fifo_t, 405
- remove_all_cfg
 - MHAPLugin::config_t, 658
- remove_item
 - MHAParser::parser_t, 622, 623
- remove_lock
 - mhamain.cpp, 969
- remove_ref
 - algo_comm_t, 173
 - MHAKernel::algo_comm_class_t, 539
- remove_var
 - algo_comm_t, 172
 - MHAKernel::algo_comm_class_t, 539
- repl_list
 - MHAParser::base_t, 576
- repl_list_t
 - MHAParser::base_t, 571
- replace
 - MHAParser::base_t::replace_t, 577
 - MHASignal::loop_wavefragment_t, 710
- replace_
 - cfg_t, 199
- replace_t
 - MHAParser::base_t::replace_t, 577
- resampling
 - MHAFilter::blockprocessing_polyphase↵_resampling_t, 459
- resampling.cpp, 974
- resampling_factors
 - MHAFilter, 92
- resampling_filter_t
 - MHAFilter::resampling_filter_t, 504
- resampling_if_t
 - MHAPLugin_Resampling::resampling_if↵_t, 665
- resampling_t
 - MHAPLugin_Resampling::resampling_t, 666
- reset
 - droptect_t, 252
 - MHA_TCP::Async_Notify, 409
 - MHA_TCP::Wakeup_Event, 434
- reset_state
 - MHAFilter::gammaflt_t, 477
- resize
 - MHAFilter::iir_filter_t, 482
- resolution
 - acTransform_wave_config, 157
- resolve
 - dynamiclib_t, 255
- resolve_and_init

- PluginLoader::mhappluginloader_t, 809
- resolve_checked
 - dynamiclib_t, 255
- result
 - cpuload_t, 212
- resynthesis_gain
 - MHAFilter::gammaflt_t, 478
- ret_size
 - MHAParser::c_ifc_parser_t, 584
- return_imag
 - fftfilterbank::fftfb_interface_t, 289
- return_imag_
 - fftfilterbank::fftfb_plug_t, 291
- return_value
 - MHA_TCP::Thread, 429
- return_wave
 - wave2spec_if_t, 868
- retv
 - MHAParser::c_ifc_parser_t, 584
- rewind
 - MHASignal::loop_wavefragment_t, 712
- rho
 - nlms_t, 783
 - prediction_error, 813
- ringbuffer
 - MHAFilter::polyphase_resampling_t, 503
- ringbuffer_t
 - MHASignal::ringbuffer_t, 726
- rm_parent_on_remove
 - MHAParser::base_t, 575
- rms
 - MHASignal::loop_wavefragment_t, 710
- rms_limit40
 - MHASignal::loop_wavefragment_t, 710
- rmsdb
 - example6_t, 280
- rmslevel
 - calibrator_variables_t, 197
 - dc::dc_t, 219
 - MHASignal::async_rmslevel_t, 695
 - Vector and matrix processing toolbox, 56, 57
- rmslevel.cpp, 974
- rmslevel_if_t, 817
 - name, 818
 - prepare, 818
 - process, 818
 - rmslevel_if_t, 818
- rmslevel_t, 819
 - fftlens, 819
 - insert, 819
 - level, 819
 - level_db, 819
 - peak, 819
 - peak_db, 819
 - process, 819
 - rmslevel_t, 819
- rmslevelmeter
 - transducers.cpp, 984
- root
 - mha_rt_fifo_t, 406
- rotated_i
 - acTransform_wave_config, 157
- rotated_p
 - acTransform_wave_config, 157
- rotated_p_max_name
 - acTransform_wave, 156
- rotated_p_name
 - acTransform_wave, 156
- route, 124
- route.cpp, 975
- route::interface_t, 820
 - algo, 821
 - cfac, 821
 - cfin, 821
 - cfout, 821
 - interface_t, 821
 - patchbay, 821
 - prepare, 821
 - prepared, 821
 - process, 821
 - release, 821
 - route_ac, 821
 - route_out, 821
 - stopped, 821
 - update, 821
- route::process_t, 822
 - process, 822
 - process_t, 822
 - sout, 822
 - sout_ac, 822
 - wout, 822
 - wout_ac, 822
- route_ac
 - route::interface_t, 821
- route_out
 - route::interface_t, 821
- rows
 - acsave::mat4head_t, 147
- rt_nlms_t, 823
 - ~rt_nlms_t, 824
 - ac, 824

- channels, 824
- F, 824
- frames, 824
- fu, 825
- fu_previous, 825
- fufit, 825
- insert, 824
- n_no_update_, 825
- name_d_, 825
- name_e_, 825
- name_u_, 825
- no_iter, 825
- ntaps, 824
- P_Sum, 825
- process, 824
- Pu, 824
- rt_nlms_t, 824
- s_E, 825
- U, 824
- Uflt, 824
- y_previous, 825
- rt_process
 - analysepath_t, 181
- rtcalibrator
 - transducers.cpp, 984
- run
 - MHA_TCP::Thread, 429
 - mhaserver_t, 693
- runtime configuration, 4
- rval
 - MHAParser::expression_t, 591
- s_E_pred_err_delay
 - prediction_error_config, 816
- s_LPC
 - prediction_error_config, 816
- s_U_delay
 - prediction_error_config, 816
- s_U_delayflt
 - prediction_error_config, 816
- s_Usmpl
 - prediction_error_config, 817
- s_Wflt
 - prediction_error_config, 816
- s_Y_delay
 - prediction_error_config, 816
- s_Y_delayflt
 - prediction_error_config, 816
- s_b
 - lpc_bl_predictor_config, 345
 - lpc_burglattice_config, 349
- s_E
 - prediction_error_config, 816
 - rt_nlms_t, 825
- s_f
 - lpc_bl_predictor_config, 345
 - lpc_burglattice_config, 349
- s_file_in
 - io_file_t, 313
- s_in
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_sound_t, 333
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 534
- s_out
 - coherence::cohflt_t, 204
 - combc_t, 208
 - fftfilterbank::fftfb_plug_t, 291
 - io_file_t, 313
 - io_parser_t, 319
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 534
- s_U
 - prediction_error_config, 816
- s_W
 - prediction_error_config, 816
- sInput
 - MHAFilter::fftfilter_t, 467
- SOCKET_ERROR
 - mha_tcp.cpp, 948
- SOCKET
 - MHA_TCP, 87
 - mha_tcp.cpp, 948
- SPP
 - timo_AC, 856
- START_BETA
 - ADM, 78
- STRERROR
 - MHA_TCP, 87
- STRLEN
 - mha_errno.c, 910
- sWeights
 - MHAFilter::fftfilter_t, 467
- safe_div
 - Complex arithmetics in the openMHA, 66
 - mha_signal.cpp, 936
 - mha_signal.hh, 945
- sample
 - lpc_config, 351
- samplerate
 - io_file_t, 312
 - io_tcp_sound_t, 332

- MHAIOPortAudio::io_portaudio_t, 522
- MHAJack::client_t, 533
- samples_AC
 - acConcat_wave, 131
- samplingrate
 - MHAOvFilter::fftfb_t, 552
- save_m
 - acsave::save_var_t, 149
- save_mat4
 - acsave::save_var_t, 148
- save_spec.cpp, 975
- save_spec_t, 826
 - basename, 827
 - prepare, 827
 - process, 827
 - save_spec_t, 826
- save_txt
 - acsave::save_var_t, 148
- save_var_t
 - acsave::save_var_t, 148
- save_vars
 - acmon::acmon_t, 137
- save_wave.cpp, 975
- save_wave_t, 827
 - basename, 828
 - prepare, 828
 - process, 828
 - save_wave_t, 828
- saveas_mat4
 - MHASignal, 119
- sc
 - MHASignal::hilbert_fftw_t, 707
 - spec2wave_t, 848
- scale
 - example5_t, 279
 - MHASignal, 116
 - MHASignal::fft_t, 706
 - MHASignal::spectrum_t, 735
 - MHASignal::waveform_t, 752
- scale_ch
 - example2_t, 271
 - example3_t, 274
 - example4_t, 278
 - plugin_interface_t, 798
- scale_channel
 - MHASignal::spectrum_t, 736
 - MHASignal::waveform_t, 752
- scale_frame
 - MHASignal::waveform_t, 753
- scale_fun_t
 - MHAOvFilter, 98
- scale_var_t
 - MHAOvFilter::scale_var_t, 567
- scafac
 - MHATableLookup::linear_table_t, 759
- scaler_t
 - gain::scaler_t, 303
- scan_plugin
 - pluginbrowser_t, 799
- scan_plugins
 - pluginbrowser_t, 799
- scheduler
 - analysepath_t, 181
 - MHAPugin_Split::posix_threads_t, 677
- schroeder_t
 - MHASignal::schroeder_t, 730
- sec2smp
 - Vector and matrix processing toolbox, 43
- select_plug
 - altplugs_t, 179
- select_source
 - MHAMultiSrc::base_t, 542
- selected_plug
 - altplugs_t, 179
- Server
 - MHA_TCP::Server, 421, 422
- server
 - io_tcp_t, 336
- server_port_open
 - io_tcp_parser_t, 328
- servername
 - MHAIOJack::io_jack_t, 516
- serversocket
 - MHA_TCP::Server, 423
- set
 - Complex arithmetics in the openMHA, 62, 63
 - MHA_TCP::Async_Notify, 409
- set_announce_port
 - mhaserver_t, 693
- set_channelcnt
 - MHAFilter::adapt_filter_t, 456
- set_connected
 - io_tcp_parser_t, 327
- set_entries
 - MHAParser::keyword_list_t, 602
- set_errnos
 - io_tcp_fwcb_t, 321
- set_error
 - mha_fifo_lw_t, 391
- set_fb_pars
 - DynComp::dc_afterburn_t, 258

- set_help
 - MHAParser::base_t, 575
- set_id_string
 - MHAParser::parser_t, 624
- set_index
 - MHAParser::keyword_list_t, 602
- set_input_domain
 - MHAPugin_Split::domain_handler_t, 669
- set_input_portnames
 - MHAJack::client_t, 532
- set_level
 - fader_wave::fader_wave_if_t, 284
- set_level_db
 - MHASignal::loop_wavefragment_t, 712
- set_level_lin
 - MHASignal::loop_wavefragment_t, 712
- set_local_port
 - io_tcp_parser_t, 325
- set_max_angle_ind
 - parser_int_dyn, 796
- set_minabs
 - mha_signal.cpp, 936
 - mha_signal.hh, 945
- set_new_peer
 - io_tcp_parser_t, 327
- set_node_id
 - MHAParser::base_t, 574
- set_output_domain
 - MHAPugin_Split::domain_handler_t, 669
- set_output_portnames
 - MHAJack::client_t, 532
- set_parse_cb
 - MHAParser::c_ifc_parser_t, 583
- set_range
 - MHAParser::kw_t, 605
 - MHAParser::range_var_t, 626
- set_server_port_open
 - io_tcp_parser_t, 326
- set_state
 - MHAFilter::complex_bandpass_t, 461
 - MHAFilter::iir_ord1_real_t, 484
- set_tau
 - MHAFilter::o1flt_lowpass_t, 490
 - MHAFilter::o1flt_maxtrack_t, 492
 - MHAFilter::o1flt_mintrack_t, 494
- set_tau_attack
 - MHAFilter::o1_ar_filter_t, 487
- set_tau_release
 - MHAFilter::o1_ar_filter_t, 487
- set_use_jack_transport
 - MHAJack::client_t, 532
- set_value
 - MHAParser::keyword_list_t, 602
- set_weights
 - MHAFilter::complex_bandpass_t, 461
 - MHAFilter::gammaflt_t, 477
- set_xfun
 - MHATableLookup::xy_table_t, 763
- set_xmax
 - MHATableLookup::linear_table_t, 758
- set_xmin
 - MHATableLookup::linear_table_t, 757
- set_xyfun
 - MHATableLookup::xy_table_t, 764
- set_yfun
 - MHATableLookup::xy_table_t, 764
- setchannels
 - dc::wideband_inhib_vars_t, 225
- setlock
 - io_file_t, 311
 - MHAParser::variable_t, 633
- sf
 - MHASndFile::sf_t, 754
 - wavwriter_t, 874
- sf_in
 - io_file_t, 313
- sf_out
 - io_file_t, 313
- sf_t
 - MHASndFile::sf_t, 754
- sf_wave_t
 - MHASndFile::sf_wave_t, 755
- sfinf_in
 - io_file_t, 313
- sfinf_out
 - io_file_t, 313
- shadowfilter_begin, 124
- shadowfilter_begin.cpp, 975
- shadowfilter_begin::cfg_t, 828
 - cfg_t, 829
 - in_spec_copy, 829
 - nch, 829
 - ntracks, 829
 - out_spec, 829
 - process, 829
- shadowfilter_begin::shadowfilter_begin_t, 830
 - basename, 831
 - nch, 831
 - ntracks, 831
 - prepare, 831
 - process, 831
 - shadowfilter_begin_t, 830

- shadowfilter_begin_t
 - shadowfilter_begin::shadowfilter_begin←_t, 830
- shadowfilter_end, 124
- shadowfilter_end.cpp, 975
- shadowfilter_end::cfg_t, 831
 - ac, 832
 - cfg_t, 832
 - gains, 832
 - in_spec, 832
 - name, 832
 - nch_out, 832
 - nfft, 832
 - ntracks, 832
 - out_spec, 832
 - process, 832
- shadowfilter_end::shadowfilter_end_t, 833
 - basename, 834
 - prepare, 834
 - process, 834
 - shadowfilter_end_t, 833
- shadowfilter_end_t
 - shadowfilter_end::shadowfilter_end←_t, 833
- shape
 - MHAOvIFilter::fftfb_t, 552
- shapes
 - MHAOvIFilter::fftfb_vars_t, 555
- shift
 - lpc, 340
 - lpc_config, 350
- shifted
 - hilbert_shifter_t, 306
- side
 - mha_channel_info_t, 373
- sign_t
 - MHASignal::schroeder_t, 730
- signal_counter
 - MHASignal, 120
- signal_out
 - MHAPugin_Split::split_t, 681
- sin125
 - speechnoise_t, 850
- sin1k
 - speechnoise_t, 850
- sin250
 - speechnoise_t, 850
- sin2k
 - speechnoise_t, 850
- sin4k
 - speechnoise_t, 850
- sin500
 - speechnoise_t, 850
- sin8k
 - speechnoise_t, 850
- sinc
 - MHAFilter, 92
- sine.cpp, 976
- sine_cfg_t, 834
 - amplitude, 835
 - channels, 835
 - mix, 835
 - phase_increment_div_2pi, 835
 - sine_cfg_t, 835
- sine_t, 835
 - ~sine_t, 836
 - channels, 837
 - frequency, 837
 - lev, 837
 - mode, 837
 - patchbay, 837
 - phase_div_2pi, 837
 - prepare, 836
 - process, 836
 - sine_t, 836
 - update_cfg, 836
- size
 - MHA_AC::ac2matrix_helper_t, 356
 - MHA_AC::acspace2matrix_t, 361
 - MHAKernel::algo_comm_class_t, 540
 - MHASignal::matrix_t, 717
 - Vector and matrix processing toolbox, 47, 48
- size_t
 - MHAParser::keyword_list_t, 601
- Sleep
 - mha_tcp.hh, 950
- slope
 - softclipper_t, 843
 - softclipper_variables_t, 845
- slope_db
 - softclip_t, 842
- smoothgains_bridge, 124
- smoothgains_bridge.cpp, 976
- smoothgains_bridge::overlapadd_if_t, 837
 - ~overlapadd_if_t, 838
 - algo, 839
 - cf_in, 839
 - cf_out, 839
 - epsilon, 839
 - irswnd, 839
 - mode, 839

- overlapadd_if_t, 838
- patchbay, 839
- plugloader, 839
- prepare, 838
- process, 838
- release, 838
- update, 839
- smoothgains_bridge::smoothspec_wrap_↵
t, 839
- proc_1, 840
- proc_2, 840
- smoothspec, 840
- smoothspec_epsilon, 840
- smoothspec_wrap_t, 840
- spec_in_copy, 840
- use_smoothspec, 840
- smoothspec
 - MHAFilter::smoothspec_t, 507
 - smoothgains_bridge::smoothspec_↵
wrap_t, 840
- smoothspec_epsilon
 - smoothgains_bridge::smoothspec_↵
wrap_t, 840
- smoothspec_t
 - MHAFilter::smoothspec_t, 506
- smoothspec_wrap_t
 - smoothgains_bridge::smoothspec_↵
wrap_t, 840
- smp2rad
 - Vector and matrix processing toolbox, 44
- smp2sec
 - Vector and matrix processing toolbox, 43
- smpl
 - prediction_error_config, 817
- sn_in
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 527
- sn_out
 - MHAJack::client_avg_t, 525
 - MHAJack::client_noncont_t, 528
- snprintf_required_length
 - mha_error_helpers, 85
- snrPost1Debug
 - noisePowProposedScale::noisePow_↵
Proposed, 789
- sock_addr
 - MHA_TCP::Server, 423
- Sockaccept_Event
 - MHA_TCP::Sockaccept_Event, 424
- Sockread_Event
 - MHA_TCP::Sockread_Event, 425
- Sockwrite_Event
 - MHA_TCP::Sockwrite_Event, 426
- softclip
 - calibrator_runtime_layer_t, 193
 - calibrator_variables_t, 197
- softclip.cpp, 976
- softclip_t, 841
 - attack, 842
 - decay, 842
 - patchbay, 842
 - prepare, 842
 - process, 842
 - slope_db, 842
 - softclip_t, 842
 - start_limit, 842
 - tfstype, 842
 - update, 842
- softclipper_t, 842
 - attack, 843
 - clipmeter, 843
 - decay, 843
 - hardlimit, 843
 - linear, 843
 - process, 843
 - slope, 843
 - softclipper_t, 843
 - threshold, 843
- softclipper_variables_t, 844
 - clipped, 845
 - hardlimit, 845
 - linear, 845
 - max_clipped, 845
 - slope, 845
 - softclipper_variables_t, 844
 - tau_attack, 844
 - tau_clip, 845
 - tau_decay, 845
 - threshold, 845
- sort_fftw2spec
 - MHASignal::fft_t, 705
- sort_spec2fftw
 - MHASignal::fft_t, 705
- sound
 - io_tcp_t, 336
- source_channel_index
 - MHAFilter::partitioned_convolution_t_↵
::index_t, 499
 - MHAFilter::transfer_function_t, 512
- sout
 - matrixmixer::cfg_t, 352
 - route::process_t, 822

- sout_ac
 - route::process_t, 822
- spec2fir
 - MHAFilter, 92
 - MHAFilter::smoothspec_t, 507
- spec2spec
 - plugindescription_t, 800
- spec2wave
 - MHASignal::fft_t, 704
 - plugindescription_t, 800
- spec2wave.cpp, 976
 - max, 977
 - min, 977
- spec2wave_if_t, 845
 - patchbay, 846
 - prepare, 846
 - process, 846
 - ramplen, 846
 - spec2wave_if_t, 846
 - update, 846
 - window_config, 846
- spec2wave_scale
 - MHASignal::fft_t, 705
- spec2wave_t, 847
 - ~spec2wave_t, 847
 - calc_out, 848
 - ft, 847
 - nfft, 848
 - npad1, 847
 - npad2, 848
 - nwndshift, 848
 - out_buf, 848
 - postwindow, 848
 - process, 847
 - ramps, 848
 - sc, 848
 - spec2wave_t, 847
 - write_buf, 848
- spec_fader_t, 848
 - ~spec_fader_t, 849
 - fr, 849
 - gains, 849
 - nch, 849
 - spec_fader_t, 849
- spec_in
 - MHAPlugin_Split::domain_handler_t, 672
 - overlapadd::overlapadd_t, 794
 - wave2spec_t, 871
- spec_in_copy
 - smoothgains_bridge::smoothspec_←
wrap_t, 840
- spec_out
 - MHAPlugin_Split::domain_handler_t, 672
 - MHAPlugin_Split::split_t, 682
 - timoConfig, 861
- specSteer1
 - acSteer_config, 153
- specSteer2
 - acSteer_config, 153
- spectrum_t
 - MHA_AC::spectrum_t, 366
 - MHAMultiSrc::spectrum_t, 544
 - MHASignal::spectrum_t, 733
- speechnoise
 - calibrator_runtime_layer_t, 194
- speechnoise.cpp, 977
 - bandw_correction, 979
 - erb_hz_f_hz, 978
 - fhz2bandno, 978
 - hz2hz, 978
 - NUM_ENTR_LTASS, 978
 - NUM_ENTR_MHAORIG, 978
 - NUM_ENTR_OLNOISE, 978
 - vLTASS_combined_lev, 979
 - vLTASS_female_lev, 979
 - vLTASS_freq, 979
 - vLTASS_male_lev, 979
 - vMHAOrigFreq, 979
 - vMHAOrigSpec, 979
 - vOlnoiseFreq, 979
 - vOlnoiseLev, 979
- speechnoise.h, 980
- speechnoise_t, 849
 - brown, 850
 - creator, 851
 - LTASS_combined, 850
 - LTASS_female, 850
 - LTASS_male, 850
 - mha, 850
 - noise_type_t, 850
 - olnoise, 850
 - pink, 850
 - sin125, 850
 - sin1k, 850
 - sin250, 850
 - sin2k, 850
 - sin4k, 850
 - sin500, 850
 - sin8k, 850
 - speechnoise_t, 851
 - TEN_SPL_250_8k, 850
 - TEN_SPL_50_16k, 850

- TEN_SPL, 850
- white, 850
- split.cpp, 980
 - default_thread_platform_string, 981
 - default_thread_platform_type, 981
 - MHAPLUGIN_OVERLOAD_OUTDOM←
AIN, 981
 - posixthreads, 981
- split_t
 - MHAPLugin_Split::split_t, 680
- splitted_part_t
 - MHAPLugin_Split::splitted_part_t, 684
- spnoise_channels
 - calibrator_variables_t, 197
- spnoise_level
 - calibrator_variables_t, 197
- spnoise_mode
 - calibrator_variables_t, 197
- spnoise_parser
 - calibrator_variables_t, 197
- spp
 - timoSmooth, 865
- srate
 - adm_if_t, 167
 - calibrator_variables_t, 197
 - MHAParser::mhaconfig_mon_t, 616
 - MHAPLugin_Resampling::resampling_if←
_t, 665
 - mhaconfig_t, 445
- srate_
 - MHAFilter::gammaflt_t, 478
- srcfile
 - MHAParser::parser_t, 624
- srcline
 - MHAParser::parser_t, 625
- start
 - fw_t, 296
 - io_file_t, 311
 - io_lib_t, 315
 - io_parser_t, 318
 - io_tcp_fwcb_t, 321
 - io_tcp_t, 335
 - MHAJack::client_t, 531
- start_event
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_fwcb_t, 322
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 534
- start_handle
 - io_file_t, 312
- io_parser_t, 319
- io_tcp_fwcb_t, 323
- MHAIOPortAudio::io_portaudio_t, 522
- MHAJack::client_t, 534
- start_limit
 - softclip_t, 842
- start_lin
 - cfg_t, 199
- start_new_session
 - wavrec_t, 872
- started
 - fw_t, 297
 - io_parser_t, 319
- starting
 - mha_drifter_fifo_t, 385
- startsample
 - io_file_t, 312
- startup_zeros
 - mha_drifter_fifo_t, 387
- stat_t
 - MHA_AC::stat_t, 368
 - MHASignal::stat_t, 737
- state
 - fw_t, 299
 - MHA_TCP::Thread, 429
 - MHAFilter::filter_t, 475
- state_cpuload
 - MHAIOJack::io_jack_t, 517
- state_parser
 - MHAIOJack::io_jack_t, 517
- state_priority
 - MHAIOJack::io_jack_t, 517
- state_scheduler
 - MHAIOJack::io_jack_t, 517
- state_t
 - fw_t, 296
- state_xruns
 - MHAIOJack::io_jack_t, 517
- staticgain
 - coherence::cohflt_t, 204
 - coherence::vars_t, 205
- status
 - MHA_TCP::Wakeup_Event, 435
- std
 - MHA_AC::stat_t, 368
- std_vector_float
 - Vector and matrix processing toolbox, 52
- std_vector_vector_complex
 - Vector and matrix processing toolbox, 53
- std_vector_vector_float
 - Vector and matrix processing toolbox, 52

- stdcomplex
 - Complex arithmetics in the openMHA, 63
- steerFile
 - acSteer, 152
- steerbf, 851
 - ~steerbf, 852
 - angle_ind, 853
 - angle_src, 853
 - bf_src, 853
 - patchbay, 853
 - prepare, 852
 - process, 852
 - release, 853
 - steerbf, 852
 - update_cfg, 853
- steerbf.cpp, 981
 - INSERT_PATCH, 981
 - PATCH_VAR, 981
- steerbf.h, 981
- steerbf_config, 853
 - _steerbf, 854
 - ~steerbf_config, 854
 - ac, 854
 - bf_src_copy, 854
 - bf_vec, 854
 - nangle, 854
 - nchan, 854
 - nfreq, 854
 - outSpec, 854
 - process, 854
 - steerbf_config, 854
- stime
 - MHA_TCP, 88
- stop
 - fw_t, 296
 - io_file_t, 311
 - io_lib_t, 315
 - io_parser_t, 319
 - io_tcp_fwcb_t, 322
 - io_tcp_t, 335
 - MHAJack::client_t, 531
 - mha_drifter_fifo_t, 385
- stop_event
 - io_file_t, 312
 - io_parser_t, 319
 - io_tcp_fwcb_t, 322
 - MHAIOPortAudio::io_portaudio_t, 522
 - MHAJack::client_t, 534
- stop_handle
 - io_file_t, 312
 - io_parser_t, 319
- io_tcp_fwcb_t, 323
- MHAIOPortAudio::io_portaudio_t, 522
- MHAJack::client_t, 534
- stopped
 - fw_t, 297
 - io_file_t, 311
 - io_parser_t, 319
 - MHAJack::client_t, 533
 - route::interface_t, 821
- store
 - mon_t, 775
- store_frame
 - acsave::cfg_t, 146
 - acsave::save_var_t, 148
- str2val
 - MHAParser::StrCnv, 109, 110
- str2val< mha_real_t >
 - MHAParser::StrCnv, 110
- str_error
 - MHAJack::client_t, 532
- strNames_AC
 - acConcat_wave_config, 132
- strdom
 - analysemhaplugin.cpp, 883
 - latex_doc_t, 337
- strict_channel_match
 - io_file_t, 312
- strict_srate_match
 - io_file_t, 312
- stride
 - comm_var_t, 210
- string_mon_t
 - MHAParser::string_mon_t, 629
- string_t
 - MHAParser::string_t, 631
- strreplace
 - MHAParser, 107
- structVersion
 - MHAIOPortAudio::device_info_t, 519
- subsample_delay_t
 - MHASignal::subsample_delay_t, 738
- subsampledelay_coeff
 - ADM, 78
- sum
 - MHASignal::stat_t, 737
 - MHASignal::waveform_t, 747, 748
- sum2
 - MHASignal::stat_t, 737
- sum_channel
 - MHASignal::waveform_t, 748
- sumsq

- MHASignal::waveform_t, 748
- sumsq_channel
 - Vector and matrix processing toolbox, 58
- sumsq_frame
 - Vector and matrix processing toolbox, 59
- svc
 - analysepath_t, 181
- symmetry_scale
 - MHAOvFilter::fspacing_t, 560
- sync
 - mha_fifo_lw_t, 391
 - mha_fifo_thread_guard_t, 399
- sysread
 - MHA_TCP::Connection, 413
- syswrite
 - MHA_TCP::Connection, 414
- T
 - MHA_TCP::OS_EVENT_TYPE, 420
- t
 - acsave::mat4head_t, 147
 - mha_tictoc_t, 435
- tAC
 - timoConfig, 860
- TEN_SPL_250_8k
 - speechnoise_t, 850
- TEN_SPL_50_16k
 - speechnoise_t, 850
- TEN_SPL
 - speechnoise_t, 850
- table
 - cpupload_t, 212
- table_t
 - MHATableLookup::table_t, 760
- tail
 - MHAFilter::fftfilterbank_t, 471
- target_channel_index
 - MHAFilter::partitioned_convolution_t↔
::index_t, 499
 - MHAFilter::transfer_function_t, 512
- tau
 - coherence::vars_t, 205
 - droptect_t, 252
 - fader_if_t, 282
- tau_attack
 - softclipper_variables_t, 844
- tau_beta
 - adm_if_t, 167
- tau_clip
 - softclipper_variables_t, 845
- tau_decay
 - softclipper_variables_t, 845
- tau_level
 - calibrator_variables_t, 197
- tau_unit
 - coherence::vars_t, 205
- tauattack
 - dc::dc_vars_t, 221
 - dc_simple::dc_vars_t, 233
- taudecay
 - dc::dc_vars_t, 222
 - dc_simple::dc_vars_t, 233
- taugain
 - DynComp::dc_afterburn_vars_t, 261
- taurmslevel
 - dc::dc_vars_t, 221
- tcp_connect_to
 - mha_tcp.cpp, 948
- tcp_connect_to_with_timeout
 - mha_tcp.cpp, 948
- tcpserver
 - mhaserver_t, 694
- termination_request
 - MHAPlugin_Split::posix_threads_t, 677
- test_error
 - io_lib_t, 316
 - MHAParser::c_ifc_parser_t, 583
 - PluginLoader::mhapluginloader_t, 809
- test_fail
 - dc_simple, 81
- test_version
 - PluginLoader::mhapluginloader_t, 809
- testalsadevice.c, 982
 - main, 982
- tftype
 - MHAPlugin::plugin_t, 662
 - softclip_t, 842
- The MHA Framework interface, 26
- The openMHA configuration language, 33
- The openMHA Plugins (programming inter-
face), 6
 - MHAPLUGIN_CALLBACKS_PREFIX, 8
 - MHAPLUGIN_CALLBACKS, 8
 - MHAPLUGIN_DOCUMENTATION, 9
- The openMHA Toolbox library, 34
- thefullname
 - MHAParser::base_t, 576
- thirdoctave_analyzer_t
 - MHAFilter::thirdoctave_analyzer_t, 509
- this_outer_out
 - MHASignal::doublebuffer_t, 703
- thr_f
 - MHA_TCP::Thread, 428

- Thread
 - MHA_TCP::Thread, 428
- thread
 - analysepath_t, 181
 - io_tcp_t, 336
 - MHAPLugin_Split::posix_threads_t, 677
 - MHAPLugin_Split::splitted_part_t, 687
- thread_arg
 - MHA_TCP::Thread, 430
- thread_attr
 - MHA_TCP::Thread, 429
- thread_finish_event
 - MHA_TCP::Thread, 429
- thread_func
 - MHA_TCP::Thread, 429
- thread_handle
 - MHA_TCP::Thread, 429
- thread_platform
 - MHAPLugin_Split::split_t, 681
- thread_platform_t
 - MHAPLugin_Split::thread_platform_t, 688
- thread_start
 - analysispath.cpp, 884
 - MHAPLugin_Split::posix_threads_t, 676
- thread_start_func
 - mha_tcp.cpp, 948
- thread_startup_function
 - MHAIOTCP.cpp, 965
- threshold
 - droptect_t, 252
 - softclipper_t, 843
 - softclipper_variables_t, 845
- tictoc
 - mhachain::plugs_t, 443
- timeout
 - MHA_TCP::OS_EVENT_TYPE, 420
 - MHA_TCP::Timeout_Watcher, 432
- Timeout_Event
 - MHA_TCP::Timeout_Event, 431
- Timeout_Watcher
 - MHA_TCP::Timeout_Watcher, 432
- timeshift
 - Vector and matrix processing toolbox, 49
- timo_AC, 854
 - alpha_frame_AC, 856
 - alpha_hat_AC, 856
 - copy, 855
 - gain_wiener_AC, 856
 - gamma_post_AC, 855
 - insert, 855
 - lambda_ceps_AC, 856
 - lambda_ml_AC, 855
 - lambda_ml_ceps_AC, 855
 - lambda_ml_smooth_AC, 855
 - lambda_spec_AC, 856
 - log_lambda_spec_AC, 856
 - max_q_AC, 856
 - max_val_AC, 856
 - pitch_set_first_AC, 856
 - pitch_set_last_AC, 856
 - SPP, 856
 - timo_AC, 855
 - winF0_AC, 856
 - xi_est_AC, 856
 - xi_ml_AC, 855
- timo_params, 856
 - alpha_const_limits_hz, 858
 - alpha_const_vals, 858
 - alpha_pitch, 858
 - beta_const, 858
 - delta_pitch, 858
 - f0_high, 857
 - f0_low, 857
 - gain_min_db, 858
 - in_cfg, 857
 - kappa_const, 858
 - lambda_thresh, 858
 - noisePow_name, 858
 - prior_q, 858
 - timo_params, 857
 - winF0, 858
 - xi_min_db, 857
 - xi_opt_db, 858
- timoConfig, 858
 - ~timoConfig, 860
 - ac, 860
 - alpha_const, 860
 - alpha_frame, 861
 - alpha_hat, 861
 - alpha_prev, 861
 - copy_AC, 860
 - fftlens, 860
 - GLRexp, 861
 - GLR, 861
 - gain_min, 860
 - gain_wiener, 861
 - gamma_post, 861
 - lambda_ceps, 861
 - lambda_ceps_prev, 861
 - lambda_ml_ceps, 861
 - lambda_ml_full, 861
 - lambda_ml_smooth, 861

- lambda_spec, 861
- log_lambda_spec, 861
- logGLRFact, 861
- max_q, 861
- max_val, 861
- mha_fft, 860
- nchan, 860
- nfreq, 860
- noisePow, 861
- ola_powspec_scale, 860
- params, 860
- pitch_set_first, 861
- pitch_set_last, 861
- powSpec, 861
- priorFact, 861
- process, 860
- q_high, 860
- q_low, 860
- spec_out, 861
- tAC, 860
- timoConfig, 860
- winF0, 860
- xi_est, 861
- xi_min, 860
- xi_ml, 861
- xiOpt, 861
- timoSmooth, 862
 - ~timoSmooth, 863
 - alpha_const_limits_hz, 865
 - alpha_const_vals, 865
 - alpha_pitch, 864
 - beta_const, 864
 - delta_pitch, 864
 - f0_high, 864
 - f0_low, 864
 - gain_min_db, 865
 - kappa_const, 864
 - lambda_thresh, 864
 - noisePow_name, 865
 - on_model_param_valuechanged, 864
 - patchbay, 865
 - prepare, 864
 - prepared, 865
 - prior_q, 865
 - process, 863
 - release, 864
 - spp, 865
 - timoSmooth, 863
 - update_cfg, 864
 - win_f0, 865
 - xi_min_db, 864
 - xi_opt_db, 865
- timoSmooth.cpp, 983
 - INSERT_PATCH, 983
 - INSERT_VAR, 983
 - PATCH_VAR, 983
- timoconfig.cpp, 982
 - CHANLOOP, 982
 - EPSILON, 982
 - LPSCALE, 982
 - OVERLAP_FACTOR, 982
 - POWSPEC_FACTOR, 982
- timoconfig.h, 982
- timosmooth.h, 983
- tmp_spec
 - MHAFilter::smoothspec_t, 508
- tmp_wave
 - MHAFilter::smoothspec_t, 508
- to_from
 - acTransform_wave, 156
 - acTransform_wave_config, 157
- total_read
 - io_file_t, 313
- transducers.cpp, 983
 - kw_index2type, 984
 - rmslevelmeter, 984
 - rtcalibrator, 984
 - vint_0123n1, 984
- transfer_function_t
 - MHAFilter::transfer_function_t, 511
- trigger_processing
 - MHAPugin_Split::split_t, 680
 - MHAPugin_Split::splitted_part_t, 686
- trim
 - MHAParser, 106
- try_accept
 - MHA_TCP::Server, 423
- try_write
 - MHA_TCP::Connection, 416
- tv1
 - mha_tictoc_t, 435
- tv2
 - mha_tictoc_t, 435
- tz
 - mha_tictoc_t, 435
- U
 - rt_nlms_t, 824
- UCL
 - AuditoryProfile::parser_t::ear_t, 187
 - AuditoryProfile::profile_t::ear_t, 190
- UPrew
 - prediction_error_config, 817

- UPrewW
 - prediction_error_config, 817
- UbufferPrew
 - prediction_error_config, 816
- Uflt
 - rt_nlms_t, 824
- uint_vector_t
 - MHASignal::uint_vector_t, 741
- underflow
 - MHAFilter::polyphase_resampling_t, 503
- unit
 - MHAOvFilter::fscale_t, 558
- unit2hz
 - MHAOvFilter::scale_var_t, 567
- unlock_channels
 - fw_vars_t, 300
- unlock_srate_fragmentsize
 - fw_vars_t, 300
- unset_fb_pars
 - DynComp::dc_afterburn_t, 258
- up
 - MHASignal::schroeder_t, 730
- up_incl
 - MHAParser::range_var_t, 627
- up_limit
 - MHAParser::range_var_t, 627
 - MHASignal::quantizer_t, 724
- up_thresh
 - acPooling_wave_config, 142
- update
 - ac2wave_if_t, 126
 - adm_if_t, 166
 - calibrator_t, 196
 - coherence::cohflt_if_t, 201
 - dc::dc_if_t, 217
 - dc::wideband_inhib_vars_t, 225
 - delay::interface_t, 237
 - DynComp::dc_afterburn_t, 259
 - DynComp::gaintable_t, 264
 - frequency_translator_t, 293
 - MHA_AC::ac2matrix_t, 357
 - MHA_AC::acspace2matrix_t, 361
 - MHA_AC::stat_t, 368
 - MHAMultiSrc::spectrum_t, 545
 - MHAMultiSrc::waveform_t, 546
 - MHAParser::mhaconfig_mon_t, 615
 - MHAPlugin_Split::split_t, 680
 - mhachain::chain_base_t, 439
 - nlms_t, 783
 - overlapadd::overlapadd_if_t, 791
 - route::interface_t, 821
 - smoothgains_bridge::overlapadd_if_t, 839
 - softclip_t, 842
 - spec2wave_if_t, 846
 - wave2spec_if_t, 868
- update_bbgain
 - gain::gain_if_t, 302
- update_burner
 - DynComp::dc_afterburn_t, 258
- update_cfg
 - acConcat_wave, 131
 - acPooling_wave, 140
 - acSteer, 152
 - acTransform_wave, 156
 - delaysum::delaysum_if_t, 239
 - doasvm_classification, 244
 - doasvm_feature_extraction, 248
 - example6_t, 280
 - fader_if_t, 282
 - fftfilterbank::fftfb_interface_t, 289
 - lpc, 340
 - lpc_bl_predictor, 343
 - lpc_burglattice, 347
 - multibandcompressor::interface_t, 779
 - noise_t, 785
 - noisePowProposedScale::interface_t, 787
 - plugin_interface_t, 798
 - prediction_error, 813
 - sine_t, 836
 - steerbf, 853
 - timoSmooth, 864
- update_coeffs
 - MHAFilter::fftfilter_t, 465
 - MHAFilter::fftfilterbank_t, 469
- update_dc
 - dc_simple::dc_if_t, 228
- update_filter
 - MHAFilter::iir_filter_t, 482
- update_frame
 - fader_wave::level_adapt_t, 286
- update_gain
 - gain::gain_if_t, 302
- update_gain_mon
 - dc_simple::dc_if_t, 228
- update_hz
 - MHAOvFilter::fscale_bw_t, 556
 - MHAOvFilter::fscale_t, 558
- update_level
 - dc_simple::dc_if_t, 228
- update_level_mon
 - dc_simple::dc_if_t, 228

- update_levels
 - multibandcompressor::plugin_signals_t, 780
- update_m
 - matrixmixer::matmix_t, 354
- update_minmax
 - gain::gain_if_t, 302
- update_monitors
 - dc::dc_if_t, 217
- update_mu
 - MHAFilter::adapt_filter_t, 456
- update_ntaps
 - MHAFilter::adapt_filter_t, 456
- update_parser
 - windowselector_t, 877
- update_proc_load
 - mhachain::plugs_t, 442
- update_ramplen
 - altplugs_t, 178
- update_recmode
 - acmon::acmon_t, 137
- update_selector_list
 - altplugs_t, 178
- update_tau_level
 - calibrator_t, 196
- updated
 - windowselector_t, 877
- updater
 - MHAOvFilter::fscale_bw_t, 556
 - MHAOvFilter::fscale_t, 558
- upper_threshold
 - acPooling_wave, 140
- upsample.cpp, 984
- upsampling_factor
 - MHAFilter::polyphase_resampling_t, 503
- upscale
 - MHASignal::quantizer_t, 724
- us_t, 865
 - antialias, 866
 - prepare, 866
 - process, 866
 - ratio, 866
 - release, 866
 - us_t, 866
- use_date
 - wavrec_t, 873
- use_frozen_
 - cfg_t, 199
- use_jack_transport
 - MHAJack::client_t, 534
- use_mat
 - acmon::ac_monitor_t, 135
- use_own_ac
 - altplugs_t, 179
- use_sine
 - cpuload_t, 212
- use_smoothspec
 - smoothgains_bridge::smoothspec_↔, wrap_t, 840
- use_wbinhib
 - dc::dc_vars_t, 222
- user
 - MHAParser::window_t, 652
- user_err_msg
 - MHAIOFile.cpp, 954
 - MHAIOJack.cpp, 957
 - MHAIOParser.cpp, 959
 - MHAIOPortAudio.cpp, 962
 - MHAITCP.cpp, 966
- user_t
 - MHAWindow::user_t, 774
- username
 - MHA_AC::ac2matrix_helper_t, 356
- userwnd
 - windowselector_t, 877
- v_G
 - prediction_error_config, 816
- vFlog
 - DynComp::gaintable_t, 266
- vGCC_ac
 - doasvm_feature_extraction_config, 249
- vGCC_con
 - acConcat_wave_config, 132
- vGCC_name
 - doasvm_classification, 244
 - doasvm_feature_extraction, 248
- vGCC
 - acConcat_wave_config, 132
 - doasvm_feature_extraction_config, 249
- vLTASS_combined_lev
 - speechnoise.cpp, 979
- vLTASS_female_lev
 - speechnoise.cpp, 979
- vLTASS_freq
 - speechnoise.cpp, 979
- vLTASS_male_lev
 - speechnoise.cpp, 979
- vMHAOrigFreq
 - speechnoise.cpp, 979
- vMHAOrigSpec
 - speechnoise.cpp, 979
- vOlnoiseFreq

- speechnoise.cpp, 979
- vOInoiseLev
 - speechnoise.cpp, 979
- val2str
 - MHAParser::StrCnv, 110, 111
- validate
 - AuditoryProfile::parser_t::fmap_t, 188
 - MHAParser::keyword_list_t, 602
 - MHAParser::kw_t, 605
 - MHAParser::range_var_t, 627
- validator_channels
 - mhasndfile.cpp, 970
- validator_length
 - mhasndfile.cpp, 970
- value
 - AuditoryProfile::parser_t::fmap_t, 188
 - MHASignal::ringbuffer_t, 726
 - MHASignal::spectrum_t, 734
 - MHASignal::waveform_t, 746, 747
 - mha_signal.hh, 945
 - Vector and matrix processing toolbox, 49–52
- value_type
 - mha_dblbuf_t, 376
 - mha_fifo_t, 396
- valuechanged
 - MHAParser::base_t, 575
- variable, 4
- variable_t
 - MHAParser::variable_t, 633
- variables, 4
 - acsave::acsave_t, 145
- varlist
 - acmon::acmon_t, 137
 - acsave::acsave_t, 145
 - acsave::cfg_t, 147
- varlist_t
 - acsave::acsave_t, 144
- vars
 - acmon::acmon_t, 137
 - analysispath_if_t, 183
 - calibrator_t, 196
 - coherence::cohflt_if_t, 201
 - MHAKernel::algo_comm_class_t, 540
- vars_t
 - coherence::vars_t, 205
 - MHAOvFilter::overlap_save_filterbank_↔t::vars_t, 565
- vbark
 - MHAOvFilter::barkscale, 98
- vbin1
 - MHAOvFilter::fftfb_t, 552
- vbin2
 - MHAOvFilter::fftfb_t, 552
- vcomplex_mon_t
 - MHAParser::vcomplex_mon_t, 635
- vcomplex_t
 - MHAParser::vcomplex_t, 637
- vec_y
 - MHATableLookup::linear_table_t, 759
- Vector and matrix processing toolbox, 36
 - assign, 48, 49
 - bin2freq, 44
 - channels, 41
 - clear, 48
 - colored_intensity, 56
 - conjugate, 59
 - copy_channel, 55
 - db2lin, 42
 - dbspl2pa, 43
 - dupvec, 45
 - dupvec_chk, 46
 - equal_dim, 46, 47
 - for_each, 42
 - freq2bin, 44
 - integrate, 47
 - lin2db, 42
 - max, 58
 - maxabs, 56, 57
 - mha_real_t, 41
 - min, 58
 - operator*=, 53, 54
 - operator^=, 55
 - operator+=, 53, 54
 - operator-=, 53
 - operator/=, 54
 - pa22dbspl, 43
 - pa2dbspl, 42
 - rad2smp, 45
 - range, 41
 - rmslevel, 56, 57
 - sec2smp, 43
 - size, 47, 48
 - smp2rad, 44
 - smp2sec, 43
 - std_vector_float, 52
 - std_vector_vector_complex, 53
 - std_vector_vector_float, 52
 - sumsq_channel, 58
 - sumsq_frame, 59
 - timeshift, 49
 - value, 49–52

- vF
 - DynComp::gaintable_t, 266
- vfloat_mon_t
 - MHAParser::vfloat_mon_t, 639
- vfloat_t
 - MHAParser::vfloat_t, 641
- vfreq
 - MHAOvFilter::barkscale, 98
- vint_0123n1
 - transducers.cpp, 984
- vint_mon_t
 - MHAParser::vint_mon_t, 643
- vint_t
 - MHAParser::vint_t, 645
- vL
 - DynComp::gaintable_t, 265
- vmax
 - gain::gain_if_t, 302
- vmin
 - gain::gain_if_t, 302
- vstring_mon_t
 - MHAParser::vstring_mon_t, 647
- vstring_t
 - MHAParser::vstring_t, 649
- vy
 - MHATableLookup::linear_table_t, 759
- W
 - MHA_TCP::OS_EVENT_TYPE, 420
 - MHAFilter::adapt_filter_state_t, 455
- w
 - ac2wave_t, 127
 - doasvm_classification, 244
 - MHAOvFilter::fftfb_t, 552
- w_out
 - combc_t, 208
- WINAPI
 - mha_plugin.hh, 931
- wIRS_fft
 - MHAFilter::fftfiler_t, 467
- wInput
 - MHAFilter::fftfiler_t, 467
- wInput_fft
 - MHAFilter::fftfiler_t, 466
- wOutput
 - MHAFilter::fftfiler_t, 467
- wOutput_fft
 - MHAFilter::fftfiler_t, 467
- wait
 - MHA_TCP::Event_Watcher, 419
- wait_for_decrease
 - mha_fifo_posix_threads_t, 393
- mha_fifo_thread_platform_t, 401
- wait_for_increase
 - mha_fifo_posix_threads_t, 393
 - mha_fifo_thread_platform_t, 401
- Wakeup_Event
 - MHA_TCP::Wakeup_Event, 434
- wave2spec
 - MHASignal::fft_t, 704
 - plugindescription_t, 800
- wave2spec.cpp, 984
 - MHAPLUGIN_OVERLOAD_OUTDOM←
AIN, 984
- wave2spec_if_t, 867
 - algo, 868
 - nfft, 868
 - nwnd, 868
 - patchbay, 868
 - prepare, 868
 - process, 868
 - return_wave, 868
 - update, 868
 - wave2spec_if_t, 868
 - window_config, 868
 - wndpos, 868
- wave2spec_scale
 - MHASignal::fft_t, 705
- wave2spec_t, 869
 - ~wave2spec_t, 870
 - calc_in, 871
 - calc_pre_wnd, 870
 - ft, 870
 - in_buf, 871
 - npad1, 870
 - npad2, 870
 - nwnd, 870
 - nwndshift, 870
 - process, 870
 - spec_in, 871
 - wave2spec_t, 870
 - window, 871
- wave2wave
 - plugindescription_t, 800
- wave_fifo
 - analysepath_t, 181
- wave_in
 - MHAPLugin_Split::domain_handler_t, 672
- wave_in1
 - overlapadd::overlapadd_t, 793
- wave_out
 - MHAPLugin_Split::domain_handler_t, 672
 - MHAPLugin_Split::split_t, 682

- wave_out1
 - overlapadd::overlapadd_t, 794
- waveform_t
 - MHA_AC::waveform_t, 369
 - MHAMultiSrc::waveform_t, 546
 - MHASignal::waveform_t, 745
- wavrec.cpp, 984
 - DEBUG, 985
- wavrec_t, 871
 - fifolen, 873
 - minwrite, 873
 - patchbay, 873
 - prefix, 873
 - prepare, 872
 - process, 872
 - record, 873
 - release, 872
 - start_new_session, 872
 - use_date, 873
 - wavrec_t, 872
- wavwriter_t, 873
 - ~wavwriter_t, 874
 - act_, 874
 - cf_, 874
 - close_session, 874
 - data, 874
 - fifo, 874
 - minw_, 874
 - process, 874
 - sf, 874
 - wavwriter_t, 874
 - write_thread, 874
 - writethread, 874
- wb_inhib_cfg_t
 - dc::wb_inhib_cfg_t, 224
- wbinhib
 - dc::dc_if_t, 217
- weights
 - dc::wb_inhib_cfg_t, 224
 - dc::wideband_inhib_vars_t, 225
 - delaysum::delaysum_if_t, 239
 - delaysum::delaysum_t, 241
- what
 - MHA_Error, 388
- white
 - speechnoise_t, 850
- wideband_inhib_vars_t
 - dc::wideband_inhib_vars_t, 225
- win_f0
 - timoSmooth, 865
- winF0
 - timo_params, 858
 - timoConfig, 860
- winF0_AC
 - timo_AC, 856
- window
 - MHAFilter::smoothspec_t, 508
 - overlapadd::overlapadd_if_t, 792
 - wave2spec_t, 871
- window_config
 - spec2wave_if_t, 846
 - wave2spec_if_t, 868
- window_t
 - MHAParser::window_t, 651
- windowselector.cpp, 985
- windowselector.h, 985
- windowselector_t, 875
 - ~windowselector_t, 876
 - get_window_data, 876
 - insert_items, 876
 - invalidate_window_data, 877
 - patchbay, 877
 - update_parser, 877
 - updated, 877
 - userwnd, 877
 - windowselector_t, 876
 - wnd, 877
 - wndexp, 877
 - wndtype, 877
- wnd
 - fader_wave::level_adapt_t, 286
 - windowselector_t, 877
- wnd_bartlett
 - MHAParser::window_t, 651
- wnd_blackman
 - MHAParser::window_t, 651
- wnd_funs
 - mha_windowparser.cpp, 950
- wnd_hamming
 - MHAParser::window_t, 651
- wnd_hann
 - MHAParser::window_t, 651
- wnd_rect
 - MHAParser::window_t, 651
- wnd_user
 - MHAParser::window_t, 651
- wndexp
 - overlapadd::overlapadd_if_t, 792
 - windowselector_t, 877
- wndlen
 - doasvm_feature_extraction_config, 249
 - MHAParser::mhaconfig_mon_t, 615

- mhaconfig_t, 445
- wndpos
 - overlapadd::overlapadd_if_t, 792
 - wave2spec_if_t, 868
- wndtype
 - windowselector_t, 877
- worker_thread_priority
 - MHAPlugin_Split::split_t, 681
- worker_thread_scheduler
 - MHAPlugin_Split::split_t, 681
- wout
 - matrixmixer::cfg_t, 352
 - route::process_t, 822
- wout_ac
 - route::process_t, 822
- write
 - MHA_TCP::Connection, 416
 - MHAFilter::blockprocessing_polyphase←
_resampling_t, 458
 - MHAFilter::polyphase_resampling_t, 502
 - MHAJack::port_t, 536
 - MHASignal::matrix_t, 720
 - MHASignal::ringbuffer_t, 727
 - MHASignal::uint_vector_t, 742
 - mha_drifter_fifo_t, 383
 - mha_fifo_lw_t, 390
 - mha_fifo_t, 396
- write_buf
 - overlapadd::overlapadd_t, 794
 - spec2wave_t, 848
- write_event
 - MHA_TCP::Connection, 417
- write_float
 - mha_parser.cpp, 925
- write_ptr
 - mha_fifo_t, 398
- write_thread
 - wavwriter_t, 874
- write_wave
 - mhasndfile.cpp, 970
 - mhasndfile.h, 971
- writeaccess
 - MHAParser::base_t, 575
- writer_started
 - mha_drifter_fifo_t, 385
- writer_xruns_in_succession
 - mha_drifter_fifo_t, 386
- writer_xruns_since_start
 - mha_drifter_fifo_t, 386
- writer_xruns_total
 - mha_drifter_fifo_t, 386
- writethread
 - wavwriter_t, 874
- Writing openMHA Plugins. A step-by-step tutorial, 10
- wtype
 - MHAParser::window_t, 652
- wtype_t
 - MHAParser::window_t, 651
- X
 - MHA_TCP::OS_EVENT_TYPE, 420
 - MHAFilter::adapt_filter_state_t, 455
- x
 - doasvm_classification, 244
- xfun
 - MHATableLookup::xy_table_t, 764
- xi_est
 - timoConfig, 861
- xi_est_AC
 - timo_AC, 856
- xi_min
 - timoConfig, 860
- xi_min_db
 - timo_params, 857
 - timoSmooth, 864
- xi_ml
 - timoConfig, 861
- xi_ml_AC
 - timo_AC, 855
- xi_opt_db
 - timo_params, 858
 - timoSmooth, 865
- xiOpt
 - noisePowProposedScale::noisePow←
Proposed, 789
 - timoConfig, 861
- xiOptDb
 - noisePowProposedScale::interface_t, 787
- xmax
 - MHATableLookup::linear_table_t, 759
- xmin
 - MHATableLookup::linear_table_t, 759
- Xs
 - MHAFilter::fftfilterbank_t, 470
- xw
 - MHAFilter::fftfilterbank_t, 470
- xy_table_t
 - MHATableLookup::xy_table_t, 762
- xyfun
 - MHATableLookup::xy_table_t, 764
- y

- doasvm_classification, [244](#)
- y0
 - dc_simple::dc_t::line_t, [231](#)
- y_previous
 - rt_nlms_t, [825](#)
- YPrew
 - prediction_error_config, [817](#)
- yfun
 - MHATableLookup::xy_table_t, [764](#)
- Yn
 - MHAFilter::complex_bandpass_t, [462](#)
 - MHAFilter::iir_ord1_real_t, [485](#)
- Ys
 - MHAFilter::fftfilterbank_t, [471](#)
- yw
 - MHAFilter::fftfilterbank_t, [470](#)
- yw_temp
 - MHAFilter::fftfilterbank_t, [471](#)
- zeros
 - ac2wave_if_t, [126](#)
- zerowindow
 - overlapadd::overlapadd_if_t, [792](#)