

The Open Master Hearing Aid (openMHA)

4.15.0

Plugin Developers' Manual



© 2005-2021 by HörTech gGmbH, Marie-Curie-Str. 2, D-26129 Oldenburg, Germany

The Open Master Hearing Aid (openMHA) – Plugin Developers' Manual
HörTech gGmbH
Marie-Curie-Str. 2
D-26129 Oldenburg

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 2019 2020 2021 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 Module Documentation	4
2.1 Concept of Variables and Data Exchange in the openMHA	4
2.2 Writing openMHA Plugins. A step-by-step tutorial	6
2.3 The MHA Framework interface	22
2.4 Communication between algorithms	23
2.5 Error handling in the openMHA	28
2.6 The openMHA configuration language	30
2.7 The openMHA Toolbox library	31
2.8 Vector and matrix processing toolbox	33
2.9 Complex arithmetics in the openMHA	58
2.10 Fast Fourier Transform functions	70
3 Namespace Documentation	78
3.1 ac2lsl Namespace Reference	78
3.2 ac_proc Namespace Reference	78
3.3 acmon Namespace Reference	79
3.4 acsave Namespace Reference	79
3.5 addsndfile Namespace Reference	79
3.6 ADM Namespace Reference	81
3.7 audiometerbackend Namespace Reference	82
3.8 AuditoryProfile Namespace Reference	84
3.9 coherence Namespace Reference	84
3.10 cpupload Namespace Reference	85
3.11 dbasync_native Namespace Reference	85
3.12 dc Namespace Reference	86
3.13 dc_simple Namespace Reference	86
3.14 delay Namespace Reference	89
3.15 delaysum Namespace Reference	89
3.16 delaysum_spec Namespace Reference	89
3.17 double2acvar Namespace Reference	90
3.18 DynComp Namespace Reference	90
3.19 equalize Namespace Reference	91
3.20 fader_wave Namespace Reference	92
3.21 fftfbpow Namespace Reference	92
3.22 fftfilter Namespace Reference	92
3.23 fftfilterbank Namespace Reference	94
3.24 fshift Namespace Reference	94
3.25 fshift_hilbert Namespace Reference	95
3.26 gain Namespace Reference	95
3.27 gsc_adaptive_stage Namespace Reference	95
3.28 gtfb_analyzer Namespace Reference	96
3.29 level_matching Namespace Reference	96
3.30 Isl2ac Namespace Reference	96
3.31 matlab_wrapper Namespace Reference	97
3.32 matrixmixer Namespace Reference	98
3.33 mconv Namespace Reference	98

3.34	MHA_AC Namespace Reference	98
3.35	mha_error_helpers Namespace Reference	99
3.36	MHA_TCP Namespace Reference	100
3.37	mha_tcp Namespace Reference	103
3.38	mhachain Namespace Reference	104
3.39	MHAEvents Namespace Reference	104
3.40	MHAFilter Namespace Reference	104
3.41	MHAIOJack Namespace Reference	109
3.42	MHAIOJackdb Namespace Reference	110
3.43	MHAIOPortAudio Namespace Reference	110
3.44	mhaioutils Namespace Reference	111
3.45	MHAJack Namespace Reference	111
3.46	MHAKernel Namespace Reference	114
3.47	MHAMultiSrc Namespace Reference	115
3.48	MHAOvFilter Namespace Reference	115
3.49	MHAOvFilter::barkscale Namespace Reference	116
3.50	MHAOvFilter::FreqScaleFun Namespace Reference	117
3.51	MHAOvFilter::ShapeFun Namespace Reference	120
3.52	MHAParser Namespace Reference	122
3.53	MHAParser::StrCnv Namespace Reference	128
3.54	MHAPlugin Namespace Reference	134
3.55	MHAPlugin_Resampling Namespace Reference	134
3.56	MHAPlugin_Split Namespace Reference	135
3.57	MHASignal Namespace Reference	136
3.58	MHASndFile Namespace Reference	150
3.59	MHATableLookup Namespace Reference	150
3.60	MHAUtils Namespace Reference	150
3.61	MHAWindow Namespace Reference	153
3.62	multibandcompressor Namespace Reference	155
3.63	noise_psd_estimator Namespace Reference	155
3.64	overlapadd Namespace Reference	156
3.65	plingploing Namespace Reference	156
3.66	PluginLoader Namespace Reference	157
3.67	plugins Namespace Reference	158
3.68	plugins::hoertech Namespace Reference	158
3.69	plugins::hoertech::acrec Namespace Reference	158
3.70	rmslevel Namespace Reference	158
3.71	rohBeam Namespace Reference	159
3.72	route Namespace Reference	160
3.73	shadowfilter_begin Namespace Reference	160
3.74	shadowfilter_end Namespace Reference	161
3.75	smooth_cepstrum Namespace Reference	161
3.76	smoothgains_bridge Namespace Reference	161
3.77	testplugin Namespace Reference	161
3.78	windnoise Namespace Reference	161
4	Class Documentation	162
4.1	ac2lsl::ac2lsl_t Class Reference	162
4.2	ac2lsl::cfg_t Class Reference	166
4.3	ac2lsl::save_var_base_t Class Reference	169
4.4	ac2lsl::save_var_t< T > Class Template Reference	171
4.5	ac2lsl::save_var_t< mha_complex_t > Class Reference	175

4.6	ac2isl::type_info Struct Reference	179
4.7	ac2osc_t Class Reference	180
4.8	ac2wave_if_t Class Reference	185
4.9	ac2wave_t Class Reference	188
4.10	ac_mul_t Class Reference	191
4.11	ac_proc::interface_t Class Reference	196
4.12	acConcat_wave Class Reference	199
4.13	acConcat_wave_config Class Reference	202
4.14	acmon::ac_monitor_t Class Reference	204
4.15	acmon::acmon_t Class Reference	207
4.16	acPooling_wave Class Reference	211
4.17	acPooling_wave_config Class Reference	215
4.18	acsave::acsave_t Class Reference	219
4.19	acsave::cfg_t Class Reference	223
4.20	acsave::mat4head_t Struct Reference	225
4.21	acsave::save_var_t Class Reference	226
4.22	acSteer Class Reference	229
4.23	acSteer_config Class Reference	232
4.24	acTransform_wave Class Reference	234
4.25	acTransform_wave_config Class Reference	237
4.26	adaptive_feedback_canceller Class Reference	240
4.27	adaptive_feedback_canceller_config Class Reference	244
4.28	addsndfile::addsndfile_if_t Class Reference	250
4.29	addsndfile::level_adapt_t Class Reference	255
4.30	addsndfile::resampled_soundfile_t Class Reference	257
4.31	addsndfile::sndfile_t Class Reference	259
4.32	addsndfile::waveform_proxy_t Class Reference	260
4.33	ADM::ADM< F > Class Template Reference	261
4.34	ADM::Delay< F > Class Template Reference	265
4.35	ADM::Linearphase_FIR< F > Class Template Reference	268
4.36	adm_if_t Class Reference	271
4.37	adm_rtconfig_t Class Reference	275
4.38	algo_comm_t Struct Reference	280
4.39	alsa_base_t Class Reference	286
4.40	alsa_dev_par_parser_t Class Reference	289
4.41	alsa_t< T > Class Template Reference	291
4.42	altconfig_t Class Reference	295
4.43	altpugs_t Class Reference	300
4.44	analysepath_t Class Reference	306
4.45	analysispath_if_t Class Reference	310
4.46	attenuate20_t Class Reference	314
4.47	audiometerbackend::audiometer_if_t Class Reference	315
4.48	audiometerbackend::level_adapt_t Class Reference	318
4.49	audiometerbackend::Inn3rdoct_t Class Reference	321
4.50	audiometerbackend::signal_gen_t Class Reference	323
4.51	audiometerbackend::sine_t Class Reference	324
4.52	AuditoryProfile::fmap_t Class Reference	325
4.53	AuditoryProfile::parser_t Class Reference	326
4.54	AuditoryProfile::parser_t::ear_t Class Reference	328
4.55	AuditoryProfile::parser_t::fmap_t Class Reference	329
4.56	AuditoryProfile::profile_t Class Reference	331

4.57	AuditoryProfile::profile_t::ear_t Class Reference	333
4.58	bbcalib_interface_t Class Reference	334
4.59	calibrator_runtime_layer_t Class Reference	336
4.60	calibrator_t Class Reference	339
4.61	calibrator_variables_t Class Reference	341
4.62	cfg_t Class Reference	344
4.63	coherence::cohflt_if_t Class Reference	348
4.64	coherence::cohflt_t Class Reference	350
4.65	coherence::vars_t Class Reference	354
4.66	combc_if_t Class Reference	356
4.67	combc_t Class Reference	358
4.68	comm_var_t Struct Reference	360
4.69	complex_scale_channel_t Class Reference	362
4.70	cpupload::cpupload_cfg_t Class Reference	364
4.71	cpupload::cpupload_if_t Class Reference	367
4.72	db_if_t Class Reference	370
4.73	db_t Class Reference	372
4.74	dbasync_native::db_if_t Class Reference	374
4.75	dbasync_native::dbasync_t Class Reference	377
4.76	dbasync_native::delay_check_t Class Reference	380
4.77	dc::dc_if_t Class Reference	381
4.78	dc::dc_t Class Reference	383
4.79	dc::dc_vars_t Class Reference	388
4.80	dc::dc_vars_validator_t Class Reference	392
4.81	dc_simple::dc_if_t Class Reference	393
4.82	dc_simple::dc_t Class Reference	397
4.83	dc_simple::dc_t::line_t Class Reference	400
4.84	dc_simple::dc_vars_t Class Reference	402
4.85	dc_simple::dc_vars_validator_t Class Reference	404
4.86	dc_simple::level_smoothen_t Class Reference	406
4.87	delay::interface_t Class Reference	409
4.88	delaysum::delaysum_wave_if_t Class Reference	411
4.89	delaysum::delaysum_wave_t Class Reference	414
4.90	delaysum_spec::delaysum_spec_if_t Class Reference	416
4.91	delaysum_spec::delaysum_t Class Reference	418
4.92	doasvm_classification Class Reference	420
4.93	doasvm_classification_config Class Reference	423
4.94	doasvm_feature_extraction Class Reference	425
4.95	doasvm_feature_extraction_config Class Reference	429
4.96	double2acvar::double2acvar_t Class Reference	432
4.97	dropgen_t Class Reference	436
4.98	droptect_t Class Reference	439
4.99	ds_t Class Reference	443
4.100	dynamiclib_t Class Reference	445
4.101	DynComp::dc_afterburn_rt_t Class Reference	449
4.102	DynComp::dc_afterburn_t Class Reference	452
4.103	DynComp::dc_afterburn_vars_t Class Reference	455
4.104	DynComp::gaintable_t Class Reference	457
4.105	equalize::cfg_t Class Reference	462
4.106	equalize::freqgains_t Class Reference	463
4.107	example1_t Class Reference	466

4.108 example2_t Class Reference	468
4.109 example3_t Class Reference	472
4.110 example4_t Class Reference	476
4.111 example5_t Class Reference	480
4.112 example6_t Class Reference	481
4.113 example7_t Class Reference	484
4.114 expression_t Class Reference	485
4.115 fader_if_t Class Reference	486
4.116 fader_wave::fader_wave_if_t Class Reference	488
4.117 fader_wave::level_adapt_t Class Reference	491
4.118 fftfbpow::fftfbpow_interface_t Class Reference	493
4.119 fftfbpow::fftfbpow_t Class Reference	496
4.120 fftfilter::fftfilter_t Class Reference	498
4.121 fftfilter::interface_t Class Reference	500
4.122 fftfilterbank::fftfb_interface_t Class Reference	503
4.123 fftfilterbank::fftfb_plug_t Class Reference	506
4.124 fshift::fshift_config_t Class Reference	508
4.125 fshift::fshift_t Class Reference	511
4.126 fshift_hilbert::frequency_translator_t Class Reference	514
4.127 fshift_hilbert::hilbert_shifter_t Class Reference	517
4.128 fw_t Class Reference	522
4.129 fw_vars_t Class Reference	531
4.130 gain::gain_if_t Class Reference	532
4.131 gain::scaler_t Class Reference	535
4.132 gsc_adaptive_stage::gsc_adaptive_stage Class Reference	536
4.133 gsc_adaptive_stage::gsc_adaptive_stage_if Class Reference	544
4.134 gtfb_analyzer::gtfb_analyzer_cfg_t Struct Reference	548
4.135 gtfb_analyzer::gtfb_analyzer_t Class Reference	552
4.136 gtfb_simd_cfg_t Class Reference	555
4.137 gtfb_simd_t Class Reference	561
4.138 gtfb_simple_rt_t Class Reference	563
4.139 gtfb_simple_t Class Reference	569
4.140 hanning_ramps_t Class Reference	573
4.141 identity_t Class Reference	575
4.142 iirfilter_t Class Reference	577
4.143 io_alsa_t Class Reference	578
4.144 io_asterisk_fwcb_t Class Reference	583
4.145 io_asterisk_parser_t Class Reference	588
4.146 io_asterisk_sound_t Class Reference	595
4.147 io_asterisk_t Class Reference	599
4.148 io_file_t Class Reference	604
4.149 io_lib_t Class Reference	610
4.150 io_parser_t Class Reference	615
4.151 io_tcp_fwcb_t Class Reference	620
4.152 io_tcp_parser_t Class Reference	624
4.153 io_tcp_sound_t Class Reference	631
4.154 io_tcp_sound_t::float_union Union Reference	636
4.155 io_tcp_t Class Reference	637
4.156 io_wrapper Class Reference	641
4.157 latex_doc_t Class Reference	643
4.158 level_matching::channel_pair Class Reference	647

4.159 level_matching::level_matching_config_t Class Reference	650
4.160 level_matching::level_matching_t Class Reference	653
4.161 levelmeter_t Class Reference	657
4.162 lpc Class Reference	660
4.163 lpc_bl_predictor Class Reference	663
4.164 lpc_bl_predictor_config Class Reference	667
4.165 lpc_burglattice Class Reference	669
4.166 lpc_burglattice_config Class Reference	672
4.167 lpc_config Class Reference	675
4.168 Isl2ac::cfg_t Class Reference	678
4.169 Isl2ac::Isl2ac_t Class Reference	680
4.170 Isl2ac::save_var_t Class Reference	685
4.171 matlab_wrapper::callback Class Reference	692
4.172 matlab_wrapper::matlab_wrapper_rt_cfg_t Class Reference	694
4.173 matlab_wrapper::matlab_wrapper_t Class Reference	696
4.174 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t Class Reference	702
4.175 matlab_wrapper::types< T > Struct Template Reference	710
4.176 matlab_wrapper::types< MHA_SPECTRUM > Struct Reference	710
4.177 matlab_wrapper::types< MHA_WAVEFORM > Struct Reference	711
4.178 matrixmixer::cfg_t Class Reference	712
4.179 matrixmixer::matmix_t Class Reference	713
4.180 mconv::MConv Class Reference	716
4.181 MHA_AC::ac2matrix_helper_t Class Reference	720
4.182 MHA_AC::ac2matrix_t Class Reference	722
4.183 MHA_AC::acspace2matrix_t Class Reference	724
4.184 MHA_AC::double_t Class Reference	728
4.185 MHA_AC::float_t Class Reference	730
4.186 MHA_AC::int_t Class Reference	732
4.187 MHA_AC::spectrum_t Class Reference	734
4.188 MHA_AC::stat_t Class Reference	736
4.189 MHA_AC::waveform_t Class Reference	738
4.190 mha_audio_descriptor_t Struct Reference	740
4.191 mha_audio_t Struct Reference	742
4.192 mha_channel_info_t Struct Reference	743
4.193 mha_complex_t Struct Reference	744
4.194 mha_complex_test_array_t Struct Reference	745
4.195 mha_dbdbuf_t< FIFO > Class Template Reference	746
4.196 mha_direction_t Struct Reference	754
4.197 mha_drifter_fifo_t< T > Class Template Reference	755
4.198 MHA_Error Class Reference	763
4.199 mha_fifo_if_t< T > Class Template Reference	766
4.200 mha_fifo_lw_t< T > Class Template Reference	769
4.201 mha_fifo_posix_threads_t Class Reference	773
4.202 mha_fifo_t< T > Class Template Reference	775
4.203 mha_fifo_thread_guard_t Class Reference	782
4.204 mha_fifo_thread_platform_t Class Reference	783
4.205 mha_real_test_array_t Struct Reference	787
4.206 mha_rt_fifo_element_t< T > Class Template Reference	787
4.207 mha_rt_fifo_t< T > Class Template Reference	789
4.208 mha_spec_t Struct Reference	793
4.209 mha_stash_environment_variable_t Class Reference	795

4.210 MHA_TCP::Async_Notify Class Reference	797
4.211 mha_tcp::buffered_socket_t Class Reference	798
4.212 MHA_TCP::Client Class Reference	800
4.213 MHA_TCP::Connection Class Reference	802
4.214 MHA_TCP::Event_Watcher Class Reference	810
4.215 MHA_TCP::OS_EVENT_TYPE Struct Reference	813
4.216 MHA_TCP::Server Class Reference	814
4.217 mha_tcp::server_t Class Reference	818
4.218 MHA_TCP::sock_init_t Class Reference	824
4.219 MHA_TCP::Sockaccept_Event Class Reference	824
4.220 MHA_TCP::Sockread_Event Class Reference	825
4.221 MHA_TCP::Sockwrite_Event Class Reference	826
4.222 MHA_TCP::Thread Class Reference	827
4.223 MHA_TCP::Timeout_Event Class Reference	832
4.224 MHA_TCP::Timeout_Watcher Class Reference	833
4.225 MHA_TCP::Wakeups_Event Class Reference	835
4.226 mha_tictoc_t Struct Reference	838
4.227 mha_wave_t Struct Reference	839
4.228 mhachain::chain_base_t Class Reference	841
4.229 mhachain::mhachain_t Class Reference	845
4.230 mhachain::plugs_t Class Reference	846
4.231 mhaconfig_t Struct Reference	850
4.232 MHAEVENTS::connector_base_t Class Reference	852
4.233 MHAEVENTS::connector_t< receiver_t > Class Template Reference	855
4.234 MHAEVENTS::emitter_t Class Reference	858
4.235 MHAEVENTS::patchbay_t< receiver_t > Class Template Reference	860
4.236 MHAFILTER::adapt_filter_param_t Class Reference	862
4.237 MHAFILTER::adapt_filter_state_t Class Reference	863
4.238 MHAFILTER::adapt_filter_t Class Reference	865
4.239 MHAFILTER::blockprocessing_polyphase_resampling_t Class Reference	867
4.240 MHAFILTER::complex_bandpass_t Class Reference	871
4.241 MHAFILTER::diff_t Class Reference	875
4.242 MHAFILTER::fftfilter_t Class Reference	876
4.243 MHAFILTER::fftfilterbank_t Class Reference	881
4.244 MHAFILTER::filter_t Class Reference	886
4.245 MHAFILTER::gamma_filt_t Class Reference	891
4.246 MHAFILTER::iir_filter_state_t Class Reference	895
4.247 MHAFILTER::iir_filter_t Class Reference	896
4.248 MHAFILTER::iir_ord1_real_t Class Reference	900
4.249 MHAFILTER::o1_ar_filter_t Class Reference	903
4.250 MHAFILTER::o1flt_lowpass_t Class Reference	908
4.251 MHAFILTER::o1flt_maxtrack_t Class Reference	910
4.252 MHAFILTER::o1flt_mintrack_t Class Reference	912
4.253 MHAFILTER::partitioned_convolution_t Class Reference	914
4.254 MHAFILTER::partitioned_convolution_t::index_t Struct Reference	919
4.255 MHAFILTER::polyphase_resampling_t Class Reference	921
4.256 MHAFILTER::resampling_filter_t Class Reference	926
4.257 MHAFILTER::smoothspec_t Class Reference	928
4.258 MHAFILTER::thirddoctave_analyzer_t Class Reference	932
4.259 MHAFILTER::transfer_function_t Struct Reference	935
4.260 MHAFILTER::transfer_matrix_t Struct Reference	938

4.261 MHAIOJack::io_jack_t Class Reference	939
4.262 MHAIOJackdb::io_jack_t Class Reference	946
4.263 MHAIOPortAudio::device_info_t Class Reference	954
4.264 MHAIOPortAudio::io_portaudio_t Class Reference	957
4.265 MHAJack::client_avg_t Class Reference	963
4.266 MHAJack::client_noncont_t Class Reference	967
4.267 MHAJack::client_t Class Reference	970
4.268 MHAJack::port_t Class Reference	980
4.269 MHAKernel::algo_comm_class_t Class Reference	984
4.270 MHAKernel::comm_var_map_t Class Reference	990
4.271 MHAMultiSrc::base_t Class Reference	991
4.272 MHAMultiSrc::channel_t Class Reference	993
4.273 MHAMultiSrc::channels_t Class Reference	993
4.274 MHAMultiSrc::spectrum_t Class Reference	994
4.275 MHAMultiSrc::waveform_t Class Reference	996
4.276 MHAOvFilter::band_descriptor_t Class Reference	997
4.277 MHAOvFilter::barkscale::bark2hz_t Class Reference	999
4.278 MHAOvFilter::barkscale::hz2bark_t Class Reference	1000
4.279 MHAOvFilter::fftfb_ac_info_t Class Reference	1001
4.280 MHAOvFilter::fftfb_t Class Reference	1002
4.281 MHAOvFilter::fftfb_vars_t Class Reference	1006
4.282 MHAOvFilter::fscale_bw_t Class Reference	1010
4.283 MHAOvFilter::fscale_t Class Reference	1012
4.284 MHAOvFilter::fspacing_t Class Reference	1014
4.285 MHAOvFilter::overlap_save_filterbank_analytic_t Class Reference	1018
4.286 MHAOvFilter::overlap_save_filterbank_t Class Reference	1019
4.287 MHAOvFilter::overlap_save_filterbank_t::vars_t Class Reference	1022
4.288 MHAOvFilter::scale_var_t Class Reference	1024
4.289 MHAParser::base_t Class Reference	1026
4.290 MHAParser::base_t::replace_t Class Reference	1039
4.291 MHAParser::bool_mon_t Class Reference	1040
4.292 MHAParser::bool_t Class Reference	1042
4.293 MHAParser::c_ifc_parser_t Class Reference	1045
4.294 MHAParser::commit_t< receiver_t > Class Template Reference	1048
4.295 MHAParser::complex_mon_t Class Reference	1050
4.296 MHAParser::complex_t Class Reference	1052
4.297 MHAParser::entry_t Class Reference	1054
4.298 MHAParser::expression_t Class Reference	1055
4.299 MHAParser::float_mon_t Class Reference	1056
4.300 MHAParser::float_t Class Reference	1058
4.301 MHAParser::int_mon_t Class Reference	1061
4.302 MHAParser::int_t Class Reference	1064
4.303 MHAParser::keyword_list_t Class Reference	1066
4.304 MHAParser::kw_t Class Reference	1070
4.305 MHAParser::mcomplex_mon_t Class Reference	1074
4.306 MHAParser::mcomplex_t Class Reference	1076
4.307 MHAParser::mfloat_mon_t Class Reference	1078
4.308 MHAParser::mfloat_t Class Reference	1080
4.309 MHAParser::mhaconfig_mon_t Class Reference	1083
4.310 MHAParser::mhaplugloader_t Class Reference	1085
4.311 MHAParser::mint_mon_t Class Reference	1090

4.312 MHAParser::mint_t Class Reference	1092
4.313 MHAParser::monitor_t Class Reference	1095
4.314 MHAParser::parser_t Class Reference	1097
4.315 MHAParser::range_var_t Class Reference	1103
4.316 MHAParser::string_mon_t Class Reference	1108
4.317 MHAParser::string_t Class Reference	1110
4.318 MHAParser::variable_t Class Reference	1113
4.319 MHAParser::vcomplex_mon_t Class Reference	1115
4.320 MHAParser::vcomplex_t Class Reference	1117
4.321 MHAParser::vfloat_mon_t Class Reference	1120
4.322 MHAParser::vfloat_t Class Reference	1122
4.323 MHAParser::vint_mon_t Class Reference	1125
4.324 MHAParser::vint_t Class Reference	1127
4.325 MHAParser::vstring_mon_t Class Reference	1129
4.326 MHAParser::vstring_t Class Reference	1131
4.327 MHAParser::window_t Class Reference	1133
4.328 mhaplug_cfg_t Class Reference	1137
4.329 MHAPlugin::cfg_node_t< runtime_cfg_t > Class Template Reference	1138
4.330 MHAPlugin::config_t< runtime_cfg_t > Class Template Reference	1141
4.331 MHAPlugin::plugin_t< runtime_cfg_t > Class Template Reference	1146
4.332 MHAPlugin_Resampling::resampling_if_t Class Reference	1152
4.333 MHAPlugin_Resampling::resampling_t Class Reference	1155
4.334 MHAPlugin_Split::domain_handler_t Class Reference	1158
4.335 MHAPlugin_Split::dummy_threads_t Class Reference	1164
4.336 MHAPlugin_Split::posix_threads_t Class Reference	1166
4.337 MHAPlugin_Split::split_t Class Reference	1171
4.338 MHAPlugin_Split::splitted_part_t Class Reference	1178
4.339 MHAPlugin_Split::thread_platform_t Class Reference	1183
4.340 MHAPlugin_Split::uni_processor_t Class Reference	1187
4.341 mhaserver_t Class Reference	1189
4.342 mhaserver_t::tcp_server_t Class Reference	1193
4.343 MHASignal::async_rmslevel_t Class Reference	1195
4.344 MHASignal::delay_spec_t Class Reference	1198
4.345 MHASignal::delay_t Class Reference	1199
4.346 MHASignal::delay_wave_t Class Reference	1202
4.347 MHASignal::doublebuffer_t Class Reference	1203
4.348 MHASignal::fft_t Class Reference	1207
4.349 MHASignal::hilbert_fftw_t Class Reference	1212
4.350 MHASignal::hilbert_t Class Reference	1214
4.351 MHASignal::loop_wavefragment_t Class Reference	1216
4.352 MHASignal::matrix_t Class Reference	1221
4.353 MHASignal::minphase_t Class Reference	1233
4.354 MHASignal::quantizer_t Class Reference	1234
4.355 MHASignal::ringbuffer_t Class Reference	1236
4.356 MHASignal::schroeder_t Class Reference	1240
4.357 MHASignal::spectrum_t Class Reference	1244
4.358 MHASignal::stat_t Class Reference	1250
4.359 MHASignal::subsample_delay_t Class Reference	1252
4.360 MHASignal::uint_vector_t Class Reference	1255
4.361 MHASignal::waveform_t Class Reference	1259
4.362 MHASndFile::sf_t Class Reference	1272

4.363 MHASndFile::sf_wave_t Class Reference	1273
4.364 MHATableLookup::linear_table_t Class Reference	1274
4.365 MHATableLookup::table_t Class Reference	1279
4.366 MHATableLookup::xy_table_t Class Reference	1281
4.367 MHAWindow::bartlett_t Class Reference	1286
4.368 MHAWindow::base_t Class Reference	1287
4.369 MHAWindow::blackman_t Class Reference	1289
4.370 MHAWindow::fun_t Class Reference	1291
4.371 MHAWindow::hamming_t Class Reference	1292
4.372 MHAWindow::hanning_t Class Reference	1293
4.373 MHAWindow::rect_t Class Reference	1295
4.374 MHAWindow::user_t Class Reference	1296
4.375 multibandcompressor::fftfb_plug_t Class Reference	1298
4.376 multibandcompressor::interface_t Class Reference	1300
4.377 multibandcompressor::plugin_signals_t Class Reference	1303
4.378 nlms_t Class Reference	1305
4.379 noise_psd_estimator::noise_psd_estimator_if_t Class Reference	1309
4.380 noise_psd_estimator::noise_psd_estimator_t Class Reference	1311
4.381 noise_t Class Reference	1315
4.382 osc2ac_t Class Reference	1318
4.383 osc_server_t Class Reference	1321
4.384 osc_variable_t Class Reference	1323
4.385 overlapadd::overlapadd_if_t Class Reference	1327
4.386 overlapadd::overlapadd_t Class Reference	1332
4.387 parser_int_dyn Class Reference	1335
4.388 plingploing::if_t Class Reference	1337
4.389 plingploing::plingploing_t Class Reference	1341
4.390 plug_t Class Reference	1346
4.391 plug_wrapper Class Reference	1347
4.392 plug_wrapperl Class Reference	1349
4.393 plugin_interface_t Class Reference	1351
4.394 pluginbrowser_t Class Reference	1353
4.395 plugindescription_t Class Reference	1356
4.396 pluginlib_t Class Reference	1357
4.397 PluginLoader::config_file_splitter_t Class Reference	1359
4.398 PluginLoader::fourway_processor_t Class Reference	1362
4.399 PluginLoader::mhaplugloader_t Class Reference	1365
4.400 pluginloader_t Class Reference	1373
4.401 plugins::hoertech::acrec::acrec_t Class Reference	1374
4.402 plugins::hoertech::acrec::acwriter_t Class Reference	1379
4.403 rmslevel::mon_t Class Reference	1385
4.404 rmslevel::rmslevel_if_t Class Reference	1387
4.405 rmslevel::rmslevel_t Class Reference	1389
4.406 rohBeam::configOptions Struct Reference	1393
4.407 rohBeam::rohBeam Class Reference	1394
4.408 rohBeam::rohConfig Class Reference	1402
4.409 route::interface_t Class Reference	1408
4.410 route::process_t Class Reference	1411
4.411 rt_nlms_t Class Reference	1413
4.412 save_spec_t Class Reference	1417
4.413 save_wave_t Class Reference	1419

4.414 shadowfilter_begin::cfg_t Class Reference	1420
4.415 shadowfilter_begin::shadowfilter_begin_t Class Reference	1422
4.416 shadowfilter_end::cfg_t Class Reference	1424
4.417 shadowfilter_end::shadowfilter_end_t Class Reference	1426
4.418 sine_cfg_t Struct Reference	1428
4.419 sine_t Class Reference	1430
4.420 smooth_cepstrum::smooth_cepstrum_if_t Class Reference	1433
4.421 smooth_cepstrum::smooth_cepstrum_t Class Reference	1438
4.422 smooth_cepstrum::smooth_params Class Reference	1445
4.423 smoothgains_bridge::overlapadd_if_t Class Reference	1448
4.424 smoothgains_bridge::smoothspec_wrap_t Class Reference	1451
4.425 softclip_t Class Reference	1453
4.426 softclipper_t Class Reference	1455
4.427 softclipper_variables_t Class Reference	1457
4.428 spec2wave_if_t Class Reference	1460
4.429 spec2wave_t Class Reference	1462
4.430 spec_fader_t Class Reference	1465
4.431 speechnoise_t Class Reference	1466
4.432 steerbf Class Reference	1469
4.433 steerbf_config Class Reference	1472
4.434 testplugin::ac_parser_t Class Reference	1474
4.435 testplugin::config_parser_t Class Reference	1477
4.436 testplugin::if_t Class Reference	1480
4.437 testplugin::signal_parser_t Class Reference	1483
4.438 us_t Class Reference	1484
4.439 wave2spec_if_t Class Reference	1486
4.440 wave2spec_t Class Reference	1491
4.441 wavrec_t Class Reference	1497
4.442 wavwriter_t Class Reference	1500
4.443 windnoise::cfg_t Class Reference	1503
4.444 windnoise::if_t Class Reference	1508
4.445 windowselector_t Class Reference	1512
5 File Documentation	1516
5.1 ac2lsl.cpp File Reference	1516
5.2 ac2osc.cpp File Reference	1517
5.3 ac2wave.cpp File Reference	1517
5.4 ac_monitor_type.cpp File Reference	1517
5.5 ac_monitor_type.hh File Reference	1517
5.6 ac_mul.cpp File Reference	1518
5.7 ac_mul.hh File Reference	1518
5.8 ac_proc.cpp File Reference	1519
5.9 acConcat_wave.cpp File Reference	1519
5.10 acConcat_wave.h File Reference	1520
5.11 acmon.cpp File Reference	1520
5.12 acPooling_wave.cpp File Reference	1520
5.13 acPooling_wave.h File Reference	1521
5.14 acrec.cpp File Reference	1521
5.15 acrec.hh File Reference	1521
5.16 acsave.cpp File Reference	1521
5.17 acSteer.cpp File Reference	1523
5.18 acSteer.h File Reference	1523

5.19	acTransform_wave.cpp File Reference	1523
5.20	acTransform_wave.h File Reference	1524
5.21	adaptive_feedback_canceller.cpp File Reference	1524
5.22	adaptive_feedback_canceller.h File Reference	1525
5.23	addsndfile.cpp File Reference	1525
5.24	adm.cpp File Reference	1526
5.25	adm.hh File Reference	1527
5.26	altconfig.cpp File Reference	1528
5.27	altconfig.hh File Reference	1528
5.28	altplugs.cpp File Reference	1528
5.29	analysemhaplugin.cpp File Reference	1529
5.30	analysispath.cpp File Reference	1530
5.31	attenuate20.cpp File Reference	1530
5.32	audiometerbackend.cpp File Reference	1531
5.33	auditory_profile.cpp File Reference	1532
5.34	auditory_profile.h File Reference	1532
5.35	browsemhaplugins.cpp File Reference	1532
5.36	coherence.cpp File Reference	1533
5.37	combinechannels.cpp File Reference	1533
5.38	compiler_id.cpp File Reference	1534
5.39	compiler_id.hh File Reference	1534
5.40	complex_filter.cpp File Reference	1535
5.41	complex_filter.h File Reference	1535
5.42	complex_scale_channel.cpp File Reference	1536
5.43	cpupload.cpp File Reference	1536
5.44	db.cpp File Reference	1536
5.45	dbasync.cpp File Reference	1536
5.46	dc.cpp File Reference	1537
5.47	dc.hh File Reference	1538
5.48	dc_afterburn.cpp File Reference	1538
5.49	dc_afterburn.h File Reference	1538
5.50	dc_simple.cpp File Reference	1539
5.51	dc_simple.hh File Reference	1539
5.52	delay.cpp File Reference	1540
5.53	delay.hh File Reference	1540
5.54	delaysum_spec.cpp File Reference	1540
5.55	delaysum_wave.cpp File Reference	1541
5.56	doasvm_classification.cpp File Reference	1541
5.57	doasvm_classification.h File Reference	1542
5.58	doasvm_feature_extraction.cpp File Reference	1542
5.59	doasvm_feature_extraction.h File Reference	1542
5.60	doc_appendix.h File Reference	1543
5.61	doc_examples.h File Reference	1543
5.62	doc_frameworks.h File Reference	1543
5.63	doc_general.h File Reference	1543
5.64	doc_kernel.h File Reference	1543
5.65	doc_matlab.h File Reference	1543
5.66	doc_mhamain.h File Reference	1543
5.67	doc_parser.h File Reference	1543
5.68	doc_plugins.h File Reference	1543
5.69	doc_system.h File Reference	1543

5.70	doc_toolbox.h File Reference	1543
5.71	double2acvar.cpp File Reference	1543
5.72	downsample.cpp File Reference	1543
5.73	dropgen.cpp File Reference	1544
5.74	droptect.cpp File Reference	1544
5.75	equalize.cpp File Reference	1544
5.76	example1.cpp File Reference	1544
5.77	example2.cpp File Reference	1544
5.78	example3.cpp File Reference	1545
5.79	example4.cpp File Reference	1545
5.80	example5.cpp File Reference	1545
5.81	example6.cpp File Reference	1545
5.82	example7.cpp File Reference	1546
5.83	example7.hh File Reference	1546
5.84	fader_spec.cpp File Reference	1546
5.85	fader_wave.cpp File Reference	1546
5.86	fftfbpow.cpp File Reference	1547
5.87	fftfilter.cpp File Reference	1547
5.88	fftfilterbank.cpp File Reference	1548
5.89	fshift.cpp File Reference	1548
5.90	fshift.hh File Reference	1548
5.91	fshift_hilbert.cpp File Reference	1549
5.92	gain.cpp File Reference	1549
5.93	gaintable.cpp File Reference	1549
5.94	gaintable.h File Reference	1550
5.95	generatemhaplugindoc.cpp File Reference	1550
5.96	gsc_adaptive_stage.cpp File Reference	1553
5.97	gsc_adaptive_stage.hh File Reference	1553
5.98	gsc_adaptive_stage_if.cpp File Reference	1553
5.99	gsc_adaptive_stage_if.hh File Reference	1553
5.100	gtfb_analyzer.cpp File Reference	1553
5.101	gtfb_simd.cpp File Reference	1555
5.102	gtfb_simple_bridge.cpp File Reference	1561
5.103	hann.cpp File Reference	1561
5.104	hann.h File Reference	1562
5.105	identity.cpp File Reference	1563
5.106	ifftshift.cpp File Reference	1563
5.107	ifftshift.h File Reference	1563
5.108	iirfilter.cpp File Reference	1563
5.109	level_matching.cpp File Reference	1564
5.110	level_matching.hh File Reference	1564
5.111	levelmeter.cpp File Reference	1564
5.112	lpc.cpp File Reference	1565
5.113	lpc.h File Reference	1566
5.114	lpc_bl_predictor.cpp File Reference	1566
5.115	lpc_bl_predictor.h File Reference	1567
5.116	lpc_burg-lattice.cpp File Reference	1567
5.117	lpc_burg-lattice.h File Reference	1568
5.118	isl2ac.cpp File Reference	1568
5.119	isl2ac.hh File Reference	1568
5.120	matlab_wrapper.cpp File Reference	1569

5.121 matlab_wrapper.hh File Reference	1569
5.122 matrixmixer.cpp File Reference	1569
5.123 mconv.cpp File Reference	1570
5.124 mha.cpp File Reference	1570
5.125 mha.hh File Reference	1570
5.126 mha_algo_comm.cpp File Reference	1578
5.127 mha_algo_comm.h File Reference	1580
5.128 mha_algo_comm.hh File Reference	1581
5.129 mha_defs.h File Reference	1582
5.130 mha_errno.c File Reference	1584
5.131 mha_errno.h File Reference	1585
5.132 mha_error.cpp File Reference	1587
5.133 mha_error.hh File Reference	1587
5.134 mha_event_emitter.h File Reference	1588
5.135 mha_events.cpp File Reference	1589
5.136 mha_events.h File Reference	1589
5.137 mha_fttbf.cpp File Reference	1589
5.138 mha_fttbf.hh File Reference	1591
5.139 mha_fifo.cpp File Reference	1592
5.140 mha_fifo.h File Reference	1592
5.141 mha_filter.cpp File Reference	1593
5.142 mha_filter.hh File Reference	1593
5.143 mha_generic_chain.cpp File Reference	1595
5.144 mha_generic_chain.h File Reference	1595
5.145 mha_git_commit_hash.cpp File Reference	1596
5.146 mha_git_commit_hash.hh File Reference	1597
5.147 mha_io_ifc.h File Reference	1597
5.148 mha_io_utils.cpp File Reference	1599
5.149 mha_io_utils.hh File Reference	1599
5.150 mha_multisrc.cpp File Reference	1600
5.151 mha_multisrc.h File Reference	1600
5.152 mha_os.cpp File Reference	1600
5.153 mha_os.h File Reference	1602
5.154 mha_parser.cpp File Reference	1607
5.155 mha_parser.hh File Reference	1611
5.156 mha_plugin.cpp File Reference	1615
5.157 mha_plugin.hh File Reference	1615
5.158 mha_profiling.c File Reference	1619
5.159 mha_profiling.h File Reference	1620
5.160 mha_ruby.cpp File Reference	1621
5.161 mha_signal.cpp File Reference	1622
5.162 mha_signal.hh File Reference	1626
5.163 mha_signal_fft.h File Reference	1636
5.164 mha_tablelookup.cpp File Reference	1636
5.165 mha_tablelookup.hh File Reference	1636
5.166 mha_tcp.cpp File Reference	1637
5.167 mha_tcp.hh File Reference	1640
5.168 mha_tcp_server.cpp File Reference	1641
5.169 mha_tcp_server.hh File Reference	1641
5.170 mha_toolbox.h File Reference	1642
5.171 mha_utils.cpp File Reference	1642

5.172 mha_utils.hh File Reference	1642
5.173 mha_windowparser.cpp File Reference	1642
5.174 mha_windowparser.h File Reference	1643
5.175 mhachain.cpp File Reference	1644
5.176 mhafw_lib.cpp File Reference	1644
5.177 mhafw_lib.h File Reference	1644
5.178 MHAIOalsa.cpp File Reference	1644
5.179 MHAIOAsterisk.cpp File Reference	1649
5.180 MHAIOFile.cpp File Reference	1654
5.181 MHAIOJack.cpp File Reference	1659
5.182 MHAIOJackdb.cpp File Reference	1663
5.183 MHAIOParser.cpp File Reference	1667
5.184 MHAIOPortAudio.cpp File Reference	1671
5.185 MHAIOTCP.cpp File Reference	1676
5.186 mhajack.cpp File Reference	1681
5.187 mhajack.h File Reference	1682
5.188 mhomain.cpp File Reference	1684
5.189 mhaplugloader.cpp File Reference	1686
5.190 mhaplugloader.h File Reference	1686
5.191 mhasndfile.cpp File Reference	1687
5.192 mhasndfile.h File Reference	1687
5.193 multibandcompressor.cpp File Reference	1688
5.194 nlms_wave.cpp File Reference	1688
5.195 noise.cpp File Reference	1690
5.196 noise_psd_estimator.cpp File Reference	1690
5.197 osc2ac.cpp File Reference	1691
5.198 overlapadd.cpp File Reference	1691
5.199 overlapadd.hh File Reference	1691
5.200 plingploing.cpp File Reference	1691
5.201 pluginbrowser.cpp File Reference	1692
5.202 pluginbrowser.h File Reference	1692
5.203 resampling.cpp File Reference	1692
5.204 rmslevel.cpp File Reference	1692
5.205 rohBeam.cpp File Reference	1693
5.206 rohBeam.hh File Reference	1693
5.207 route.cpp File Reference	1694
5.208 save_spec.cpp File Reference	1694
5.209 save_wave.cpp File Reference	1694
5.210 shadowfilter_begin.cpp File Reference	1695
5.211 shadowfilter_end.cpp File Reference	1695
5.212 sine.cpp File Reference	1695
5.213 smooth_cepstrum.cpp File Reference	1695
5.214 smooth_cepstrum.hh File Reference	1696
5.215 smoothgains_bridge.cpp File Reference	1696
5.216 softclip.cpp File Reference	1697
5.217 spec2wave.cpp File Reference	1697
5.218 speechnoise.cpp File Reference	1697
5.219 speechnoise.h File Reference	1701
5.220 split.cpp File Reference	1702
5.221 steerbf.cpp File Reference	1703
5.222 steerbf.h File Reference	1704

5.223 testalsadevice.c File Reference	1704
5.224 testplugin.cpp File Reference	1704
5.225 transducers.cpp File Reference	1705
5.226 upsample.cpp File Reference	1706
5.227 wave2spec.cpp File Reference	1706
5.228 wave2spec.hh File Reference	1706
5.229 wavrec.cpp File Reference	1707
5.230 windnoise.cpp File Reference	1707
5.231 windnoise.hh File Reference	1707
5.232 windowselector.cpp File Reference	1707
5.233 windowselector.h File Reference	1707
Index	1709

1 Overview

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

1.1 Structure

The openMHA can be split into four major components :

- **The openMHA command line application (MHA)** (p. 30)
- Signal processing plugins
- Audio input-output (IO) plugins (see [io_file_t](#) (p. 604), [MHAIQJack](#) (p. 109), [io_parser_t](#) (p. 615), [io_tcp_parser_t](#) (p. 624))
- **The openMHA toolbox library** (p. 31)

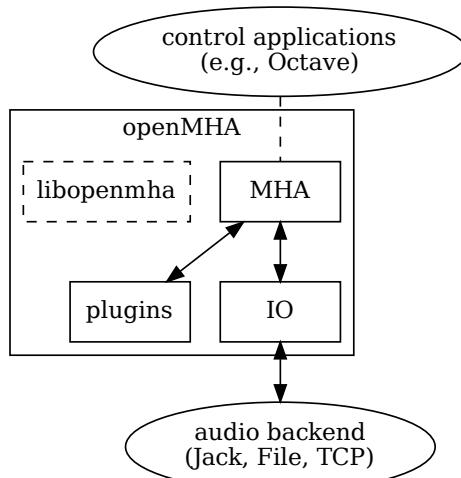


Figure 1 openMHA structure

The openMHA command line application (MHA) (p. 30) acts as a plugin host. It can load signal processing plugins as well as audio input-output (IO) plugins. Additionally, it provides the command line configuration interface and a TCP/IP based configuration interface. Several IO plugins exist: For real-time signal processing, commonly the openMHA [MHAIQJack](#) (p. 109) plugin (see plugins' manual) is used, which provides an interface to the Jack Audio Connection Kit (JACK). Other IO plugins provide audio file access or TCP/IP-based processing.

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

1.2 Platform Services and Conventions

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

1.2.1 Audio Signal Domains

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

Similar to some other platforms, the openMHA allows also arbitrary data to be exchanged between plugins through a mechanism called **algorithm communication variables** (p. 23) or short "AC vars". This mechanism is commonly used to share data such as filter coefficients or filter states.

1.2.2 Real-Time Safe Complex Configuration Changes

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

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

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

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

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

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

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

1.2.3 Plugins can Themselves Host Other Plugins

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

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

1.2.4 Central Calibration

The purpose of hearing aid signal processing is to enhance the sound for hearing impaired listeners. Hearing impairment generally means that people suffering from it have increased hearing thresholds, i.e. soft sounds that are audible for normal hearing listeners may be imperceptible for hearing impaired listeners. To provide accurate signal enhancement for hearing impaired people, hearing aid signal processing algorithms have to be able to determine the absolute physical sound pressure level corresponding to a digital signal given to any openMHA plugin for processing. Inside the openMHA, we achieve this with the following convention: The single-precision floating point time-domain sound signal samples, that are processed inside the openMHA plugins in blocks of short durations, have the physical pressure unit Pascal ($1\text{Pa} = 1\text{N/m}^2$). With this convention in place, all plugins can determine the absolute physical sound pressure level from the sound samples that they process: E.g. plugins can compute the rms (root mean squared) sound pressure in Pascal of the current block by computing $\text{rms} = \sqrt{\sum_i x_i^2}$, where x_i refer to the samples of the audio signal in a single audio channel, and the corresponding free-field sound pressure level L in dB SPL FF as $L = 20 \log_{10}(rms/20\mu\text{Pa})$. ($20\mu\text{Pa}$ is the sound pressure at 0dB SPL FF).

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

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

Due to the dependency of the calibration on the hardware used, it is the responsibility of the user of the openMHA to perform calibration measurements and adapt the openMHA settings to make sure that this calibration convention is met. We provide the plugin `transducers` which can be configured to perform the necessary signal adjustments.

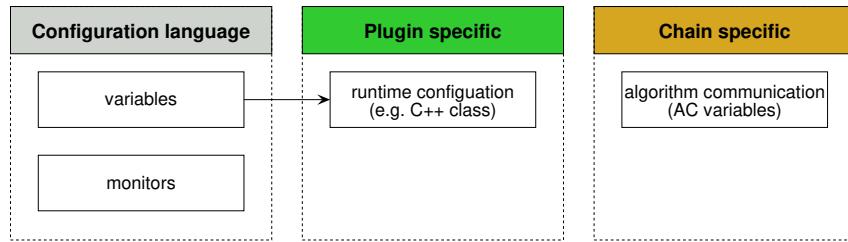
2 Module Documentation

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

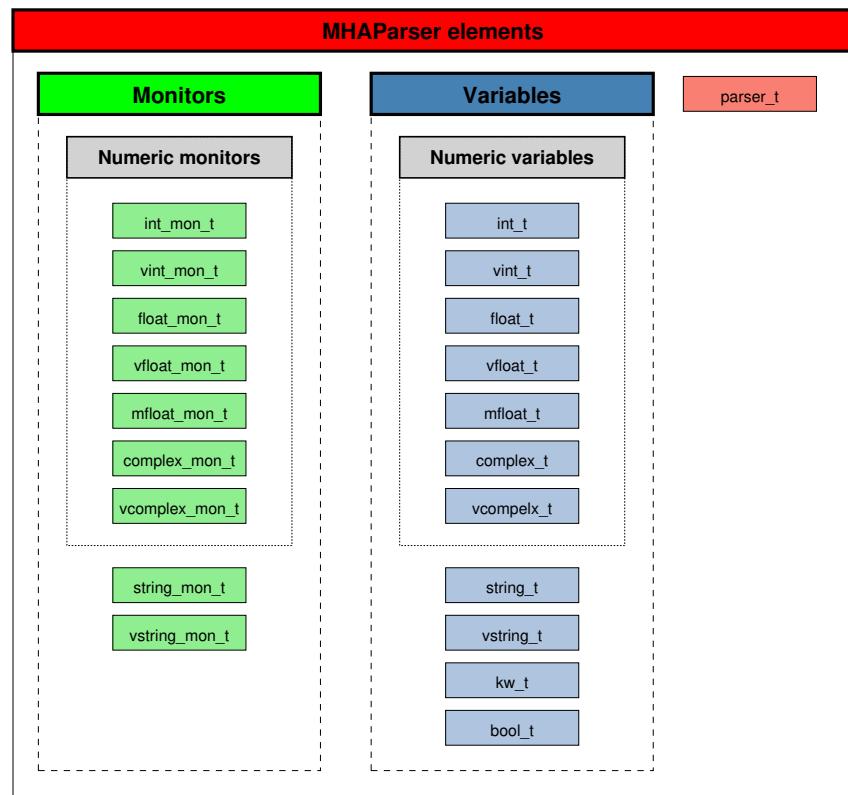
Accessibility of configuration variables and data exchange between plugins (processing blocks) are an important issue in the openMHA. In general, variable types in the openMHA are distinguished by their different access methods. The variable types in the openMHA are:

- **Configuration variables** : Read and write accesses are possible through the openMHA configuration language interface. Configuration variables are implemented as C++ classes with a public data member of the underlying C type. Configuration variables can be read and modified from ‘outside’ using the configuration language. The plugin which provides the configuration variable can use the exposed data member directly. All accesses through the openMHA configuration language are checked for data type, valid range, and access restrictions.
- **Monitor variables** : Read access is possible through the openMHA configuration language. Write access is only possible from the C++ code. Internally, monitor variables have a similar C++ class interface as configuration variables.
- **AC variables (algorithm communication variables)** ([p. 23](#)): Any C or C++ data structure can be shared within an openMHA chain. Access management and name space is realised in openMHA chain plugin ('mhachain'). AC variables are not available to the openMHA configuration language interface, although a read-only converter plugin `acmon` is available.
- **Runtime configuration** : Algorithms usually derive more parameters (runtime configuration) from the openMHA configuration language variables. When a configuration variable changes through configuration language write access, then the runtime configuration has to be recomputed. Plugin developers are encouraged to encapsulate the runtime configuration in a C++ class, which recomputes the runtime configuration from configuration variables in the constructor. The openMHA supports lock-free and thread-safe replacement of the runtime configuration instance (see `example5.cpp` ([p. 15](#)) and references therein).

**Figure 2 Variable types in the openMHA**

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

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

**Figure 3 MHAParser elements**

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

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

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

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

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.

2.2.1 example1.cpp

The example plugin file `example1.cpp` (p. 1544) 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 **MHAParser::parser_t** (p. 1146) 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 MHAParser::parser_t<int> {
public:
    example1_t(algo_comm_t & ac,
               const std::string & chain_name,
               const std::string & algo_name)
        : MHAParser::parser_t<int>("", ac)
    /* Do nothing in constructor */
    void release(void)
    /* Do nothing in release */
}
```

Every plugin implementation should include the '**mha_plugin.hh**' (p. 1615) 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. 1615).

The class `plugin1_t` inherits from the class **MHAParser::parser_t** (p. 1146), which then inherits from **MHAParser::parser_t** (p. 1097) – the configuration language interface in the method "parse". Our plugin class therefore exports the working "parse" method inherited from **MHAParser::parser_t** (p. 1097), 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.

2.2.1.1 The prepare method

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

Parameters

<i>signal_info</i>	Contains information about the input signal's parameters, see mhaconfig_t (p. 850).
--------------------	--

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.

2.2.1.2 The signal processing method

```
mha_wave_t * process(mha_wave_t * signal)
{
    unsigned int channel = 0; // channels and frames counting starts with 0
    float factor = 0.1f;
    unsigned int frame;
    // Scale channel number "channel" by "factor":
    for(frame = 0; frame < signal->num_frames; frame++) {
        // Waveform channels are stored interleaved.
        signal->buf[signal->num_channels * frame + channel] *= factor;
    }
    // Algorithms may process data in-place and return the input signal
    // structure as their output signal:
    return signal;
};
```

Parameters

<i>signal</i>	Pointer to the input signal structure mha_wave_t (p. 839).
---------------	---

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. 839) 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).

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

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

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

This macro takes care of accessing the C++ class from the C functions required as the plugin's interface. It implements the C funtions and calls the corresponding C++ instance methods. Plugin classes should be derived from the template class **MHAPlugin::plugin_t** (p. 1146) to be compatible with the C interface wrapper.

This macro also catches C++ exceptions of type **MHA_Error** (p. 763), 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 **MH←A_Error** (p. 763) 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. 763). 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.

2.2.2 example2.cpp

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

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

```
class example2_t : public MHAPlugin::plugin_t<int> {
    MHAParser::int_t scale_ch;
    MHAParser::float_t factor;
public:
    example2_t(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. 1146) for intergration with the openMHA configuration language. The two data members serve as externally visible configuration variables. All methods of this class have a non-empty implementation.

2.2.2.1 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. 1099) method.

2.2.2.2 The prepare method

```

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

```

Parameters

<i>signal_info</i>	– contains information about the input signal's parameters, see mhaconfig_t (p. 850).
--------------------	--

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.

2.2.2.3 The release method

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

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

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

2.2.2.4 The signal processing method

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

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

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

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

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

Fortunately, for some data types on PC systems, case (c) cannot happen. These are 32bit wide data types with a 4-byte alignment. Therefore, the values in **MHAParser::int_t** (p. 1064) and **MHAParser::float_t** (p. 1058) 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.

2.2.3 example3.cpp

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

The third example 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.

2.2.3.1 Data member declarations

```
class example3_t : public MHAParser::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.

2.2.3.2 Method declarations

```
/* Callbacks triggered by Events */
void on_scale_ch_writeaccess();
void on_scale_ch_valuechanged();
void on_scale_ch_readaccess();
void on_prereadaccess();
public:
    example3_t(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.

2.2.3.3 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.

2.2.3.4 The prepare method

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

The prepare method checks 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.

2.2.3.5 The release method

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

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

2.2.3.6 The signal processing method

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

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

2.2.3.7 The callback methods

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

```

}
MHAPLUGIN_CALLBACKS(example3,example3_t, wave, wave)

```

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

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

2.2.4 example4.cpp

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

2.2.4.1 The Prepare method

```

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

```

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

2.2.4.2 The signal processing method

```

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

```

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

The `mha_spec_t` (p. 793) instance stores the complex (`mha_complex_t` (p. 744)) spectral signal for positive frequencies only (since the waveform signal is always real). The `num_frames` member of `mha_spec_t` (p. 793) actually denotes the number of STFT bins.

Please note that different from `mha_wave_t` (p. 839), a multichannel signal in `mha_spec_t` (p. 793) is stored non-interleaved in the signal buffer.

Some arithmetic operations are defined on struct `mha_complex_t` (p. 744) to facilitate efficient complex computations. The `*=` operator used here (defined for real and for complex arguments) is one of them.

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

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

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

2.2.5 example5.cpp

Many algorithms use complex operations to transform the user space variables into run time configurations. If this takes a noticeable time (e.g. more than 100-500 μ sec), the update of the runtime configuration can not take place in the real time processing thread. Furthermore, the parallel access to complex structures may cause unpredictable results if variables are read while only parts of them are written to memory (cf. section **Consistency** (p. 10)). To handle these situations, a special C++ template class **MHAPlugin::plugin_t** (p. 1146) 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. 480) is the parameter of the template class **MHAPlugin::plugin_t** (p. 1146). 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. 1146), 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 **MHAEevents::patchbay_t** (p. 860) 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: */
    MHAEevents::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. 763) 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. 763) exception.

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

In this example, the run time configuration class **example5_t** (p. 480) 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. 860) 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 scaling a spectral signal",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. 763) 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. 1145) stores the configuration in a FiFo queue of runtime configurations. Once they are inserted in the FiFo, the **MHAPLUGIN::plugin_t** (p. 1146) 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. 1618) defines all ANSI-C interface functions and passes them to the corresponding C++ class member functions (partly defined by the **MHAPLUGIN::plugin_t** (p. 1146) template class). All exceptions of type **MHA_Error** (p. 763) are caught and transformed into an appropriate error code and error message.

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

2.2.6 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 'acsav' plugin can be used. 'acmon' is a plugin which converts AC variables into parsable monitor variables.

In the constructor of the plugin class the variable `rmsdb` is registered under the name `example6_rmslev` as a one-dimensional AC variable of type float. For registration of other types, read access and other detailed informations please see **Communication between algorithms** (p. 23).

```
example6_t::example6_t(const algo_comm_t& iac,
                      const std::string&,const std::string&)
: MHAPLUGIN::plugin_t<cfg_t>("Example rms level meter plugin",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, than it is necessary to use different names for this  
 * variable (i.e. prefixing the name with the algorithm name  
 * passed to MHAInit).  
 */  
ac.insert_var_float( ac.handle, "example6_rmslev", &rmsdb );  
}
```

2.2.7 Debugging openMHA plugins

Suppose you would want to step through the code of your openMHA plugin with a debugger. This example details how to use the linux gdb debugger to inspect the `example6_t::prepare()` (p. 482) and `example6_t::process()` (p. 482) routines of `example6.cpp` (p. 17) 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. 482)) does not yet work, as the symbol lives in the openMHA plugin that has not yet been loaded. Specifying by line number works, however. Specifying the breakpoint by symbol also works once the plugin is loaded (i.e. when the debugger stops in the first break point). You can set the breakpoints like this (example shown here is run in `gdb` version 7.11.1):

```
(gdb) run ?read:debugexample6.cfg
Starting program: {openMHA_directory}/bin/mha ?read:debugexample6.cfg
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
The Open Master Hearing Aid (openMHA) server
Copyright (c) 2005-2021 HoerTech gGmbH, D-26129 Oldenburg, Germany
```

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

```

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

```

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

```

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

```

2.2.8 Writing unit tests for openMHA plugins

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

2.2.8.1 example7 As an example, unit tests for plugin `example7.cpp` (p. 1546) are written, which is functionally the same as plugin `example1.cpp` (p. 1544) (see section **example1.cpp** (p. 6)). In order to write unit tests for your plugin it must have its class/function declarations in a header file (.hh) so you can include it in the unit test file. The class/function definitions are contained in the respective source file (.cpp).

The unit tests are written using a test fixture class (here: `example7_testing`) which will be inherited by the individual tests (`TEST_F`). This enables us to use the members in `example7_testing` in multiple tests without the need for redundant declarations.

```

#include "mha_algo_comm.hh"
#include "mha_signal.hh"
#include "example7.hh"
#include <gtest/gtest.h>
class example7_testing : public ::testing::Test {
public:
    // AC variable space
    MHAKernel::algo_comm_class_t acspace{};
    // C handle to AC variable space
    algo_comm_t ac {acspace.get_c_handle()};
    // example input to prepare method
    mhaconfig_t signal_properties {
        .channels = 2U,
        .domain = MHA_WAVEFORM,
        .fragsize = 10U,
        .wndlen = 0U,
        .fftlen = 0U,
        .srate = 44100.0f
    };

```

```
//Plugin instance
example7_t ex7{ac,"thread","algo"};
MHASignal::waveform_t wave_input{signal_properties.fragsize,signal_properties.channels};
};
```

The test fixture class is derived from the `::testing::Test` class declared in `gtest.h`. The constructor of `example7_t` (p. 484) needs three parameters, namely a handle to the algorithm communication variable space and two strings. A container for audio signals for repeatedly passing blocks of the input signal to the plugin under test is also allocated by the test fixture class. It is defined as an instance of `MHASignal::waveform_t` (p. 1259) with the name `wave_input` and its values are zero upon initialization.

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

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

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

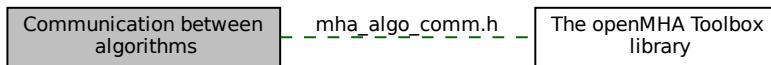
In this test the goal is to assess the main feature of the plugin (`example7_t` (p. 484)), which is the same as in `example1_t` (p. 466), namely altering the signal's first channel by a constant factor of 0.1. The variable `wave_input` is the signal that will be processed by the plugin. In order to assert the success, the elements in `wave_input` are set to a constant value of 1, because they are 0 upon initialization. During the `process()` function all elements of the first channel of `wave_input` are multiplied by the factor 0.1. Before `process()` is called the value assigned to `wave_input` is checked via the method `EXPECT_FLOAT_EQ` provided by GoogleTest. The values of `wave_input` are retrieved by the method `value()` (p. 46) by passing the desired sample position and channel number as second and third input parameter, respectively. Here, we checked the values of two frames in each channel to show the difference before and after processing; the frame indices were chosen randomly. After calling `process()`, the values contained in `wave_input` are checked again to make sure that the plugin worked as intended.

2.3 The MHA Framework interface

2.4 Communication between algorithms

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.

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. 1244) class into the AC space.*
- class **MHA_AC::waveform_t**
*Insert a **MHASignal::waveform_t** (p. 1259) 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.

2.4.1 Detailed Description

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

An algorithm communication handle (**algo_comm_t** (p. 280)) is passed at initialisation time to the constructor of each plugin class **constructor** (p. 1146). This handle contains a reference handle, **algo_comm_t::handle** (p. 281), and a number of function pointers, **algo_comm_t::insert_var** (p. 281) etc.. An algorithm communication variable is accessed through objects of type **comm_var_t** (p. 360).

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

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

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

1. Plugins publishing AC variables:

- insert all variables during prepare()

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

2.4.2 Function Documentation

2.4.2.1 get_var_spectrum() `mha_spec_t MHA_AC::get_var_spectrum (algo_comm_t ac, const std::string & name)`

Convert an AC variable into a spectrum.

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

Parameters

<code>ac</code>	AC handle
<code>name</code>	Name of the variable

Returns

Spectrum structure

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

Convert an AC variable into a waveform.

This function reads an AC variable and tries to convert it into a valid 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

2.4.2.3 **get_var_int()**

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

Return value of an integer scalar AC variable.

Parameters

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

Returns

Variable value

2.4.2.4 **get_var_float()**

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

Return value of an floating point scalar AC variable.

Parameters

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

Returns

Variable value

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

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

Parameters

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

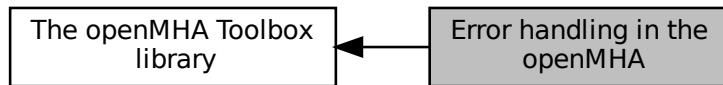
Returns

Variable value

2.5 Error handling in the openMHA

Errors are reported to the user via the **MHA_Error** (p. 763) 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. 763).*
- #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. 763) with the values.*

Functions

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

2.5.1 Detailed Description

Errors are reported to the user via the **MHA_Error** (p. 763) exception.

2.5.2 Macro Definition Documentation

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

Throw an openMHA error with a text message.

Parameters

x	Text message.
---	---------------

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

Assertion macro, which throws an **MHA_Error** (p. 763).

Parameters

x	Boolean expression which should be true.
---	--

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

Equality assertion macro, which throws an **MHA_Error** (p. 763) with the values.

Parameters

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

2.5.3 Function Documentation

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

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

2.6 The openMHA configuration language

openMHA Plugins that should use the openMHA configuration language for their configuration have to be implemented in C++ and need to include **mha_parser.hh** (p. 1611). All required classes and functions for parser access are declared in the namespace **MHAParser** (p. 122). The plugin class should be derived from the class **MHAParser::parser_t** (p. 1097) (or **MHAParser::MHAParser::plugin_t** (p. 1146)), 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. 1099).

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. 1611). All required classes and functions for parser access are declared in the namespace **MHAParser** (p. 122). The plugin class should be derived from the class **MHAParser::parser_t** (p. 1097) (or **MHAParser::MHAParser::plugin_t** (p. 1146)), 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. 1099).

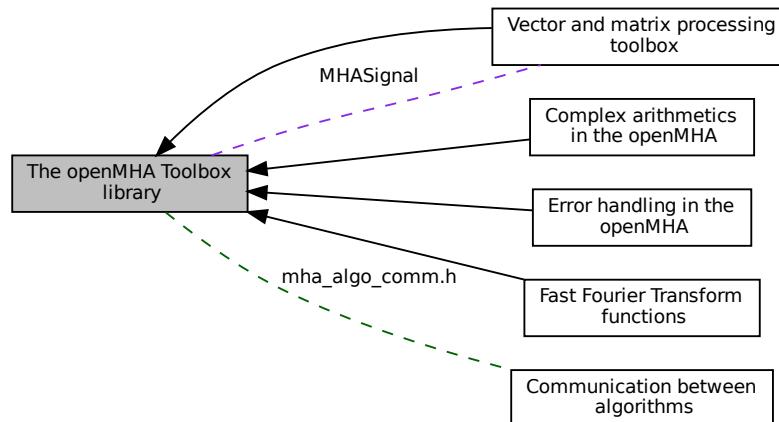
The openMHA Plugin template class **MHAParser::MHAParser::plugin_t** (p. 1146) together with the Plugin macro **MHAPARSER_CALLBACKS** (p. 1618) provide the callback mappings and correct inheritance. If your plugin is based on that template class, you simply have to use the **insert_item** command to 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. 122) namespace.

2.7 The openMHA Toolbox library

The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins. It contains the openMHA script language classes.

Collaboration diagram for The openMHA Toolbox library:



Modules

- **Error handling in the openMHA**

Errors are reported to the user via the [MHA_Error](#) (p. 763) exception.

- **Vector and matrix processing toolbox**

The vector and matrix processing toolbox consists of a number of classes defined in the namespace [MHASignal](#) (p. 136), and many functions and operators for use with the structures [mha_wave_t](#) (p. 839) and [mha_spec_t](#) (p. 793).

- **Complex arithmetics in the openMHA**

- **Fast Fourier Transform functions**

Files

- file **mha_algo_comm.h**

Header file for Algorithm Communication.

- file **mha_filter.hh**

Header file for IIR filter classes.

- file **mha_signal.hh**

Header file for audio signal handling and processing classes.

- file **mha_tablelookup.hh**

Header file for table lookup classes.

Namespaces

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

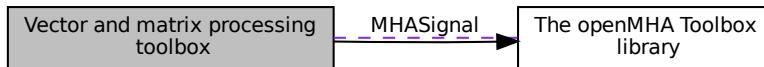
2.7.1 Detailed Description

The openMHA toolbox is a static C++ library which makes it more comfortable to develop openMHA plugins. It contains the openMHA script language classes.

2.8 Vector and matrix processing toolbox

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 136), and many functions and operators for use with the structures **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793).

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_audio_descriptor_t**
*Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 850)).*
- struct **mha_audio_t**
*An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793)).*
- class **MHASignal::spectrum_t**
*a signal processing class for spectral data (based on **mha_spec_t** (p. 793))*
- class **MHASignal::waveform_t**
*signal processing class for waveform data (based on **mha_wave_t** (p. 839))*
- class **MHASignal::doublebuffer_t**
Double-buffering class.
- class **MHASignal::hilbert_t**
Hilbert transformation of a waveform segment.
- class **MHASignal::minphase_t**
Minimal phase function.
- class **MHASignal::uint_vector_t**
Vector of unsigned values, used for size and index description of n-dimensional matrixes.

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

Typedefs

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

Functions

- **mha_wave_t range** (**mha_wave_t** s, unsigned int k0, unsigned int len)
Return a time interval from a waveform chunk.
- **mha_spec_t channels** (**mha_spec_t** s, unsigned int ch_start, unsigned int nch)
Return a channel interval from a spectrum.
- **mha_real_t MHASignal::bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srat)
conversion from fft bin index to frequency
- **mha_real_t MHASignal::freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srat)
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. 839) 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. 839) 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. 793) structure into a std::vector< std::vector<mha_complex_t> > (outer vector represents channels).*
- **mha_wave_t & operator+=** (**mha_wave_t** &, const **mha_real_t** &)
Addition operator.
- **mha_wave_t & operator+=** (**mha_wave_t** &, const **mha_wave_t** &)
Addition operator.
- **mha_wave_t & operator-=** (**mha_wave_t** &, const **mha_wave_t** &)
Subtraction operator.
- **mha_spec_t & operator-=** (**mha_spec_t** &, const **mha_spec_t** &)
Subtraction operator.
- **mha_wave_t & operator*=** (**mha_wave_t** &, const **mha_real_t** &)
Element-wise multiplication operator.
- **mha_wave_t & operator*=** (**mha_wave_t** &, const **mha_wave_t** &)
Element-wise multiplication operator.
- **mha_spec_t & operator*=** (**mha_spec_t** &, const **mha_real_t** &)
Element-wise multiplication operator.
- **mha_spec_t & operator*=** (**mha_spec_t** &, const **mha_wave_t** &)
Element-wise multiplication operator.
- **mha_spec_t & operator*=** (**mha_spec_t** &, const **mha_spec_t** &)
Element-wise multiplication operator.
- **mha_spec_t & operator/=** (**mha_spec_t** &, const **mha_spec_t** &)
Element-wise division operator.
- **mha_wave_t & operator/=** (**mha_wave_t** &, const **mha_wave_t** &)
Element-wise division operator.
- **mha_spec_t & operator+=** (**mha_spec_t** &, const **mha_spec_t** &)
Addition operator.
- **mha_spec_t & operator+=** (**mha_spec_t** &, const **mha_real_t** &)
Addition operator.
- **mha_wave_t & operator^=** (**mha_wave_t** &self, const **mha_real_t** &arg)
Exponent operator.

- void **MHASignal::copy_channel** (**mha_spec_t** &self, const **mha_spec_t** &src, unsigned sch, unsigned dch)
Copy one channel of a source signal.
- void **MHASignal::copy_channel** (**mha_wave_t** &self, const **mha_wave_t** &src, unsigned src_channel, unsigned dest_channel)
Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen)
Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** *sqfreq_response=nullptr)
Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs** (const **mha_spec_t** &s, unsigned int channel)
Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel** (const **mha_wave_t** &s, unsigned int channel)
Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s, unsigned int channel)
Find maximal absolute value.
- **mha_real_t MHASignal::maxabs** (const **mha_wave_t** &s)
Find maximal absolute value.
- **mha_real_t MHASignal::max** (const **mha_wave_t** &s)
Find maximal value.
- **mha_real_t MHASignal::min** (const **mha_wave_t** &s)
Find minimal value.
- **mha_real_t MHASignal::sumsqr_channel** (const **mha_wave_t** &s, unsigned int channel)
Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsqr_frame** (const **mha_wave_t** &s, unsigned int frame)
Calculate sum over all channels of squared values.
- void **conjugate** (**mha_spec_t** &self)
*Replace (!) the value of this **mha_spec_t** (p. 793) with its conjugate.*

2.8.1 Detailed Description

The vector and matrix processing toolbox consists of a number of classes defined in the namespace **MHASignal** (p. 136), and many functions and operators for use with the structures **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793).

2.8.2 Typedef Documentation

2.8.2.1 **mha_real_t** `typedef float mha_real_t`

openMHA type for real numbers

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

2.8.3 Function Documentation

2.8.3.1 **range()** `mha_wave_t range (`

<code> mha_wave_t s,</code>
<code> unsigned int k0,</code>
<code> unsigned int len)</code>

Return a time interval from a waveform chunk.

A waveform chunk containing a time interval 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

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

Returns

Waveform structure representing the sub-interval.

2.8.3.2 **channels()** `mha_spec_t channels (`

<code> mha_spec_t s,</code>
<code> unsigned int ch_start,</code>
<code> unsigned int nch)</code>

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.

2.8.3.3 bin2freq() `mha_real_t MHASignal::bin2freq (`

```
    mha_real_t bin,
    unsigned fftlen,
    mha_real_t srate ) [inline]
```

conversion from fft bin index to frequency

Parameters

<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

2.8.3.4 freq2bin() `mha_real_t MHASignal::freq2bin (`

```
    mha_real_t freq,
    unsigned fftlen,
    mha_real_t srate ) [inline]
```

conversion from frequency to fft bin index

Parameters

<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

2.8.3.5 smp2rad() `mha_real_t MHASignal::smp2rad (`

```
    mha_real_t samples,
    unsigned bin,
    unsigned fftlen ) [inline]
```

conversion from delay in samples to phase shift

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

Parameters

<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 nyqvist bin cannot be delayed)
<i>ffflen</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 nyqvist, returns 0 phase shift.

2.8.3.6 rad2smp() `mha_real_t MHASignal::rad2smp (`

```
    mha_real_t phase_shift,
    unsigned bin,
    unsigned fftlen ) [inline]
```

conversion from phase shift to delay in samples

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

Parameters

<i>phase_shift</i>	phase shift in radiant
<i>bin</i>	index of fft bin, index 0 has dc (index 0 and nyqvist bin cannot be delayed)
<i>ffflen</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.

2.8.3.7 dupvec() `template<class elem_type >`

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

Duplicate last vector element to match desired size.

Parameters

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

Return values

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

2.8.3.8 dupvec_chk() `template<class elem_type >`

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

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

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

Parameters

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

Return values

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

2.8.3.9 equal_dim() [1/6] `bool equal_dim (`
`const mha_wave_t & a,`
`const mha_wave_t & b) [inline]`

Test for equal dimension of waveform structures.

2.8.3.10 equal_dim() [2/6] `bool equal_dim (`
`const mha_wave_t & a,`
`const mhaconfig_t & b) [inline]`

Test for match of waveform dimension with mhaconfig structure.

2.8.3.11 equal_dim() [3/6] `bool equal_dim (`
`const mha_spec_t & a,`
`const mha_spec_t & b) [inline]`

Test for equal dimension of spectrum structures.

2.8.3.12 equal_dim() [4/6] `bool equal_dim (`
`const mha_spec_t & a,`
`const mhaconfig_t & b) [inline]`

Test for match of spectrum dimension with mhaconfig structure.

2.8.3.13 equal_dim() [5/6] `bool equal_dim (`
`const mha_wave_t & a,`
`const mha_spec_t & b) [inline]`

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 839) use interleaved data order, while spectrum structures **mha_spec_t** (p. 793) use non-interleaved.

```
2.8.3.14 equal_dim() [6/6] bool equal_dim (
    const mha_spec_t & a,
    const mha_wave_t & b ) [inline]
```

Test for equal dimension of waveform/spectrum structures.

Warning

Waveform structures **mha_wave_t** (p. 839) use interleaved data order, while spectrum structures **mha_spec_t** (p. 793) use non-interleaved.

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

Numeric integration of a signal vector (real values)

Parameters

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

```
2.8.3.16 integrate() [2/2] void integrate (
    mha_spec_t & s )
```

Numeric integration of a signal vector (complex values)

Parameters

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

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

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

Return size of a waveform structure.

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

Return size of a spectrum structure.

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

Set all values of waveform to zero.

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

Set all values of waveform to zero.

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

Set all values of spectrum to zero.

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

Set all values of spectrum to zero.

2.8.3.25 assign() [1/3] `void assign (`
 `mha_wave_t self,`
 `mha_real_t val)` [inline]

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

Parameters

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

2.8.3.26 assign() [2/3] void assign (

```
    mha_wave_t self,  
    const mha_wave_t & val )
```

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

Parameters

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

2.8.3.27 assign() [3/3] void assign (

```
    mha_spec_t self,  
    const mha_spec_t & val )
```

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

Parameters

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

2.8.3.28 timeshift() 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

2.8.3.29 **value()** [1/8]

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

Access an element of a waveform structure.

Parameters

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

Returns

Reference to element

2.8.3.30 **value()** [2/8]

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

Constant access to an element of a waveform structure.

Parameters

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

Returns

Reference to element

2.8.3.31 value() [3/8] `mha_complex_t& value (`
`mha_spec_t * s,`
`unsigned int fr,`
`unsigned int ch)` [inline]

Access to an element of a spectrum.

Parameters

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

Returns

Reference to element

2.8.3.32 value() [4/8] `const mha_complex_t& value (`
`const mha_spec_t * s,`
`unsigned int fr,`
`unsigned int ch)` [inline]

Constant access to an element of a spectrum.

Parameters

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

Returns

Reference to element

2.8.3.33 value() [5/8] `mha_real_t& value (`
`mha_wave_t & s,`
`unsigned int fr,`
`unsigned int ch)` [inline]

Access to an element of a waveform structure.

Parameters

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

Returns

Reference to element

```
2.8.3.34 value() [6/8] const mha_real_t& value (
    const mha_wave_t & s,
    unsigned int fr,
    unsigned int ch ) [inline]
```

Constant access to an element of a waveform structure.

Parameters

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

Returns

Reference to element

```
2.8.3.35 value() [7/8] mha_complex_t& value (
    mha_spec_t & s,
    unsigned int fr,
    unsigned int ch ) [inline]
```

Access to an element of a spectrum.

Parameters

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

Returns

Reference to element

```
2.8.3.36 value() [8/8] const mha_complex_t& value (
    const mha_spec_t & s,
    unsigned int fr,
    unsigned int ch ) [inline]
```

Constant access to an element of a spectrum.

Parameters

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

Returns

Reference to element

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

Converts a **mha_wave_t** (p. 839) structure into a `std::vector<float>` (interleaved order).

Warning

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

```
2.8.3.38 std_vector_vector_float() std::vector<std::vector<float>> std_vector_<-
vector_float (
    const mha_wave_t & )
```

Converts a **mha_wave_t** (p. 839) 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.

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

Converts a **mha_spec_t** (p. 793) 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.

2.8.3.40 operator+=() [1/4] `mha_wave_t& operator+= (`
 `mha_wave_t & ,`
 `const mha_real_t &)`

Addition operator.

2.8.3.41 operator+=() [2/4] `mha_wave_t& operator+= (`
 `mha_wave_t & ,`
 `const mha_wave_t &)`

Addition operator.

2.8.3.42 operator-=() [1/2] `mha_wave_t& operator-= (`
 `mha_wave_t & ,`
 `const mha_wave_t &)`

Subtraction operator.

2.8.3.43 operator-=() [2/2] `mha_spec_t& operator-= (`
 `mha_spec_t & ,`
 `const mha_spec_t &)`

Subtraction operator.

```
2.8.3.44 operator*() [1/5] mha_wave_t& operator*= (
    mha_wave_t & ,
    const mha_real_t & )
```

Element-wise multiplication operator.

```
2.8.3.45 operator*() [2/5] mha_wave_t& operator*= (
    mha_wave_t & ,
    const mha_wave_t & )
```

Element-wise multiplication operator.

```
2.8.3.46 operator*() [3/5] mha_spec_t& operator*= (
    mha_spec_t & ,
    const mha_real_t & )
```

Element-wise multiplication operator.

```
2.8.3.47 operator*() [4/5] mha_spec_t& operator*= (
    mha_spec_t & ,
    const mha_wave_t & )
```

Element-wise multiplication operator.

```
2.8.3.48 operator*() [5/5] mha_spec_t& operator*= (
    mha_spec_t & ,
    const mha_spec_t & )
```

Element-wise multiplication operator.

2.8.3.49 operator/() [1/2] `mha_spec_t& operator/= (`
 `mha_spec_t & ,`
 `const mha_spec_t &)`

Element-wise division operator.

2.8.3.50 operator/() [2/2] `mha_wave_t& operator/= (`
 `mha_wave_t & ,`
 `const mha_wave_t &)`

Element-wise division operator.

2.8.3.51 operator+=() [3/4] `mha_spec_t& operator+= (`
 `mha_spec_t & ,`
 `const mha_spec_t &)`

Addition operator.

2.8.3.52 operator+=() [4/4] `mha_spec_t& operator+= (`
 `mha_spec_t & ,`
 `const mha_real_t &)`

Addition operator.

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

Exponent operator.

Warning

This overwrites the xor operator!

2.8.3.54 copy_channel() [1/2] `void MHASignal::copy_channel (`
 `mha_spec_t & self,`
 `const mha_spec_t & src,`
 `unsigned sch,`
 `unsigned dch)`

Copy one channel of a source signal.

Parameters

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

```
2.8.3.55 copy_channel() [2/2] void MHASignal::copy_channel (
    mha_wave_t & self,
    const mha_wave_t & src,
    unsigned src_channel,
    unsigned dest_channel )
```

Copy one channel of a source signal.

Parameters

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

```
2.8.3.56 rmslevel() [1/2] mha_real_t MHASignal::rmslevel (
    const mha_spec_t & s,
    unsigned int channel,
    unsigned int fftlen )
```

Return RMS level of a spectrum channel.

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

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

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

2.8.3.57 colored_intensity() `mha_real_t MHASignal::colored_intensity (`

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

Colored spectrum intensity.

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

Parameters

<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>	An array with one squared weighting factor for every fft bin. Array length must be equal to <i>s</i> ->num_frames. nullptr can be given for equal weighting of all frequencies.

Returns

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

2.8.3.58 maxabs() [1/3] `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

2.8.3.59 rmslevel() [2/2] `mha_real_t MHASignal::rmslevel (`
`const mha_wave_t & s,`
`unsigned int channel)`

Return RMS level of a waveform channel.

Parameters

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

Returns

RMS level in Pa

2.8.3.60 maxabs() [2/3] `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

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

Find maximal absolute value.

Parameters

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

Returns

maximum absolute value

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

Find maximal value.

Parameters

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

Returns

maximum absolute value

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

Find minimal value.

Parameters

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

Returns

maximum absolute value

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

Calculate sum of squared values in one channel.

Parameters

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

Returns

$$\sum x^2$$

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

Calculate sum over all channels of squared values.

Parameters

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

Returns

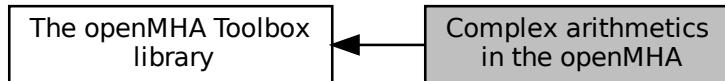
$$\sum x^2$$

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

Replace (!) the value of this **mha_spec_t** (p. 793) with its conjugate.

2.9 Complex arithmetics in the openMHA

Collaboration diagram for Complex arithmetics in the openMHA:



Classes

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

Functions

- **mha_complex_t & set (mha_complex_t &self, mha_real_t real, mha_real_t imag=0)**
*Assign real and imaginary parts to a **mha_complex_t** (p. 744) variable.*
- **mha_complex_t mha_complex (mha_real_t real, mha_real_t imag=0)**
*Create a new **mha_complex_t** (p. 744) 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. 744) 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. 744).*
- **mha_complex_t & expi (mha_complex_t &self, mha_real_t angle)**
*replaces the value of the given **mha_complex_t** (p. 744) with $\exp(i \cdot b)$.*
- **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. 744) 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)**
- **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. 744) with its conjugate.*
- **mha_complex_t _conjugate (const mha_complex_t &self)**

- **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.

2.9.1 Detailed Description

2.9.2 Function Documentation

2.9.2.1 set() [1/2] `mha_complex_t& set (`
`mha_complex_t & self,`
`mha_real_t real,`
`mha_real_t imag = 0) [inline]`

Assign real and imaginary parts to a **mha_complex_t** (p. 744) variable.

Parameters

<i>self</i>	The mha_complex_t (p. 744) 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.

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

Create a new **mha_complex_t** (p. 744) with specified real and imaginary parts.

Parameters

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

Returns

The new value.

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

Assign a **mha_complex_t** (p. 744) variable from a std::complex.

Parameters

<i>self</i>	The mha_complex_t (p. 744) variable whose value is about to change.
<i>stdcomplex</i>	The new complex value.

Returns

A reference to the changed variable.

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

Create a std::complex from **mha_complex_t** (p. 744).

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

replaces the value of the given **mha_complex_t** (p. 744) with $\exp(i \cdot b)$.

Parameters

<i>self</i>	The mha_complex_t (p. 744) variable whose value is about to change.
<i>angle</i>	The angle in the complex plane [rad].

Returns

A reference to the changed variable.

2.9.2.6 angle() `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.

2.9.2.7 operator+=() `[1/2] mha_complex_t& operator+= (`
`mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

Addition of two complex numbers, overwriting the first.

2.9.2.8 operator+() `[1/2] mha_complex_t operator+ (`
`const mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

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

2.9.2.9 operator+=() [2/2] `mha_complex_t& operator+= (`
 `mha_complex_t & self,`
 `mha_real_t other_real) [inline]`

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

2.9.2.10 operator+() [2/2] `mha_complex_t operator+ (`
 `const mha_complex_t & self,`
 `mha_real_t other_real) [inline]`

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

2.9.2.11 operator-=() [1/2] `mha_complex_t& operator-= (`
 `mha_complex_t & self,`
 `const mha_complex_t & other) [inline]`

Subtraction of two complex numbers, overwriting the first.

2.9.2.12 operator-() [1/3] `mha_complex_t operator- (`
 `const mha_complex_t & self,`
 `const mha_complex_t & other) [inline]`

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

2.9.2.13 operator-=() [2/2] `mha_complex_t& operator-= (`
 `mha_complex_t & self,`
 `mha_real_t other_real) [inline]`

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

2.9.2.14 operator-() [2/3] `mha_complex_t operator- (`
`const mha_complex_t & self,`
`mha_real_t other_real) [inline]`

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

2.9.2.15 operator*=(()) [1/2] `mha_complex_t& operator*= (`
`mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

Multiplication of two complex numbers, overwriting the first.

2.9.2.16 operator*() [1/2] `mha_complex_t operator* (`
`const mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

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

2.9.2.17 operator*=(()) [2/2] `mha_complex_t& operator*= (`
`mha_complex_t & self,`
`mha_real_t other_real) [inline]`

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

2.9.2.18 expi() [2/2] `mha_complex_t& expi (`
`mha_complex_t & self,`
`mha_real_t angle,`
`mha_real_t factor) [inline]`

replaces (!) the value of the given **mha_complex_t** (p. 744) with a $\ast \exp(i\ast b)$

Parameters

self	The mha_complex_t (p. 744) variable whose value is about to change.
angle	The imaginary exponent.
factor	The absolute value of the result.

Returns

A reference to the changed variable.

2.9.2.19 operator*() [2/2] `mha_complex_t operator* (`
`const mha_complex_t & self,`
`mha_real_t other_real) [inline]`

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

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

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

Returns

The square of the absolute value of self.

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

Compute the absolute value of a complex value.

Returns

The absolute value of self.

2.9.2.22 operator/() [1/2] `mha_complex_t& operator/= (`
`mha_complex_t & self,`
`mha_real_t other_real) [inline]`

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

2.9.2.23 operator/() [1/2] `mha_complex_t operator/ (`
`const mha_complex_t & self,`
`mha_real_t other_real) [inline]`

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

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

Safe division of two complex numbers, overwriting the first. If $\text{abs}(\text{divisor}) < \text{eps}$, then divisor is replaced by eps. $\text{eps2} = \text{eps} * \text{eps}$.

2.9.2.25 operator/=(()) [2/2] `mha_complex_t& operator/= (`
`mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

Division of two complex numbers, overwriting the first.

2.9.2.26 operator/() [2/2] `mha_complex_t operator/ (`
`const mha_complex_t & self,`
`const mha_complex_t & other) [inline]`

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

2.9.2.27 operator-() [3/3] `mha_complex_t operator- (`
`const mha_complex_t & self) [inline]`

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

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

Compare two complex numbers for equality.

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

Compare two complex numbers for inequality.

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

Replace (!) the value of this **mha_complex_t** (p. 744) with its conjugate.

```
2.9.2.31 _conjugate() 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.

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

Replace the value of this complex with its reciprocal.

2.9.2.33 `_reciprocal()` `mha_complex_t _reciprocal (`
`const mha_complex_t & self) [inline]`

compute the reciprocal of this complex value.

Returns

A temporary object holding the reciprocal value.

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

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

2.9.2.35 `normalize()` [2/2] `void normalize (`
`mha_complex_t & self,`
`mha_real_t margin) [inline]`

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

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

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

Parameters

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

Returns

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

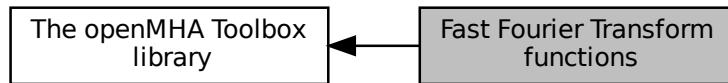
2.9.2.37 operator<() `bool operator< (`

```
    const mha_complex_t & x,  
    const mha_complex_t & y ) [inline]
```

Compares the absolute values of two complex numbers.

2.10 Fast Fourier Transform functions

Collaboration diagram for Fast Fourier Transform functions:



Typedefs

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

Functions

- `mha_fft_t mha_fft_new (unsigned int n)`
Create a new FFT handle.
- `void mha_fft_free (mha_fft_t h)`
Destroy an FFT handle.
- `void mha_fft_wave2spec (mha_fft_t h, const mha_wave_t *in, mha_spec_t *out)`
Transform waveform segment into spectrum.
- `void mha_fft_wave2spec (mha_fft_t h, const mha_wave_t *in, mha_spec_t *out, bool swaps)`
Transform waveform segment into spectrum.
- `void mha_fft_spec2wave (mha_fft_t h, const mha_spec_t *in, mha_wave_t *out)`
Transform spectrum into waveform segment.
- `void mha_fft_spec2wave (mha_fft_t h, const mha_spec_t *in, mha_wave_t *out, unsigned int offset)`
Transform 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.
- `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.

2.10.1 Detailed Description

2.10.2 Typedef Documentation

2.10.2.1 **mha_fft_t** typedef void* mha_fft_t

Handle for an FFT object.

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

2.10.3 Function Documentation

2.10.3.1 **mha_fft_new()** **mha_fft_t** mha_fft_new (unsigned int n)

Create a new FFT handle.

Parameters

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

Create a new FFT handle.

Parameters

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

Return values

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

2.10.3.2 mha_fft_free() `void mha_fft_free (`
 `mha_fft_t h)`

Destroy an FFT handle.

Parameters

<i>h</i>	Handle to be destroyed.
----------	-------------------------

Destroy an FFT handle.

Parameters

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

2.10.3.3 mha_fft_wave2spec() [1/2] `void mha_fft_wave2spec (`
 `mha_fft_t h,`
 `const mha_wave_t * in,`
 `mha_spec_t * out)`

Transform waveform segment into spectrum.

Parameters

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

Transform 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)

```
2.10.3.4 mha_fft_wave2spec() [2/2] void mha_fft_wave2spec (
    mha_fft_t h,
    const mha_wave_t * in,
    mha_spec_t * out,
    bool swaps )
```

Transform 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.

```
2.10.3.5 mha_fft_spec2wave() [1/2] void mha_fft_spec2wave (
```

```
    mha_fft_t h,
    const mha_spec_t * in,
    mha_wave_t * out )
```

Transform 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.

Transform 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)

2.10.3.6 mha_fft_spec2wave() [2/2]

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

Transform 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. *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.

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 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.

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

Complex to complex FFT (forward).

sIn and *sOut* need to have nfft bins (please note that **mha_spec_t** (p. 793) 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.

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

Complex to complex FFT (backward).

sIn and *sOut* need to have nfft bins (please note that **mha_spec_t** (p. 793) 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.

2.10.3.9 mha_fft_forward_scale()

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

Complex to complex FFT (forward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 793) 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.

2.10.3.10 mha_fft_backward_scale()

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

Complex to complex FFT (backward).

sIn and sOut need to have nfft bins (please note that **mha_spec_t** (p. 793) 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.

2.10.3.11 mha_fft_wave2spec_scale() void mha_fft_wave2spec_scale (

mha_fft_t	<i>h</i> ,
const mha_wave_t *	<i>in</i> ,
mha_spec_t *	<i>out</i>)

Transform 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.

2.10.3.12 mha_fft_spec2wave_scale() void mha_fft_spec2wave_scale (

mha_fft_t	<i>h</i> ,
const mha_spec_t *	<i>in</i> ,
mha_wave_t *	<i>out</i>)

Transform 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.

3 Namespace Documentation

3.1 ac2isl Namespace Reference

All types for the **ac2isl** (p. 78) plugins live in this namespace.

Classes

- class **ac2isl_t**
*Plugin class of **ac2isl** (p. 78).*
- class **cfg_t**
*Runtime configuration class of the **ac2isl** (p. 78) plugin.*
- class **save_var_base_t**
Interface for ac to Isl bridge variable.
- class **save_var_t**
Implementation for all ac to Isl bridges except complex types.
- class **save_var_t< mha_complex_t >**
*Template specialization of the **ac2isl** (p. 78) bridge to take care of complex numbers.*
- struct **type_info**

Variables

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

3.1.1 Detailed Description

All types for the **ac2isl** (p. 78) plugins live in this namespace.

3.1.2 Variable Documentation

3.1.2.1 **types** const std::map<int, **type_info**> ac2isl::types

3.2 ac_proc Namespace Reference

Classes

- class **interface_t**

3.3 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**

3.3.1 Detailed Description

Namespace for displaying ac variables as parser monitors.

3.4 acsave Namespace Reference

Classes

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

3.5 addsndfile Namespace Reference

Classes

- class **addsndfile_if_t**
- class **level_adapt_t**
- class **resampled_soundfile_t**
Reads sound from file and resamples it if necessary and wanted.
- class **sndfile_t**
- class **waveform_proxy_t**
*Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. [257](#)).*

Typedefs

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

Enumerations

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

*Specifies the resampling mode in **resampled_soundfile_t**.*

Functions

- static unsigned **resampled_num_frames** (unsigned num_source_frames, float source_rate, float target_rate, **addsndfile_resampling_mode_t** resampling_mode)

3.5.1 Typedef Documentation

3.5.1.1 level_adaptor `typedef MHAPlugin::config_t< level_adapt_t> addsndfile::level_adaptor`

3.5.1.2 wave_reader `typedef MHAPlugin::plugin_t< sndfile_t> addsndfile::wave_reader`

3.5.2 Enumeration Type Documentation

3.5.2.1 addsndfile_resampling_mode_t `enum addsndfile::addsndfile_resampling_mode_t`

Specifies the resampling mode in **resampled_soundfile_t** (p. 257).

Enumerator

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

3.5.3 Function Documentation

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

3.6 ADM Namespace Reference

Classes

- class **ADM**
Adaptive differential microphone, working for speech frequency range.
- class **Delay**
A delay-line class.
- 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

3.6.1 Function Documentation

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

compute IIR coefficient for subsample delay

Parameters

<i>samples</i>	Constraint: $0.0 \leq \text{samples} < 1.0$; 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

3.6.2 Variable Documentation

3.6.2.1 PI const double ADM::PI = 3.14159265358979312

3.6.2.2 C const double ADM::C = 340

3.6.2.3 DELAY_FREQ const double ADM::DELAY_FREQ = 2000

3.6.2.4 START_BETA const double ADM::START_BETA = 0.5

3.7 audiometerbackend Namespace Reference

Classes

- class **audiometer_if_t**
- class **level_adapt_t**
- class **Inn3rdcoct_t**
- class **signal_gen_t**
- class **sine_t**

TypeDefs

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

Functions

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

3.7.1 Typedef Documentation

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

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

3.7.2 Function Documentation

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

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

3.8 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.

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

3.9 coherence Namespace Reference

Classes

- class **cohfilt_if_t**
- class **cohfilt_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)

3.9.1 Function Documentation

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

3.10 cpuload Namespace Reference

Classes

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

3.11 dbasync_native Namespace Reference

Classes

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

Enumerations

- enum { **INVALID_THREAD_PRIORITY** = 999999999 }

Functions

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

3.11.1 Enumeration Type Documentation

3.11.1.1 anonymous enum anonymous enum

Enumerator

INVALID_THREAD_PRIORITY	<input type="button" value=""/>
-------------------------	---------------------------------

3.11.2 Function Documentation

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

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

3.12 dc Namespace Reference

Classes

- class `dc_if_t`
- class `dc_t`
- class `dc_vars_t`
- class `dc_vars_validator_t`

3.13 dc_simple Namespace Reference

Classes

- class `dc_if_t`
interface class
- class `dc_t`
Runtime config class for `dc_simple` (p. 86) plugin.
- class `dc_vars_t`
class for `dc_simple` (p. 86) plugin which registers variables to `MHAParser` (p. 122).
- class `dc_vars_validator_t`
Helper class to check sizes of configuration variable vectors.
- class `level_smoothen_t`

Typedefs

- `typedef MHAPlugin::plugin_t< dc_t > DC`
- `typedef MHAPlugin::config_t< level_smoothen_t > LEVEL`

Functions

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

3.13.1 Typedef Documentation

3.13.1.1 DC `typedef MHAPlugin::plugin_t< dc_t > dc_simple::DC`

3.13.1.2 LEVEL `typedef MHAPlugin::config_t< level_smoothen_t > dc_simple::LEVEL`

3.13.2 Function Documentation

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

Checks size of vector.

Parameters

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

Exceptions

MHA_Error (p. 763)	if the size of <i>v</i> is neither <i>s</i> nor 1.
---------------------------	--

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

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

Parameters

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

Returns

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

Exceptions

MHA_Error (p. 763)	if size of <i>v</i> is neither <i>s</i> nor 1.
---------------------------	--

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

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

Parameters

in	<i>x</i>	The value to check.
in	<i>comment</i>	Optional explanation for error message.

Exceptions

<i>MHA_Error</i> (p. 763)	if <i>x</i> == 0.
--	-------------------

3.14 delay Namespace Reference

Classes

- class **interface_t**

3.15 delaysum Namespace Reference

This namespace contains the delaysum plugin.

Classes

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

3.15.1 Detailed Description

This namespace contains the delaysum plugin.

3.16 delaysum_spec Namespace Reference

Classes

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

3.17 double2acvar Namespace Reference

Classes

- class **double2acvar_t**
*Plugin interface class for **double2acvar** (p. 90).*

3.18 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. 452) 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.

3.18.1 Detailed Description

dynamic compression related classes and functions

3.18.2 Function Documentation

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

One-dimensional linear interpolation.

Parameters

<i>vX</i>	Vector with input samples.
<i>vY</i>	Vector with values at input samples.
<i>X</i>	Input value to be interpolated.

Return values

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

3.18.2.2 interp2() *mha_real_t* DynComp::interp2 (

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

Linear interpolation in a two-dimensional field.

Parameters

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

Return values

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

3.19 equalize Namespace Reference**Classes**

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

3.20 fader_wave Namespace Reference

Classes

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

Typedefs

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

3.20.1 Typedef Documentation

3.20.1.1 level_adaptor `typedef MHAPlugin::plugin_t< level_adapt_t > fader_wave::level_adaptor`

3.21 fftfbpow Namespace Reference

Namespace for the fftfbpow plugin.

Classes

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

3.21.1 Detailed Description

Namespace for the fftfbpow plugin.

3.22 fftfilter Namespace Reference

Classes

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

Functions

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

3.22.1 Function Documentation

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

Return the length of the longest vector in irs.

Parameters

<i>irs</i>	"Matrix" of floats parser variable
------------	------------------------------------

Returns

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

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

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

Parameters

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

Returns

the length of the longest impulse response vector in irs.

3.23 fftfilterbank Namespace Reference**Classes**

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

3.24 fshift Namespace Reference

All types for the fshift plugin live in this namespace.

Classes

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

Functions

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

3.24.1 Detailed Description

All types for the fshift plugin live in this namespace.

3.24.2 Function Documentation**3.24.2.1 fft_find_bin()** `int fshift::fft_find_bin (`

```
    mha_real_t frequency,
    unsigned fftlen,
    mha_real_t srate )
```

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

Parameters

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

Returns

Bin number of the FFT bin corresponding to frequency

3.25 fshift_hilbert Namespace Reference

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

Classes

- class `frequency_translator_t`
- class `hilbert_shifter_t`

3.25.1 Detailed Description

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

3.26 gain Namespace Reference

Classes

- class `gain_if_t`
- class `scaler_t`

3.27 gsc_adaptive_stage Namespace Reference

Classes

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

Variables

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

3.27.1 Variable Documentation

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

Small constant to ensure no division by zero occurs.

3.28 gtfb_analyzer Namespace Reference

Classes

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

3.29 level_matching Namespace Reference

Classes

- `class channel_pair`
- `class level_matching_config_t`
- `class level_matching_t`

3.30 Isl2ac Namespace Reference

Classes

- `class cfg_t`
Runtime configuration class of the `Isl2ac` (p. 96) plugin.
- `class Isl2ac_t`
Plugin class of `Isl2ac` (p. 96).
- `class save_var_t`
LSL to AC bridge variable.

Enumerations

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

3.30.1 Enumeration Type Documentation

3.30.1.1 **overrun_behavior** enum ls12ac::overrun_behavior [strong]

Enumerator

Discard	
Ignore	

3.31 matlab_wrapper Namespace Reference

Namespace where all classes of the matlab wrapper plugin live.

Classes

- class **callback**
Utility class connecting a user_config_t instance to its corresponding configuration parser.
- class **matlab_wrapper_rt_cfg_t**
Thin wrapper around the emxArray containing the user defined configuration variables.
- class **matlab_wrapper_t**
Matlab wraper plugin interface class.
- struct **types**
- struct **types< MHA_SPECTRUM >**
- struct **types< MHA_WAVEFORM >**

3.31.1 Detailed Description

Namespace where all classes of the matlab wrapper plugin live.

3.32 matrixmixer Namespace Reference

Classes

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

3.33 mconv Namespace Reference

Classes

- class **MConv**

3.34 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. 1244) class into the AC space.*
- class **stat_t**
- class **waveform_t**

*Insert a **MHASignal::waveform_t** (p. 1259) 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.

3.34.1 Detailed Description

Functions and classes for Algorithm Communication (AC) support.

3.35 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.

3.35.1 Function Documentation

3.35.1.1 **digits()** `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> .
----------	---

3.35.1.2 snprintf_required_length() `unsigned mha_error_helpers::snprintf_required_←
length (`
`const char * formatstring,`
`...)`

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

Parameters

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

Returns

the number of bytes required by printf without the terminating nul

3.36 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. 802) handles Communication between client and server, is used on both sides.
- class **Event_Watcher**
OS-independent event watcher, uses select on Unix and WaitForMultipleObjects on Windows.
- struct **OS_EVENT_TYPE**
- class **Server**
- class **sock_init_t**
- class **Sockaccept_Event**

- class **Sockread_Event**
Watch socket for incoming data.
- class **Sockwrite_Event**
- class **Thread**
A very simple class for portable threads.
- class **Timeout_Event**
- class **Timeout_Watcher**
OS-independent event watcher with internal fixed-end-time timeout.
- class **Wakeup_Event**
A base class for asynchronous wakeup events.

TypeDefs

- typedef int **SOCKET**

Functions

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

Variables

- class **MHA_TCP::sock_init_t** **sock_initializer**

3.36.1 Detailed Description

A Namespace for TCP helper classes.

3.36.2 Typedef Documentation

3.36.2.1 SOCKET `typedef int MHA_TCP::SOCKET`

3.36.3 Function Documentation

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

Portable conversion from error number to error string.

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

Portable conversion from hostname error number to error string.

3.36.3.3 N_ERRNO() `int MHA_TCP::N_ERRNO ()`

Portable access to last network error number.

3.36.3.4 H_ERRNO() `int MHA_TCP::H_ERRNO ()`

Portable access to last hostname error number.

3.36.3.5 G_ERRNO() `int MHA_TCP::G_ERRNO ()`

Portable access to last non-network error number.

3.36.3.6 dtime() [1/2] `double MHA_TCP::dtime ()`

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

3.36.3.7 dtime() [2/2] `double MHA_TCP::dtime (const struct timeval & tv)`

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

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

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

3.36.4 Variable Documentation

3.36.4.1 sock_initializer `class MHA_TCP::sock_init_t MHA_TCP::sock_initializer`

3.37 mha_tcp Namespace Reference

namespace for network communication classes of MHA

Classes

- class **buffered_socket_t**

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

- class **server_t**

Class for accepting TCP connections from clients.

3.37.1 Detailed Description

namespace for network communication classes of MHA

3.38 mhachain Namespace Reference

Classes

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

3.39 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.

3.39.1 Detailed Description

Collection of event handling classes.

3.40 MHAFilter Namespace Reference

Namespace for IIR and FIR filter classes.

Classes

- class **adapt_filter_param_t**
- class **adapt_filter_state_t**
- class **adapt_filter_t**

Adaptive filter.
- class **blockprocessing_polyphase_resampling_t**

A class that does polyphase resampling and takes into account block processing.
- class **complex_bandpass_t**

Complex bandpass filter.
- class **diff_t**

Differentiator class (non-normalized)
- class **fftfilter_t**

FFT based FIR filter implementation.
- class **fftfilterbank_t**

FFT based FIR filterbank implementation.
- class **filter_t**

Generic IIR filter class.
- class **gamma_flt_t**

Class for gammatone filter.
- class **iir_filter_state_t**
- class **iir_filter_t**

IIR filter class wrapper for integration into parser structure.
- class **iir_ord1_real_t**

First order recursive filter.
- class **o1_ar_filter_t**

First order attack-release lowpass filter.
- class **o1flt_lowpass_t**

First order low pass filter.
- class **o1flt_maxtrack_t**

First order maximum tracker.
- class **o1flt_mintrack_t**

First order minimum tracker.
- class **partitioned_convolution_t**

A filter class for partitioned convolution.
- class **polyphase_resampling_t**

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

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

Smooth spectral gains, create a windowed impulse response.
- class **thirddoctave_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

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

3.40.1 Detailed Description

Namespace for IIR and FIR filter classes.

3.40.2 Function Documentation

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

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

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

Parameters

<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

3.40.2.3 butter_stop_ord1() `void MHAFilter::butter_stop_ord1 (`

```
    double * A,
    double * B,
    double f1,
    double f2,
    double fs )
```

Setup a first order butterworth band stop filter.

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

Return values

<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

3.40.2.4 fir_lp() `std::vector< float > MHAFilter::fir_lp (`

```
    float f_pass_,
    float f_stop_,
    float fs_,
    unsigned order_ )
```

Setup a nth order fir low pass filter.

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

Returns

vector containing filter coefficients

Precondition

`f_pass_` must be smaller or equal to `f_stop_`.

Parameters

<code>f_pass_</code>	Upper passband frequency
<code>f_stop_</code>	Lower stopband frequency
<code>fs_</code>	sample frequency

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

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

Parameters

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

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

greatest common divisor

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

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

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

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

Computes rational resampling factor from two sampling rates.

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

Parameters

<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

MHA_Error (p. 763)	if no rational resampling factor can be found.
--	--

3.41 MHAIOJack Namespace Reference

JACK IO.

Classes

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

3.41.1 Detailed Description

JACK IO.

3.42 MHAIOJackdb Namespace Reference

Classes

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

3.43 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)

3.43.1 Function Documentation

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

3.44 mhaioutils Namespace Reference

Functions

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

3.44.1 Function Documentation

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

3.45 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)
Return the JACK port latency of ports.

3.45.1 Detailed Description

Classes and functions for openMHA and JACK interaction.

3.45.2 Function Documentation

3.45.2.1 **io()** void MHAJack::io (

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

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

3.45.2.2 **get_port_capture_latency()** std::vector< unsigned int > MHAJack::get_port_capture_latency (

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

Return the JACK port latency of ports.

Parameters

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

Returns

Latency vector (one entry for each port)

3.45.2.3 **get_port_capture_latency_int()** std::vector< int > MHAJack::get_port_capture_latency_int (

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

Return the JACK port latency of ports.

Parameters

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

Returns

Latency vector (one entry for each port)

```
3.45.2.4 get_port_playback_latency() std::vector< unsigned int > MHAJack::get_port_playback_latency (
    const std::vector< std::string > & ports )
```

Return the JACK port latency of ports.

Parameters

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

Returns

Latency vector (one entry for each port)

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

3.46 MHAKernel Namespace Reference

Classes

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

Functions

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

3.46.1 Function Documentation

```
3.46.1.1 algo_comm_safe_cast() MHAKernel::algo_comm_class_t * MHAKernel::algo_←
comm_safe_cast (
    void * h )
```

3.47 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**

3.47.1 Detailed Description

Collection of classes for selecting audio chunks from multiple sources.

3.48 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 **MHAOvlFilter::fftfb_t** (p. 1002).*
- class **scale_var_t**

Typedefs

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

3.48.1 Detailed Description

Namespace for overlapping FFT based filter bank classes and functions.

3.48.2 Typedef Documentation

3.48.2.1 **scale_fun_t** `typedef mha_real_t() MHAOvlFilter::scale_fun_t(mha_real_t)`

3.49 MHAOvlFilter::barkscale Namespace Reference

Classes

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

Variables

- `mha_real_t vfreq [BARKSCALE_ENTRIES]`
- `mha_real_t vbark [BARKSCALE_ENTRIES]`

3.49.1 Variable Documentation

3.49.1.1 `vfreq mha_real_t MHAOvlFilter::barkscale::vfreq`

3.49.1.2 `vbark mha_real_t MHAOvlFilter::barkscale::vbark`

3.50 MHAOvlFilter::FreqScaleFun Namespace Reference

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

Functions

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

3.50.1 Detailed Description

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

3.50.2 Function Documentation

3.50.2.1 hz2hz() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2hz (mha_real_t x)`

Dummy scale transformation Hz to Hz.

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

Parameters

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

Returns

Frequency in Hz

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

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

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

3.50.2.5 hz2bark() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2bark (mha_real_t x)`

Transformation to bark scale.

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

Parameters

x	Input frequency in Hz
---	-----------------------

Returns

Critical band rate in Bark

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

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

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

3.50.2.9 hz2log() `mha_real_t MHAOvlFilter::FreqScaleFun::hz2log (mha_real_t x)`

Third octave frequency scale.

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

Parameters

x	Frequency in Hz
---	-----------------

Returns

Third octaves relative to 1000 Hz

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

3.51 MHAOvlFilter::ShapeFun Namespace Reference

Shape functions for overlapping filters.

Functions

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

3.51.1 Detailed Description

Shape functions for overlapping filters.

3.51.2 Function Documentation

```
3.51.2.1 rect() mha_real_t MHAOvlFilter::ShapeFun::rect (
    mha_real_t x )
```

Filter shape function for rectangular filters.

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

Parameters

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

Returns

Weigth function in the range [0,1]

3.51.2.2 linear() `mha_real_t MHAOvlFilter::ShapeFun::linear (mha_real_t x)`

Filter shape function for sawtooth filters.

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

Parameters

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

Returns

Weigth function in the range [0,1]

3.51.2.3 hann() `mha_real_t MHAOvlFilter::ShapeFun::hann (mha_real_t x)`

Filter shape function for hanning shaped filters.

This function creates hanning window shaped filters.

Parameters

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

Returns

Weigth function in the range [0,1]

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

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

3.52 MHParse 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 **mpluginloader_t**
Class to create a plugin loader in a parser, including the load logic.
- class **mint_mon_t**
Matrix of ints monitor.
- class **mint_t**
Matrix variable with int value.
- 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)

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)

Variables

- const typedef char *(* **c_parse_err_t**)(void *, int)

3.52.1 Detailed Description

Name space for the openMHA-Parser configuration language.

This namespace contains all classes which are needed for the implementation of the openMHA configuration language. For details on the script language itself please see section **The openMHA configuration language** (p. 30).

3.52.2 List of valid MHAParser items

- **Sub-parser:** `parser_t` (p. 1097)
- **Variables:**
Numeric variables: `int_t` (p. 1064), `vint_t` (p. 1127), `float_t` (p. 1058), `vfloat_t` (p. 1122),
`mfloat_t` (p. 1080)
Other variables: `string_t` (p. 1110), `vstring_t` (p. 1131), `kw_t` (p. 1070), `bool_t` (p. 1042)
- **Monitors:**
Numeric monitors: `int_mon_t` (p. 1061), `vint_mon_t` (p. 1125), `float_mon_t` (p. 1056),
`vfloat_mon_t` (p. 1120)
`mfloat_mon_t` (p. 1078)
`mcomplex_mon_t` (p. 1074)
Other monitors: `bool_mon_t` (p. 1040), `string_mon_t` (p. 1108), `vstring_mon_t` (p. 1129)

Members can be inserted into the configuration namespace by using `MHAParser::insert_item()` or the `insert_member()` (p. 1615) macro.

3.52.3 Typedef Documentation

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

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

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

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

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

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

3.52.4 Function Documentation

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

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

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

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

3.52.4.5 cfg_dump_short() std::string MHAParser::cfg_dump_short (

```
base_t * p,  
const std::string & pref )
```

3.52.4.6 all_dump() std::string MHAParser::all_dump (

```
base_t * p,  
const std::string & pref )
```

3.52.4.7 mon_dump() std::string MHAParser::mon_dump (

```
base_t * p,  
const std::string & pref )
```

3.52.4.8 all_ids() std::string MHAParser::all_ids (

```
base_t * p,  
const std::string & pref,  
const std::string & id = "" )
```

3.52.4.9 strreplace() 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

3.52.4.10 envreplace() void MHAParser::envreplace (

```
std::string & s )
```

3.52.5 Variable Documentation

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

3.53 MHAParser::StrCnv Namespace Reference

String converter namespace.

Functions

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

- std::string **val2str** (const int &)

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

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

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

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

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

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

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

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

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

Convert to string.

3.53.1 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.

3.53.2 Function Documentation

3.53.2.1 num_brackets()

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

count number of brackets

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

Parameters

s	String
---	--------

Returns

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

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

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

Convert from string.

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

Convert from string.

3.53.2.5 str2val() [3/8] void MHAParser::StrCnv::str2val (const std::string & s, mha_complex_t & v)

Convert from string.

3.53.2.6 str2val() [4/8] void MHAParser::StrCnv::str2val (const std::string & s, int & v)

Convert from string.

3.53.2.7 str2val() [5/8] void MHAParser::StrCnv::str2val (const std::string & s, **MHAParser::keyword_list_t** & v)

Convert from string.

3.53.2.8 str2val() [6/8] void MHAParser::StrCnv::str2val (const std::string & s, std::string & v)

Convert from string.

3.53.2.9 str2val() [7/8] template<class arg_t > void MHAParser::StrCnv::str2val (const std::string & s, std::vector< arg_t > & val)

Converter for vector types.

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

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

3.53.2.11 str2val() [8/8] template<class arg_t >
void MHAParser::StrCnv::str2val (const std::string & s,
std::vector< std::vector< arg_t > > & val)

Converter for matrix types.

3.53.2.12 val2str() [1/13] std::string MHAParser::StrCnv::val2str (const bool & v)

Convert to string.

3.53.2.13 val2str() [2/13] std::string MHAParser::StrCnv::val2str (const float & v)

Convert to string.

3.53.2.14 val2str() [3/13] std::string MHAParser::StrCnv::val2str (const mha_complex_t & v)

Convert to string.

3.53.2.15 val2str() [4/13] std::string MHAParser::StrCnv::val2str (const int & v)

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

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

Convert to string.

3.54 MHAPlugin Namespace Reference

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

Classes

- class **`cfg_node_t`**

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

- class **`config_t`**

Template class for thread safe configuration.

- class **`plugin_t`**

The template class for C++ openMHA plugins.

3.54.1 Detailed Description

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

3.55 MHAPlugin_Resampling Namespace Reference

Classes

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

3.56 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. 1178) 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

- enum { **INVALID_THREAD_PRIORITY** = 999999999 }
Invalid thread priority.

3.56.1 Detailed Description

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

3.56.2 Enumeration Type Documentation

3.56.2.1 anonymous enum anonymous enum

Invalid thread priority.

Enumerator

INVALID_THREAD_PRIORITY	<input type="checkbox"/>
-------------------------	--------------------------

3.57 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. 793))*
- 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. 839))*

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. 839).*
- **mha_real_t lin2db** (**mha_real_t** x, **mha_real_t** eps)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t lin2db** (**mha_real_t** x)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t db2lin** (**mha_real_t** x)
Conversion from dB scale to linear (no SPL reference)
- **mha_real_t sq2db** (**mha_real_t** x, **mha_real_t** eps=0.0f)
conversion from squared values to dB (no SPL reference)
- **mha_real_t db2sq** (**mha_real_t** x)
conversion from dB to squared values (no SPL reference)
- **mha_real_t pa2dbspl** (**mha_real_t** x, **mha_real_t** eps)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t pa2dbspl** (**mha_real_t** x)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t dbspl2pa** (**mha_real_t** x)
Conversion from dB SPL to linear Pascal scale.
- **mha_real_t pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=0.0f)
Conversion from squared Pascal scale to dB SPL.
- **mha_real_t dbspl2pa2** (**mha_real_t** x)
conversion from dB SPL to squared Pascal scale
- **mha_real_t smp2sec** (**mha_real_t** n, **mha_real_t** srate)
conversion from samples to seconds
- **mha_real_t sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
conversion from seconds to samples
- **mha_real_t bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** srate)
conversion from fft bin index to frequency
- **mha_real_t freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** srate)
conversion from frequency to fft bin index
- **mha_real_t smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
conversion from delay in samples to phase shift
- **mha_real_t rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)
conversion from phase shift to delay in samples
- template<class elem_type >
std::vector< elem_type > dupvec (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size.
- template<class elem_type >
std::vector< elem_type > dupvec_chk (std::vector< elem_type > vec, unsigned n)
Duplicate last vector element to match desired size, check for dimension.
- void **copy_channel** (**mha_spec_t** &self, const **mha_spec_t** &src, unsigned sch, unsigned dch)
Copy one channel of a source signal.

- void **copy_channel** (**mha_wave_t** &self, const **mha_wave_t** &src, unsigned src_channel, unsigned dest_channel)

Copy one channel of a source signal.
- **mha_real_t rmslevel** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen)

Return RMS level of a spectrum channel.
- **mha_real_t colored_intensity** (const **mha_spec_t** &s, unsigned int channel, unsigned int fftlen, **mha_real_t** *sqfreq_response=nullptr)

Colored spectrum intensity.
- **mha_real_t maxabs** (const **mha_spec_t** &s, unsigned int channel)

Find maximal absolute value.
- **mha_real_t rmslevel** (const **mha_wave_t** &s, unsigned int channel)

Return RMS level of a waveform channel.
- **mha_real_t maxabs** (const **mha_wave_t** &s, unsigned int channel)

Find maximal absolute value.
- **mha_real_t maxabs** (const **mha_wave_t** &s)

Find maximal absolute value.
- **mha_real_t max** (const **mha_wave_t** &s)

Find maximal value.
- **mha_real_t min** (const **mha_wave_t** &s)

Find minimal value.
- **mha_real_t sumsqr_channel** (const **mha_wave_t** &s, unsigned int channel)

Calculate sum of squared values in one channel.
- **mha_real_t sumsqr_frame** (const **mha_wave_t** &s, unsigned int frame)

Calculate sum over all channels of squared values.
- void **scale** (**mha_spec_t** *dest, const **mha_wave_t** *src)
- void **limit** (**mha_wave_t** &s, const **mha_real_t** &min, const **mha_real_t** &max)

Limit the singal in the waveform buffer to the range [min, max].
- template<class elem_type >
 elem_type **kth_smallest** (elem_type array[], unsigned n, unsigned k)

Fast search for the kth smallest element of an array.
- template<class elem_type >
 elem_type **median** (elem_type array[], unsigned n)

Fast median search.
- template<class elem_type >
 elem_type **mean** (const std::vector< elem_type > &data, elem_type start_val)

Calculate average of elements in a vector.
- template<class elem_type >
 std::vector< elem_type > **quantile** (std::vector< elem_type > data, const std::vector< elem_type > &p)

Calculate quantile of elements in a vector.
- void **saveas_mat4** (const **mha_spec_t** &data, const std::string &varname, FILE *fh)

Save a openMHA spectrum as a variable in a Matlab4 file.
- void **saveas_mat4** (const **mha_wave_t** &data, const std::string &varname, FILE *fh)

Save a openMHA waveform as a variable in a Matlab4 file.
- void **saveas_mat4** (const std::vector< **mha_real_t** > &data, const std::string &varname, FILE *fh)

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

- void **copy_permuted** (**mha_wave_t** *dest, const **mha_wave_t** *src)
Copy contents of a waveform to a permuted waveform.

Variables

- unsigned long int **signal_counter** = 0
Signal counter to produce signal ID strings.

3.57.1 Detailed Description

Namespace for audio signal handling and processing classes.

3.57.2 Function Documentation

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

Apply a function to each element of a **mha_wave_t** (p. 839).

Parameters

<i>s</i>	Pointer to a mha_wave_t (p. 839) structure
<i>fun</i>	Function to be applied (one argument)

3.57.2.2 lin2db() [1/2] **mha_real_t** MHASignal::lin2db (
 mha_real_t *x*,
 mha_real_t *eps*) [inline]

Conversion from linear scale to dB (no SPL reference)

Parameters

<i>x</i>	Linear input
<i>eps</i>	minimum linear value (if $x < \text{eps}$ --> convert <i>eps</i> instead), $\text{eps} < 0$ not allowed

Returns

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

Exceptions

MHA_Error (p. 763)	if $\text{eps} < 0$
--	---------------------

3.57.2.3 lin2db() [2/2] `mha_real_t MHASignal::lin2db (mha_real_t x) [inline]`

Conversion from linear scale to dB (no SPL reference)

Parameters

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

Returns

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

3.57.2.4 db2lin() `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.
----------	-----------

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

conversion from squared values to dB (no SPL reference)

Parameters

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

Returns

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

Exceptions

MHA_Error (p. 763)	if <i>eps</i> < 0
--	-------------------

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

conversion from dB to squared values (no SPL reference)

Parameters

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

```
3.57.2.7 pa2dbspl() [1/2] mha_real_t MHASignal::pa2dbspl (
    mha_real_t x,
    mha_real_t eps ) [inline]
```

Conversion from linear Pascal scale to dB SPL.

Parameters

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

Precondition

<code>eps >= 0</code>

Returns

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

Exceptions

MHA_Error (p. 763)	if $\text{eps} < 0$
---------------------------	---------------------

3.57.2.8 pa2dbspl() [2/2] `mha_real_t MHASignal::pa2dbspl (mha_real_t x)` [inline]

Conversion from linear Pascal scale to dB SPL.

Parameters

<code>x</code>	Linear input
----------------	--------------

Returns

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

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

Conversion from dB SPL to linear Pascal scale.

Parameters

<code>x</code>	Linear input.
----------------	---------------

3.57.2.10 pa22dbspl() `mha_real_t MHASignal::pa22dbspl (`

```
mha_real_t x,
mha_real_t eps = 0.0f ) [inline]
```

Conversion from squared Pascal scale to dB SPL.

Parameters

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

Returns

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

Exceptions

<code>MHA_Error</code> (p. 763)	if <code>eps < 0</code>
---------------------------------	----------------------------

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

conversion from dB SPL to squared Pascal scale

Parameters

<code>x</code>	dB SPL input
----------------	--------------

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

conversion from samples to seconds

Parameters

<code>n</code>	number of samples
<code>srate</code>	sampling rate / Hz

3.57.2.13 sec2smp() `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

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

3.57.2.15 limit() `void MHASignal::limit (`
 `mha_wave_t & s,`
 `const mha_real_t & min,`
 `const mha_real_t & max)`

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

Parameters

<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

3.57.2.16 kth_smallest() `template<class elem_type >`
`elem_type MHASignal::kth_smallest (`

```
elem_type array[],  
unsigned n,  
unsigned k )
```

Fast search for the kth smallest element of an array.

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

Parameters

array	Element array
-------	---------------

Postcondition

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

Parameters

n	number of elements in array
---	-----------------------------

Precondition

$n \geq 1$

Parameters

k	The k'th smalles element is returned: k = 0 returns the minimum, k = $(n-1)/2$ returns the median, k=(n-1) returns the maximum
---	--

Precondition

$k < n$

Returns

The kth smallest array element

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

Fast median search.

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

Parameters

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

Postcondition

The order of elements in the array is altered. $\text{array}[(n-1)/2]$ then holds the median.

Parameters

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

Precondition

$n \geq 1$

Returns

The median of the array elements

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

Calculate average of elements in a vector.

Parameters

<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

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

Calculate quantile of elements in a vector.

Parameters

<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.

```
3.57.2.20 saveas_mat4() [1/3] void MHASignal::saveas_mat4 (
    const mha_spec_t & data,
    const std::string & varname,
    FILE * fh )
```

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

Parameters

<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.

```
3.57.2.21 saveas_mat4() [2/3] void MHASignal::saveas_mat4 (
    const mha_wave_t & data,
    const std::string & varname,
    FILE * fh )
```

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

Parameters

<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.

3.57.2.22 `saveas_mat4()` [3/3] void MHASignal::saveas_mat4 (const std::vector< mha_real_t > & data, const std::string & varname, FILE * fh)

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

Parameters

<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.

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

Copy contents of a waveform to a permuted waveform.

Parameters

<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.

3.57.3 Variable Documentation

3.57.3.1 `signal_counter` unsigned long int MHASignal::signal_counter = 0

Signal counter to produce signal ID strings.

3.58 MHASndFile Namespace Reference

Classes

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

3.59 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.

3.59.1 Detailed Description

Namespace for table lookup classes.

3.60 MHAUtils Namespace Reference

Functions

- bool **is_multiple_of** (const unsigned big, const unsigned small)
- bool **is_power_of_two** (const unsigned n)
- bool **is_multiple_of_by_power_of_two** (const unsigned big, const unsigned small)
- std::string **strip** (const std::string &line)
- std::string **remove** (const std::string &str_, char c)
- bool **is_denormal** (mha_real_t x)
Get the normal-ness of a mha_real_t.
- bool **is_denormal** (const **mha_complex_t** &x)
Get the normal-ness of a complex number.
- bool **is_denormal** (const std::complex< **mha_real_t** > &x)
Get the normal-ness of a complex number.
- **mha_real_t spl2hl** (**mha_real_t** f)
Get the offset of between dB(SPL) and dB(HL) for a given frequency according to ISO 389-7-2005 (freefield); e.g.

3.60.1 Function Documentation

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

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

3.60.1.3 `is_multiple_of_by_power_of_two()` `bool MHAUtils::is_multiple_of_by_power←`
`_of_two (`
 `const unsigned big,`
 `const unsigned small) [inline]`

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

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

3.60.1.6 `is_denormal()` [1/3] `bool MHAUtils::is_denormal (`
 `mha_real_t x) [inline]`

Get the normal-ness of a `mha_real_t`.

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

Parameters

<i>x</i>	A mha_real_t floating point number
----------	------------------------------------

Returns

True if *x* is denormal, false otherwise

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

Get the normal-ness of a complex number.

Overload for **mha_complex_t** (p. 744). Returns true iff one or both of real and imaginary part are denormal

Parameters

<i>x</i>	[in] A mha_complex_t (p. 744) number
----------	--------------------------------------

Returns

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

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

Get the normal-ness of a complex number.

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

Parameters

<i>x</i>	[in] A mha_complex_t (p. 744) number
----------	--------------------------------------

Returns

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

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

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

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

Parameters

in	<code>f</code>	The frequency in Hz for which the offset shall be returned
----	----------------	--

Returns

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

Exceptions

<code>MHA_Error</code> (p. 763)	if $f < 0$
--	------------

3.61 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.

3.61.1 Detailed Description

Collection of Window types.

3.61.2 Function Documentation

3.61.2.1 **rect()** float MHAWindow::rect (float x)

Rectangular window function.

3.61.2.2 bartlett() float MHAWindow::bartlett (float x)

Bartlett window function.

3.61.2.3 hanning() float MHAWindow::hanning (float x)

Hanning window function.

3.61.2.4 hamming() float MHAWindow::hamming (float x)

Hamming window function.

3.61.2.5 blackman() float MHAWindow::blackman (float x)

Blackman window function.

3.62 multibandcompressor Namespace Reference

Classes

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

3.63 noise_psd_estimator Namespace Reference

Classes

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

3.64 overlapadd Namespace Reference

Classes

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

3.65 plingploing Namespace Reference

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

Classes

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

Functions

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

3.65.1 Detailed Description

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

3.65.2 Function Documentation

3.65.2.1 **drand()** double plingploing::drand (

```
double a,
double b )
```

3.66 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. 850) structures, and report differences as an error.*

3.66.1 Function Documentation

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

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

Compare two **mhaconfig_t** (p. 850) structures, and report differences as an error.

Parameters

<i>req</i>	Expected mhaconfig_t (p. 850) structure
<i>avail</i>	Available mhaconfig_t (p. 850) structure
<i>pref</i>	Prefix for error messages

3.67 plugins Namespace Reference

Namespaces

- **hoertech**

3.68 plugins::hoertech Namespace Reference

Namespaces

- **acrec**

3.69 plugins::hoertech::acrec Namespace Reference

Classes

- class **acrec_t**
Plugin interface class of plugin acrec.
- class **acwriter_t**
acwriter_t (p. 1379) decouples signal processing from writing to disk.

Functions

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

3.69.1 Function Documentation

3.69.1.1 to_iso8601() std::string plugins::hoertech::acrec::to_iso8601 (time_t tm)

3.70 rmslevel Namespace Reference

Classes

- class **mon_t**
- class **rmslevel_if_t**
Interface class of the rmslevel plugin.
- class **rmslevel_t**
Run-time configuration class of the rmslevel plugin.

Enumerations

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

3.70.1 Enumeration Type Documentation

3.70.1.1 **UNIT** enum **rmslevel::UNIT** [strong]

Enumerator

SPL	
HL	

3.71 rohBeam Namespace Reference

Classes

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

Functions

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

Variables

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

3.71.1 Function Documentation

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

Cylindrical bessel function of the first kind of order 0.

Parameters

x	the argument of the function
---	------------------------------

Returns $j_0(x)$ **3.71.2 Variable Documentation****3.71.2.1 scalarify** auto rohBeam::scalarify = [](auto t){return t(0);}**3.71.2.2 CONST_C** constexpr float rohBeam::CONST_C = 343.0115f [constexpr]**3.71.2.3 refL** constexpr int rohBeam::refL = 0 [constexpr]**3.71.2.4 refR** constexpr int rohBeam::refR = 3 [constexpr]**3.72 route Namespace Reference****Classes**

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

3.73 shadowfilter_begin Namespace Reference**Classes**

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

3.74 shadowfilter_end Namespace Reference

Classes

- class `cfg_t`
- class `shadowfilter_end_t`

3.75 smooth_cepstrum Namespace Reference

Classes

- class `smooth_cepstrum_if_t`
- class `smooth_cepstrum_t`
- class `smooth_params`

3.76 smoothgains_bridge Namespace Reference

Classes

- class `overlapadd_if_t`
- class `smoothspec_wrap_t`

3.77 testplugin Namespace Reference

Classes

- class `ac_parser_t`
- class `config_parser_t`
- class `if_t`
- class `signal_parser_t`

3.78 windnoise Namespace Reference

namespace for plugin windnoise which detects and cancels wind noise

Classes

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

3.78.1 Detailed Description

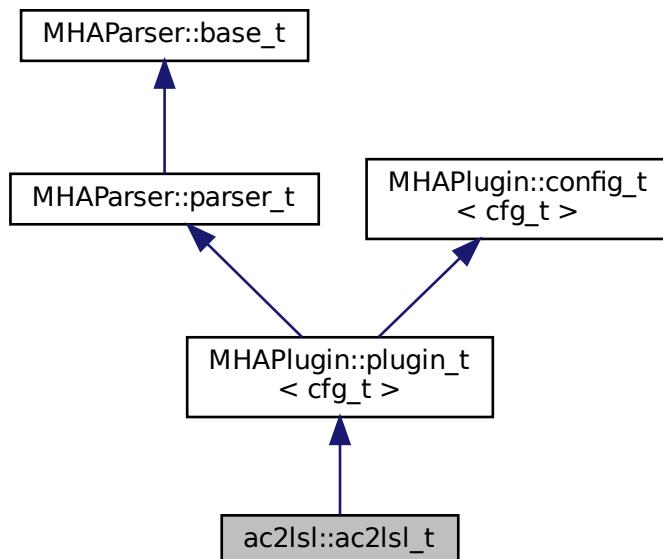
namespace for plugin windnoise which detects and cancels wind noise

4 Class Documentation

4.1 ac2lsl::ac2lsl_t Class Reference

Plugin class of **ac2lsl** (p. 78).

Inheritance diagram for ac2lsl::ac2lsl_t:



Public Member Functions

- **ac2lsl_t (algo_comm_t iac, const char *chain, const char *algo)**
- **void prepare (mhaconfig_t &)**
*Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 165).*
- **mha_wave_t * process (mha_wave_t *s)**
Processing fct for waveforms.
- **mha_spec_t * process (mha_spec_t *s)**
Processing fct for spectra.
- **void process ()**
Process function.
- **void release ()**
Release fct.

Private Member Functions

- std::vector< std::string > **get_all_names_from_ac_space** (const **algo_comm_t** & **ac**)
const
Retrieves all variable names from the AC space.
- void **update** ()
Construct new runtime configuration.

Private Attributes

- **MHAParser::vstring_t vars**
- **MHAParser::string_t source_id**
- **MHAParser::bool_t rt_strict**
- **MHAParser::bool_t activate**
- **MHAParser::int_t skip**
- **MHAEvents::patchbay_t< ac2lsl_t > patchbay**
- bool **is_first_run**

Additional Inherited Members

4.1.1 Detailed Description

Plugin class of **ac2lsl** (p. 78).

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ac2lsl_t() ac2lsl::ac2lsl_t::ac2lsl_t (

```
algo_comm_t iac,
const char * chain,
const char * algo )
```

4.1.3 Member Function Documentation

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

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 165).

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1148).

4.1.3.2 `process() [1/3]` `mha_wave_t* ac2lsl::ac2lsl_t::process (mha_wave_t * s) [inline]`

Processing fct for waveforms.

Calls **process(void)** (p. 164).

4.1.3.3 `process() [2/3]` `mha_spec_t* ac2lsl::ac2lsl_t::process (mha_spec_t * s) [inline]`

Processing fct for spectra.

Calls **process(void)** (p. 164).

4.1.3.4 `process() [3/3]` `void ac2lsl::ac2lsl_t::process ()`

Process function.

Checks once if the plugin is run in a real-time thread and throws if rt_strict is true, then forwards to **cfg_t::process()** (p. 168).

4.1.3.5 `release()` `void ac2lsl::ac2lsl_t::release () [virtual]`

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1149).

4.1.3.6 `get_all_names_from_ac_space()` `std::vector< std::string > ac2lsl::ac2lsl_t::get_all_names_from_ac_space (const algo_comm_t & ac) const [private]`

Retrieves all variable names from the AC space.

Parameters

<i>ac</i>	AC space
-----------	----------

Returns

Vector of variable names

4.1.3.7 update() `void ac2lsl::ac2lsl_t::update () [private]`

Construct new runtime configuration.

4.1.4 Member Data Documentation**4.1.4.1 vars** `MHAParser::vstring_t ac2lsl::ac2lsl_t::vars [private]`**4.1.4.2 source_id** `MHAParser::string_t ac2lsl::ac2lsl_t::source_id [private]`**4.1.4.3 rt_strict** `MHAParser::bool_t ac2lsl::ac2lsl_t::rt_strict [private]`**4.1.4.4 activate** `MHAParser::bool_t ac2lsl::ac2lsl_t::activate [private]`**4.1.4.5 skip** `MHAParser::int_t ac2lsl::ac2lsl_t::skip [private]`

4.1.4.6 patchbay `MHAEEvents::patchbay_t< ac2lsl_t> ac2lsl::ac2lsl_t::patchbay`
[private]

4.1.4.7 is_first_run `bool ac2lsl::ac2lsl_t::is_first_run` [private]

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

- `ac2lsl.cpp`

4.2 ac2lsl::cfg_t Class Reference

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

Public Member Functions

- `cfg_t` (const `algo_comm_t &ac_`, unsigned `skip_`, const std::string & `source_id`, const std::vector< std::string > & `varnames_`, double `rate`)
C'tor of ac2lsl (p. 78) run time configuration.
- void `process` ()

Private Member Functions

- void `create_or_replace_var` (const std::string & `name`, const `comm_var_t &v`)
- void `check_vars` ()
- void `update_varlist` ()

Private Attributes

- std::map< std::string, std::unique_ptr< `save_var_base_t` > > `varlist`
Maps variable name to unique ptr's of ac to Isl bridges.
- unsigned `skipcnt`
Counter of frames to skip.
- const unsigned `skip`
Number of frames to skip after each send.
- const double `srate`
Sampling rate of the stream.
- const std::string `source_id`
User configurable source id.
- const `algo_comm_t & ac`
Handle to the ac space.

4.2.1 Detailed Description

Runtime configuration class of the **ac2lsl** (p. 78) plugin.

4.2.2 Constructor & Destructor Documentation

```
4.2.2.1 cfg_t() cfg_t::cfg_t (
    const algo_comm_t & ac_,
    unsigned skip_,
    const std::string & source_id,
    const std::vector< std::string > & varnames_,
    double rate )
```

C'tor of **ac2lsl** (p. 78) run time configuration.

Parameters

<i>ac_</i>	AC space, source of data to send over LSL
<i>skip_</i>	Number of frames to skip after each send
<i>source_id_</i>	LSL identifier for this data stream
<i>varnames_</i>	Names of AC variables to send over LSL
<i>rate</i>	Rate with which chunks of data are sent to the LSL stream. Usually the rate with which process calls happen, but may be lower due to the subsampling caused by <i>skip_</i>

4.2.3 Member Function Documentation

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

4.2.3.2 check_vars() void ac2lsl::cfg_t::check_vars () [private]

4.2.3.3 update_varlist() void cfg_t::update_varlist () [private]

4.2.3.4 process() void cfg_t::process ()

4.2.4 Member Data Documentation

4.2.4.1 varlist std::map<std::string, std::unique_ptr< **save_var_base_t**> > ac2lsl::cfg_t::varlist [private]

Maps variable name to unique ptr's of ac to Isl bridges.

4.2.4.2 skipcnt unsigned ac2lsl::cfg_t::skipcnt [private]

Counter of frames to skip.

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

Number of frames to skip after each send.

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

Sampling rate of the stream.

4.2.4.5 source_id const std::string ac2isl::cfg_t::source_id [private]

User configurable source id.

4.2.4.6 ac const algo_comm_t& ac2isl::cfg_t::ac [private]

Handle to the ac space.

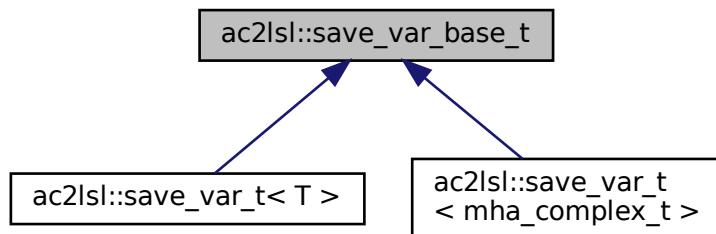
The documentation for this class was generated from the following file:

- ac2isl.cpp

4.3 ac2isl::save_var_base_t Class Reference

Interface for ac to Isl bridge variable.

Inheritance diagram for ac2isl::save_var_base_t:

**Public Member Functions**

- virtual void **send_frame** ()=0
- virtual void * **get_buf_address** () const noexcept=0
- virtual void **set_buf_address** (void *data)=0
- virtual Isl::stream_info **info** () const noexcept=0
- virtual unsigned **data_type** () const noexcept=0
- virtual unsigned **num_entries** () const noexcept=0
- virtual ~**save_var_base_t** ()=default

4.3.1 Detailed Description

Interface for ac to Isl bridge variable.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ~save_var_base_t() virtual ac2isl::save_var_base_t::~save_var_base_t ()
[virtual], [default]

4.3.3 Member Function Documentation

4.3.3.1 send_frame() virtual void ac2isl::save_var_base_t::send_frame () [pure
virtual]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 178), and **ac2isl::save_var_t<
T >** (p. 174).

4.3.3.2 get_buf_address() virtual void* ac2isl::save_var_base_t::get_buf_address ()
const [pure virtual], [noexcept]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 177), and **ac2isl::save_var_t<
T >** (p. 173).

4.3.3.3 set_buf_address() virtual void ac2isl::save_var_base_t::set_buf_address (void * data) [pure virtual]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 177), and **ac2isl::save_var_t<
T >** (p. 173).

4.3.3.4 info() virtual lsl::stream_info ac2isl::save_var_base_t::info () const
[pure virtual], [noexcept]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 177), and **ac2isl::save_var_t< T >** (p. 174).

4.3.3.5 data_type() virtual unsigned ac2isl::save_var_base_t::data_type () const
[pure virtual], [noexcept]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 178), and **ac2isl::save_var_t< T >** (p. 174).

4.3.3.6 num_entries() virtual unsigned ac2isl::save_var_base_t::num_entries ()
const [pure virtual], [noexcept]

Implemented in **ac2isl::save_var_t< mha_complex_t >** (p. 177), and **ac2isl::save_var_t< T >** (p. 174).

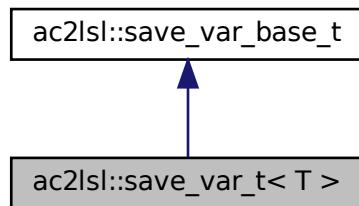
The documentation for this class was generated from the following file:

- **ac2isl.cpp**

4.4 ac2isl::save_var_t< T > Class Template Reference

Implementation for all ac to Isl bridges except complex types.

Inheritance diagram for ac2isl::save_var_t< T >:



Public Member Functions

- **save_var_t** (const std::string &name_, const std::string &type_, unsigned num_entries_, const **mha_real_t** rate_, const Isl::channel_format_t format_, const std::string &source_id_, void *data_, const unsigned **data_type_**)

C'tor of generic ac to Isl bridge.
- virtual void * **get_buf_address** () const noexcept override

Get buffer address as void pointer.
- virtual void **set_buf_address** (void *data) override

Cast the input pointer to the appropriate type and set the buffer address.
- virtual Isl::stream_info **info** () const noexcept override

Get stream info object from stream outlet.
- virtual unsigned **num_entries** () const noexcept override

Get number of entries in the stream object.
- virtual unsigned **data_type** () const noexcept override

Get data type id according MHA convention.
- virtual ~**save_var_t** ()=default
- virtual void **send_frame** () override

Send a frame to Isl.

Private Attributes

- Isl::stream_outlet **stream**

LSL stream outlet.
- T * **buf**

Pointer to data buffer of the ac variable.
- const unsigned **data_type_**

Data type id according to MHA convention.

4.4.1 Detailed Description

```
template<typename T>
class ac2isl::save_var_t< T >
```

Implementation for all ac to Isl bridges except complex types.

4.4.2 Constructor & Destructor Documentation

```
4.4.2.1 save_var_t() template<typename T >
ac2isl::save_var_t< T >:: save_var_t (
    const std::string & name_,
    const std::string & type_,
    unsigned num_entries_,
    const mha_real_t rate_,
    const lsl::channel_format_t format_,
    const std::string & source_id_,
    void * data_,
    const unsigned data_type_ ) [inline]
```

C'tor of generic ac to Isl bridge.

Parameters

<i>info</i>	LSL stream info object containing metadata
<i>data</i>	Pointer to data buffer of the ac variable
<i>data_type</i>	Type id of the stream, in mha convention. Should be set to one if not a vector.

```
4.4.2.2 ~save_var_t() template<typename T >
virtual ac2isl::save_var_t< T >::~ save_var_t ( ) [virtual], [default]
```

4.4.3 Member Function Documentation

```
4.4.3.1 get_buf_address() template<typename T >
virtual void* ac2isl::save_var_t< T >::get_buf_address ( ) const [inline], [override],
[virtual], [noexcept]
```

Get buffer address as void pointer.

Returns

Adress of the data buffer

Implements [ac2isl::save_var_base_t](#) (p. 170).

```
4.4.3.2 set_buf_address() template<typename T >
virtual void ac2isl::save_var_t< T >::set_buf_address (
    void * data ) [inline], [override], [virtual]
```

Cast the input pointer to the appropriate type and set the buffer address.

Parameters

<code>data</code>	New buffer address
-------------------	--------------------

Implements `ac2lsl::save_var_base_t` (p. 170).

4.4.3.3 info() template<typename T >
 virtual lsl::stream_info `ac2lsl::save_var_t< T >::info` () const [inline], [override], [virtual], [noexcept]

Get stream info object from stream outlet.

Implements `ac2lsl::save_var_base_t` (p. 170).

4.4.3.4 num_entries() template<typename T >
 virtual unsigned `ac2lsl::save_var_t< T >::num_entries` () const [inline], [override], [virtual], [noexcept]

Get number of entries in the stream object.

Implements `ac2lsl::save_var_base_t` (p. 171).

4.4.3.5 data_type() template<typename T >
 virtual unsigned `ac2lsl::save_var_t< T >::data_type` () const [inline], [override], [virtual], [noexcept]

Get data type id according MHA convention.

Implements `ac2lsl::save_var_base_t` (p. 171).

4.4.3.6 send_frame() template<typename T >
 virtual void `ac2lsl::save_var_t< T >::send_frame` () [inline], [override], [virtual]

Send a frame to lsl.

Implements `ac2lsl::save_var_base_t` (p. 170).

4.4.4 Member Data Documentation

4.4.4.1 stream template<typename T >

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

LSL stream outlet.

Interface to lsl

4.4.4.2 buf template<typename T >

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

Pointer to data buffer of the ac variable.

4.4.4.3 data_type_ template<typename T >

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

Data type id according to MHA convention.

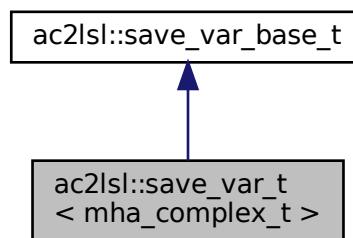
The documentation for this class was generated from the following file:

- ac2lsl.cpp

4.5 ac2lsl::save_var_t< mha_complex_t > Class Reference

Template specialization of the **ac2lsl** (p. 78) bridge to take care of complex numbers.

Inheritance diagram for ac2lsl::save_var_t< mha_complex_t >:



Public Member Functions

- **save_var_t** (const std::string &name_, const std::string &type_, const unsigned num_entries_, const **mha_real_t** rate_, const Isl::channel_format_t format_, const std::string &source_id_, void *data_)

C'tor of specialization for complex types.
- virtual void * **get_buf_address** () const noexcept override
- virtual void **set_buf_address** (void *data) override
- virtual Isl::stream_info **info** () const noexcept override

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

Get number of entries in the stream object.
- virtual unsigned **data_type** () const noexcept override

Cast the input pointer to the appropriate type and set the buffer address.
- virtual ~**save_var_t** ()=default
- virtual void **send_frame** () override

Send a frame of complex types.

Private Attributes

- Isl::stream_outlet **stream**

LSL stream outlet.
- **mha_complex_t** * **buf**

Pointer to data buffer of the ac variable.

4.5.1 Detailed Description

Template specialization of the **ac2isl** (p. 78) bridge to take care of complex numbers.

This specialization is needed because Isl does not support complex numbers. Order is [re(0), im(0), re(1), im(1),]

4.5.2 Constructor & Destructor Documentation

```
4.5.2.1 save_var_t() ac2lsl::save_var_t< mha_complex_t >:: save_var_t (
    const std::string & name_,
    const std::string & type_,
    const unsigned num_entries_,
    const mha_real_t rate_,
    const lsl::channel_format_t format_,
    const std::string & source_id_,
    void * data_ ) [inline]
```

C'tor of specialization for complex types.

See generic c'tor for details.

```
4.5.2.2 ~save_var_t() virtual ac2lsl::save_var_t< mha_complex_t >::~ save_var_t
( ) [virtual], [default]
```

4.5.3 Member Function Documentation

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

Implements [ac2lsl::save_var_base_t](#) (p. 170).

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

Implements [ac2lsl::save_var_base_t](#) (p. 170).

```
4.5.3.3 info() virtual lsl::stream_info ac2lsl::save_var_t< mha_complex_t >::info
( ) const [inline], [override], [virtual], [noexcept]
```

Get buffer address as void pointer.

Returns

Adress of the data buffer

Implements [ac2lsl::save_var_base_t](#) (p. 170).

4.5.3.4 num_entries() `virtual unsigned ac2lsl::save_var_t< mha_complex_t >::num_entries () const [inline], [override], [virtual], [noexcept]`

Get number of entries in the stream object.

Implements `ac2lsl::save_var_base_t` (p. 171).

4.5.3.5 data_type() `virtual unsigned ac2lsl::save_var_t< mha_complex_t >::data_type () const [inline], [override], [virtual], [noexcept]`

Cast the input pointer to the appropriate type and set the buffer address.

Parameters

<code>data</code>	New buffer address
-------------------	--------------------

Implements `ac2lsl::save_var_base_t` (p. 171).

4.5.3.6 send_frame() `virtual void ac2lsl::save_var_t< mha_complex_t >::send_frame () [inline], [override], [virtual]`

Send a frame of complex types.

Complex numbers are stored as alternating real and imaginary parts. An array of complex numbers in memory can be reinterpreted as a vector of real numbers that correspond to real and imaginary parts. LSL does not support complex types directly. Send one vector containing {buf[0].re,buf[0].im,buf[1].re,buf[1].im,...} instead.

Implements `ac2lsl::save_var_base_t` (p. 170).

4.5.4 Member Data Documentation

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

LSL stream outlet.

Interface to lsl

4.5.4.2 buf `mha_complex_t* ac2lsl::save_var_t< mha_complex_t >::buf` [private]

Pointer to data buffer of the ac variable.

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

- `ac2lsl.cpp`

4.6 ac2lsl::type_info Struct Reference

Public Attributes

- `const std::string name`
- `const lsl::channel_format_t format`

4.6.1 Member Data Documentation

4.6.1.1 name `const std::string ac2lsl::type_info::name`

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

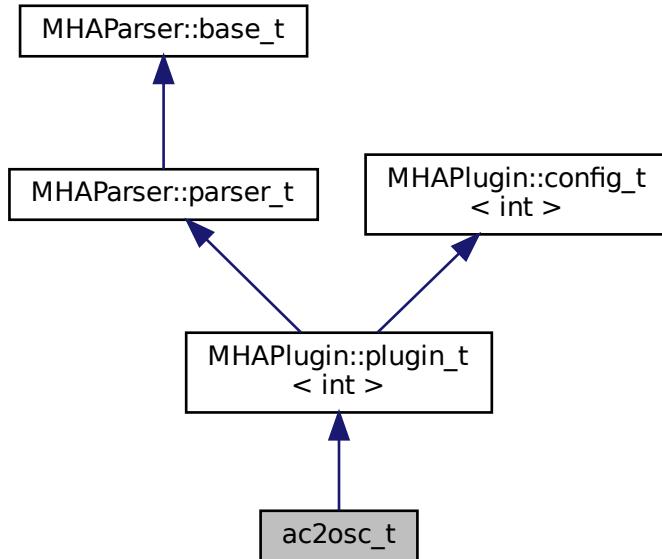
The documentation for this struct was generated from the following file:

- `ac2lsl.cpp`

4.7 ac2osc_t Class Reference

Plugin class of the ac2osc plugin.

Inheritance diagram for ac2osc_t:



Public Member Functions

- **ac2osc_t (algo_comm_t iac, const char *chain, const char *algo)**
C'tor of plugin class.
- **void prepare (mhaconfig_t &)**
- **mha_wave_t * process (mha_wave_t *s)**
Processing fct for waveforms.
- **mha_spec_t * process (mha_spec_t *s)**
Processing fct for spectra.
- **void process ()**
Process function.
- **void release ()**
Release frees osc related memory, does cleanup.

Private Member Functions

- **void send_osc_float ()**
- **void update_mode ()**
Start/Stop sending of messages.

Private Attributes

- **MHAParser::string_t host**
OSC server host name.
- **MHAParser::string_t port**
OSC server port.
- **MHAParser::int_t ttl**
Time-to-live of UDP packages.
- **MHAParser::vstring_t vars**
List of AC variables to be saved, empty for all.
- **MHAParser::kw_t mode**
Record mode.
- **MHAParser::int_t skip**
number of frames to skip after sending
- **MHAParser::bool_t rt_strict**
abort if used in real-time thread?
- std::unique_ptr< **MHA_AC::acspace2matrix_t** > **acspace**
- **MHAEvents::patchbay_t**< **ac2osc_t** > **patchbay**
- bool **b_record**
- uint8_t * **rtmem**
- float **framerate**
- int **skipcnt**
- lo_address **lo_addr**
- bool **is_first_run**

Additional Inherited Members

4.7.1 Detailed Description

Plugin class of the ac2osc plugin.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ac2osc_t() ac2osc_t::ac2osc_t (

```
    algo_comm_t iac,
    const char * chain,
    const char * algo )
```

C'tor of plugin class.

4.7.3 Member Function Documentation

4.7.3.1 `prepare()` `void ac2osc_t::prepare (mhaconfig_t & cf) [virtual]`

Implements `MHAPlugin::plugin_t< int >` (p. [1148](#)).

4.7.3.2 `process()` [1/3] `mha_wave_t* ac2osc_t::process (mha_wave_t * s) [inline]`

Processing fct for waveforms.

Calls `process(void)` (p. [182](#)).

4.7.3.3 `process()` [2/3] `mha_spec_t* ac2osc_t::process (mha_spec_t * s) [inline]`

Processing fct for spectra.

Calls `process(void)` (p. [182](#)).

4.7.3.4 `process()` [3/3] `void ac2osc_t::process ()`

Process function.

Checks once if the plugin is run in a real-time thread and throws if rt_strict is true, sends osc messages according to config.

4.7.3.5 `release()` `void ac2osc_t::release () [virtual]`

Release frees osc related memory, does cleanup.

Reimplemented from `MHAPlugin::plugin_t< int >` (p. [1149](#)).

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

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

Start/Stop sending of messages.

4.7.4 Member Data Documentation

4.7.4.1 host `MHAParser::string_t` ac2osc_t::host [private]

OSC server host name.

4.7.4.2 port `MHAParser::string_t` ac2osc_t::port [private]

OSC server port.

4.7.4.3 ttl `MHAParser::int_t` ac2osc_t::ttl [private]

Time-to-live of UDP packages.

4.7.4.4 vars `MHAParser::vstring_t` ac2osc_t::vars [private]

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

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

Record mode.

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

number of frames to skip after sending

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

abort if used in real-time thread?

4.7.4.8 acspace `std::unique_ptr< MHA_AC::acspace2matrix_t> ac2osc_t::acspace [private]`**4.7.4.9 patchbay** `MHAEVENTS::patchbay_t< ac2osc_t> ac2osc_t::patchbay [private]`**4.7.4.10 b_record** `bool ac2osc_t::b_record [private]`**4.7.4.11 rtmem** `uint8_t* ac2osc_t::rtmem [private]`**4.7.4.12 framerate** `float ac2osc_t::framerate [private]`

4.7.4.13 skipcnt int ac2osc_t::skipcnt [private]

4.7.4.14 lo_addr lo_address ac2osc_t::lo_addr [private]

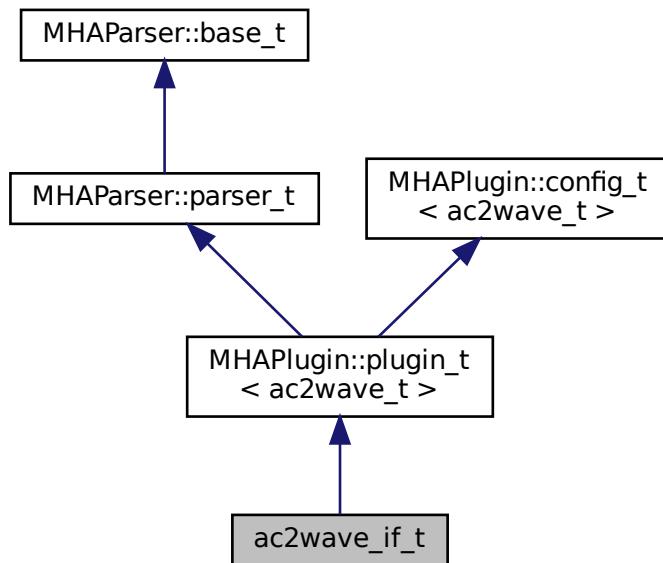
4.7.4.15 is_first_run bool ac2osc_t::is_first_run [private]

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

- **ac2osc.cpp**

4.8 ac2wave_if_t Class Reference

Inheritance diagram for ac2wave_if_t:



Public Member Functions

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

Private Member Functions

- `void update ()`

Private Attributes

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

Additional Inherited Members

4.8.1 Constructor & Destructor Documentation

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

4.8.2 Member Function Documentation

4.8.2.1 process() [1/2] `mha_wave_t * ac2wave_if_t::process (mha_spec_t *)`

4.8.2.2 process() [2/2] `mha_wave_t * ac2wave_if_t::process (mha_wave_t * s)`

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

Implements `MHAParser::plugin_t< ac2wave_t >` (p. 1148).

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

Reimplemented from `MHAParser::plugin_t< ac2wave_t >` (p. 1149).

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

4.8.3 Member Data Documentation

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

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

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

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

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

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

4.8.3.7 prepared `bool ac2wave_if_t::prepared [private]`

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

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

- **ac2wave.cpp**

4.9 ac2wave_t Class Reference

Public Member Functions

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

Private Attributes

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

4.9.1 Constructor & Destructor Documentation

```
4.9.1.1 ac2wave_t() 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_ )
```

4.9.2 Member Function Documentation

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

4.9.3 Member Data Documentation

```
4.9.3.1 frames unsigned int ac2wave_t::frames [private]
```

4.9.3.2 channels `unsigned int ac2wave_t::channels [private]`

4.9.3.3 w `mha_wave_t ac2wave_t::w [private]`

4.9.3.4 ac `algo_comm_t ac2wave_t::ac [private]`

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

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

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

4.9.3.8 gain_in `mha_real_t ac2wave_t::gain_in [private]`

4.9.3.9 gain_ac `mha_real_t ac2wave_t::gain_ac [private]`

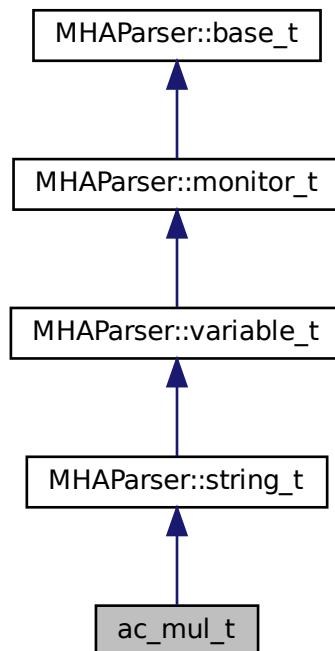
The documentation for this class was generated from the following file:

- **ac2wave.cpp**

4.10 ac_mul_t Class Reference

The class which implements the **ac_mul_t** (p. 191) plugin.

Inheritance diagram for ac_mul_t:



Public Member Functions

- **ac_mul_t (algo_comm_t, std::string, std::string)**
Plugin constructor.
- **void prepare_ (mhaconfig_t &)**
*Prepare method, called **prepare_()** (p. 193) with trailing underscore because **ac_mul_t** (p. 191) does not inherit from **plugin_t<>**.*
- **void release_ ()**
- **mha_wave_t * process (mha_wave_t *)**
- **mha_spec_t * process (mha_spec_t *)**

Private Member Functions

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

Private Attributes

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

Additional Inherited Members

4.10.1 Detailed Description

The class which implements the **ac_mul_t** (p. 191) plugin.

Different from most other plugins, the **ac_mul** plugin's interface class does not inherit from **plugin_t<>**, but from **MHAParser::string_t** (p. 1110). This way, it does not get inserted into the MHA configuration tree as a parser node which can have multiple variables, but as a string variable.

The **ac_mul_t** (p. 191) variable multiplies two AC variables element-wise.

4.10.2 Constructor & Destructor Documentation

```
4.10.2.1 ac_mul_t() ac_mul_t::ac_mul_t (
    algo_comm_t iac,
    std::string chain,
    std::string ialgo )
```

Plugin constructor.

4.10.3 Member Function Documentation

```
4.10.3.1 prepare_() void ac_mul_t::prepare_ (
    mhaconfig_t & )
```

Prepare method, called **prepare_()** (p. 193) with trailing underscore because **ac_mul_t** (p. 191) does not inherit from plugin_t<>.

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

```
4.10.3.2 release_() void ac_mul_t::release_ ( )
```

```
4.10.3.3 process() [1/3] mha_wave_t * ac_mul_t::process (
    mha_wave_t * s )
```

```
4.10.3.4 process() [2/3] mha_spec_t * ac_mul_t::process (
    mha_spec_t * s )
```

```
4.10.3.5 scan_syntax() void ac_mul_t::scan_syntax ( ) [private]
```

4.10.3.6 `get_arg_type_and_dimension()` [1/2] void ac_mul_t::get_arg_type_and_dimension () [private]

4.10.3.7 `get_arg_type_and_dimension()` [2/2] void ac_mul_t::get_arg_type_and_dimension (const std::string & name, **val_type_t** & vt, unsigned int & num_frames, unsigned int & num_channels) [private]

4.10.3.8 `process()` [3/3] void ac_mul_t::process () [private]

4.10.3.9 `process_rr()` void ac_mul_t::process_rr () [private]

4.10.3.10 `process_rc()` void ac_mul_t::process_rc () [private]

4.10.3.11 `process_cr()` void ac_mul_t::process_cr () [private]

4.10.3.12 `process_cc()` void ac_mul_t::process_cc () [private]

4.10.4 Member Data Documentation

4.10.4.1 `ac algo_comm_t` ac_mul_t::ac [private]

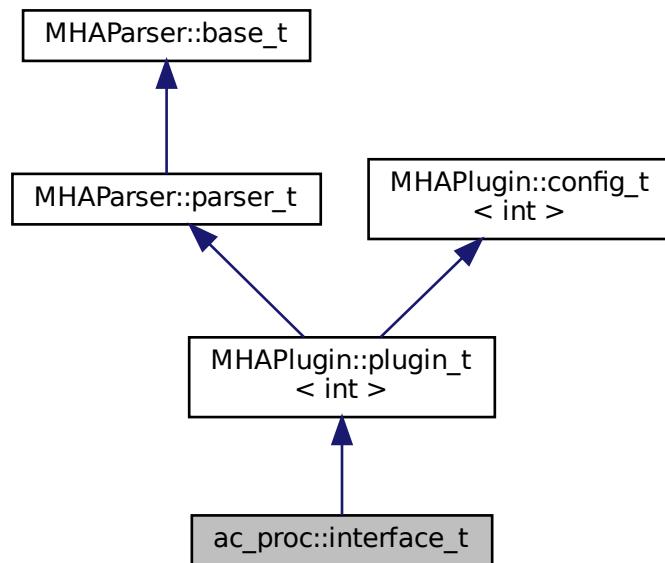
4.10.4.2 algo std::string ac_mul_t::algo [private]**4.10.4.3 argt** arg_type_t ac_mul_t::argt [private]**4.10.4.4 str_a** std::string ac_mul_t::str_a [private]**4.10.4.5 str_b** std::string ac_mul_t::str_b [private]**4.10.4.6 res_r** MHA_AC::waveform_t* ac_mul_t::res_r [private]**4.10.4.7 res_c** MHA_AC::spectrum_t* ac_mul_t::res_c [private]**4.10.4.8 num_frames** unsigned int ac_mul_t::num_frames [private]**4.10.4.9 num_channels** unsigned int ac_mul_t::num_channels [private]

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

- ac_mul.hh
- ac_mul.cpp

4.11 ac_proc::interface_t Class Reference

Inheritance diagram for ac_proc::interface_t:



Public Member Functions

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

Private Attributes

- `std::string algo`
- `MHAParser::mhapluginloader_t plug`
- `MHAParser::string_t input`
- `MHAParser::bool_t permute`
- `MHA_AC::waveform_t * s_out`
- `MHASignal::waveform_t * s_in_perm`
- `bool b_permute`
- `mha_wave_t s_in`

Additional Inherited Members

4.11.1 Constructor & Destructor Documentation

```
4.11.1.1 interface_t() ac_proc::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

4.11.2 Member Function Documentation

```
4.11.2.1 prepare() void ac_proc::interface_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< int >** (p. [1148](#)).

```
4.11.2.2 release() void ac_proc::interface_t::release ( ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< int >** (p. [1149](#)).

```
4.11.2.3 process() [1/3] void ac_proc::interface_t::process ( )
```

4.11.2.4 process() [2/3] `mha_spec_t * ac_proc::interface_t::process (mha_spec_t * s)`

4.11.2.5 process() [3/3] `mha_wave_t * ac_proc::interface_t::process (mha_wave_t * s)`

4.11.3 Member Data Documentation

4.11.3.1 algo `std::string ac_proc::interface_t::algo` [private]

4.11.3.2 plug `MHAParser::mhapluginloader_t ac_proc::interface_t::plug` [private]

4.11.3.3 input `MHAParser::string_t ac_proc::interface_t::input` [private]

4.11.3.4 permute `MHAParser::bool_t ac_proc::interface_t::permute` [private]

4.11.3.5 s_out `MHA_AC::waveform_t* ac_proc::interface_t::s_out` [private]

4.11.3.6 s_in_perm `MHASignal::waveform_t* ac_proc::interface_t::s_in_perm` [private]

4.11.3.7 **b_permute** bool ac_proc::interface_t::b_permute [private]

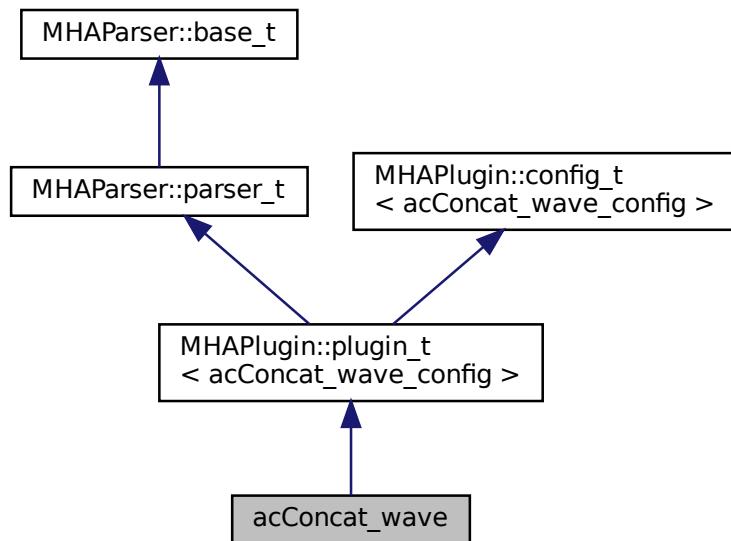
4.11.3.8 **s_in mha_wave_t** ac_proc::interface_t::s_in [private]

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

- **ac_proc.cpp**

4.12 acConcat_wave Class Reference

Inheritance diagram for acConcat_wave:



Public Member Functions

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

Public Attributes

- **MHAParser::int_t num_AC**
- **MHAParser::string_t prefix_names_AC**
- **MHAParser::vint_t samples_AC**
- **MHAParser::string_t name_con_AC**
- **MHAParser::int_t numchannels**

Private Member Functions

- void **update_cfg ()**

Private Attributes

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

Additional Inherited Members**4.12.1 Constructor & Destructor Documentation**

4.12.1.1 acConcat_wave() `acConcat_wave::acConcat_wave (algo_comm_t & ac, const std::string & chain_name, const std::string & algo_name)`

Constructs our plugin.

4.12.1.2 ~acConcat_wave() `acConcat_wave::~acConcat_wave ()`

4.12.2 Member Function Documentation

4.12.2.1 process() `mha_wave_t * acConcat_wave::process (mha_wave_t * signal)`

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

4.12.2.2 prepare() `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. [1148](#)).

4.12.2.3 release() void acConcat_wave::release (
void) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin_t< acConcat_wave_config >** (p. [1149](#)).

4.12.2.4 update_cfg() void acConcat_wave::update_cfg () [private]

4.12.3 Member Data Documentation

4.12.3.1 num_AC MHAParser::int_t acConcat_wave::num_AC

4.12.3.2 prefix_names_AC MHAParser::string_t acConcat_wave::prefix_names_AC

4.12.3.3 samples_AC MHAParser::vint_t acConcat_wave::samples_AC

4.12.3.4 name_con_AC MHAParser::string_t acConcat_wave::name_con_AC

4.12.3.5 numchannels `MHAParser::int_t acConcat_wave::numchannels`**4.12.3.6 patchbay** `MHAEvents::patchbay_t< acConcat_wave> acConcat_wave::patchbay`
[private]

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

- `acConcat_wave.h`
- `acConcat_wave.cpp`

4.13 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`

4.13.1 Constructor & Destructor Documentation**4.13.1.1 acConcat_wave_config()** `acConcat_wave_config::acConcat_wave_config (`
`algo_comm_t & ac,`
`const mhaconfig_t in_cfg,`
`acConcat_wave * _concat)`

4.13.1.2 ~acConcat_wave_config() acConcat_wave_config::~acConcat_wave_config ()

4.13.2 Member Function Documentation

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

4.13.3 Member Data Documentation

4.13.3.1 ac algo_comm_t& acConcat_wave_config::ac

4.13.3.2 strNames_AC std::vector<std::string> acConcat_wave_config::strNames_AC

4.13.3.3 numSamples_AC std::vector<int> acConcat_wave_config::numSamples_AC

4.13.3.4 vGCC mha_wave_t acConcat_wave_config::vGCC

4.13.3.5 vGCC_con MHA_AC::waveform_t* acConcat_wave_config::vGCC_con

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

- acConcat_wave.h
- acConcat_wave.cpp

4.14 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

4.14.1 Detailed Description

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

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ac_monitor_t() `acmon::ac_monitor_t::ac_monitor_t (`

<code>MHAParser::parser_t & parent,</code>
<code>const std::string & name_,</code>
<code>algo_comm_t ac,</code>
<code>bool use_matrix)</code>

Converts AC variable to parser monitor.

Parameters

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

4.14.3 Member Function Documentation

4.14.3.1 getvar() `void acmon::ac_monitor_t::getvar (`

<code>algo_comm_t ac)</code>

Update values of monitor.

Parameters

<code>ac</code>	Handle to algorithm communication space
-----------------	---

4.14.4 Member Data Documentation

4.14.4.1 name `std::string acmon::ac_monitor_t::name`

name of AC variable and parser monitor

4.14.4.2 dimstr `std::string acmon::ac_monitor_t::dimstr`

columns x rows

4.14.4.3 mon `MHAParser::vfloat_mon_t acmon::ac_monitor_t::mon`

Monitor used for real vectors.

4.14.4.4 mon_mat `MHAParser::mfloat_mon_t acmon::ac_monitor_t::mon_mat`

Monitor used for real matrices.

4.14.4.5 mon_complex `MHAParser::vcomplex_mon_t acmon::ac_monitor_t::mon_complex`

monitor used for complex vectors

4.14.4.6 mon_mat_complex `MHAParser::mcomplex_mon_t acmon::ac_monitor_t::mon_←
mat_complex`

monitor used for complex matrices

4.14.4.7 p_parser `MHAParser::parser_t& acmon::ac_monitor_t::p_parser`

parent parser to insert monitor into

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

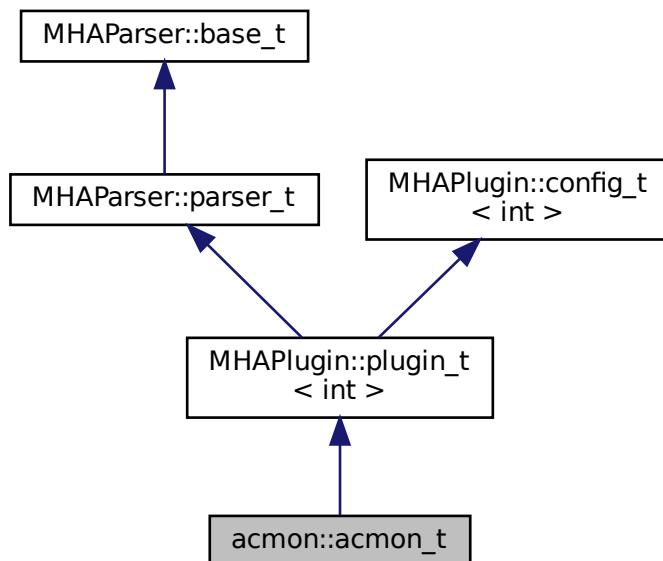
if true, use matrix monitor, else use vector monitor

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

- `ac_monitor_type.hh`
- `ac_monitor_type.cpp`

4.15 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

4.15.1 Constructor & Destructor Documentation

4.15.1.1 acmon_t() acmon::acmon_t::acmon_t (

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

4.15.1.2 ~acmon_t() acmon::acmon_t::~acmon_t ()

4.15.2 Member Function Documentation

4.15.2.1 `prepare()` void acmon::acmon_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPlugIn::plugin_t< int >** (p. 1148).

4.15.2.2 `release()` void acmon::acmon_t::release () [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin_t< int >** (p. 1149).

4.15.2.3 `process()` [1/2] mha_spec_t * acmon::acmon_t::process (
 mha_spec_t * s)

4.15.2.4 `process()` [2/2] mha_wave_t * acmon::acmon_t::process (
 mha_wave_t * s)

4.15.2.5 `save_vars()` void acmon::acmon_t::save_vars () [private]

4.15.2.6 `update_recmode()` void acmon::acmon_t::update_recmode () [private]

4.15.3 Member Data Documentation

4.15.3.1 `ac` algo_comm_t acmon::acmon_t::ac [private]

4.15.3.2 varlist `MHAParser::vstring_mon_t acmon::acmon_t::varlist` [private]

4.15.3.3 dimensions `MHAParser::vstring_mon_t acmon::acmon_t::dimensions` [private]

4.15.3.4 dispmode `MHAParser::kw_t acmon::acmon_t::dispmode` [private]

4.15.3.5 recmode `MHAParser::kw_t acmon::acmon_t::recmode` [private]

4.15.3.6 vars `std::vector< ac_monitor_t*> acmon::acmon_t::vars` [private]

4.15.3.7 patchbay `MHAEvents::patchbay_t< acmon_t> acmon::acmon_t::patchbay` [private]

4.15.3.8 chain `std::string acmon::acmon_t::chain` [private]

4.15.3.9 algo `std::string acmon::acmon_t::algo` [private]

4.15.3.10 b_cont `bool acmon::acmon_t::b_cont` [private]

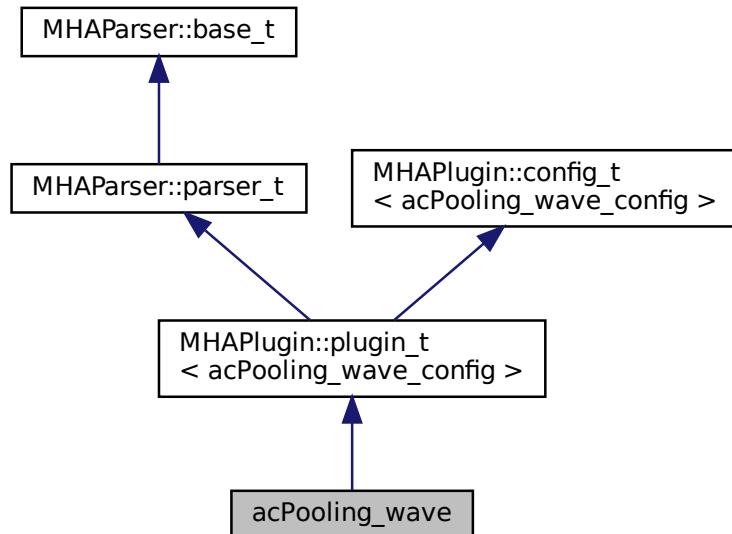
4.15.3.11 b_snapshot bool acmon::acmon_t::b_snapshot [private]

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

- `acmon.cpp`

4.16 acPooling_wave Class Reference

Inheritance diagram for acPooling_wave:



Public Member Functions

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

Public Attributes

- `MHAParser::int_t numsamples`
- `MHAParser::int_t pooling_wndlen`
- `MHAParser::kw_t pooling_type`
- `MHAParser::float_t upper_threshold`
- `MHAParser::float_t lower_threshold`
- `MHAParser::int_t neighbourhood`
- `MHAParser::float_t alpha`
- `MHAParser::string_t p_name`
- `MHAParser::string_t p_biased_name`
- `MHAParser::string_t pool_name`
- `MHAParser::string_t max_pool_ind_name`
- `MHAParser::string_t like_ratio_name`
- `MHAParser::vfloat_t prob_bias`

Private Member Functions

- `void update_cfg ()`

Private Attributes

- `MHAEvents::patchbay_t< acPooling_wave > patchbay`

Additional Inherited Members**4.16.1 Constructor & Destructor Documentation**

4.16.1.1 acPooling_wave() `acPooling_wave::acPooling_wave (`
`algo_comm_t & ac,`
`const std::string & chain_name,`
`const std::string & algo_name)`

Constructs our plugin.

4.16.1.2 ~acPooling_wave() `acPooling_wave::~acPooling_wave ()`

4.16.2 Member Function Documentation

4.16.2.1 process() `mha_wave_t * acPooling_wave::process (mha_wave_t * signal)`

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

4.16.2.2 prepare() `void acPooling_wave::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------------	---

Implements `MHAPlugIn::plugin_t< acPooling_wave_config >` (p. [1148](#)).

4.16.2.3 release() `void acPooling_wave::release (void) [inline], [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< acPooling_wave_config >` (p. [1149](#)).

4.16.2.4 update_cfg() `void acPooling_wave::update_cfg () [private]`

4.16.3 Member Data Documentation

4.16.3.1 `numsamples` `MHAParser::int_t acPooling_wave::numsamples`

4.16.3.2 `pooling_wndlen` `MHAParser::int_t acPooling_wave::pooling_wndlen`

4.16.3.3 `pooling_type` `MHAParser::kw_t acPooling_wave::pooling_type`

4.16.3.4 `upper_threshold` `MHAParser::float_t acPooling_wave::upper_threshold`

4.16.3.5 `lower_threshold` `MHAParser::float_t acPooling_wave::lower_threshold`

4.16.3.6 `neighbourhood` `MHAParser::int_t acPooling_wave::neighbourhood`

4.16.3.7 `alpha` `MHAParser::float_t acPooling_wave::alpha`

4.16.3.8 `p_name` `MHAParser::string_t acPooling_wave::p_name`

4.16.3.9 `p_biased_name` `MHAParser::string_t acPooling_wave::p_biased_name`

4.16.3.10 pool_name `MHAParser::string_t acPooling_wave::pool_name`

4.16.3.11 max_pool_ind_name `MHAParser::string_t acPooling_wave::max_pool_ind_name`

4.16.3.12 like_ratio_name `MHAParser::string_t acPooling_wave::like_ratio_name`

4.16.3.13 prob_bias `MHAParser::vfloat_t acPooling_wave::prob_bias`

4.16.3.14 patchbay `MHAEvents::patchbay_t< acPooling_wave> acPooling_wave::patchbay` [private]

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

- `acPooling_wave.h`
- `acPooling_wave.cpp`

4.17 acPooling_wave_config Class Reference

Public Member Functions

- `acPooling_wave_config (algo_comm_t & ac, const mhaconfig_t in_cfg, acPooling_wave *_pooling)`
- `~acPooling_wave_config ()`
- `mha_wave_t * process (mha_wave_t *)`
- `void insert ()`

Public Attributes

- `algo_comm_t & ac`
- `std::string raw_p_name`
- `MHA_AC::waveform_t p`
- `MHA_AC::waveform_t p_biased`
- `MHA_AC::waveform_t p_max`
- `MHA_AC::waveform_t like_ratio`
- `mha_wave_t c`
- `unsigned int pooling_ind`
- `unsigned int pooling_option`
- `unsigned int pooling_size`
- `float up_thresh`
- `float low_thresh`
- `int neigh`
- `float alpha`
- `MHASignal::waveform_t pool`
- `MHASignal::waveform_t prob_bias_func`

4.17.1 Constructor & Destructor Documentation**4.17.1.1 acPooling_wave_config()** `acPooling_wave_config::acPooling_wave_config (`

```
    algo_comm_t & ac,
    const mhaconfig_t in_cfg,
    acPooling_wave * _pooling )
```

4.17.1.2 ~acPooling_wave_config() `acPooling_wave_config::~acPooling_wave_config (``)`**4.17.2 Member Function Documentation****4.17.2.1 process()** `mha_wave_t * acPooling_wave_config::process (` `mha_wave_t * wave)`

4.17.2.2 insert() void acPooling_wave_config::insert ()

4.17.3 Member Data Documentation

4.17.3.1 ac algo_comm_t& acPooling_wave_config::ac

4.17.3.2 raw_p_name std::string acPooling_wave_config::raw_p_name

4.17.3.3 p MHA_AC::waveform_t acPooling_wave_config::p

4.17.3.4 p_biased MHA_AC::waveform_t acPooling_wave_config::p_biased

4.17.3.5 p_max MHA_AC::waveform_t acPooling_wave_config::p_max

4.17.3.6 like_ratio MHA_AC::waveform_t acPooling_wave_config::like_ratio

4.17.3.7 c mha_wave_t acPooling_wave_config::c

4.17.3.8 pooling_ind unsigned int acPooling_wave_config::pooling_ind

4.17.3.9 pooling_option `unsigned int acPooling_wave_config::pooling_option`

4.17.3.10 pooling_size `unsigned int acPooling_wave_config::pooling_size`

4.17.3.11 up_thresh `float acPooling_wave_config::up_thresh`

4.17.3.12 low_thresh `float acPooling_wave_config::low_thresh`

4.17.3.13 neigh `int acPooling_wave_config::neigh`

4.17.3.14 alpha `float acPooling_wave_config::alpha`

4.17.3.15 pool `MHASignal::waveform_t acPooling_wave_config::pool`

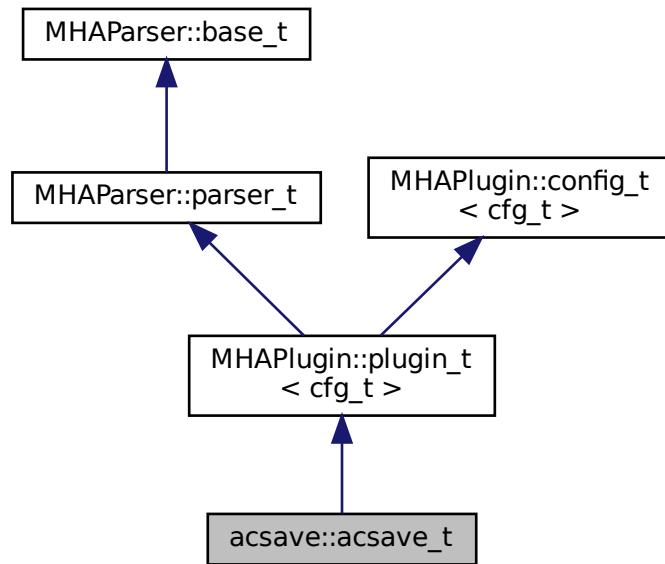
4.17.3.16 prob_bias_func `MHASignal::waveform_t acPooling_wave_config::prob_bias_func`

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

- **acPooling_wave.h**
- **acPooling_wave.cpp**

4.18 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

4.18.1 Member Typedef Documentation

4.18.1.1 `varlist_t` `typedef std::vector< save_var_t*> acsave::acsave_t::varlist_t [private]`

4.18.2 Constructor & Destructor Documentation

4.18.2.1 `acsave_t()` `acsave::acsave_t::acsave_t (`
 `const algo_comm_t & iac,`
 `const std::string & ith,`
 `const std::string & ial)`

4.18.3 Member Function Documentation

4.18.3.1 `prepare()` void acsave::acsave_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1148).

4.18.3.2 `release()` void acsave::acsave_t::release () [virtual]

Reimplemented from **MHAPlugin::plugin_t< cfg_t >** (p. 1149).

4.18.3.3 `process()` [1/3] mha_spec_t * acsave::acsave_t::process (
 mha_spec_t * s)

4.18.3.4 `process()` [2/3] mha_wave_t * acsave::acsave_t::process (
 mha_wave_t * s)

4.18.3.5 `event_start_recording()` void acsave::acsave_t::event_start_recording ()

4.18.3.6 `event_stop_and_flush()` void acsave::acsave_t::event_stop_and_flush ()

4.18.3.7 `process()` [3/3] void acsave::acsave_t::process () [private]

4.18.4 Member Data Documentation

4.18.4.1 bflush `MHAParser::bool_t acsave::acsave_t::bflush [private]`

4.18.4.2 fileformat `MHAParser::kw_t acsave::acsave_t::fileformat [private]`

4.18.4.3 fname `MHAParser::string_t acsave::acsave_t::fname [private]`

4.18.4.4 reclen `MHAParser::float_t acsave::acsave_t::reclen [private]`

4.18.4.5 variables `MHAParser::vstring_t acsave::acsave_t::variables [private]`

4.18.4.6 varlist `varlist_t acsave::acsave_t::varlist [private]`

4.18.4.7 chain `std::string acsave::acsave_t::chain [private]`

4.18.4.8 algo `std::string acsave::acsave_t::algo [private]`

4.18.4.9 b_prepared `bool acsave::acsave_t::b_prepared [private]`

4.18.4.10 b_flushed bool acsave::acsave_t::b_flushed [private]

4.18.4.11 patchbay MHAEvents::patchbay_t< acsave_t> acsave::acsave_t::patchbay [private]

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

- **acsave.cpp**

4.19 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**

4.19.1 Constructor & Destructor Documentation

4.19.1.1 cfg_t() cfg_t::cfg_t (

```
const algo_comm_t & iac,
unsigned int imax_frames,
std::vector< std::string > & var_names )
```

4.19.1.2 ~cfg_t() `cfg_t::~cfg_t ()`**4.19.2 Member Function Documentation****4.19.2.1 store_frame()** `void cfg_t::store_frame ()`

This function is called in the processing thread.

4.19.2.2 flush_data() `void cfg_t::flush_data (`
`const std::string & filename,`
`unsigned int fmt)`

This function is called in the configuration thread.

Parameters

<i>filename</i>	Output file name
<i>fmt</i>	Output file format

4.19.3 Member Data Documentation**4.19.3.1 ac** `algo_comm_t` `acsave::cfg_t::ac [private]`**4.19.3.2 nvars** `unsigned int` `acsave::cfg_t::nvars [private]`**4.19.3.3 varlist** `save_var_t**` `acsave::cfg_t::varlist [private]`

4.19.3.4 rec_frames unsigned int acsave::cfg_t::rec_frames [private]

4.19.3.5 max_frames unsigned int acsave::cfg_t::max_frames [private]

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

- **acsave.cpp**

4.20 acsave::mat4head_t Struct Reference

Public Attributes

- int32_t **t**
- int32_t **rows**
- int32_t **cols**
- int32_t **imag**
- int32_t **namelen**

4.20.1 Member Data Documentation

4.20.1.1 t int32_t acsave::mat4head_t::t

4.20.1.2 rows int32_t acsave::mat4head_t::rows

4.20.1.3 cols int32_t acsave::mat4head_t::cols

4.20.1.4 `imag` `int32_t acsave::mat4head_t::imag`**4.20.1.5 `namelen`** `int32_t acsave::mat4head_t::namelen`

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

- `acsave.cpp`

4.21 `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`

4.21.1 Constructor & Destructor Documentation

4.21.1.1 `save_var_t()` acsave::save_var_t::save_var_t (

```
    const std::string & nm,
    int n,
    const algo_comm_t & iac )
```

4.21.1.2 `~save_var_t()` acsave::save_var_t::~save_var_t ()

4.21.2 Member Function Documentation

4.21.2.1 `store_frame()` void acsave::save_var_t::store_frame ()

4.21.2.2 `save_txt()` void acsave::save_var_t::save_txt (

```
    FILE * fh,
    unsigned int writeframes )
```

4.21.2.3 `save_mat4()` void acsave::save_var_t::save_mat4 (

```
    FILE * fh,
    unsigned int writeframes )
```

4.21.2.4 `save_m()` void acsave::save_var_t::save_m (

```
    FILE * fh,
    unsigned int writeframes )
```

4.21.3 Member Data Documentation

4.21.3.1 `data` double* acsave::save_var_t::data

4.21.3.2 `name` std::string acsave::save_var_t::name [private]

4.21.3.3 `nframes` unsigned int acsave::save_var_t::nframes [private]

4.21.3.4 `ndim` unsigned int acsave::save_var_t::ndim [private]

4.21.3.5 `maxframe` unsigned int acsave::save_var_t::maxframe [private]

4.21.3.6 `ac` algo_comm_t acsave::save_var_t::ac [private]

4.21.3.7 `framecnt` unsigned int acsave::save_var_t::framecnt [private]

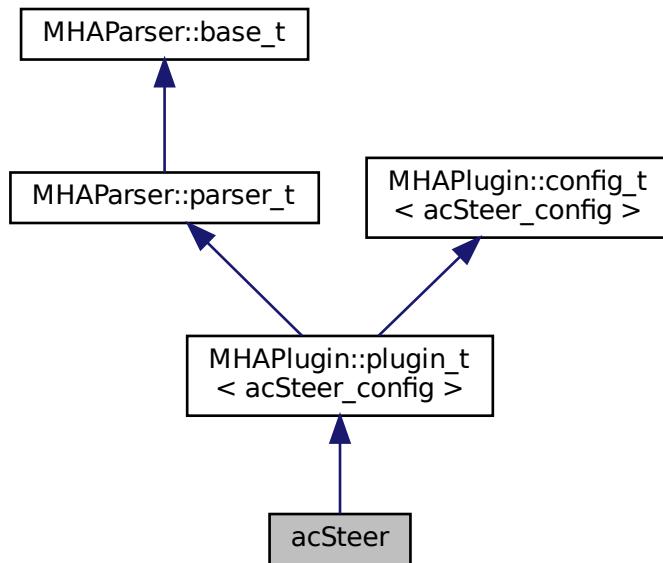
4.21.3.8 `b_complex` bool acsave::save_var_t::b_complex [private]

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

- **acsave.cpp**

4.22 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**4.22.1 Constructor & Destructor Documentation****4.22.1.1 acSteer()** acSteer::acSteer (

```
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs our plugin.

4.22.1.2 ~acSteer() acSteer::~acSteer ()**4.22.2 Member Function Documentation****4.22.2.1 process()** mha_spec_t * acSteer::process (

```
    mha_spec_t * signal )
```

This method is a NOOP.

4.22.2.2 prepare() 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 **MHAParser::plugin_t< acSteer_config >** (p. 1148).

4.22.2.3 release() `void acSteer::release (void) [inline], [virtual]`

Reimplemented from **MHAParser::plugin_t< acSteer_config >** (p. 1149).

4.22.2.4 update_cfg() `void acSteer::update_cfg () [private]`

4.22.3 Member Data Documentation

4.22.3.1 steerFile `MHAParser::string_t acSteer::steerFile`

4.22.3.2 acSteerName1 `MHAParser::string_t acSteer::acSteerName1`

4.22.3.3 acSteerName2 `MHAParser::string_t acSteer::acSteerName2`

4.22.3.4 nsteerchan `MHAParser::int_t acSteer::nsteerchan`

4.22.3.5 nrefmic `MHAParser::int_t acSteer::nrefmic`

4.22.3.6 patchbay `MHAEvents::patchbay_t< acSteer> acSteer::patchbay [private]`

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

- `acSteer.h`
- `acSteer.cpp`

4.23 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`

4.23.1 Constructor & Destructor Documentation

4.23.1.1 acSteer_config() `acSteer_config::acSteer_config (`
`algo_comm_t & ac,`
`const mhaconfig_t in_cfg,`
`acSteer * acSteer)`

4.23.1.2 ~acSteer_config() acSteer_config::~acSteer_config ()

4.23.2 Member Function Documentation

4.23.2.1 insert() void acSteer_config::insert ()

4.23.3 Member Data Documentation

4.23.3.1 nchan unsigned int acSteer_config::nchan

4.23.3.2 nfreq unsigned int acSteer_config::nfreq

4.23.3.3 nsteerchan unsigned int acSteer_config::nsteerchan

4.23.3.4 nrefmic unsigned int acSteer_config::nrefmic

4.23.3.5 nangle unsigned int acSteer_config::nangle

4.23.3.6 specSteer1 MHA_AC::spectrum_t acSteer_config::specSteer1

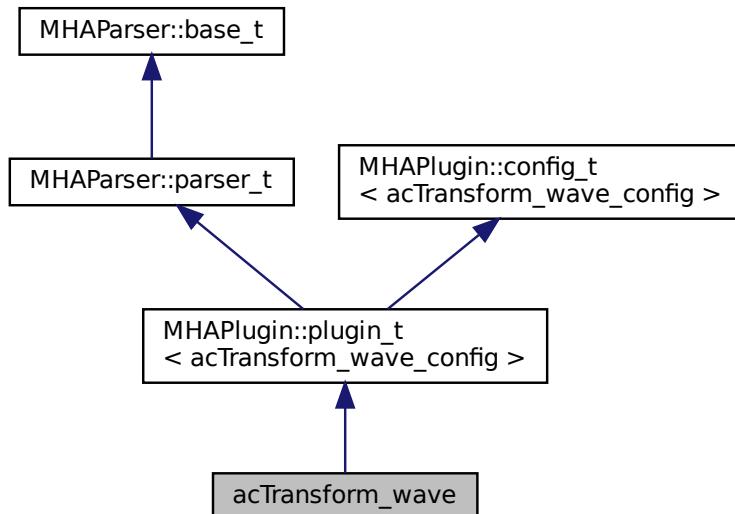
4.23.3.7 **specSteer2** `MHA_AC::spectrum_t acSteer_config::specSteer2`

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

- `acSteer.h`
- `acSteer.cpp`

4.24 **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

- MHParse::string_t ang_name
- MHParse::string_t raw_p_name
- MHParse::string_t raw_p_max_name
- MHParse::string_t rotated_p_name
- MHParse::string_t rotated_p_max_name
- MHParse::int_t numsamples
- MHParse::bool_t to_from

Private Member Functions

- void update_cfg ()

Private Attributes

- MHAEvents::patchbay_t< acTransform_wave > patchbay

Additional Inherited Members

4.24.1 Constructor & Destructor Documentation

4.24.1.1 acTransform_wave() acTransform_wave::acTransform_wave (
 algo_comm_t & ac,
 const std::string & chain_name,
 const std::string & algo_name)

Constructs our plugin.

4.24.1.2 ~acTransform_wave() acTransform_wave::~acTransform_wave ()

4.24.2 Member Function Documentation

4.24.2.1 process() `mha_wave_t * acTransform_wave::process (`
`mha_wave_t * signal)`

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

4.24.2.2 prepare() `void acTransform_wave::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	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. 1148).

4.24.2.3 release() `void acTransform_wave::release (`
`void) [inline], [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< acTransform_wave_config >` (p. 1149).

4.24.2.4 update_cfg() `void acTransform_wave::update_cfg () [private]`

4.24.3 Member Data Documentation

4.24.3.1 ang_name `MHAParser::string_t acTransform_wave::ang_name`

4.24.3.2 raw_p_name `MHAParser::string_t acTransform_wave::raw_p_name`

4.24.3.3 raw_p_max_name `MHAParser::string_t acTransform_wave::raw_p_max_name`

4.24.3.4 rotated_p_name `MHAParser::string_t acTransform_wave::rotated_p_name`

4.24.3.5 rotated_p_max_name `MHAParser::string_t acTransform_wave::rotated_p_max_name`

4.24.3.6 numsamples `MHAParser::int_t acTransform_wave::numsamples`

4.24.3.7 to_from `MHAParser::bool_t acTransform_wave::to_from`

4.24.3.8 patchbay `MHAEvents::patchbay_t< acTransform_wave> acTransform_wave::patchbay [private]`

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

- `acTransform_wave.h`
- `acTransform_wave.cpp`

4.25 acTransform_wave_config Class Reference

Public Member Functions

- `acTransform_wave_config (algo_comm_t & ac, const mhaconfig_t in_cfg, acTransform_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`

4.25.1 Constructor & Destructor Documentation

4.25.1.1 `acTransform_wave_config()` `acTransform_wave_config::acTransform_wave_config (`

```
algo_comm_t & ac,
const mhaconfig_t in_cfg,
acTransform_wave * _transform )
```

4.25.1.2 `~acTransform_wave_config()` `acTransform_wave_config::~acTransform_wave_config ()`

4.25.2 Member Function Documentation

4.25.2.1 `process()` `mha_wave_t * acTransform_wave_config::process (`

```
mha_wave_t * wave )
```

4.25.3 Member Data Documentation

4.25.3.1 ac algo_comm_t& acTransform_wave_config::ac

4.25.3.2 ang_name std::string acTransform_wave_config::ang_name

4.25.3.3 raw_p_name std::string acTransform_wave_config::raw_p_name

4.25.3.4 raw_p_max_name std::string acTransform_wave_config::raw_p_max_name

4.25.3.5 rotated_p MHA_AC::waveform_t acTransform_wave_config::rotated_p

4.25.3.6 rotated_i MHA_AC::int_t acTransform_wave_config::rotated_i

4.25.3.7 offset unsigned int acTransform_wave_config::offset

4.25.3.8 resolution unsigned int acTransform_wave_config::resolution

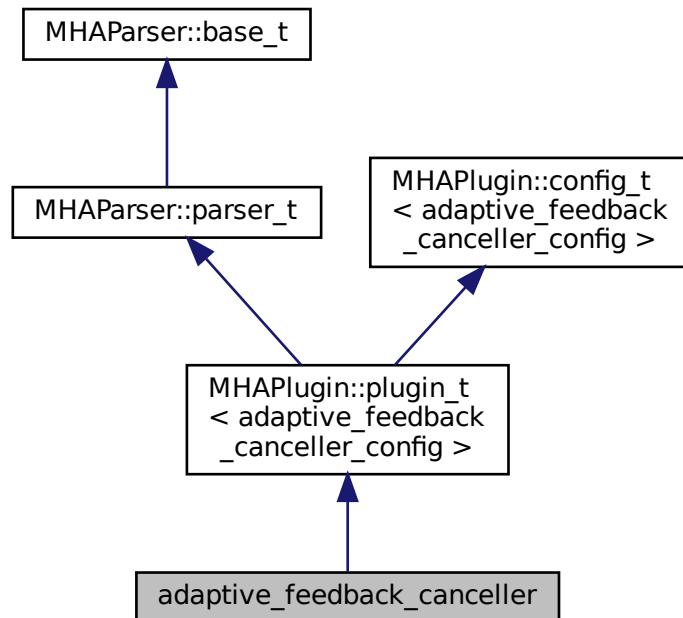
4.25.3.9 to_from unsigned int acTransform_wave_config::to_from

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

- **acTransform_wave.h**
- **acTransform_wave.cpp**

4.26 adaptive_feedback_canceller Class Reference

Inheritance diagram for adaptive_feedback_canceller:



Public Member Functions

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

Public Attributes

- **`MHAParser::float_t rho`**
- **`MHAParser::float_t c`**
- **`MHAParser::int_t ntaps`**

- `MHAParser::vfloat_t gains`
- `MHAParser::string_t name_e`
- `MHAParser::string_t name_f`
- `MHAParser::string_t name_lpc`
- `MHAParser::int_t lpc_order`
- `MHAParser::vint_t afc_delay`
- `MHAParser::vint_t delay_w`
- `MHAParser::vint_t delay_d`
- `MHAParser::int_t n_no_update`

Private Member Functions

- `void update_cfg ()`

Private Attributes

- `MHAEvents::patchbay_t< adaptive_feedback_canceller > patchbay`

Additional Inherited Members

4.26.1 Constructor & Destructor Documentation

```
4.26.1.1 adaptive_feedback_canceller() adaptive_feedback_canceller::adaptive_←
feedback_canceller (
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs our plugin.

```
4.26.1.2 ~adaptive_feedback_canceller() adaptive_feedback_canceller::~adaptive_←
feedback_canceller ( )
```

4.26.2 Member Function Documentation

4.26.2.1 process() `mha_wave_t * adaptive_feedback_canceller::process (`
`mha_wave_t * signal)`

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

4.26.2.2 prepare() `void adaptive_feedback_canceller::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------------	---

Implements **MHAPlugIn::plugin_t< adaptive_feedback_canceller_config >** (p. [1148](#)).

4.26.2.3 release() `void adaptive_feedback_canceller::release (`
`void) [inline], [virtual]`

Reimplemented from **MHAPlugIn::plugin_t< adaptive_feedback_canceller_config >** (p. [1149](#)).

4.26.2.4 update_cfg() `void adaptive_feedback_canceller::update_cfg () [private]`

4.26.3 Member Data Documentation

4.26.3.1 rho `MHAParser::float_t adaptive_feedback_canceller::rho`

4.26.3.2 c `MHAParser::float_t` `adaptive_feedback_canceller::c`

4.26.3.3 ntaps `MHAParser::int_t` `adaptive_feedback_canceller::ntaps`

4.26.3.4 gains `MHAParser::vfloat_t` `adaptive_feedback_canceller::gains`

4.26.3.5 name_e `MHAParser::string_t` `adaptive_feedback_canceller::name_e`

4.26.3.6 name_f `MHAParser::string_t` `adaptive_feedback_canceller::name_f`

4.26.3.7 name_lpc `MHAParser::string_t` `adaptive_feedback_canceller::name_lpc`

4.26.3.8 lpc_order `MHAParser::int_t` `adaptive_feedback_canceller::lpc_order`

4.26.3.9 afc_delay `MHAParser::vint_t` `adaptive_feedback_canceller::afc_delay`

4.26.3.10 delay_w `MHAParser::vint_t` `adaptive_feedback_canceller::delay_w`

4.26.3.11 `delay_d` `MHAParser::vint_t adaptive_feedback_canceller::delay_d`

4.26.3.12 `n_no_update` `MHAParser::int_t adaptive_feedback_canceller::n_no_update`

4.26.3.13 `patchbay` `MHAEvents::patchbay_t< adaptive_feedback_canceller> adaptive->-feedback_canceller::patchbay [private]`

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

- `adaptive_feedback_canceller.h`
- `adaptive_feedback_canceller.cpp`

4.27 `adaptive_feedback_canceller_config` Class Reference

Public Member Functions

- `adaptive_feedback_canceller_config (algo_comm_t & ac, const mhaconfig_t in_cfg, adaptive_feedback_canceller *afc)`
- `~adaptive_feedback_canceller_config ()`
- `mha_wave_t * process (mha_wave_t *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`
- `double PSD_val`

- `MHASignal::waveform_t v_G`
- `MHASignal::waveform_t s_U`
- `MHASignal::delay_t s_E_afc_delay`
- `MHASignal::delay_t s_W`
- `MHASignal::ringbuffer_t s_Wflt`
- `MHASignal::delay_t s_U_delay`
- `MHASignal::ringbuffer_t s_U_delayflt`
- `MHASignal::waveform_t F_Uflt`
- `MHASignal::delay_t s_Y_delay`
- `MHASignal::ringbuffer_t s_Y_delayflt`
- `MHASignal::ringbuffer_t UbufferPrew`
- `mha_wave_t s_LPC`
- `mha_wave_t UPrew`
- `mha_wave_t YPrew`
- `mha_wave_t EPrew`
- `mha_wave_t UPrewW`
- `mha_wave_t smpl`
- `mha_wave_t * s_Usmpl`

4.27.1 Constructor & Destructor Documentation

4.27.1.1 adaptive_feedback_canceller_config() `adaptive_feedback_canceller_config::adaptive_feedback_canceller_config (algo_comm_t & ac,`
`const mhaconfig_t in_cfg,`
`adaptive_feedback_canceller * afc)`

4.27.1.2 ~adaptive_feedback_canceller_config() `adaptive_feedback_canceller_config::~adaptive_feedback_canceller_config ()`

4.27.2 Member Function Documentation

4.27.2.1 process() `mha_wave_t * adaptive_feedback_canceller_config::process (`
`mha_wave_t * s_Y,`
`mha_real_t rho,`
`mha_real_t c)`

4.27.2.2 insert() `void adaptive_feedback_canceller_config::insert ()`

4.27.3 Member Data Documentation

4.27.3.1 ac `algo_comm_t adaptive_feedback_canceller_config::ac [private]`

4.27.3.2 ntaps `unsigned int adaptive_feedback_canceller_config::ntaps [private]`

4.27.3.3 frames `unsigned int adaptive_feedback_canceller_config::frames [private]`

4.27.3.4 channels `unsigned int adaptive_feedback_canceller_config::channels [private]`

4.27.3.5 s_E `MHA_AC::waveform_t adaptive_feedback_canceller_config::s_E [private]`

4.27.3.6 F `MHA_AC::waveform_t adaptive_feedback_canceller_config::F [private]`

4.27.3.7 Pu `MHASignal::waveform_t adaptive_feedback_canceller_config::Pu` [private]

Power of input signal delayline.

4.27.3.8 name_d_ `std::string adaptive_feedback_canceller_config::name_d_` [private]**4.27.3.9 name_lpc_** `std::string adaptive_feedback_canceller_config::name_lpc_` [private]**4.27.3.10 n_no_update_** `int adaptive_feedback_canceller_config::n_no_update_` [private]**4.27.3.11 no_iter** `int adaptive_feedback_canceller_config::no_iter` [private]**4.27.3.12 iter** `int adaptive_feedback_canceller_config::iter` [private]**4.27.3.13 PSD_val** `double adaptive_feedback_canceller_config::PSD_val` [private]**4.27.3.14 v_G** `MHASignal::waveform_t adaptive_feedback_canceller_config::v_G` [private]**4.27.3.15 s_U** `MHASignal::waveform_t adaptive_feedback_canceller_config::s_U` [private]

4.27.3.16 s_E_afc_delay `MHASignal::delay_t` adaptive_feedback_canceller_config←
 ::s_E_afc_delay [private]

4.27.3.17 s_W `MHASignal::delay_t` adaptive_feedback_canceller_config::s_W [private]

4.27.3.18 s_Wfilt `MHASignal::ringbuffer_t` adaptive_feedback_canceller_config::s_Wfilt←
 [private]

4.27.3.19 s_U_delay `MHASignal::delay_t` adaptive_feedback_canceller_config::s_U_delay←
 [private]

4.27.3.20 s_U_delayfilt `MHASignal::ringbuffer_t` adaptive_feedback_canceller_config←
 ::s_U_delayfilt [private]

4.27.3.21 F_Ufilt `MHASignal::waveform_t` adaptive_feedback_canceller_config::F_Ufilt
 [private]

4.27.3.22 s_Y_delay `MHASignal::delay_t` adaptive_feedback_canceller_config::s_Y_delay←
 [private]

4.27.3.23 s_Y_delayfilt `MHASignal::ringbuffer_t` adaptive_feedback_canceller_config←
 ::s_Y_delayfilt [private]

4.27.3.24 UbufferPrew `MHASignal::ringbuffer_t adaptive_feedback_canceller_config::UbufferPrew` [private]

4.27.3.25 s_LPC `mha_wave_t adaptive_feedback_canceller_config::s_LPC` [private]

4.27.3.26 UPrew `mha_wave_t adaptive_feedback_canceller_config::UPrew` [private]

4.27.3.27 YPrew `mha_wave_t adaptive_feedback_canceller_config::YPrew` [private]

4.27.3.28 EPrew `mha_wave_t adaptive_feedback_canceller_config::EPrew` [private]

4.27.3.29 UPrewW `mha_wave_t adaptive_feedback_canceller_config::UPrewW` [private]

4.27.3.30 smpl `mha_wave_t adaptive_feedback_canceller_config::smpl` [private]

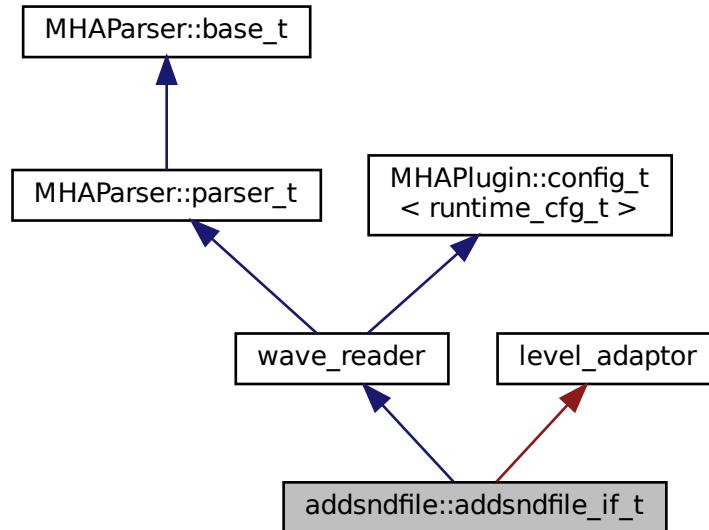
4.27.3.31 s_Usmpl `mha_wave_t* adaptive_feedback_canceller_config::s_Usmpl` [private]

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

- `adaptive_feedback_canceller.h`
- `adaptive_feedback_canceller.cpp`

4.28 addsndfile::addsndfile_if_t Class Reference

Inheritance diagram for addsndfile::addsndfile_if_t:



Public Member Functions

- **addsndfile_if_t (algo_comm_t, const char *, const char *)**
- **mha_wave_t * process (mha_wave_t *)**
- **void prepare (mhaconfig_t &)**
- **void release ()**

Private Member Functions

- **void update ()**
- **void change_mode ()**
- **void set_level ()**
- **void scan_dir ()**

Private Attributes

- `MHAParser::string_t filename`
- `MHAParser::string_t path`
- `MHAParser::bool_t loop`
- `MHAParser::float_t level`
- `MHAParser::kw_t levelmode`
- `MHAParser::kw_t resamplingmode`
- `MHAParser::vint_t channels`
- `MHAParser::kw_t mode`
- `MHAParser::float_t ramplen`
- `MHAParser::int_t startpos`
- `MHAParser::vint_mon_t mapping`
- `MHAParser::int_mon_t numchannels`
- `MHAParser::int_mon_t mhachannels`
- `MHAParser::int_mon_t active`
- `MHAParser::string_t search_pattern`
- `MHAParser::vstring_mon_t search_result`
- `unsigned int uint_mode`
- `MHAEvents::patchbay_t< addsndfile_if_t > patchbay`

Additional Inherited Members

4.28.1 Constructor & Destructor Documentation

```
4.28.1.1 addsndfile_if_t() addsndfile::addsndfile_if_t::addsndfile_if_t (
    algo_comm_t iac,
    const char * ,
    const char * )
```

4.28.2 Member Function Documentation

```
4.28.2.1 process() mha_wave_t * addsndfile::addsndfile_if_t::process (
    mha_wave_t * s )
```

4.28.2.2 `prepare()` void addsndfile::addsndfile_if_t::prepare (mhaconfig_t & tf) [virtual]

Implements **MHAPlugIn::plugin_t< runtime_cfg_t >** (p. [1148](#)).

4.28.2.3 `release()` void addsndfile::addsndfile_if_t::release () [virtual]

Reimplemented from **MHAPlugIn::plugin_t< runtime_cfg_t >** (p. [1149](#)).

4.28.2.4 `update()` void addsndfile::addsndfile_if_t::update () [private]

4.28.2.5 `change_mode()` void addsndfile::addsndfile_if_t::change_mode () [private]

4.28.2.6 `set_level()` void addsndfile::addsndfile_if_t::set_level () [private]

4.28.2.7 `scan_dir()` void addsndfile::addsndfile_if_t::scan_dir () [private]

4.28.3 Member Data Documentation

4.28.3.1 `filename` MHAParser::string_t addsndfile::addsndfile_if_t::filename [private]

4.28.3.2 `path` MHAParser::string_t addsndfile::addsndfile_if_t::path [private]

4.28.3.3 `loop` `MHAParser::bool_t addsndfile::addsndfile_if_t::loop` [private]

4.28.3.4 `level` `MHAParser::float_t addsndfile::addsndfile_if_t::level` [private]

4.28.3.5 `levelmode` `MHAParser::kw_t addsndfile::addsndfile_if_t::levelmode` [private]

4.28.3.6 `resamplingmode` `MHAParser::kw_t addsndfile::addsndfile_if_t::resamplingmode` [private]

4.28.3.7 `channels` `MHAParser::vint_t addsndfile::addsndfile_if_t::channels` [private]

4.28.3.8 `mode` `MHAParser::kw_t addsndfile::addsndfile_if_t::mode` [private]

4.28.3.9 `ramplen` `MHAParser::float_t addsndfile::addsndfile_if_t::ramplen` [private]

4.28.3.10 `startpos` `MHAParser::int_t addsndfile::addsndfile_if_t::startpos` [private]

4.28.3.11 `mapping` `MHAParser::vint_mon_t addsndfile::addsndfile_if_t::mapping` [private]

4.28.3.12 numchannels `MHAParser::int_mon_t addsndfile::addsndfile_if_t::numchannels` [private]

4.28.3.13 mhachannels `MHAParser::int_mon_t addsndfile::addsndfile_if_t::mhachannels` [private]

4.28.3.14 active `MHAParser::int_mon_t addsndfile::addsndfile_if_t::active` [private]

4.28.3.15 search_pattern `MHAParser::string_t addsndfile::addsndfile_if_t::search←_pattern` [private]

4.28.3.16 search_result `MHAParser::vstring_mon_t addsndfile::addsndfile_if_t←::search_result` [private]

4.28.3.17 uint_mode `unsigned int addsndfile::addsndfile_if_t::uint_mode` [private]

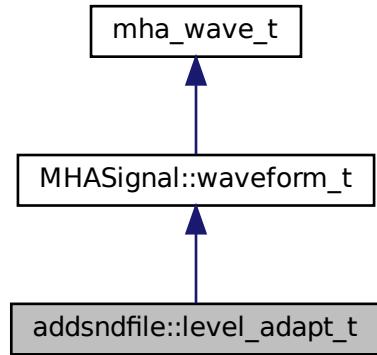
4.28.3.18 patchbay `MHAEEvents::patchbay_t< addsndfile_if_t> addsndfile::addsndfile_if_t::patchbay` [private]

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

- `addsndfile.cpp`

4.29 `addsndfile::level_adapt_t` Class Reference

Inheritance diagram for `addsndfile::level_adapt_t`:



Public Member Functions

- `level_adapt_t (mhaconfig_t cf, mha_real_t adapt_len, mha_real_t l_new_, mha_real_t l_old_)`
- `void update_frame ()`
- `mha_real_t get_level () const`
- `bool can_update () const`

Private Attributes

- `unsigned int llen`
- `unsigned int pos`
- `MHAWindow::fun_t wnd`
- `mha_real_t l_new`
- `mha_real_t l_old`

Additional Inherited Members

4.29.1 Constructor & Destructor Documentation

```
4.29.1.1 level_adapt_t() addsndfile::level_adapt_t::level_adapt_t (
    mhaconfig_t cf,
    mha_real_t adapt_len,
    mha_real_t l_new_,
    mha_real_t l_old_ )
```

4.29.2 Member Function Documentation

4.29.2.1 update_frame() void addsndfile::level_adapt_t::update_frame ()

4.29.2.2 get_level() mha_real_t addsndfile::level_adapt_t::get_level () const [inline]

4.29.2.3 can_update() bool addsndfile::level_adapt_t::can_update () const [inline]

4.29.3 Member Data Documentation

4.29.3.1 ilen unsigned int addsndfile::level_adapt_t::ilen [private]

4.29.3.2 pos unsigned int addsndfile::level_adapt_t::pos [private]

4.29.3.3 wnd MHAWindow::fun_t addsndfile::level_adapt_t::wnd [private]

4.29.3.4 l_new mha_real_t addsndfile::level_adapt_t::l_new [private]**4.29.3.5 l_old mha_real_t addsndfile::level_adapt_t::l_old [private]**

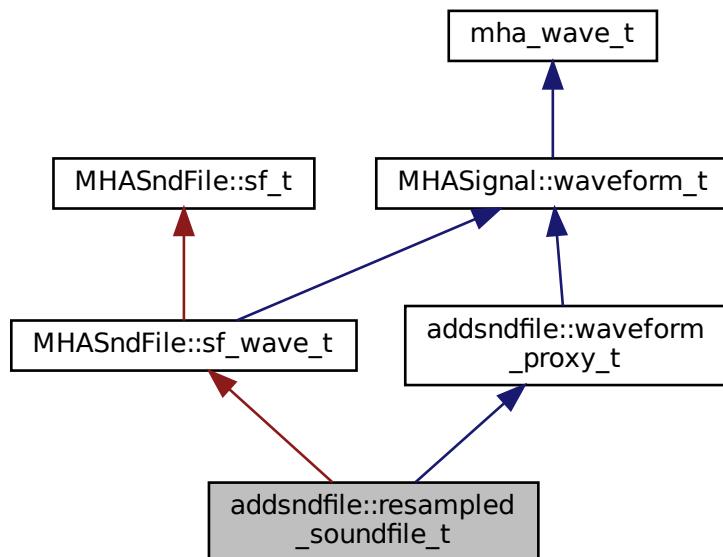
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

4.30 addsndfile::resampled_soundfile_t Class Reference

Reads sound from file and resamples it if necessary and wanted.

Inheritance diagram for addsndfile::resampled_soundfile_t:



Public Member Functions

- **resampled_soundfile_t (const std::string &name, float mha_sampling_rate, addsndfile_resampling_mode_t resampling_mode)**

Reads sound from file and resamples if necessary and wanted.

Additional Inherited Members

4.30.1 Detailed Description

Reads sound from file and resamples it if necessary and wanted.

Sound data can then be used by addsndfile.

4.30.2 Constructor & Destructor Documentation

```
4.30.2.1 resampled_soundfile_t() addsndfile::resampled_soundfile_t::resampled_<br>
soundfile_t ( <br>
    const std::string & name,<br>
    float mha_sampling_rate,<br>
    addsndfile_resampling_mode_t resampling_mode )
```

Reads sound from file and resamples if necessary and wanted.

If the sound file does not specify a sampling rate, then the sound data is always used without resampling.

Parameters

<i>name</i>	Sound file name
<i>mha_sampling_rate</i>	The sampling rate of the MHA signal processing at the point of the addsndfile plugin
<i>resampling_mode</i>	DONT_RESAMPLE_STRICT: Do not resample, just use the samples from the sound file at the current sample rate, even if the sample rate of the sound file differs. DONT_RESAMPLE_PERMISSIVE: Do not resample, if the sample rate of the MHA differs from the sample rate of the sound file, raise an error. DO_RESAMPLE: Resample.

Exceptions

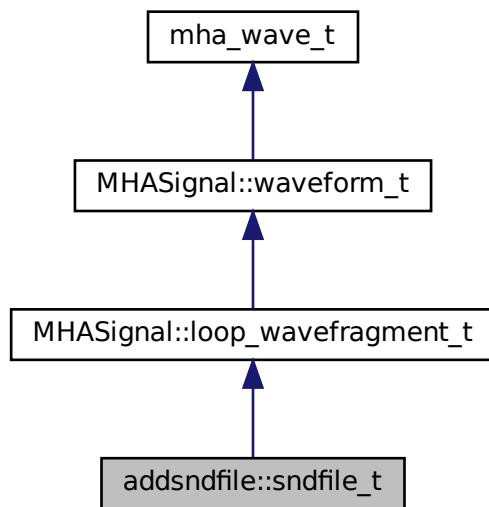
MHA_Error (p. 763)	If the sampling rate of the file does not match the sampling rate of the MHA and DONT_RESAMPLE_STRICT was requested. If resampling failed (e.g. due to non-rational quotient of MHA sampling rate and sound file sampling rate).
--	--

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

- addsndfile.cpp

4.31 addsndfile::sndfile_t Class Reference

Inheritance diagram for addsndfile::sndfile_t:



Public Member Functions

- **sndfile_t** (const std::string &name, bool loop, unsigned int level_mode, std::vector< int > channels_, unsigned int nchannels, std::vector< int > &mapping, int &numchannels, unsigned int startpos, float mha_sampling_rate, **addsndfile_resampling_mode_t** resampling_mode)

Additional Inherited Members

4.31.1 Constructor & Destructor Documentation

```
4.31.1.1 sndfile_t() addsndfile::sndfile_t::sndfile_t (
    const std::string & name,
    bool loop,
    unsigned int level_mode,
    std::vector< int > channels_,
    unsigned int nchannels,
    std::vector< int > & mapping,
    int & numchannels,
    unsigned int startpos,
    float mha_sampling_rate,
    addsndfile_resampling_mode_t resampling_mode )
```

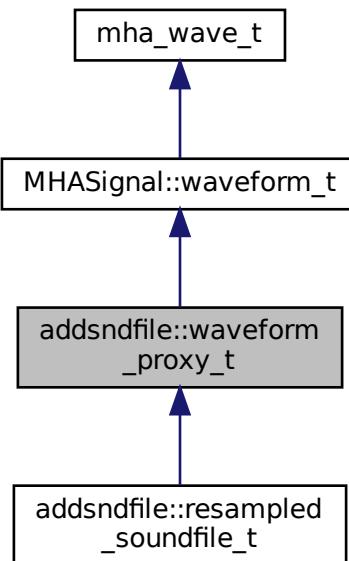
The documentation for this class was generated from the following file:

- **addsndfile.cpp**

4.32 addsndfile::waveform_proxy_t Class Reference

Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. [257](#)).

Inheritance diagram for addsndfile::waveform_proxy_t:



Public Member Functions

- **waveform_proxy_t** (unsigned frames, unsigned **channels**)

Additional Inherited Members

4.32.1 Detailed Description

Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in **resampled_soundfile_t** (p. [257](#)).

4.32.2 Constructor & Destructor Documentation

4.32.2.1 waveform_proxy_t()

```
addsndfile::waveform_proxy_t::waveform_proxy_t (
    unsigned frames,
    unsigned channels ) [inline]
```

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

- **addsndfile.cpp**

4.33 ADM::ADM< F > Class Template Reference

Adaptive differential microphone, working for speech frequency range.

Public Member Functions

- **ADM** (F fs, F dist, unsigned lp_order, const F *lp_alphas, unsigned decomb_order, const F *decomb_alphas, F tau_beta=F(50e-3), F mu_beta=F(1e-4))
Create Adaptive Differential Microphone.
- F **process** (const F &front, const F &back, const F &external_beta=F(-1), bool update←_beta=true)
ADM (p. [261](#)) 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`

4.33.1 Detailed Description

```
template<class F>
class ADM::ADM< F >
```

Adaptive differential microphone, working for speech frequency range.

4.33.2 Constructor & Destructor Documentation

```
4.33.2.1 ADM() template<class F >
ADM::ADM< F >:: ADM (
    F fs,
    F dist,
    unsigned lp_order,
    const F * lp_alphas,
    unsigned decomb_order,
    const F * decomb_alphas,
    F tau_beta = F(50e-3),
    F mu_beta = F(1e-4) )
```

Create Adaptive Differential Microphone.

Parameters

<code>fs</code>	Sampling rate / Hz
<code>dist</code>	Distance between physical microphones / m
<code>lp_order</code>	Filter order of FIR lowpass filter used for adaptation

Parameters

<i>lp_alpha</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_order</i> ≤ 1 deactivates filter.
<i>decomb_alpha</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>	adaptation speed

4.33.3 Member Function Documentation

4.33.3.1 process() template<class F >
F ADM::ADM< F >::process (const F & *front*, const F & *back*, const F & *external_beta* = F(-1), bool *update_beta* = true) [inline]

ADM (p. 261) 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.
<i>update_beta</i>	Perform the beta adaptation step?

Returns

The computed output sample

4.33.3.2 `beta()` template<class F >
F **ADM::ADM**< F >::beta () const [inline]

4.33.4 Member Data Documentation

4.33.4.1 `m_delay_front` template<class F >
Delay<F> **ADM::ADM**< F >::m_delay_front [private]

4.33.4.2 `m_delay_back` template<class F >
Delay<F> **ADM::ADM**< F >::m_delay_back [private]

4.33.4.3 `m_lp_bf` template<class F >
Linearphase_FIR<F> **ADM::ADM**< F >::m_lp_bf [private]

4.33.4.4 `m_lp_result` template<class F >
Linearphase_FIR<F> **ADM::ADM**< F >::m_lp_result [private]

4.33.4.5 `m_decomb` template<class F >
Linearphase_FIR<F> **ADM::ADM**< F >::m_decomb [private]

4.33.4.6 `m_beta` template<class F >
F **ADM::ADM**< F >::m_beta [private]

4.33.4.7 m_mu_beta template<class F >
F ADM::ADM< F >::m_mu_beta [private]

4.33.4.8 m_powerfilter_coeff template<class F >
F ADM::ADM< F >::m_powerfilter_coeff [private]

4.33.4.9 m_powerfilter_norm template<class F >
F ADM::ADM< F >::m_powerfilter_norm [private]

4.33.4.10 m_powerfilter_state template<class F >
F ADM::ADM< F >::m_powerfilter_state [private]

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

- adm.hh

4.34 ADM::Delay< F > Class Template Reference

A delay-line class.

Public Member Functions

- **Delay** (F samples, F f_design, F fs)
Create a signal delay object.
- **~Delay ()**
- **F process** (const F &in_sample)
Apply delay to signal.

Private Attributes

- **unsigned m_fullsamples**
Integer part of delay.
- **F m_coeff**
coefficient for 1st order IIR lowpass filter which does the subsample delay
- **F m_norm**
normalization for the IIR subsample delay filter
- **F * m_state**
Ringbuffer: Delayline.
- **unsigned m_now_in**
current position for inserting new samples into m_state ringbuffer

4.34.1 Detailed Description

```
template<class F>
class ADM::Delay< F >
```

A delay-line class.

It can delay samples in a single audio channel. It stores samples while they are delayed until they have reached their target delay. It can also do subsample-delays for a limited frequency range below fs/4.

4.34.2 Constructor & Destructor Documentation

```
4.34.2.1 Delay() 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>	design frequency (in Hz). Subsampledelay is exact for this frequency and approximate for different frequencies
<i>fs</i>	sampling frequency (in Hz).

4.34.2.2 ~Delay() template<class F >
ADM::Delay< F >::~ Delay

4.34.3 Member Function Documentation

4.34.3.1 process() template<class F >
F ADM::Delay< F >::process (const F & *in_sample*) [inline]

Apply delay to signal.

Whenever a new audio sample enters the delay line, a previous audio sample, now delayed, is returned by this method. Sub-sample-delays are implemented by applying a first-order recursive lowpass filter. This method needs to be called repeatedly, once for each incoming audio sample in correct order for a block of audio with multiple samples (oldest first, newest last).

Parameters

<i>in_sample</i>	The current input signal sample
------------------	---------------------------------

Returns

The output sample, which is one of the previously received input samples except for the sub-sample delay.

4.34.4 Member Data Documentation

4.34.4.1 m_fullsamples template<class F >
unsigned ADM::Delay< F >::m_fullsamples [private]

Integer part of delay.

4.34.4.2 m_coeff template<class F >
F **ADM::Delay**< F >::m_coeff [private]

coefficient for 1st order IIR lowpass filter which does the subsample delay

4.34.4.3 m_norm template<class F >
F **ADM::Delay**< F >::m_norm [private]

normalization for the IIR subsample delay filter

4.34.4.4 m_state template<class F >
F* **ADM::Delay**< F >::m_state [private]

Ringbuffer: Delayline.

4.34.4.5 m_now_in template<class F >
unsigned **ADM::Delay**< F >::m_now_in [private]

current position for inserting new samples into m_state ringbuffer

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

- adm.hh

4.35 ADM::Linearpase_FIR< F > Class Template Reference

An efficient linear-phase fir filter implementation.

Public Member Functions

- **Linearpase_FIR** (unsigned order, const F *alphas)
Create linear-phase FIR filter.
- **~Linearpase_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

4.35.1 Detailed Description

```
template<class F>
class ADM::Linearphase_FIR< F >
```

An efficient linear-phase fir filter implementation.

4.35.2 Constructor & Destructor Documentation

```
4.35.2.1 Linearphase_FIR() template<class F >
ADM::Linearphase_FIR< F >:: Linearphase_FIR (
    unsigned order,
    const F * alphas )
```

Create linear-phase FIR filter.

Parameters

<code>order</code>	filter order of this FIR filter. restriction: must be even.
<code>alphas</code>	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.)

```
4.35.2.2 ~Linearphase_FIR() template<class F >
ADM::Linearphase_FIR< F >::~ Linearphase_FIR
```

4.35.3 Member Function Documentation

4.35.3.1 process() template<class F >
F ADM::Linearphase_FIR< F >::process (const F & *in_sample*) [inline]

Filter one sample with this linear-phase FIR filter.

Parameters

<i>in_sample</i>	the current input sample
------------------	--------------------------

Returns

the computed output sample

4.35.4 Member Data Documentation

4.35.4.1 m_order template<class F >
unsigned ADM::Linearphase_FIR< F >::m_order [private]

The filter order of this linear-phase FIR filter.

4.35.4.2 m_alphas template<class F >
F* ADM::Linearphase_FIR< F >::m_alphas [private]

FIR filter coefficients.

Only m_order / 2 + 1 coefficients need to be stored since coefficients of linear-phase FIR filters are symmetric

4.35.4.3 m_output template<class F >
F* ADM::Linearphase_FIR< F >::m_output [private]

Ringbuffer for building future output.

```
4.35.4.4 m_now template<class F>
unsigned ADM::Linearphase_FIR< F >::m_now [private]
```

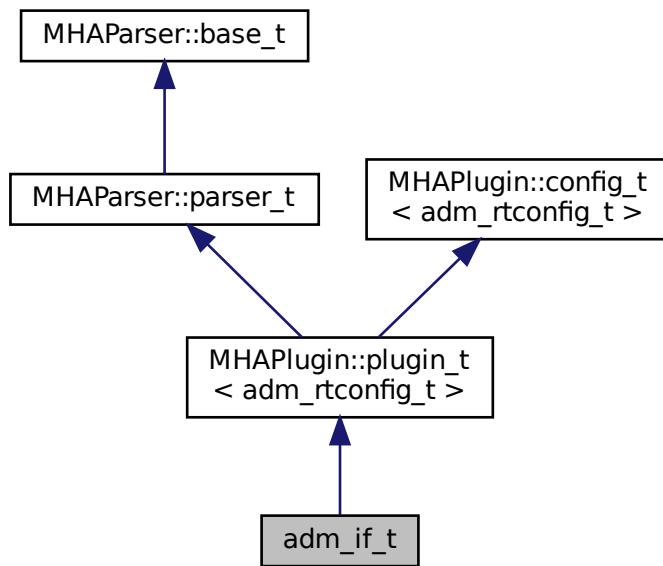
current start of ringbuffer

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

- **adm.hh**

4.36 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**
- **MHAParser::int_t adaptation_ratio**
- unsigned **input_channels**
- int **framecnt**
- **mha_real_t srate**
- **MHAEvents::patchbay_t< adm_if_t > patchbay**

Additional Inherited Members

4.36.1 Constructor & Destructor Documentation

4.36.1.1 adm_if_t() `adm_if_t::adm_if_t (`
 `const algo_comm_t & ac,`
 `const std::string & thread_name,`
 `const std::string & algo_name)`

4.36.2 Member Function Documentation

4.36.2.1 process() `mha_wave_t * adm_if_t::process (`
`mha_wave_t * in)`

4.36.2.2 prepare() `void adm_if_t::prepare (`
`mhaconfig_t & cfg) [virtual]`

Implements `MHAPlugIn::plugin_t< adm_rtconfig_t >` (p. [1148](#)).

4.36.2.3 release() `void adm_if_t::release () [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< adm_rtconfig_t >` (p. [1149](#)).

4.36.2.4 update() `void adm_if_t::update () [private]`

4.36.2.5 is_prepared() `bool adm_if_t::is_prepared () [inline], [private]`

4.36.3 Member Data Documentation

4.36.3.1 out MHASignal::waveform_t* adm_if_t::out [private]

4.36.3.2 front_channels MHAParser::vint_t adm_if_t::front_channels [private]

4.36.3.3 rear_channels `MHAParser::vint_t adm_if_t::rear_channels [private]`

4.36.3.4 distances `MHAParser::vfloat_t adm_if_t::distances [private]`

4.36.3.5 lp_order `MHAParser::int_t adm_if_t::lp_order [private]`

4.36.3.6 decomb_order `MHAParser::int_t adm_if_t::decomb_order [private]`

4.36.3.7 bypass `MHAParser::int_t adm_if_t::bypass [private]`

4.36.3.8 beta `MHAParser::float_t adm_if_t::beta [private]`

4.36.3.9 mu_beta `MHAParser::vfloat_t adm_if_t::mu_beta [private]`

4.36.3.10 tau_beta `MHAParser::vfloat_t adm_if_t::tau_beta [private]`

4.36.3.11 coeff_lp `MHAParser::vfloat_mon_t adm_if_t::coeff_lp [private]`

4.36.3.12 coeff_decomb `MHAParser::vfloat_mon_t adm_if_t::coeff_decomb` [private]

4.36.3.13 adaptation_ratio `MHAParser::int_t adm_if_t::adaptation_ratio` [private]

4.36.3.14 input_channels `unsigned adm_if_t::input_channels` [private]

4.36.3.15 framecnt `int adm_if_t::framecnt` [private]

4.36.3.16 srate `mha_real_t adm_if_t::srate` [private]

4.36.3.17 patchbay `MHAEvents::patchbay_t< adm_if_t> adm_if_t::patchbay` [private]

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

- `adm.cpp`

4.37 **adm_rtconfig_t** Class Reference

Public Types

- `typedef ADM::ADM< mha_real_t > adm_t`

Public Member Functions

- **adm_rtconfig_t** (unsigned nchannels_in, unsigned nchannels_out, int adaptation_ratio_, const std::vector< int > & **front_channels**, const std::vector< int > & **rear_channels**, const **mha_real_t** fs, const std::vector< **mha_real_t** > &distances, const int lp_order, const int decomb_order, const std::vector< **mha_real_t** > &tau_beta, const std::vector< **mha_real_t** > &mu_beta)

Construct new ADMs.

- virtual ~**adm_rtconfig_t** ()
- size_t **num_adms** () const
- **adm_t** & **adm** (unsigned index)

Returns adm object number index.

- int **front_channel** (unsigned index) const

Returns index of front channel for adm number index.

- int **rear_channel** (unsigned index) const

Returns index of rear channel for adm number index.

- int **get_adaptation_ratio** () const

Private Member Functions

- void **check_index** (unsigned index) const
- Index checking for all internal arrays.*

Private Attributes

- std::vector< int > **front_channels**
Indices of channels containing the signals from the front microphones.
- std::vector< int > **rear_channels**
Indices of channels containing the signals from the rear microphones.
- int **adaptation_ratio**
Prescale.
- **MHASignal::waveform_t** * **lp_coeffs**
Lowpass filter coefficients.
- std::vector< **MHASignal::waveform_t** * > **decomb_coeffs**
Decomb-Filter coefficients.
- std::vector< **adm_t** * > **adms**
ADMs.

4.37.1 Member Typedef Documentation

4.37.1.1 adm_t `typedef ADM::ADM< mha_real_t> adm_rtconfig_t::adm_t`

4.37.2 Constructor & Destructor Documentation

4.37.2.1 adm_rtconfig_t() `adm_rtconfig_t::adm_rtconfig_t (`

```
    unsigned nchannels_in,
    unsigned nchannels_out,
    int adaptation_ratio_,
    const std::vector< int > & front_channels,
    const std::vector< int > & rear_channels,
    const mha_real_t fs,
    const std::vector< mha_real_t > & distances,
    const int lp_order,
    const int decomb_order,
    const std::vector< mha_real_t > & tau_beta,
    const std::vector< mha_real_t > & mu_beta )
```

Construct new ADMs.

Used when configuration changes.

Parameters

<i>nchannels_in</i>	Number of input channels
<i>nchannels_out</i>	Number of output channels
<i>adaptation_ratio_</i>	Update beta every <i>adaptation_ratio</i> frames
<i>front_channels</i>	Parser's <i>front_channels</i> setting
<i>rear_channels</i>	Parser's <i>front_channels</i> 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

4.37.2.2 ~adm_rtconfig_t() `adm_rtconfig_t::~adm_rtconfig_t () [virtual]`

4.37.3 Member Function Documentation

4.37.3.1 `check_index()` `void adm_rtconfig_t::check_index (`
`unsigned index) const [inline], [private]`

Index checking for all internal arrays.

Exceptions

MHA_Error (p. 763)	if index out of range.
--	------------------------

4.37.3.2 `num_adms()` `size_t adm_rtconfig_t::num_adms () const [inline]`

4.37.3.3 `adm()` `adm_t& adm_rtconfig_t::adm (`
`unsigned index) [inline]`

Returns adm object number index.

4.37.3.4 `front_channel()` `int adm_rtconfig_t::front_channel (`
`unsigned index) const [inline]`

Returns index of front channel for adm number index.

4.37.3.5 `rear_channel()` `int adm_rtconfig_t::rear_channel (`
`unsigned index) const [inline]`

Returns index of rear channel for adm number index.

4.37.3.6 get_adaptation_ratio() int adm_rtconfig_t::get_adaptation_ratio () const [inline]

4.37.4 Member Data Documentation

4.37.4.1 front_channels std::vector<int> adm_rtconfig_t::front_channels [private]

Indices of channels containing the signals from the front microphones.

4.37.4.2 rear_channels std::vector<int> adm_rtconfig_t::rear_channels [private]

Indices of channels containing the signals from the rear microphones.

4.37.4.3 adaptation_ratio int adm_rtconfig_t::adaptation_ratio [private]

Prescale.

4.37.4.4 lp_coeffs MHASignal::waveform_t* adm_rtconfig_t::lp_coeffs [private]

Lowpass filter coefficients.

4.37.4.5 decomb_coeffs std::vector< MHASignal::waveform_t*> adm_rtconfig_t::decomb_coeffs [private]

Decomb-Filter coefficients.

4.37.4.6 adms std::vector< **adm_t** *> adm_rtconfig_t::adms [private]

ADMs.

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

- **adm.cpp**

4.38 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(* **insert_var_double**)(void *, const char *, double *)
- 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_var_double**)(void *, const char *, double *)
- 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.

4.38.1 Detailed Description

A reference handle for algorithm communication variables.

This structure contains a control handle and a set of function pointers for sharing variables within one processing chain. See also section **Communication between algorithms** (p. 23).

4.38.2 Member Data Documentation

4.38.2.1 handle algo_comm_t::handle

AC variable control handle.

4.38.2.2 insert_var 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. 360)

Returns

Error code or zero on success

4.38.2.3 insert_var_int 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

4.38.2.4 insert_var_float 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

4.38.2.5 insert_var_double int(* algo_comm_t::insert_var_double) (void *, const char *, double *)**4.38.2.6 remove_var 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

4.38.2.7 remove_ref 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

4.38.2.8 is_var 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

4.38.2.9 get_var algo_comm_t::get_var

Get the variable handle of an AC variable.

This function returns the variable handle **comm_var_t** (p. 360) 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

4.38.2.10 get_var_int 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

4.38.2.11 get_var_float 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

4.38.2.12 get_var_double int(* algo_comm_t::get_var_double) (void *, const char *, double *)**4.38.2.13 get_entries** 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.

4.38.2.14 `get_error` `algo_comm_t::get_error`

Convert AC error codes into human readable error messages.

Parameters

<i>e</i>	Error code
----------	------------

Returns

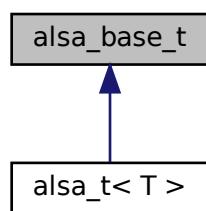
Error message

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

- `mha.hh`
- `mha_algo_comm.cpp`

4.39 `alsa_base_t` Class Reference

Inheritance diagram for `alsa_base_t`:



Public Member Functions

- **alsa_base_t ()**
- virtual ~**alsa_base_t ()**=default
- virtual void **start ()=0**
start puts alsa device in usable state
- virtual void **stop ()=0**
stop informs alsa device that we do not need any more samples / will not provide any more samples
- virtual bool **read (mha_wave_t **)=0**
*read audio samples from the device into an internal **mha_wave_t** (p. 839) buffer, then update the pointer given as parameter to point to the internal structure.*
- virtual bool **write (mha_wave_t *)=0**
write audio samples from the given waveform buffer to the sound device.

Public Attributes

- **snd_pcm_t * pcm**
The underlying alsa handle to this sound card.

4.39.1 Constructor & Destructor Documentation

4.39.1.1 **alsa_base_t()** `alsa_base_t::alsa_base_t () [inline]`

4.39.1.2 **~alsa_base_t()** `virtual alsa_base_t::~alsa_base_t () [virtual], [default]`

4.39.2 Member Function Documentation

4.39.2.1 **start()** `virtual void alsa_base_t::start () [pure virtual]`

start puts alsa device in usable state

Implemented in **alsa_t< T >** (p. 293).

4.39.2.2 stop() `virtual void alsa_base_t::stop () [pure virtual]`

stop informs alsa device that we do not need any more samples / will not provide any more samples

Implemented in `alsa_t< T >` (p. [293](#)).

4.39.2.3 read() `virtual bool alsa_base_t::read (`
`mha_wave_t **) [pure virtual]`

read audio samples from the device into an internal `mha_wave_t` (p. [839](#)) buffer, then update the pointer given as parameter to point to the internal structure.

Converts sound samples from the integer data type provided by the sound card to floating-point values needed by the MHA in the range [-1.0,1.0]

Implemented in `alsa_t< T >` (p. [293](#)).

4.39.2.4 write() `virtual bool alsa_base_t::write (`
`mha_wave_t *) [pure virtual]`

write audio samples from the given waveform buffer to the sound device.

converts the floating point values coming from the MHA to the integer samples required by the sound card.

Implemented in `alsa_t< T >` (p. [293](#)).

4.39.3 Member Data Documentation**4.39.3.1 pcm** `snd_pcm_t* alsa_base_t::pcm`

The underlying alsa handle to this sound card.

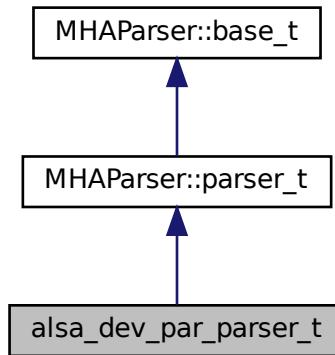
The documentation for this class was generated from the following file:

- **MHAIoalsa.cpp**

4.40 `alsa_dev_par_parser_t` Class Reference

Parser variables corresponding to one alsa device.

Inheritance diagram for `alsa_dev_par_parser_t`:



Public Member Functions

- **`alsa_dev_par_parser_t` (`snd_pcm_stream_t stream_dir`)**
Constructor inserts the parser variables into this sub-parser.

Public Attributes

- **`MHParse::string_t device`**
Name of the device in the alsa world, like "hw:0.0", "default", etc.
- **`MHParse::int_t nperiods`**
Number of buffers of fragsize to hold in the alsa buffer.
- **`snd_pcm_stream_t stream_dir`**
Remember the direction (capture/playback) of this device.

Additional Inherited Members

4.40.1 Detailed Description

Parser variables corresponding to one alsa device.

ALSA separates audio capture and audio playback into two different devices that have to be opened separately. This class encapsulates the parser variables that pertain to one such direction.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 `alsa_dev_par_parser_t()` `alsa_dev_par_parser_t::alsa_dev_par_parser_t (snd_pcm_stream_t stream_dir)`

Constructor inserts the parser variables into this sub-parser.

Parameters

<code>stream_dir</code>	capture or playback
-------------------------	---------------------

4.40.3 Member Data Documentation

4.40.3.1 `device` `MHAParser::string_t alsa_dev_par_parser_t::device`

Name of the device in the alsal world, like "hw:0.0", "default", etc.

4.40.3.2 `nperiods` `MHAParser::int_t alsa_dev_par_parser_t::nperiods`

Number of buffers of fragsize to hold in the alsal buffer.

Usually 2, the minimum possible.

4.40.3.3 `stream_dir` `snd_pcm_stream_t alsa_dev_par_parser_t::stream_dir`

Remember the direction (capture/playback) of this device.

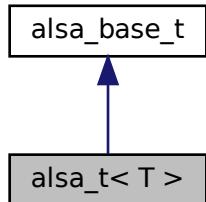
The documentation for this class was generated from the following file:

- **MHAIOalsa.cpp**

4.41 `alsa_t< T >` Class Template Reference

Our representation of one alsal device.

Inheritance diagram for `alsa_t< T >`:



Public Member Functions

- `alsa_t (const alsadev_par_parser_t &par, unsigned int rate, unsigned int fragsize, unsigned int channels)`
Constructor receives the parameters for this device.
- `~alsa_t ()`
Destructor closes the sound device.
- `void start () override`
start puts alsal device in usable state
- `void stop () override`
stop informs alsal device that we do not need any more samples / will not provide any more samples
- `bool read (mha_wave_t **) override`
read audio samples from the device into an internal `mha_wave_t` (p. 839) buffer, then update the pointer given as parameter to point to the internal structure.
- `bool write (mha_wave_t *) override`
write audio samples from the given waveform buffer to the sound device.

Private Attributes

- `unsigned int channels`
- `unsigned int fragsize`
- `T * buffer`
- `std::vector< mha_real_t > frame_data`
- `MHASignal::waveform_t wave`
internal buffer to store sound samples coming from the sound card.
- `const mha_real_t gain`
- `const mha_real_t invgain`
- `snd_pcm_format_t pcm_format`

Additional Inherited Members

4.41.1 Detailed Description

```
template<typename T>
class alsa_t< T >
```

Our representation of one alsa device.

We can start and stop the device, and depending on the direction, read or write samples.

4.41.2 Constructor & Destructor Documentation

```
4.41.2.1 alsa_t() template<typename T >
alsa_t< T >:: alsa_t (
    const alsa_dev_par_parser_t & par,
    unsigned int rate,
    unsigned int fragsize,
    unsigned int channels )
```

Constructor receives the parameters for this device.

It opens the sound device using the alsa library and selects the given parameters, but does not yet start the sound device to perform real I/O.

Parameters

<i>par</i>	our parser variable aggregator (containing direction, device name, and number of periods to place in alsa buffer)
<i>rate</i>	sampling rate in Hz
<i>fragsize</i>	samples per block per channel
<i>channels</i>	number of audio channels to open

```
4.41.2.2 ~alsa_t() template<typename T >
alsa_t< T >::~ alsa_t
```

Destructor closes the sound device.

4.41.3 Member Function Documentation

4.41.3.1 **start()** template<typename T >

```
void alsa_t< T >::start [override], [virtual]
```

start puts alsa device in usable state

Implements **alsa_base_t** (p. [287](#)).

4.41.3.2 **stop()** template<typename T >

```
void alsa_t< T >::stop [override], [virtual]
```

stop informs alsa device that we do not need any more samples / will not provide any more samples

Implements **alsa_base_t** (p. [287](#)).

4.41.3.3 **read()** template<typename T >

```
bool alsa_t< T >::read (
    mha_wave_t ** s ) [override], [virtual]
```

read audio samples from the device into an internal **mha_wave_t** (p. [839](#)) buffer, then update the pointer given as parameter to point to the internal structure.

Converts sound samples from the integer data type provided by the sound card to floating-point values needed by the MHA in the range [-1.0,1.0]

Implements **alsa_base_t** (p. [288](#)).

4.41.3.4 **write()** template<typename T >

```
bool alsa_t< T >::write (
    mha_wave_t * s ) [override], [virtual]
```

write audio samples from the given waveform buffer to the sound device.

converts the floating point values coming from the MHA to the integer samples required by the sound card.

Implements **alsa_base_t** (p. [288](#)).

4.41.4 Member Data Documentation

4.41.4.1 channels template<typename T >
unsigned int **alsa_t**< T >::channels [private]

4.41.4.2 fragsize template<typename T >
unsigned int **alsa_t**< T >::fragsize [private]

4.41.4.3 buffer template<typename T >
T* **alsa_t**< T >::buffer [private]

4.41.4.4 frame_data template<typename T >
std::vector< **mha_real_t**> **alsa_t**< T >::frame_data [private]

4.41.4.5 wave template<typename T >
MHASignal::waveform_t **alsa_t**< T >::wave [private]

internal buffer to store sound samples coming from the sound card.

4.41.4.6 gain template<typename T >
const **mha_real_t** **alsa_t**< T >::gain [private]

4.41.4.7 invgain template<typename T >
const **mha_real_t** **alsa_t**< T >::invgain [private]

```
4.41.4.8 pcm_format template<typename T >
snd_pcm_format_t alsa_t< T >::pcm_format [private]
```

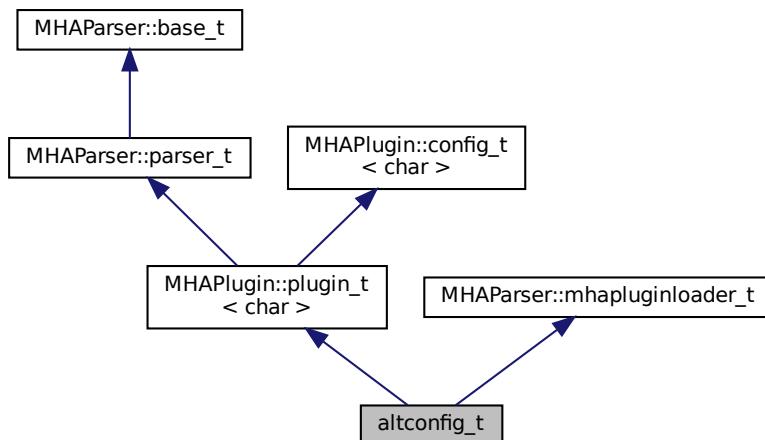
The documentation for this class was generated from the following file:

- **MHAIoalsa.cpp**

4.42 altconfig_t Class Reference

Single class implementing plugin altconfig.

Inheritance diagram for altconfig_t:



Public Member Functions

- **altconfig_t (algo_comm_t iac, const char *chain, const char *algo)**
Constructor initializes an instance of the altconfig plugin.
- **void prepare (mhaconfig_t &cf)**
Invoked by MHA when this plugin should prepare for signal processing.
- **void release ()**
Invoked by MHA when this plugin should call its release function.

Private Member Functions

- void **on_set_algos ()**
Callback executed when the configuration variable "algos" is written to at run time.
- void **on_set_select ()**
Callback executed when the configuration variable "select" is successfully written to at run time.
- void **event_select_all ()**
Callback executed when the configuration variable "selectall" is written to at run time.
- std::map< std::string, MHParse::string_t > **save_state ()**
Save the old state of the user-defined sub-parsers for eventual restoration later.
- void **restore_state (std::map< std::string, MHParse::string_t > &state, std::map< std::string, MHParse::string_t > &failed_state)**
Restore the old parser state from a saved state.

Private Attributes

- MHParse::vstring_t **parser_algos**
- MHParse::kw_t **select_plug**
- MHParse::bool_t **selectall**
- std::map< std::string, MHParse::string_t > **configs**
Storage for alternative configuration commands.
- MHAEvents::patchbay_t< altconfig_t > **patchbay**
Configuration event broker.

Additional Inherited Members

4.42.1 Detailed Description

Single class implementing plugin altconfig.

altconfig loads another plugin and can send configuration commands to that plugin. altconfig does not need a separate runtime configuration class, template parameter char is used as a placeholder. Uses parser variable names "algorithms" and "select" even though the alternatives that can be selected in this plugin are not algorithms or plugins but configuration commands in order to be interface-compatible with plugin altplugs. mhacontrol has a UI area that automatically fills with the alternatives found in either altplugs or altconfig if the complete MHA loads exactly one plugin with this interface.

4.42.2 Constructor & Destructor Documentation

```
4.42.2.1 altconfig_t() altconfig_t::altconfig_t (
    algo_comm_t iac,
    const char * chain,
    const char * algo )
```

Constructor initializes an instance of the altconfig plugin.

Parameters

<i>iac</i>	Algorithm communication variable space, not used by this plugin except to initialize base class.
<i>algo</i>	Configured name of this plugin.

4.42.3 Member Function Documentation

4.42.3.1 `prepare()` `void altconfig_t::prepare (mhaconfig_t & cf) [inline], [virtual]`

Invoked by MHA when this plugin should prepare for signal processing.

altconfig delegates to the loaded plugin and does not need to do more work to prepare.

Parameters

<i>cf</i>	signal dimensions, forwarded to loaded plugin which may change the signal dimensions.
-----------	---

Implements `MHAPlugin::plugin_t< char >` (p. [1148](#)).

4.42.3.2 `release()` `void altconfig_t::release () [inline], [virtual]`

Invoked by MHA when this plugin should call its release function.

altconfig delegates to the loaded plugin.

Reimplemented from `MHAPlugin::plugin_t< char >` (p. [1149](#)).

4.42.3.3 `on_set_algos()` `void altconfig_t::on_set_algos () [private]`

Callback executed when the configuration variable "algorithms" is written to at run time.

Adds the configuration variables that store the alternative configuration commands based on the new names and sets the allowed values of configuration variable "select"

4.42.3.4 on_set_select() void altconfig_t::on_set_select () [private]

Callback executed when the configuration variable "select" is successfully written to at run time.

Causes the execution of the stored command for the selected condition in the context of the loaded plugin.

4.42.3.5 event_select_all() void altconfig_t::event_select_all () [private]

Callback executed when the configuration variable "selectall" is written to at run time.

When set to yes, iterates once through all stored configurations in the order they appear in configuration variable algos.

4.42.3.6 save_state() std::map< std::string, MHAParser::string_t > altconfig_t::save_state () [private]

Save the old state of the user-defined sub-parsers for eventual restoration later.

operator= does not suffice to store/restore the old state because we need to re-insert the string_t's that were removed, but a copy of the string_t keeps a pointer to its parent and complains if we try to insert_item it again. So we just copy the relevant data into a new string_t.

Parameters

<i>old_state</i>	old parser state
------------------	------------------

Returns

Copy of the old state

4.42.3.7 restore_state() void altconfig_t::restore_state (std::map< std::string, MHAParser::string_t > & state, std::map< std::string, MHAParser::string_t > & failed_state) [private]

Restore the old parser state from a saved state.

Parameters

<i>state</i>	old parser state
--------------	------------------

4.42.4 Member Data Documentation

4.42.4.1 parser_algos `MHAParser::vstring_t altconfig_t::parser_algos [private]`

4.42.4.2 select_plug `MHAParser::kw_t altconfig_t::select_plug [private]`

4.42.4.3 selectall `MHAParser::bool_t altconfig_t::selectall [private]`

4.42.4.4 configs `std::map<std::string, MHAParser::string_t> altconfig_t::configs [private]`

Storage for alternative configuration commands.

New entries are registered with the plugin's parser when parser_algos is updated.

4.42.4.5 patchbay `MHAEvents::patchbay_t< altconfig_t> altconfig_t::patchbay [private]`

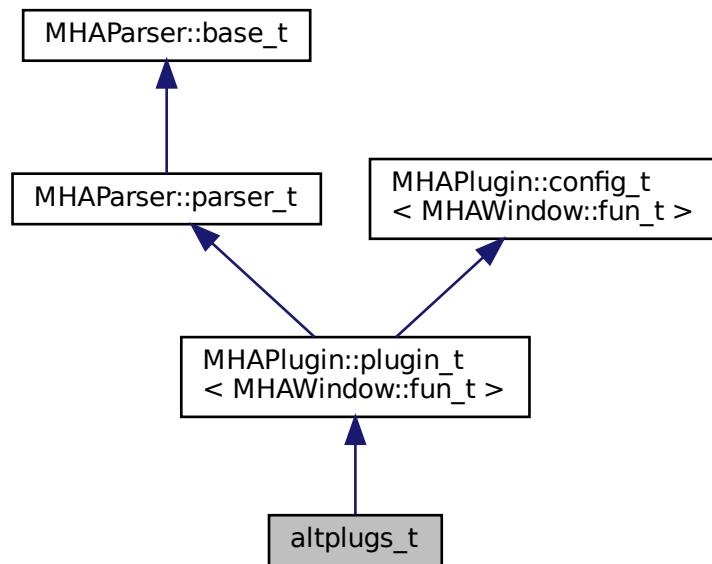
Configuration event broker.

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

- **altconfig.hh**
- **altconfig.cpp**

4.43 altplugs_t Class Reference

Inheritance diagram for altplugs_t:



Public Member Functions

- `altplugs_t (algo_comm_t iac, const char *chain, const char *algo)`
- `void prepare (mhaconfig_t &)`
- `void release ()`
- `void process (mha_wave_t *, mha_wave_t **)`
- `void process (mha_spec_t *, mha_wave_t **)`
- `void process (mha_wave_t *, mha_spec_t **)`
- `void process (mha_spec_t *, mha_spec_t **)`
- `virtual std::string parse (const std::string &arg)`
- `virtual void parse (const char *a1, char *a2, unsigned int a3)`

Private Member Functions

- `void event_set_plugs ()`
- `void event_add_plug ()`
- `void event_delete_plug ()`
- `void event_select_plug ()`
- `void update_selector_list ()`
- `void update_ramplen ()`
- `void proc_ramp (mha_wave_t *s)`

Private Attributes

- `MHAParser::bool_t use_own_ac`
- `MHAParser::vstring_t parser_plugs`
- `MHAParser::string_t add_plug`
- `MHAParser::string_t delete_plug`
- `MHAParser::float_t ramplen`
- `MHAParser::kw_t select_plug`
- `MHAParser::parser_t current`
- `MHAParser::vstring_mon_t nondefault_labels`
- `std::vector< mhaplug_cfg_t * > plugs`
- `mhaplug_cfg_t * selected_plug`
- `MHAEvents::patchbay_t< altplugs_t > patchbay`
- `MHASignal::waveform_t * fallback_wave`
- `MHASignal::spectrum_t * fallback_spec`
- `mhaconfig_t cfin`
- `mhaconfig_t cout`
- `bool prepared`
- `bool added_via_plugs`
- `unsigned int ramp_counter`
- `unsigned int ramp_len`

Additional Inherited Members

4.43.1 Constructor & Destructor Documentation

```
4.43.1.1 altplugs_t() altplugs_t::altplugs_t (
    algo_comm_t iac,
    const char * chain,
    const char * algo )
```

4.43.2 Member Function Documentation

```
4.43.2.1 prepare() void altplugs_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAPlugin::plugin_t< MHAWindow::fun_t >` (p. 1148).

4.43.2.2 release() void altplugs_t::release () [virtual]

Reimplemented from **MHAParser::base_t**<**MHAWindow::fun_t**> (p. 1149).

4.43.2.3 process() [1/4] void altplugs_t::process (

```
mha_wave_t * sIn,  
mha_wave_t ** sOut )
```

4.43.2.4 process() [2/4] void altplugs_t::process (

```
mha_spec_t * sIn,  
mha_wave_t ** sOut )
```

4.43.2.5 process() [3/4] void altplugs_t::process (

```
mha_wave_t * sIn,  
mha_spec_t ** sOut )
```

4.43.2.6 process() [4/4] void altplugs_t::process (

```
mha_spec_t * sIn,  
mha_spec_t ** sOut )
```

4.43.2.7 parse() [1/2] std::string altplugs_t::parse (

```
const std::string & arg ) [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1031).

4.43.2.8 parse() [2/2] virtual void altplugs_t::parse (

```
const char * a1,  
char * a2,  
unsigned int a3 ) [inline], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1031).

4.43.2.9 event_set_plugs() void altplugs_t::event_set_plugs () [private]

4.43.2.10 event_add_plug() void altplugs_t::event_add_plug () [private]

4.43.2.11 event_delete_plug() void altplugs_t::event_delete_plug () [private]

4.43.2.12 event_select_plug() void altplugs_t::event_select_plug () [private]

4.43.2.13 update_selector_list() void altplugs_t::update_selector_list () [private]

4.43.2.14 update_ramplen() void altplugs_t::update_ramplen () [private]

4.43.2.15 proc_ramp() void altplugs_t::proc_ramp (
 mha_wave_t * s) [private]

4.43.3 Member Data Documentation

4.43.3.1 use_own_ac MHAParser::bool_t altplugs_t::use_own_ac [private]

4.43.3.2 parser_plugs `MHAParser::vstring_t` `altplugs_t::parser_plugs` [private]

4.43.3.3 add_plug `MHAParser::string_t` `altplugs_t::add_plug` [private]

4.43.3.4 delete_plug `MHAParser::string_t` `altplugs_t::delete_plug` [private]

4.43.3.5 ramplen `MHAParser::float_t` `altplugs_t::ramplen` [private]

4.43.3.6 select_plug `MHAParser::kw_t` `altplugs_t::select_plug` [private]

4.43.3.7 current `MHAParser::parser_t` `altplugs_t::current` [private]

4.43.3.8 nondefault_labels `MHAParser::vstring_mon_t` `altplugs_t::nondefault_labels` [private]

4.43.3.9 plugs `std::vector< mhaplug_cfg_t*>` `altplugs_t::plugs` [private]

4.43.3.10 selected_plug `mhaplug_cfg_t*` `altplugs_t::selected_plug` [private]

4.43.3.11 patchbay `MHAEEvents::patchbay_t< altplugs_t> altplugs_t::patchbay [private]`

4.43.3.12 fallback_wave `MHASignal::waveform_t* altplugs_t::fallback_wave [private]`

4.43.3.13 fallback_spec `MHASignal::spectrum_t* altplugs_t::fallback_spec [private]`

4.43.3.14 cfin `mhaconfig_t altplugs_t::cfin [private]`

4.43.3.15 cfout `mhaconfig_t altplugs_t::cfout [private]`

4.43.3.16 prepared `bool altplugs_t::prepared [private]`

4.43.3.17 added_via_plugs `bool altplugs_t::added_via_plugs [private]`

4.43.3.18 ramp_counter `unsigned int altplugs_t::ramp_counter [private]`

4.43.3.19 ramp_len `unsigned int altplugs_t::ramp_len [private]`

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

- **altplugs.cpp**

4.44 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_if_t<** **mha_real_t** **>** wave_fifo
- **mha_fifo_if_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

4.44.1 Constructor & Destructor Documentation

```
4.44.1.1 analysepath_t() 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 )
```

4.44.1.2 ~analysepath_t() analysepath_t::~analysepath_t () [virtual]

4.44.2 Member Function Documentation

4.44.2.1 rt_process() void analysepath_t::rt_process (
 mha_wave_t * outer_input)

4.44.2.2 svc() int analysepath_t::svc () [virtual]

4.44.3 Member Data Documentation

4.44.3.1 inner_process_wave2wave MHAProc_wave2wave_t analysepath_t::inner_<-
process_wave2wave [private]

4.44.3.2 inner_process_wave2spec `MHAProc_wave2spec_t` `analysepath_t::inner_<process_wave2spec [private]`

4.44.3.3 inner_input `MHASignal::waveform_t` `analysepath_t::inner_input [private]`

4.44.3.4 libdata `void*` `analysepath_t::libdata [private]`

4.44.3.5 wave_fifo `mha_fifo_lf_t< mha_real_t>` `analysepath_t::wave_fifo [private]`

4.44.3.6 ac_fifo `mha_fifo_lf_t< MHA_AC::acspace2matrix_t>` `analysepath_t::ac_fifo [private]`

4.44.3.7 inner_ac_copy `MHA_AC::acspace2matrix_t` `analysepath_t::inner_ac_copy [private]`

4.44.3.8 outer_ac_copy `MHA_AC::acspace2matrix_t` `analysepath_t::outer_ac_copy [private]`

4.44.3.9 outer_ac `algo_comm_t` `analysepath_t::outer_ac [private]`

4.44.3.10 inner_out_domain `mha_domain_t` `analysepath_t::inner_out_domain [private]`

4.44.3.11 inner_error `MHA_Error analysepath_t::inner_error [private]`

4.44.3.12 has_inner_error `bool analysepath_t::has_inner_error [private]`

4.44.3.13 flag_terminate_inner_thread `bool analysepath_t::flag_terminate_inner_thread [private]`

4.44.3.14 input_to_process `int analysepath_t::input_to_process [private]`

4.44.3.15 ProcessMutex `pthread_mutex_t analysepath_t::ProcessMutex [private]`

4.44.3.16 attr `pthread_attr_t analysepath_t::attr [private]`

4.44.3.17 priority `struct sched_param analysepath_t::priority [private]`

4.44.3.18 scheduler `int analysepath_t::scheduler [private]`

4.44.3.19 thread `pthread_t analysepath_t::thread [private]`

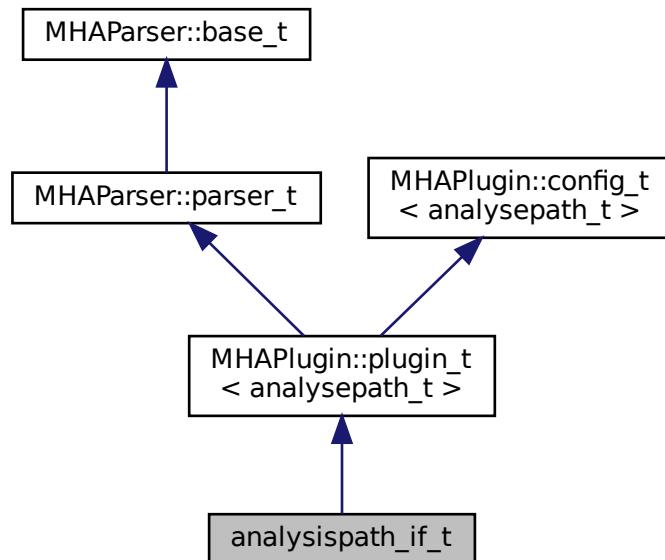
4.44.3.20 `cond_to_process` pthread_cond_t analysepath_t::cond_to_process [private]

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

- `analysispath.cpp`

4.45 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

4.45.1 Constructor & Destructor Documentation

4.45.1.1 analysispath_if_t() `analysispath_if_t::analysispath_if_t (algo_comm_t iac,
std::string th,
std::string al)`

4.45.1.2 ~analysispath_if_t() `analysispath_if_t::~analysispath_if_t ()`

4.45.2 Member Function Documentation

4.45.2.1 process() `mha_wave_t * analysispath_if_t::process (mha_wave_t * s)`

4.45.2.2 `prepare()` `void analysispath_if_t::prepare (mhaconfig_t & conf) [virtual]`

Implements `MHAPlugIn::plugin_t< analysepath_t >` (p. 1148).

4.45.2.3 `release()` `void analysispath_if_t::release () [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< analysepath_t >` (p. 1149).

4.45.2.4 `loadlib()` `void analysispath_if_t::loadlib () [private]`

4.45.3 Member Data Documentation

4.45.3.1 `patchbay` `MHAEvents::patchbay_t< analysispath_if_t > analysispath_if_t::patchbay [private]`

4.45.3.2 `libname` `MHAParser::string_t analysispath_if_t::libname [private]`

4.45.3.3 `fragsize` `MHAParser::int_t analysispath_if_t::fragsize [private]`

4.45.3.4 `fifolen` `MHAParser::int_t analysispath_if_t::fifolen [private]`

4.45.3.5 priority `MHAParser::int_t analysispath_if_t::priority [private]`

4.45.3.6 vars `MHAParser::vstring_t analysispath_if_t::vars [private]`

4.45.3.7 plug `plug_t* analysispath_if_t::plug [private]`

4.45.3.8 chain `std::string analysispath_if_t::chain [private]`

4.45.3.9 algo `std::string analysispath_if_t::algo [private]`

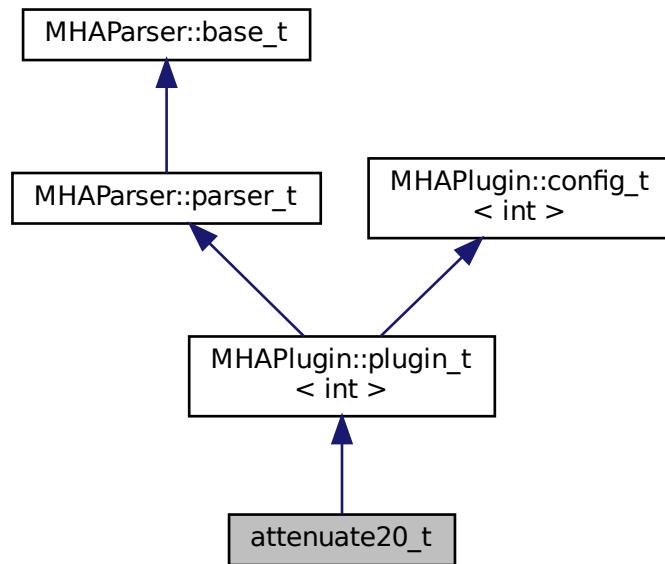
4.45.3.10 acspace_template `MHA_AC::acspace2matrix_t* analysispath_if_t::acspace<template> [private]`

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

- `analysispath.cpp`

4.46 attenuate20_t Class Reference

Inheritance diagram for attenuate20_t:



Public Member Functions

- `attenuate20_t (algo_comm_t & ac, const std::string &chain_name, const std::string &algo_name)`
- `void release (void) override`
- `void prepare (mhaconfig_t &signal_info) override`
- `mha_wave_t * process (mha_wave_t *signal)`

Additional Inherited Members

4.46.1 Constructor & Destructor Documentation

4.46.1.1 `attenuate20_t()` `attenuate20_t::attenuate20_t (`
 `algo_comm_t & ac,`
 `const std::string & chain_name,`
 `const std::string & algo_name) [inline]`

4.46.2 Member Function Documentation

4.46.2.1 `release()` `void attenuate20_t::release (void) [inline], [override], [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< int >` (p. 1149).

4.46.2.2 `prepare()` `void attenuate20_t::prepare (mhaconfig_t & signal_info) [inline], [override], [virtual]`

Implements `MHAPlugIn::plugin_t< int >` (p. 1148).

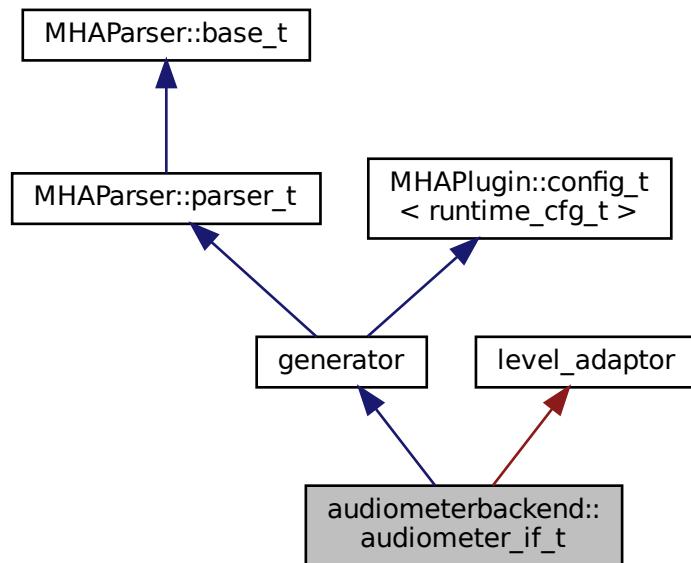
4.46.2.3 `process()` `mha_wave_t* attenuate20_t::process (mha_wave_t * signal) [inline]`

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

- `attenuate20.cpp`

4.47 audiometerbackend::audiometer_if_t Class Reference

Inheritance diagram for audiometerbackend::audiometer_if_t:



Public Member Functions

- `audiometer_if_t (algo_comm_t, const char *, const char *)`
- `mha_wave_t * process (mha_wave_t *)`
- `void prepare (mhaconfig_t &)`

Private Member Functions

- `void update ()`
- `void change_mode ()`
- `void set_level ()`

Private Attributes

- `MHAParser::int_t freq`
- `MHAParser::kw_t sigtype`
- `MHAParser::float_t level`
- `MHAParser::kw_t mode`
- `MHAParser::float_t ramplen`
- `MHASignal::loop_wavefragment_t::playback_mode_t pmode`
- `std::vector< int > outchannel`
- `MHAEvents::patchbay_t< audiometer_if_t > patchbay`

Additional Inherited Members

4.47.1 Constructor & Destructor Documentation

4.47.1.1 audiometer_if_t() `audiometerbackend::audiometer_if_t::audiometer_if_t (`
`algo_comm_t iac,`
`const char * ,`
`const char *)`

4.47.2 Member Function Documentation

4.47.2.1 process() `mha_wave_t * audiometerbackend::audiometer_if_t::process (mha_wave_t * s)`

4.47.2.2 prepare() `void audiometerbackend::audiometer_if_t::prepare (mhaconfig_t & tf) [virtual]`

Implements `MHAPlugIn::plugin_t< runtime_cfg_t >` (p. 1148).

4.47.2.3 update() `void audiometerbackend::audiometer_if_t::update () [private]`

4.47.2.4 change_mode() `void audiometerbackend::audiometer_if_t::change_mode () [private]`

4.47.2.5 set_level() `void audiometerbackend::audiometer_if_t::set_level () [private]`

4.47.3 Member Data Documentation

4.47.3.1 freq `MHAParser::int_t audiometerbackend::audiometer_if_t::freq [private]`

4.47.3.2 sigtype `MHAParser::kw_t audiometerbackend::audiometer_if_t::sigtype [private]`

4.47.3.3 level `MHAParser::float_t audiometerbackend::audiometer_if_t::level [private]`

4.47.3.4 mode `MHAParser::kw_t audiometerbackend::audiometer_if_t::mode` [private]

4.47.3.5 ramplen `MHAParser::float_t audiometerbackend::audiometer_if_t::ramplen` [private]

4.47.3.6 pmode `MHASignal::loop_wavefragment_t::playback_mode_t audiometerbackend::audiometer_if_t::pmode` [private]

4.47.3.7 outchannel `std::vector<int> audiometerbackend::audiometer_if_t::outchannel` [private]

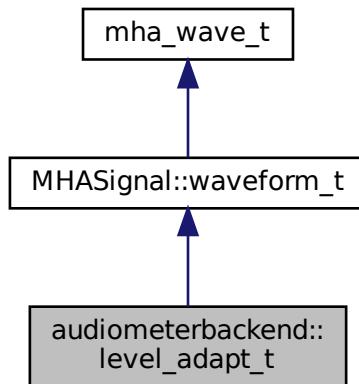
4.47.3.8 patchbay `MHAEvents::patchbay_t< audiometer_if_t> audiometerbackend::audiometer_if_t::patchbay` [private]

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

- `audiometerbackend.cpp`

4.48 audiometerbackend::level_adapt_t Class Reference

Inheritance diagram for audiometerbackend::level_adapt_t:



Public Member Functions

- `level_adapt_t (mhaconfig_t cf, mha_real_t adapt_len, mha_real_t l_new_, mha_real_t l_old_)`
- `void update_frame ()`
- `mha_real_t get_level () const`
- `bool can_update () const`

Private Attributes

- `unsigned int ilen`
- `unsigned int pos`
- `MHAWindow::fun_t wnd`
- `mha_real_t l_new`
- `mha_real_t l_old`

Additional Inherited Members

4.48.1 Constructor & Destructor Documentation

```
4.48.1.1 level_adapt_t() audiometerbackend::level_adapt_t::level_adapt_t (
    mhaconfig_t cf,
    mha_real_t adapt_len,
    mha_real_t l_new_,
    mha_real_t l_old_ )
```

4.48.2 Member Function Documentation

```
4.48.2.1 update_frame() void audiometerbackend::level_adapt_t::update_frame ( )
```

```
4.48.2.2 get_level() mha_real_t audiometerbackend::level_adapt_t::get_level ( )
const [inline]
```

4.48.2.3 can_update() `bool audiometerbackend::level_adapt_t::can_update () const [inline]`

4.48.3 Member Data Documentation

4.48.3.1 ilen `unsigned int audiometerbackend::level_adapt_t::ilen [private]`

4.48.3.2 pos `unsigned int audiometerbackend::level_adapt_t::pos [private]`

4.48.3.3 wnd `MHAWindow::fun_t audiometerbackend::level_adapt_t::wnd [private]`

4.48.3.4 l_new `mha_real_t audiometerbackend::level_adapt_t::l_new [private]`

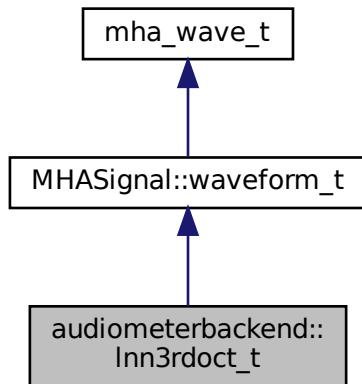
4.48.3.5 l_old `mha_real_t audiometerbackend::level_adapt_t::l_old [private]`

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

- **audiometerbackend.cpp**

4.49 audiometerbackend::Inn3rdoct_t Class Reference

Inheritance diagram for audiometerbackend::Inn3rdoct_t:



Public Member Functions

- **Inn3rdoct_t** (unsigned int fs, unsigned int f, float bw, unsigned int niter)

Private Member Functions

- void **iterate_Inn** ()
- void **bandpass** ()

Private Attributes

- unsigned int **_fmin**
- unsigned int **_fmax**
- std::random_device **dev**
- std::default_random_engine **rng**
- std::uniform_real_distribution<float> **random**

Additional Inherited Members

4.49.1 Constructor & Destructor Documentation

```
4.49.1.1 lnn3rdoct_t() audiometerbackend::lnn3rdoct_t::lnn3rdoct_t (  
    unsigned int fs,  
    unsigned int f,  
    float bw,  
    unsigned int niter )
```

4.49.2 Member Function Documentation

```
4.49.2.1 iterate_lnn() void audiometerbackend::lnn3rdoct_t::iterate_lnn ( ) [private]
```

```
4.49.2.2 bandpass() void audiometerbackend::lnn3rdoct_t::bandpass ( ) [private]
```

4.49.3 Member Data Documentation

```
4.49.3.1 _fmin unsigned int audiometerbackend::lnn3rdoct_t::_fmin [private]
```

```
4.49.3.2 _fmax unsigned int audiometerbackend::lnn3rdoct_t::_fmax [private]
```

```
4.49.3.3 dev std::random_device audiometerbackend::lnn3rdoct_t::dev [private]
```

```
4.49.3.4 rng std::default_random_engine audiometerbackend::lnn3rdoct_t::rng [private]
```

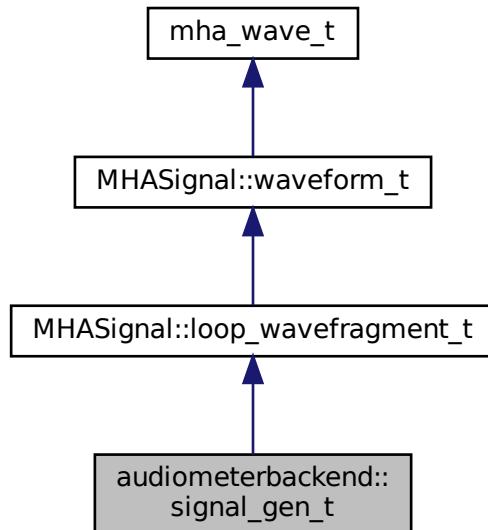
4.49.3.5 random std::uniform_real_distribution<float> audiometerbackend::lnn3rdoct←_t::random [private]

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

- audiometerbackend.cpp

4.50 audiometerbackend::signal_gen_t Class Reference

Inheritance diagram for audiometerbackend::signal_gen_t:



Public Member Functions

- **signal_gen_t** (int f, int fs, unsigned int sigtype)

Additional Inherited Members

4.50.1 Constructor & Destructor Documentation

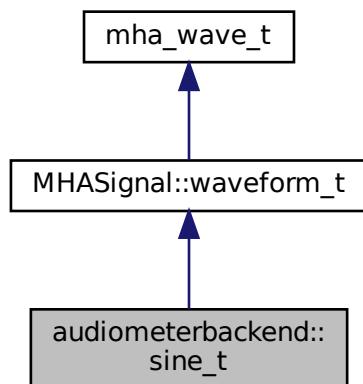
```
4.50.1.1 signal_gen_t() audiometerbackend::signal_gen_t::signal_gen_t (
    int f,
    int fs,
    unsigned int sigtype )
```

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

- **audiometerbackend.cpp**

4.51 audiometerbackend::sine_t Class Reference

Inheritance diagram for audiometerbackend::sine_t:



Public Member Functions

- **sine_t** (unsigned int fs, unsigned int f)

Additional Inherited Members

4.51.1 Constructor & Destructor Documentation

```
4.51.1.1 sine_t() sine_t::sine_t (
    unsigned int fs,
    unsigned int f )
```

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

- **audiometerbackend.cpp**

4.52 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

4.52.1 Detailed Description

A class to store frequency dependent data (e.g., HTL and UCL).

4.52.2 Member Function Documentation

```
4.52.2.1 get_frequencies() std::vector< mha_real_t > AuditoryProfile::fmap_t::get_frequencies ( ) const
```

Return configured frequencies.

4.52.2.2 get_values() std::vector< mha_real_t > AuditoryProfile::fmap_t::get_values () const

Return stored values corresponding to the frequencies.

4.52.2.3 isempty() bool AuditoryProfile::fmap_t::isempty () const [inline]

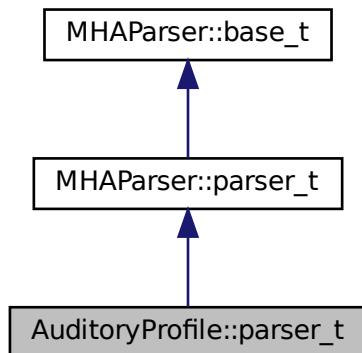
The documentation for this class was generated from the following files:

- [auditory_profile.h](#)
- [auditory_profile.cpp](#)

4.53 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

4.53.1 Detailed Description

Class to make the auditory profile accessible through the parser interface.

4.53.2 Constructor & Destructor Documentation

4.53.2.1 `parser_t()` `AuditoryProfile::parser_t::parser_t()`

4.53.3 Member Function Documentation

4.53.3.1 `get_current_profile()` `AuditoryProfile::profile_t AuditoryProfile::parser_t::get_current_profile()`

4.53.4 Member Data Documentation

4.53.4.1 `L` `AuditoryProfile::parser_t::ear_t AuditoryProfile::parser_t::L [private]`

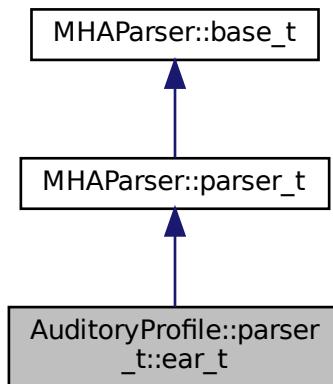
4.53.4.2 R `AuditoryProfile::parser_t::ear_t` `AuditoryProfile::parser_t::R` [private]

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

- `auditory_profile.h`
- `auditory_profile.cpp`

4.54 `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**4.54.1 Constructor & Destructor Documentation**

4.54.1.1 ear_t() `AuditoryProfile::parser_t::ear_t::ear_t ()`

4.54.2 Member Function Documentation

4.54.2.1 get_ear() `AuditoryProfile::profile_t::ear_t AuditoryProfile::parser_t::ear_t::get_ear () const`

4.54.3 Member Data Documentation

4.54.3.1 HTL `AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::HTL [private]`

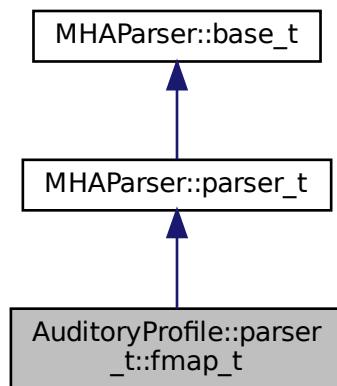
4.54.3.2 UCL `AuditoryProfile::parser_t::fmap_t AuditoryProfile::parser_t::ear_t::UCL [private]`

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

- `auditory_profile.h`
- `auditory_profile.cpp`

4.55 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

4.55.1 Constructor & Destructor Documentation

4.55.1.1 fmap_t() `AuditoryProfile::parser_t::fmap_t::fmap_t (`
 `const std::string & name,`
 `const std::string & help)`

4.55.2 Member Function Documentation

4.55.2.1 get_fmap() `AuditoryProfile::fmap_t AuditoryProfile::parser_t::fmap_t::get_fmap () const`

4.55.2.2 validate() `void AuditoryProfile::parser_t::fmap_t::validate () [private]`

4.55.3 Member Data Documentation

4.55.3.1 patchbay `MHAEVENTS::patchbay_t< AuditoryProfile::parser_t::fmap_t>` `AuditoryProfile::parser_t::fmap_t::patchbay` [private]

4.55.3.2 f `MHAPARSER::vfloat_t` `AuditoryProfile::parser_t::fmap_t::f` [private]

4.55.3.3 value `MHAPARSER::vfloat_t` `AuditoryProfile::parser_t::fmap_t::value` [private]

4.55.3.4 name_ `std::string` `AuditoryProfile::parser_t::fmap_t::name_` [private]

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

- `auditory_profile.h`
- `auditory_profile.cpp`

4.56 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.

4.56.1 Detailed Description

The Auditory Profile class.

See definition of auditory profile

Currently only the audiogram data is stored.

4.56.2 Member Function Documentation

Return ear information of channel number.

4.56.3 Member Data Documentation

4.56.3.1 L AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::L

Left ear data.

4.56.3.2 R AuditoryProfile::profile_t::ear_t AuditoryProfile::profile_t::R

Right ear data.

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

- **auditory_profile.h**

4.57 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**

4.57.1 Detailed Description

Class for ear-dependent parameters, e.g., audiograms or unilateral loudness scaling.

4.57.2 Member Function Documentation

4.57.2.1 **convert_empty2normal()** void AuditoryProfile::profile_t::ear_t::convert← _empty2normal ()

4.57.3 Member Data Documentation

4.57.3.1 HTL `AuditoryProfile::fmap_t` `AuditoryProfile::profile_t::ear_t::HTL`

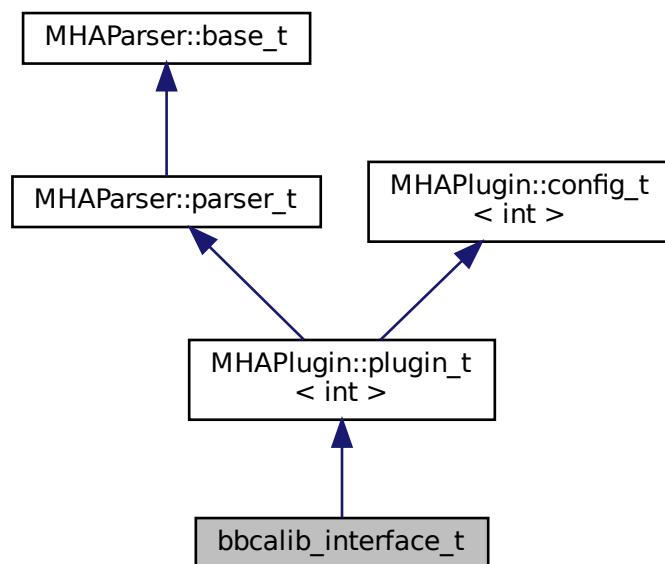
4.57.3.2 UCL `AuditoryProfile::fmap_t` `AuditoryProfile::profile_t::ear_t::UCL`

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

- `auditory_profile.h`
- `auditory_profile.cpp`

4.58 `bbcalib_interface_t` Class Reference

Inheritance diagram for `bbcalib_interface_t`:



Public Member Functions

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

Private Attributes

- **calibrator_t calib_in**
- **calibrator_t calib_out**
- **MHAParser::mhapluginloader_t plugloader**

Additional Inherited Members

4.58.1 Constructor & Destructor Documentation

4.58.1.1 `bbcalib_interface_t()` `bbcalib_interface_t::bbcalib_interface_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

4.58.1.2 `~bbcalib_interface_t()` `bbcalib_interface_t::~bbcalib_interface_t ()`

4.58.2 Member Function Documentation

4.58.2.1 `process()` `mha_wave_t * bbcalib_interface_t::process (`
 `mha_wave_t * s)`

4.58.2.2 `prepare()` `void bbcalib_interface_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPlugin::plugin_t< int >** (p. 1148).

4.58.2.3 `release()` `void bbcalib_interface_t::release () [virtual]`

Reimplemented from **MHAPlugIn::plugin_t< int >** (p. 1149).

4.58.3 Member Data Documentation

4.58.3.1 `calib_in` `calibrator_t bbcalib_interface_t::calib_in [private]`

4.58.3.2 `calib_out` `calibrator_t bbcalib_interface_t::calib_out [private]`

4.58.3.3 `plugloader` `MHAParser::mhapluginloader_t bbcalib_interface_t::plugloader [private]`

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

- `transducers.cpp`

4.59 `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 `firfir2ffflen` (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**

4.59.1 Constructor & Destructor Documentation

4.59.1.1 calibrator_runtime_layer_t() `calibrator_runtime_layer_t::calibrator_runtime_layer_t (`
`bool is_input,`
`const mhaconfig_t & tf,`
`calibrator_variables_t & vars)`

4.59.2 Member Function Documentation

4.59.2.1 process() `mha_real_t calibrator_runtime_layer_t::process (`
`mha_wave_t ** s)`

4.59.2.2 firfirlen() `unsigned int calibrator_runtime_layer_t::firfirlen (`
`const std::vector< std::vector< float > > & fir) [static], [private]`

4.59.2.3 firfir2ffflen() `unsigned int calibrator_runtime_layer_t::firfir2ffflen (`
`unsigned int fragsize,`
`const std::vector< std::vector< float > > & fir) [static], [private]`

4.59.3 Member Data Documentation

4.59.3.1 fir `MHAFilter::fftfilter_t` `calibrator_runtime_layer_t::fir` [private]

4.59.3.2 quant `MHASignal::quantizer_t` `calibrator_runtime_layer_t::quant` [private]

4.59.3.3 gain `MHASignal::waveform_t` `calibrator_runtime_layer_t::gain` [private]

4.59.3.4 softclip `softclipper_t` `calibrator_runtime_layer_t::softclip` [private]

4.59.3.5 b_is_input `bool` `calibrator_runtime_layer_t::b_is_input` [private]

4.59.3.6 b_use_fir `bool` `calibrator_runtime_layer_t::b_use_fir` [private]

4.59.3.7 b_use_clipping `bool` `calibrator_runtime_layer_t::b_use_clipping` [private]

4.59.3.8 speechnoise `MHASignal::loop_wavefragment_t` `calibrator_runtime_layer_t::speechnoise` [private]

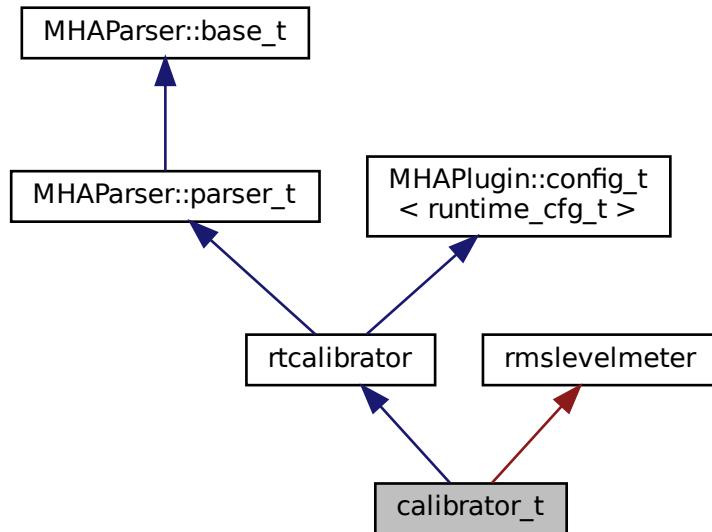
```
4.59.3.9 pmode MHASignal::loop_wavefragment_t::playback_mode_t calibrator_<-
runtime_layer_t::pmode [private]
```

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

- `transducers.cpp`

4.60 calibrator_t Class Reference

Inheritance diagram for calibrator_t:



Public Member Functions

- `calibrator_t (algo_comm_t, bool is_input)`
- `void prepare (mhaconfig_t &f)`
- `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**4.60.1 Constructor & Destructor Documentation**

4.60.1.1 calibrator_t() `calibrator_t::calibrator_t (algo_comm_t iac, bool is_input)`

4.60.2 Member Function Documentation

4.60.2.1 prepare() `void calibrator_t::prepare (mhaconfig_t & tf) [inline], [virtual]`

Implements **MHAPlugin::plugin_t< runtime_cfg_t >** (p. [1148](#)).

4.60.2.2 release() `void calibrator_t::release () [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. [1149](#)).

4.60.2.3 process() `mha_wave_t * calibrator_t::process (mha_wave_t * s)`

4.60.2.4 update() void calibrator_t::update () [private]

4.60.2.5 update_tau_level() void calibrator_t::update_tau_level () [private]

4.60.2.6 read_levels() void calibrator_t::read_levels () [private]

4.60.3 Member Data Documentation

4.60.3.1 b_is_input bool calibrator_t::b_is_input [private]

4.60.3.2 patchbay MHAEvents::patchbay_t< calibrator_t> calibrator_t::patchbay [private]

4.60.3.3 vars calibrator_variables_t calibrator_t::vars [private]

4.60.3.4 prepared bool calibrator_t::prepared [private]

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

- **transducers.cpp**

4.61 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`

4.61.1 Constructor & Destructor Documentation

```
4.61.1.1 calibrator_variables_t() calibrator_variables_t::calibrator_variables_t ( 
    bool is_input,
    MHAParser::parser_t & parent )
```

4.61.2 Member Data Documentation

```
4.61.2.1 peaklevel MHAParser::vfloat_t calibrator_variables_t::peaklevel
```

```
4.61.2.2 fir MHAParser::mfloat_t calibrator_variables_t::fir
```

4.61.2.3 nbits `MHAParser::int_t calibrator_variables_t::nbits`

4.61.2.4 tau_level `MHAParser::float_t calibrator_variables_t::tau_level`

4.61.2.5 spnoise_mode `MHAParser::kw_t calibrator_variables_t::spnoise_mode`

4.61.2.6 spnoise_channels `MHAParser::vint_t calibrator_variables_t::spnoise_<→channels`

4.61.2.7 spnoise_level `MHAParser::float_t calibrator_variables_t::spnoise_level`

4.61.2.8 rmslevel `MHAParser::vfloat_mon_t calibrator_variables_t::rmslevel`

4.61.2.9 spnoise_parser `MHAParser::parser_t calibrator_variables_t::spnoise_<→parser`

4.61.2.10 srate `MHAParser::float_mon_t calibrator_variables_t::srate`

4.61.2.11 fragsize `MHAParser::int_mon_t calibrator_variables_t::fragsize`

4.61.2.12 num_channels `MHAParser::int_mon_t calibrator_variables_t::num_channels`

4.61.2.13 config_parser `MHAParser::parser_t calibrator_variables_t::config_parser`

4.61.2.14 softclip `softclipper_variables_t calibrator_variables_t::softclip`

4.61.2.15 do_clipping `MHAParser::bool_t calibrator_variables_t::do_clipping`

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

- `transducers.cpp`

4.62 cfg_t Class Reference

Public Member Functions

- `cfg_t` (unsigned int ichannel, unsigned int numchannels, const `mha_complex_t` &iscale)
- `cfg_t` (unsigned int, unsigned int)
- `cfg_t` (`mhaconfig_t` chcfg, `mha_real_t` newlev, bool replace, `mha_real_t` len, int seed)
- `void process` (`mha_wave_t` *)
- `void process` (`mha_spec_t` *)
- `cfg_t` (`mha_real_t` tau_attack, `mha_real_t` tau_decay, unsigned int nch, `mha_real_t` start_limit, `mha_real_t` slope_db, `mha_real_t` fs)

Public Attributes

- unsigned int `channel`
- `mha_complex_t` `scale`
- `mha_real_t` `start_lin`
- `mha_real_t` `alpha`
- `MHAFilter::o1flt_lowpass_t` `attack`
- `MHAFilter::o1flt_maxtrack_t` `decay`

Private Attributes

- `mha_real_t gain_wave_`
- `mha_real_t gain_spec_`
- `bool replace_`
- `bool use_frozen_`
- `MHASignal::waveform_t frozen_noise_`
- `unsigned int pos`
- `std::default_random_engine rng`
- `std::uniform_real_distribution< mha_real_t > rand_dist`

4.62.1 Constructor & Destructor Documentation

4.62.1.1 cfg_t() [1/4] `cfg_t::cfg_t (`
 `unsigned int ichannel,`
 `unsigned int numchannels,`
 `const mha_complex_t & iscale)`

4.62.1.2 cfg_t() [2/4] `cfg_t::cfg_t (`
 `unsigned int ichannel,`
 `unsigned int numchannels)`

4.62.1.3 cfg_t() [3/4] `cfg_t::cfg_t (`
 `mhaconfig_t chcfg,`
 `mha_real_t newlev,`
 `bool replace,`
 `mha_real_t len,`
 `int seed)`

4.62.1.4 cfg_t() [4/4] `cfg_t::cfg_t (`
 `mha_real_t tau_attack,`
 `mha_real_t tau_decay,`
 `unsigned int nch,`
 `mha_real_t start_limit,`
 `mha_real_t slope_db,`
 `mha_real_t fs)`

4.62.2 Member Function Documentation

4.62.2.1 process() [1/2] void cfg_t::process (

```
mha_wave_t * s ) [inline]
```

4.62.2.2 process() [2/2] void cfg_t::process (

```
mha_spec_t * s ) [inline]
```

4.62.3 Member Data Documentation

4.62.3.1 channel unsigned int cfg_t::channel

4.62.3.2 scale mha_complex_t cfg_t::scale

4.62.3.3 gain_wave_ mha_real_t cfg_t::gain_wave_ [private]

4.62.3.4 gain_spec_ mha_real_t cfg_t::gain_spec_ [private]

4.62.3.5 replace_ bool cfg_t::replace_ [private]

4.62.3.6 use_frozen_ bool cfg_t::use_frozen_ [private]

4.62.3.7 frozen_noise_ MHASignal::waveform_t cfg_t::frozen_noise_ [private]

4.62.3.8 pos unsigned int cfg_t::pos [private]

4.62.3.9 rng std::default_random_engine cfg_t::rng [private]

4.62.3.10 rand_dist std::uniform_real_distribution< mha_real_t> cfg_t::rand_dist [private]

4.62.3.11 start_lin mha_real_t cfg_t::start_lin

4.62.3.12 alpha mha_real_t cfg_t::alpha

4.62.3.13 attack MHAFilter::olflt_lowpass_t cfg_t::attack

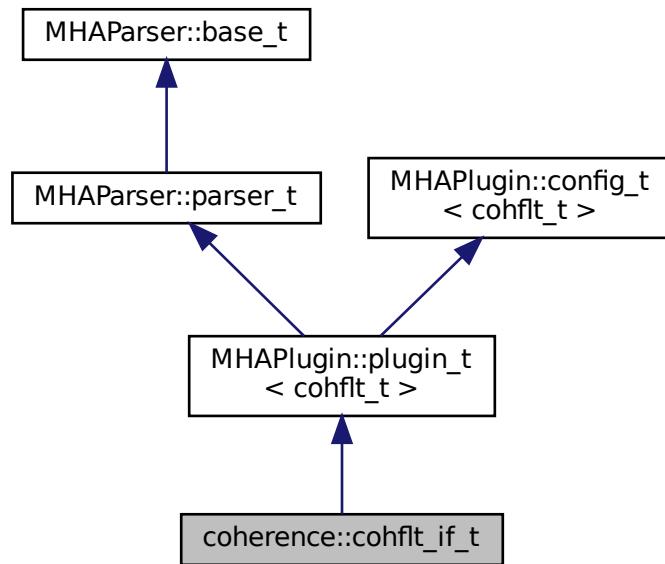
4.62.3.14 decay MHAFilter::olflt_maxtrack_t cfg_t::decay

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

- **complex_scale_channel.cpp**
- **example6.cpp**
- **noise.cpp**
- **softclip.cpp**

4.63 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

4.63.1 Constructor & Destructor Documentation

4.63.1.1 cohflt_if_t() coherence::cohflt_if_t::cohflt_if_t (

```
const algo_comm_t & ac,
const std::string & th,
const std::string & al )
```

4.63.2 Member Function Documentation

4.63.2.1 prepare() void coherence::cohflt_if_t::prepare (

```
mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cohflt_t >** (p. 1148).

4.63.2.2 release() void coherence::cohflt_if_t::release () [virtual]

Reimplemented from **MHAPlugin::plugin_t< cohflt_t >** (p. 1149).

4.63.2.3 process() mha_spec_t * coherence::cohflt_if_t::process (

```
mha_spec_t * s )
```

4.63.2.4 update() void coherence::cohflt_if_t::update () [private]

4.63.3 Member Data Documentation

4.63.3.1 patchbay `MHAEvents::patchbay_t< cohflt_if_t> coherence::cohflt_if_t::patchbay` [private]

4.63.3.2 vars `vars_t coherence::cohflt_if_t::vars` [private]

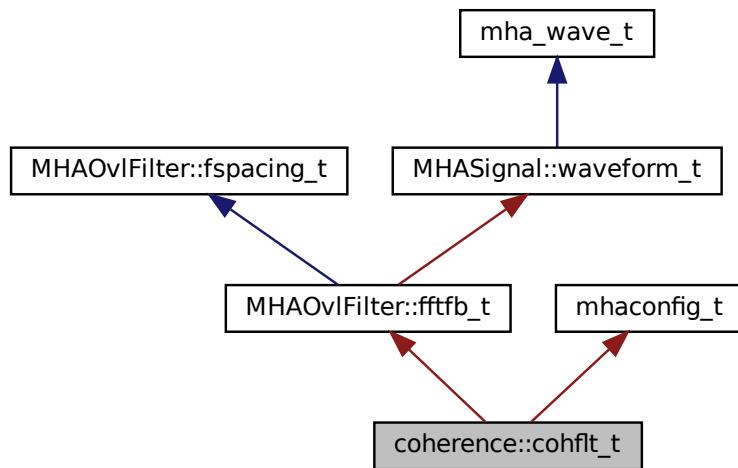
4.63.3.3 algo `const std::string coherence::cohflt_if_t::algo` [private]

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

- **coherence.cpp**

4.64 coherence::cohflt_t Class Reference

Inheritance diagram for coherence::cohflt_t:



Public Member Functions

- **cohflt_t** (`vars_t &v, const mhaconfig_t &icf, algo_comm_t iac, const std::string &name)`)
- **mha_spec_t * process** (`mha_spec_t *`)
- **void insert()**

Private Attributes

- unsigned int **nbands**
- bool **avg_ipd**
- **mha_complex_t** **cg**
- float **g**
- float **c_scale**
- float **c_min**
- **MHASignal::waveform_t** **alpha**
- float **limit**
- **MHAFilter::o1flt_lowpass_t** **lp1r**
- **MHAFilter::o1flt_lowpass_t** **lp1i**
- **MHA_AC::spectrum_t** **coh_c**
- **MHA_AC::waveform_t** **coh_rlp**
- **MHASignal::waveform_t** **gain**
- **MHASignal::delay_wave_t** **gain_delay**
- **MHASignal::spectrum_t** **s_out**
- bool **bInvert**
- **MHAFilter::o1flt_lowpass_t** **lp1ltg**
- bool **b_ltg**
- std::vector< float > **staticgain**

Additional Inherited Members

4.64.1 Constructor & Destructor Documentation

```
4.64.1.1 cohflt_t() coherence::cohflt_t::cohflt_t (
    vars_t & v,
    const mhaconfig_t & icf,
    algo_comm_t iac,
    const std::string & name )
```

4.64.2 Member Function Documentation

```
4.64.2.1 process() mha_spec_t * coherence::cohflt_t::process (
    mha_spec_t * s )
```

4.64.2.2 insert() void coherence::cohflt_t::insert ()

4.64.3 Member Data Documentation

4.64.3.1 nbands unsigned int coherence::cohflt_t::nbands [private]

4.64.3.2 avg_ipd bool coherence::cohflt_t::avg_ipd [private]

4.64.3.3 cg mha_complex_t coherence::cohflt_t::cg [private]

4.64.3.4 g float coherence::cohflt_t::g [private]

4.64.3.5 c_scale float coherence::cohflt_t::c_scale [private]

4.64.3.6 c_min float coherence::cohflt_t::c_min [private]

4.64.3.7 alpha MHASignal::waveform_t coherence::cohflt_t::alpha [private]

4.64.3.8 limit float coherence::cohflt_t::limit [private]

4.64.3.9 lp1r `MHAFilter::olflt_lowpass_t` coherence::cohflt_t::lp1r [private]

4.64.3.10 lp1i `MHAFilter::olflt_lowpass_t` coherence::cohflt_t::lp1i [private]

4.64.3.11 coh_c `MHA_AC::spectrum_t` coherence::cohflt_t::coh_c [private]

4.64.3.12 coh_rlp `MHA_AC::waveform_t` coherence::cohflt_t::coh_rlp [private]

4.64.3.13 gain `MHASignal::waveform_t` coherence::cohflt_t::gain [private]

4.64.3.14 gain_delay `MHASignal::delay_wave_t` coherence::cohflt_t::gain_delay [private]

4.64.3.15 s_out `MHASignal::spectrum_t` coherence::cohflt_t::s_out [private]

4.64.3.16 bInvert `bool` coherence::cohflt_t::bInvert [private]

4.64.3.17 lp1ltg `MHAFilter::olflt_lowpass_t` coherence::cohflt_t::lp1ltg [private]

4.64.3.18 b_ltg bool coherence::cohflt_t::b_ltg [private]

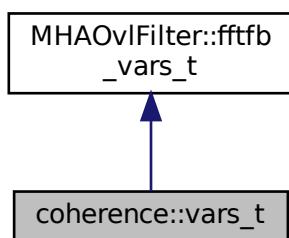
4.64.3.19 staticgain std::vector<float> coherence::cohflt_t::staticgain [private]

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

- **coherence.cpp**

4.65 coherence::vars_t Class Reference

Inheritance diagram for coherence::vars_t:



Public Member Functions

- **vars_t (MHParse::parser_t *)**

Public Attributes

- **MHParse::kw_t tau_unit**
- **MHParse::vfloat_t tau**
- **MHParse::vfloat_t alpha**
- **MHParse::float_t limit**
- **MHParse::vfloat_t mapping**
- **MHParse::kw_t average**
- **MHParse::bool_t invert**
- **MHParse::bool_t ltgcomp**
- **MHParse::vfloat_t ltgtau**
- **MHParse::vfloat_t staticgain**
- **MHParse::int_t delay**

4.65.1 Constructor & Destructor Documentation

4.65.1.1 vars_t() coherence::vars_t::vars_t (

```
    MHAParser::parser_t * p )
```

4.65.2 Member Data Documentation

4.65.2.1 tau_unit MHAParser::kw_t coherence::vars_t::tau_unit

4.65.2.2 tau MHAParser::vfloat_t coherence::vars_t::tau

4.65.2.3 alpha MHAParser::vfloat_t coherence::vars_t::alpha

4.65.2.4 limit MHAParser::float_t coherence::vars_t::limit

4.65.2.5 mapping MHAParser::vfloat_t coherence::vars_t::mapping

4.65.2.6 average MHAParser::kw_t coherence::vars_t::average

4.65.2.7 invert `MHAParser::bool_t coherence::vars_t::invert`

4.65.2.8 ltgcomp `MHAParser::bool_t coherence::vars_t::ltgcomp`

4.65.2.9 ltgtau `MHAParser::vfloat_t coherence::vars_t::ltgtau`

4.65.2.10 staticgain `MHAParser::vfloat_t coherence::vars_t::staticgain`

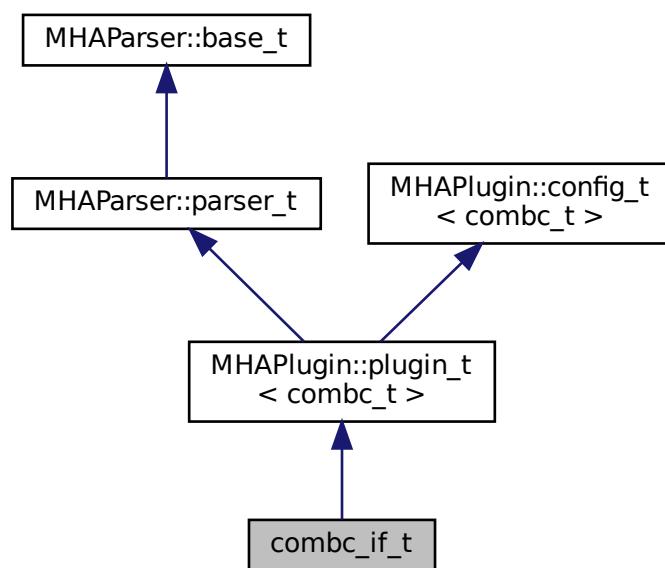
4.65.2.11 delay `MHAParser::int_t coherence::vars_t::delay`

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

- `coherence.cpp`

4.66 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

4.66.1 Constructor & Destructor Documentation

```
4.66.1.1 combc_if_t() combc_if_t::combc_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.66.2 Member Function Documentation

```
4.66.2.1 prepare() void combc_if_t::prepare (
    mhaconfig_t & chcfg ) [virtual]
```

Implements **MHAPlugin::plugin_t<combc_t>** (p. 1148).

```
4.66.2.2 process() [1/2] mha_wave_t * combc_if_t::process (
    mha_wave_t * s )
```

4.66.2.3 process() [2/2] `mha_spec_t * combc_if_t::process (mha_spec_t * s)`

4.66.3 Member Data Documentation

4.66.3.1 outchannels `MHAParser::int_t combc_if_t::outchannels [private]`

4.66.3.2 interleaved `MHAParser::bool_t combc_if_t::interleaved [private]`

4.66.3.3 channel_gain_name `MHAParser::string_t combc_if_t::channel_gain_name [private]`

4.66.3.4 element_gain_name `MHAParser::string_t combc_if_t::element_gain_name [private]`

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

- `combinechannels.cpp`

4.67 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_**

4.67.1 Constructor & Destructor Documentation

```
4.67.1.1 combc_t() 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 )
```

4.67.2 Member Function Documentation

```
4.67.2.1 process() [1/2] mha_wave_t * combc_t::process (
    mha_wave_t * s )
```

```
4.67.2.2 process() [2/2] mha_spec_t * combc_t::process (
    mha_spec_t * s )
```

4.67.3 Member Data Documentation

4.67.3.1 ac_ algo_comm_t combc_t::ac_ [private]

4.67.3.2 interleaved_ bool combc_t::interleaved_ [private]

4.67.3.3 nbands unsigned int combc_t::nbands [private]

4.67.3.4 w_out MHASignal::waveform_t combc_t::w_out [private]

4.67.3.5 s_out MHASignal::spectrum_t combc_t::s_out [private]

4.67.3.6 channel_gains_ std::vector< mha_real_t> combc_t::channel_gains_ [private]

4.67.3.7 element_gain_name_ std::string combc_t::element_gain_name_ [private]

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

- **combinechannels.cpp**

4.68 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.

4.68.1 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.

4.68.2 Member Data Documentation

4.68.2.1 data_type 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

4.68.2.2 num_entries comm_var_t::num_entries

Number of entries.

4.68.2.3 stride comm_var_t::stride

length of one row (C interpretation) or of one column (Fortran interpretation)

4.68.2.4 data comm_var_t::data

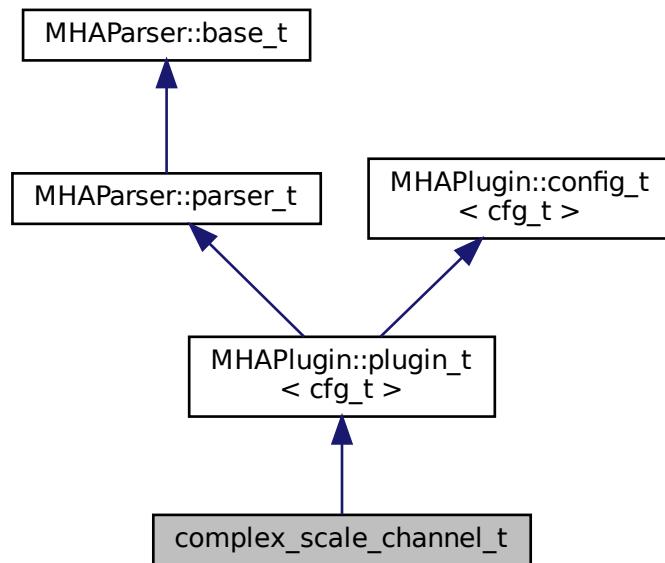
Pointer to variable data.

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

- **mha.hh**
- **mha_algo_comm.cpp**

4.69 complex_scale_channel_t Class Reference

Inheritance diagram for complex_scale_channel_t:



Public Member Functions

- `complex_scale_channel_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< complex_scale_channel_t > patchbay`
- `MHAParser::int_t scale_ch`
- `MHAParser::complex_t factor`

Additional Inherited Members

4.69.1 Constructor & Destructor Documentation

```
4.69.1.1 complex_scale_channel_t() complex_scale_channel_t::complex_scale_channel_t (const algo_comm_t & iac, const std::string & , const std::string & )
```

4.69.2 Member Function Documentation

```
4.69.2.1 process() mha_spec_t * complex_scale_channel_t::process (mha_spec_t * spec )
```

4.69.2.2 `prepare()` `void complex_scale_channel_t::prepare (mhaconfig_t & cfg) [virtual]`

Implements `MHAPlugIn::plugin_t< cfg_t >` (p. 1148).

4.69.2.3 `update_cfg()` `void complex_scale_channel_t::update_cfg () [private]`

4.69.3 Member Data Documentation

4.69.3.1 `patchbay` `MHAEvents::patchbay_t< complex_scale_channel_t > complex_scale←_channel_t::patchbay [private]`

4.69.3.2 `scale_ch` `MHAParser::int_t complex_scale_channel_t::scale_ch [private]`

4.69.3.3 `factor` `MHAParser::complex_t complex_scale_channel_t::factor [private]`

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

- `complex_scale_channel.cpp`

4.70 cpupload::cpupload_cfg_t Class Reference

Public Member Functions

- **`cpupload_cfg_t (mha_real_t factor_, size_t table_size_, bool use_sine_)`**
Ctor of the runtime configuration class.
- **`void process (unsigned fac_)`**
Process callback. Does not actually change signal.

Private Member Functions

- void **calc_sine** ()
- void **write_to_table** ()

Private Attributes

- float **phase** =0
Phase of the sine.
- volatile float **result** =0
Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.
- bool **use_sine** =false
Use sine or do table operation.
- float **factor** =0
cpu load factor. Values > 1 increase cpu load, values < 1 decrease it
- std::vector< float > **table**
Table with arbitrary values to operate on. Unused if use_sine=true.

4.70.1 Constructor & Destructor Documentation

```
4.70.1.1 cpupload_cfg_t() cpupload::cpupload_cfg_t::cpupload_cfg_t (
    mha_real_t factor_,
    size_t table_size_,
    bool use_sine_ )
```

Ctor of the runtime configuration class.

Parameters

<i>factor_</i>	cpu load factor. Values > 1 increase cpu load, values < 1 decrease it
<i>table_size_</i>	
<i>use_sine_</i>	

4.70.2 Member Function Documentation

4.70.2.1 process() void cpupload::cpupload_cfg_t::process (unsigned *fac_*)

Process callback. Does not actually change signal.

4.70.2.2 calc_sine() void cpupload::cpupload_cfg_t::calc_sine () [private]

4.70.2.3 write_to_table() void cpupload::cpupload_cfg_t::write_to_table () [private]

4.70.3 Member Data Documentation

4.70.3.1 phase float cpupload::cpupload_cfg_t::phase =0 [private]

Phase of the sine.

4.70.3.2 result volatile float cpupload::cpupload_cfg_t::result =0 [private]

Result of sin(phase). Volatile to prevent compiler from optimizing away the calculation.

4.70.3.3 use_sine bool cpupload::cpupload_cfg_t::use_sine =false [private]

Use sine or do table operation.

4.70.3.4 factor float cpupload::cpupload_cfg_t::factor =0 [private]

cpu load factor. Values > 1 increase cpu load, values < 1 decrease it

4.70.3.5 **table** std::vector<float> cpupload::cpupload_cfg_t::table [private]

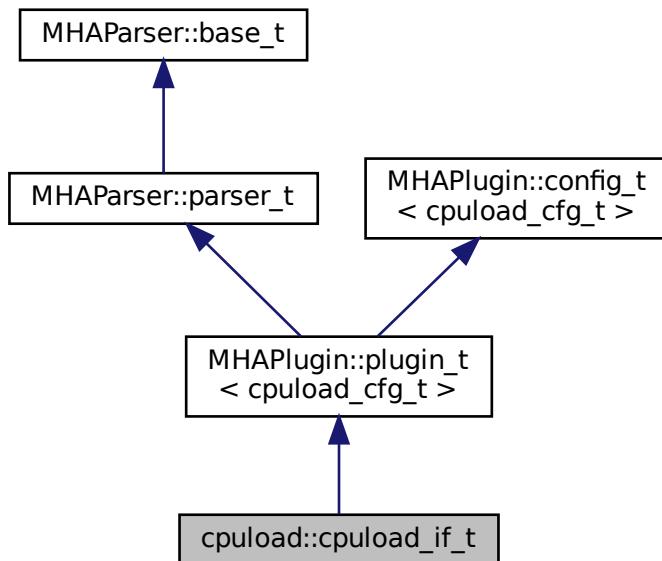
Table with arbitrary values to operate on. Unused if use_sine=true.

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

- **cpupload.cpp**

4.71 cpupload::cpupload_if_t Class Reference

Inheritance diagram for cpupload::cpupload_if_t:



Public Member Functions

- **cpupload_if_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 update ()**

Private Attributes

- `MHAParser::float_t factor`
- `MHAParser::int_t table_size`
- `MHAParser::bool_t use_sine`
- `MHAEvents::patchbay_t< cpupload_if_t > patchbay`

Additional Inherited Members

4.71.1 Constructor & Destructor Documentation

4.71.1.1 `cpupload_if_t()` `cpupload::cpupload_if_t::cpupload_if_t (`
 `algo_comm_t iac,`
 `const char * ,`
 `const char *)`

4.71.2 Member Function Documentation

4.71.2.1 `process() [1/2]` `mha_spec_t * cpupload::cpupload_if_t::process (`
 `mha_spec_t * s)`

4.71.2.2 `process() [2/2]` `mha_wave_t * cpupload::cpupload_if_t::process (`
 `mha_wave_t * s)`

4.71.2.3 `prepare()` `void cpupload::cpupload_if_t::prepare (`
 `mhaconfig_t & cf) [virtual]`

Implements `MHAParser::MHAParser()` (p. [1148](#)).

4.71.2.4 update() void cpupload::cpupload_if_t::update () [private]

4.71.3 Member Data Documentation

4.71.3.1 factor `MHAParser::float_t` cpupload::cpupload_if_t::factor [private]

4.71.3.2 table_size `MHAParser::int_t` cpupload::cpupload_if_t::table_size [private]

4.71.3.3 use_sine `MHAParser::bool_t` cpupload::cpupload_if_t::use_sine [private]

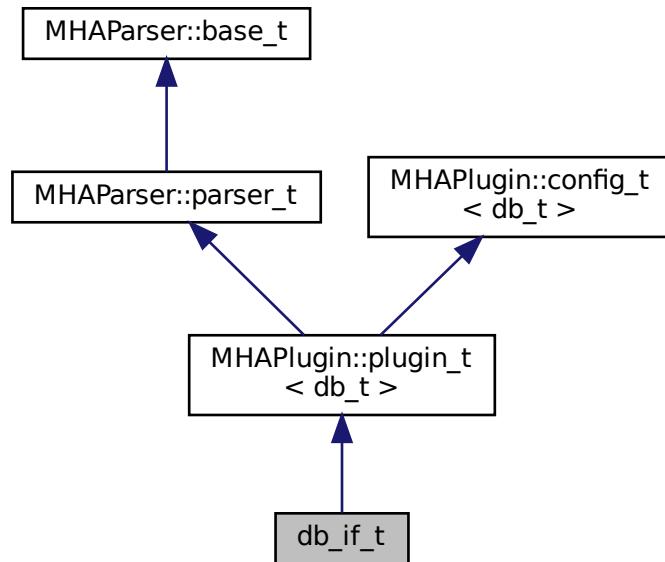
4.71.3.4 patchbay `MHAEvents::patchbay_t< cpupload_if_t>` cpupload::cpupload_if_t::patchbay [private]

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

- `cpupload.cpp`

4.72 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

4.72.1 Constructor & Destructor Documentation

4.72.1.1 db_if_t() db_if_t::db_if_t (algo_comm_t iac, std::string th, std::string al)

4.72.1.2 ~db_if_t() db_if_t::~db_if_t ()

4.72.2 Member Function Documentation

4.72.2.1 process() mha_wave_t * db_if_t::process (mha_wave_t * s)

4.72.2.2 prepare() void db_if_t::prepare (mhaconfig_t & conf) [virtual]

Implements **MHAPlugin::plugin_t< db_t >** (p. 1148).

4.72.2.3 release() void db_if_t::release () [virtual]

Reimplemented from **MHAPlugin::plugin_t< db_t >** (p. 1149).

4.72.3 Member Data Documentation

4.72.3.1 patchbay `MHAEVENTS::patchbay_t < db_if_t > db_if_t::patchbay [private]`

4.72.3.2 fragsize `MHAPARSER::int_t db_if_t::fragsize [private]`

4.72.3.3 plugloader `MHAPARSER::mhaplugloader_t db_if_t::plugloader [private]`

4.72.3.4 chain `std::string db_if_t::chain [private]`

4.72.3.5 algo `std::string db_if_t::algo [private]`

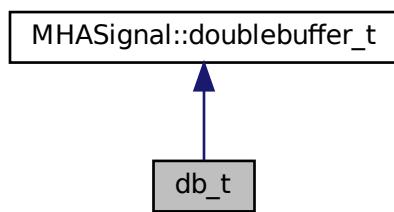
4.72.3.6 bypass `bool db_if_t::bypass [private]`

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

- `db.cpp`

4.73 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::mhaplugloader_t** &plug)
- **mha_wave_t * inner_process (mha_wave_t *)**

Private Attributes

- **MHAParser::mhaplugloader_t & plugloader**

Additional Inherited Members

4.73.1 Constructor & Destructor Documentation

```
4.73.1.1 db_t() db_t::db_t (
    unsigned int outer_fragsize,
    unsigned int inner_fragsize,
    unsigned int nch_in,
    unsigned int nch_out,
    MHAParser::mhaplugloader_t & plug )
```

4.73.2 Member Function Documentation

```
4.73.2.1 inner_process() mha_wave_t * db_t::inner_process (
    mha_wave_t * s ) [virtual]
```

Implements **MHASignal::doublebuffer_t** (p. 1205).

4.73.3 Member Data Documentation

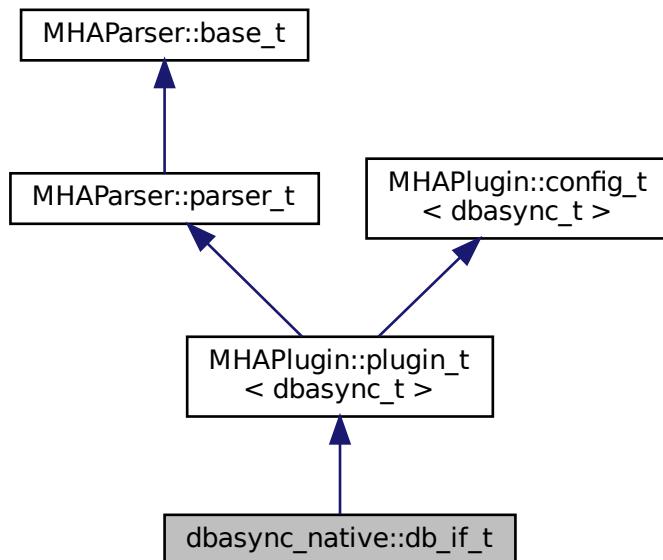
4.73.3.1 **plugloader** `MHAParser::mhaplugloader_t& db_t::plugloader [private]`

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

- `db.cpp`

4.74 `dbasync_native::db_if_t` Class Reference

Inheritance diagram for `dbasync_native::db_if_t`:



Public Member Functions

- `db_if_t (algo_comm_t, const std::string &, const std::string &)`
- `mha_wave_t * process (mha_wave_t *)`
- `void prepare (mhaconfig_t &)`
- `void release ()`
- `~db_if_t ()=default`

Private Attributes

- **MHAKernel::algo_comm_class_t sub_ac**
- **MHAParser::mhaplugloader_t plugloader**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t delay**
- **MHAParser::kw_t worker_thread_scheduler**
Scheduler used for worker thread.
- **MHAParser::int_t worker_thread_priority**
Priority of worker thread.
- **MHAParser::string_mon_t framework_thread_scheduler**
Scheduler of the signal processing thread.
- **MHAParser::int_mon_t framework_thread_priority**
Priority of signal processing thread.
- std::string **chain**
- std::string **algo**

Additional Inherited Members

4.74.1 Constructor & Destructor Documentation

```
4.74.1.1 db_if_t() db_if_t::db_if_t (
    algo_comm_t iac,
    const std::string & th,
    const std::string & al )
```

```
4.74.1.2 ~db_if_t() dbasync_native::db_if_t::~db_if_t ( ) [default]
```

4.74.2 Member Function Documentation

```
4.74.2.1 process() mha_wave_t * db_if_t::process (
    mha_wave_t * s )
```

4.74.2.2 `prepare()` `void db_if_t::prepare (mhaconfig_t & conf) [virtual]`

Implements `MHAPlugIn::plugin_t< dbasync_t >` (p. 1148).

4.74.2.3 `release()` `void db_if_t::release () [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< dbasync_t >` (p. 1149).

4.74.3 Member Data Documentation

4.74.3.1 `sub_ac` `MHAKernel::algo_comm_class_t dbasync_native::db_if_t::sub_ac [private]`

4.74.3.2 `plugloader` `MHAParser::mhapluginloader_t dbasync_native::db_if_t::plugloader [private]`

4.74.3.3 `fragsize` `MHAParser::int_t dbasync_native::db_if_t::fragsize [private]`

4.74.3.4 `delay` `MHAParser::int_t dbasync_native::db_if_t::delay [private]`

4.74.3.5 `worker_thread_scheduler` `MHAParser::kw_t dbasync_native::db_if_t::worker->_thread_scheduler [private]`

Scheduler used for worker thread.

4.74.3.6 worker_thread_priority `MHAParser::int_t dbasync_native::db_if_t::worker_thread_priority [private]`

Priority of worker thread.

4.74.3.7 framework_thread_scheduler `MHAParser::string_mon_t dbasync_native::db_if_t::framework_thread_scheduler [private]`

Scheduler of the signal processing thread.

4.74.3.8 framework_thread_priority `MHAParser::int_mon_t dbasync_native::db_if_t::framework_thread_priority [private]`

Priority of signal processing thread.

4.74.3.9 chain `std::string dbasync_native::db_if_t::chain [private]`

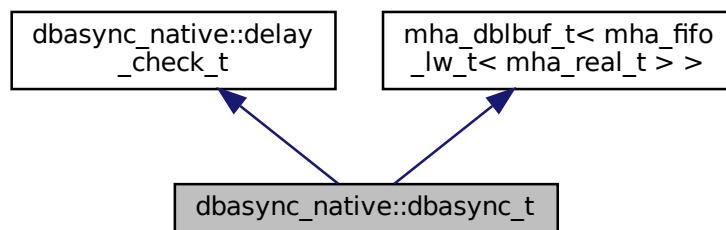
4.74.3.10 algo `std::string dbasync_native::db_if_t::algo [private]`

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

- `dbasync.cpp`

4.75 dbasync_native::dbasync_t Class Reference

Inheritance diagram for dbasync_native::dbasync_t:



Public Member Functions

- **dbasync_t** (unsigned int nchannels_in, unsigned int nchannels_out, unsigned int outer_fragsize, unsigned int inner_fragsize, int **delay**, const std::string &thread_scheduler, int thread_priority, **MHAParser::mhapluginloader_t** &plug)
- **~dbasync_t ()**
- **mha_wave_t * outer_process (mha_wave_t *)**
- int **svc ()**

Private Attributes

- **MHAParser::mhapluginloader_t & plugloader**
- **MHASignal::waveform_t inner_input**
- **MHASignal::waveform_t outer_output**
- pthread_attr_t **attr**
- struct sched_param **priority**
- int **scheduler**
- pthread_t **thread**

Additional Inherited Members

4.75.1 Constructor & Destructor Documentation

4.75.1.1 dbasync_t() dbasync_native::dbasync_t::dbasync_t (

```
    unsigned int nchannels_in,
    unsigned int nchannels_out,
    unsigned int outer_fragsize,
    unsigned int inner_fragsize,
    int delay,
    const std::string & thread_scheduler,
    int thread_priority,
    MHAParser::mhapluginloader_t & plug )
```

4.75.1.2 ~dbasync_t() dbasync_native::dbasync_t::~dbasync_t ()

4.75.2 Member Function Documentation

4.75.2.1 outer_process() `mha_wave_t * dbasync_native::dbasync_t::outer_process (mha_wave_t * outer_input)`

4.75.2.2 svc() `int dbasync_native::dbasync_t::svc ()`

4.75.3 Member Data Documentation

4.75.3.1 plugloader `MHAParser::mhaplugloader_t& dbasync_native::dbasync_t::plugloader [private]`

4.75.3.2 inner_input `MHASignal::waveform_t dbasync_native::dbasync_t::inner_input [private]`

4.75.3.3 outer_output `MHASignal::waveform_t dbasync_native::dbasync_t::outer_output [private]`

4.75.3.4 attr `pthread_attr_t dbasync_native::dbasync_t::attr [private]`

4.75.3.5 priority `struct sched_param dbasync_native::dbasync_t::priority [private]`

4.75.3.6 scheduler `int dbasync_native::dbasync_t::scheduler [private]`

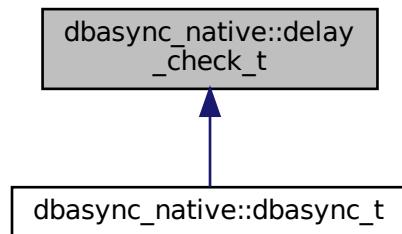
4.75.3.7 thread pthread_t dbasync_native::dbasync_t::thread [private]

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

- **dbasync.cpp**

4.76 dbasync_native::delay_check_t Class Reference

Inheritance diagram for dbasync_native::delay_check_t:

**Protected Member Functions**

- **delay_check_t** (int delay, unsigned inner_fragsize, unsigned outer_fragsize)

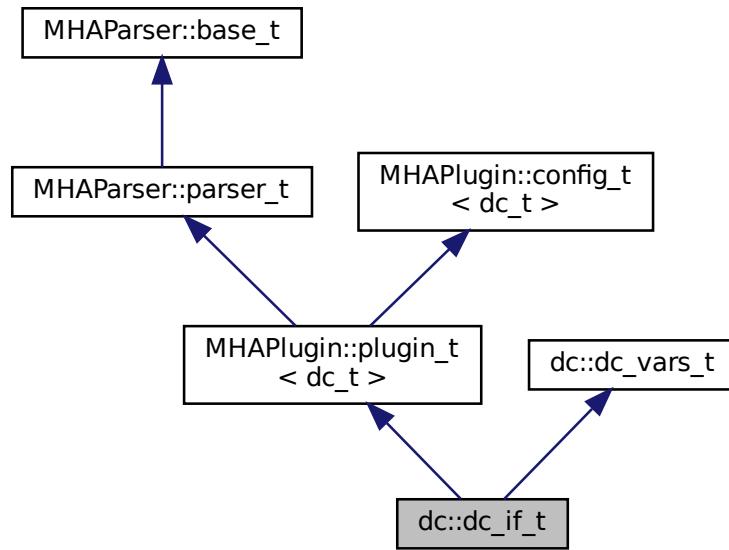
4.76.1 Constructor & Destructor Documentation**4.76.1.1 delay_check_t()** dbasync_native::delay_check_t::delay_check_t (
 int *delay*,
 unsigned *inner_fragsize*,
 unsigned *outer_fragsize*) [protected]

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

- **dbasync.cpp**

4.77 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`
- `MHAEvents::patchbay_t< dc_if_t > patchbay`

Additional Inherited Members

4.77.1 Constructor & Destructor Documentation

4.77.1.1 `dc_if_t()` `dc_if_t::dc_if_t (`
 `const algo_comm_t & ac_,`
 `const std::string & th_,`
 `const std::string & al_)`

4.77.2 Member Function Documentation

4.77.2.1 `prepare()` `void dc_if_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements `MHAPlugin::plugin_t< dc_t >` (p. [1148](#)).

4.77.2.2 `process() [1/2]` `mha_wave_t * dc_if_t::process (`
 `mha_wave_t * s_in)`

4.77.2.3 `process() [2/2]` `mha_spec_t * dc_if_t::process (`
 `mha_spec_t * s_in)`

4.77.2.4 `update_monitors()` `void dc_if_t::update_monitors () [private]`

Called from within the processing routines: updates the monitor variables.

4.77.2.5 update() void dc_if_t::update () [private]

Called by MHA configuration change event mechanism: creates new runtime configuration.

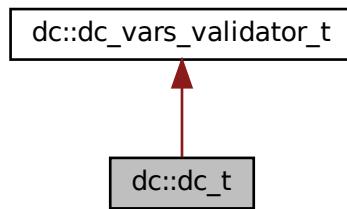
4.77.3 Member Data Documentation**4.77.3.1 algo** std::string dc::dc_if_t::algo [private]**4.77.3.2 patchbay** MHAEvents::patchbay_t< dc_if_t> dc::dc_if_t::patchbay [private]

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

- **dc.hh**
- **dc.cpp**

4.78 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, const std::string &algo, const std::vector< mha_real_t > &rmslevel_state={}, const std::vector< mha_real_t > &attack_state={}, const std::vector< mha_real_t > &decay_state={})`
- `mha_wave_t * process (mha_wave_t *)`
- `mha_spec_t * process (mha_spec_t *)`
- `void explicit_insert ()`
- `unsigned get_nbands () const`
`Number of frequency bands accessor.`
- `unsigned get_nch () const`
`Total number of channels accessor.`
- `const MHASignal::waveform_t & get_level_in_db () const`
- `const MHASignal::waveform_t & get_level_in_db_adjusted () const`
- `std::vector< mha_real_t > get_rmslevel_filter_state () const`
- `std::vector< mha_real_t > get_attack_filter_state () const`
- `std::vector< mha_real_t > get_decay_filter_state () const`

Private Attributes

- `std::vector< MHATableLookup::linear_table_t > gt`
- `std::vector< mha_real_t > offset`
- `MHAFilter::o1flt_lowpass_t rmslevel`
- `MHAFilter::o1flt_lowpass_t attack`
- `MHAFilter::o1flt_maxtrack_t decay`
- `bool bypass`
- `bool log_interp`
- `unsigned int naudiochannels`
- `unsigned int nbands`
- `unsigned int nch`
- `MHA_AC::waveform_t level_in_db`
- `MHA_AC::waveform_t level_in_db_adjusted`
- `unsigned int fftlen`

Additional Inherited Members

4.78.1 Constructor & Destructor Documentation

```
4.78.1.1 dc_t() 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,
    const std::string & algo,
    const std::vector< mha_real_t > & rmslevel_state = {},
    const std::vector< mha_real_t > & attack_state = {},
    const std::vector< mha_real_t > & decay_state = {} )
```

4.78.2 Member Function Documentation

4.78.2.1 process() [1/2] mha_wave_t * dc_t::process (mha_wave_t * s)

4.78.2.2 process() [2/2] mha_spec_t * dc_t::process (mha_spec_t * s)

4.78.2.3 explicit_insert() void dc_t::explicit_insert ()

4.78.2.4 get_nbands() unsigned dc::dc_t::get_nbands () const [inline]

Number of frequency bands accessor.

4.78.2.5 get_nch() unsigned dc::dc_t::get_nch () const [inline]

Total number of channels accessor.

4.78.2.6 `get_level_in_db()` const `MHASignal::waveform_t&` `dc::dc_t::get_level_in_db()` const [inline]

4.78.2.7 `get_level_in_db_adjusted()` const `MHASignal::waveform_t&` `dc::dc_t::get_level_in_db_adjusted()` const [inline]

4.78.2.8 `get_rmslevel_filter_state()` `std::vector< mha_real_t >` `dc::dc_t::get_rmslevel_filter_state()` const [inline]

4.78.2.9 `get_attack_filter_state()` `std::vector< mha_real_t >` `dc::dc_t::get_attack_filter_state()` const [inline]

4.78.2.10 `get_decay_filter_state()` `std::vector< mha_real_t >` `dc::dc_t::get_decay_filter_state()` const [inline]

4.78.3 Member Data Documentation

4.78.3.1 `gt` `std::vector< MHATableLookup::linear_table_t >` `dc::dc_t::gt` [private]

4.78.3.2 `offset` `std::vector< mha_real_t >` `dc::dc_t::offset` [private]

4.78.3.3 `rmslevel` `MHAFilter::olflt_lowpass_t` `dc::dc_t::rmslevel` [private]

4.78.3.4 attack `MHAFilter::olflt_lowpass_t` `dc::dc_t::attack` [private]

4.78.3.5 decay `MHAFilter::olflt_maxtrack_t` `dc::dc_t::decay` [private]

4.78.3.6 bypass `bool` `dc::dc_t::bypass` [private]

4.78.3.7 log_interp `bool` `dc::dc_t::log_interp` [private]

4.78.3.8 naudiochannels `unsigned int` `dc::dc_t::naudiochannels` [private]

4.78.3.9 nbands `unsigned int` `dc::dc_t::nbands` [private]

4.78.3.10 nch `unsigned int` `dc::dc_t::nch` [private]

4.78.3.11 level_in_db `MHA_AC::waveform_t` `dc::dc_t::level_in_db` [private]

4.78.3.12 level_in_db_adjusted `MHA_AC::waveform_t` `dc::dc_t::level_in_db_adjusted` [private]

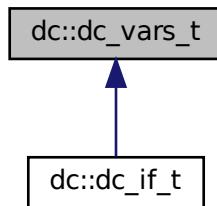
4.78.3.13 **fftlen** `unsigned int dc::dc_t::fftlen [private]`

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

- `dc.hh`
- `dc.cpp`

4.79 `dc::dc_vars_t` Class Reference

Inheritance diagram for `dc::dc_vars_t`:



Public Member Functions

- `dc_vars_t (MHParse::parser_t &)`

Public Attributes

- `MHParse::mfloat_t gtdata`
- `MHParse::vfloat_t gtmin`
- `MHParse::vfloat_t gtstep`
- `MHParse::vfloat_t taurmslevel`
- `MHParse::vfloat_t tauattack`
- `MHParse::vfloat_t taudecay`
- `MHParse::vfloat_t offset`
- `MHParse::string_t filterbank`
- `std::string cf_name`
- `std::string ef_name`
- `std::string bw_name`
- `MHParse::string_t chname`
- `MHParse::bool_t bypass`

- MHParse::bool_t log_interp
- MHParse::string_t clientid
- MHParse::string_t gainrule
- MHParse::string_t preset
- MHParse::int_mon_t modified
- MHParse::vfloat_mon_t input_level
- MHParse::vfloat_mon_t filtered_level
- MHParse::vfloat_mon_t center_frequencies
- MHParse::vfloat_mon_t edge_frequencies
- MHParse::vfloat_mon_t band_weights

4.79.1 Constructor & Destructor Documentation

4.79.1.1 dc_vars_t() dc_vars_t::dc_vars_t (
MHParse::parser_t & p) [explicit]

4.79.2 Member Data Documentation

4.79.2.1 gtdata MHParse::mfloat_t dc::dc_vars_t::gtdata

4.79.2.2 gtmin MHParse::vfloat_t dc::dc_vars_t::gtmin

4.79.2.3 gtstep MHParse::vfloat_t dc::dc_vars_t::gtstep

4.79.2.4 taurmslevel MHParse::vfloat_t dc::dc_vars_t::taurmslevel

4.79.2.5 tauattack `MHAParser::vfloat_t dc::dc_vars_t::tauattack`

4.79.2.6 taudecay `MHAParser::vfloat_t dc::dc_vars_t::taudecay`

4.79.2.7 offset `MHAParser::vfloat_t dc::dc_vars_t::offset`

4.79.2.8 filterbank `MHAParser::string_t dc::dc_vars_t::filterbank`

4.79.2.9 cf_name `std::string dc::dc_vars_t::cf_name`

4.79.2.10 ef_name `std::string dc::dc_vars_t::ef_name`

4.79.2.11 bw_name `std::string dc::dc_vars_t::bw_name`

4.79.2.12 chname `MHAParser::string_t dc::dc_vars_t::chname`

4.79.2.13 bypass `MHAParser::bool_t dc::dc_vars_t::bypass`

4.79.2.14 log_interp `MHAParser::bool_t dc::dc_vars_t::log_interp`

4.79.2.15 clientid `MHAParser::string_t dc::dc_vars_t::clientid`

4.79.2.16 gainrule `MHAParser::string_t dc::dc_vars_t::gainrule`

4.79.2.17 preset `MHAParser::string_t dc::dc_vars_t::preset`

4.79.2.18 modified `MHAParser::int_mon_t dc::dc_vars_t::modified`

4.79.2.19 input_level `MHAParser::vfloat_mon_t dc::dc_vars_t::input_level`

4.79.2.20 filtered_level `MHAParser::vfloat_mon_t dc::dc_vars_t::filtered_level`

4.79.2.21 center_frequencies `MHAParser::vfloat_mon_t dc::dc_vars_t::center_frequencies`

4.79.2.22 edge_frequencies `MHAParser::vfloat_mon_t dc::dc_vars_t::edge_frequencies`

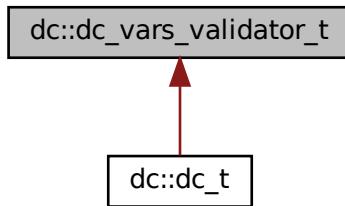
4.79.2.23 **band_weights** `MHAParser::vfloat_mon_t dc::dc_vars_t::band_weights`

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

- `dc.hh`
- `dc.cpp`

4.80 **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)`

4.80.1 Constructor & Destructor Documentation

4.80.1.1 `dc_vars_validator_t()` `dc_vars_validator_t::dc_vars_validator_t (`

```

dc_vars_t & v,
unsigned int s,
mha_domain_t domain )
  
```

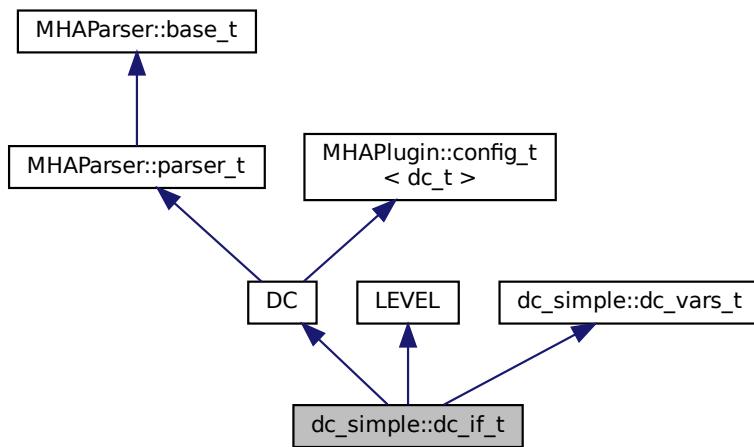
The documentation for this class was generated from the following files:

- `dc.hh`
- `dc.cpp`

4.81 dc_simple::dc_if_t Class Reference

interface class

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 &)`
- `void release ()`
- `mha_spec_t * process (mha_spec_t *s)`
- `mha_wave_t * process (mha_wave_t *s)`

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

4.81.1 Detailed Description

interface class

4.81.2 Constructor & Destructor Documentation

```
4.81.2.1 dc_if_t() dc_simple::dc_if_t::dc_if_t (
    const algo_comm_t & ac_,
    const std::string & th_,
    const std::string & al_ )
```

4.81.3 Member Function Documentation

```
4.81.3.1 prepare() void dc_simple::dc_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements `MHAParser::parser_t< dc_t >` (p. 1148).

4.81.3.2 release() void dc_simple::dc_if_t::release () [virtual]

Reimplemented from **MHAPlugin::plugin_t<dc_t>** (p. 1149).

4.81.3.3 process() [1/2] mha_spec_t * dc_simple::dc_if_t::process (mha_spec_t * s)

4.81.3.4 process() [2/2] mha_wave_t * dc_simple::dc_if_t::process (mha_wave_t * s)

4.81.3.5 update_dc() void dc_simple::dc_if_t::update_dc () [private]

4.81.3.6 update_level() void dc_simple::dc_if_t::update_level () [private]

4.81.3.7 has Been Modified() void dc_simple::dc_if_t::has_been_modified () [inline], [private]

4.81.3.8 read Modified() void dc_simple::dc_if_t::read_modified () [inline], [private]

4.81.3.9 update_level_mon() void dc_simple::dc_if_t::update_level_mon () [private]

4.81.3.10 update_gain_mon() void dc_simple::dc_if_t::update_gain_mon () [private]

4.81.4 Member Data Documentation

4.81.4.1 clientid `MHAParser::string_t` `dc_simple::dc_if_t::clientid` [private]

4.81.4.2 gainrule `MHAParser::string_t` `dc_simple::dc_if_t::gainrule` [private]

4.81.4.3 preset `MHAParser::string_t` `dc_simple::dc_if_t::preset` [private]

4.81.4.4 modified `MHAParser::int_mon_t` `dc_simple::dc_if_t::modified` [private]

4.81.4.5 mon_l `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::mon_l` [private]

4.81.4.6 mon_g `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::mon_g` [private]

4.81.4.7 filterbank `MHAParser::string_t` `dc_simple::dc_if_t::filterbank` [private]

4.81.4.8 center_frequencies `MHAParser::vfloat_mon_t` `dc_simple::dc_if_t::center_<= frequencies` [private]

4.81.4.9 edge_frequencies `MHAPARSER::VFLOAT_MON_T dc_simple::dc_if_t::edge_frequencies` [private]

4.81.4.10 patchbay `MHAEVENTS::PATCHBAY_T< dc_if_t> dc_simple::dc_if_t::patchbay` [private]

4.81.4.11 prepared `bool dc_simple::dc_if_t::prepared` [private]

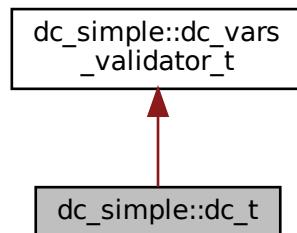
The documentation for this class was generated from the following files:

- `dc_simple.hh`
- `dc_simple.cpp`

4.82 dc_simple::dc_t Class Reference

Runtime config class for **dc_simple** (p. 86) plugin.

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_ratem, unsigned int nch, unsigned int fftlen_)
- **mha_spec_t * process** (**mha_spec_t** *s, **mha_wave_t** *level_db)
- **mha_wave_t * process** (**mha_wave_t** *s, **mha_wave_t** *level_db)

Public Attributes

- std::vector< float > **mon_I**
- 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

4.82.1 Detailed Description

Runtime config class for **dc_simple** (p. 86) plugin.

4.82.2 Constructor & Destructor Documentation

4.82.2.1 **dc_t()** `dc_simple::dc_t::dc_t (`

```
    const dc_vars_t & vars,
    mha_real_t filter_ratem,
    unsigned int nch,
    unsigned int fftlen_ )
```

4.82.3 Member Function Documentation

4.82.3.1 process() [1/2] `mha_spec_t * dc_simple::dc_t::process (`
`mha_spec_t * s,`
`mha_wave_t * level_db)`

4.82.3.2 process() [2/2] `mha_wave_t * dc_simple::dc_t::process (`
`mha_wave_t * s,`
`mha_wave_t * level_db)`

4.82.4 Member Data Documentation

4.82.4.1 expansion_threshold `std::vector< mha_real_t > dc_simple::dc_t::expansion_threshold` [private]

4.82.4.2 limiter_threshold `std::vector< mha_real_t > dc_simple::dc_t::limiter_threshold` [private]

4.82.4.3 compression `std::vector< line_t > dc_simple::dc_t::compression` [private]

4.82.4.4 expansion `std::vector< line_t > dc_simple::dc_t::expansion` [private]

4.82.4.5 limiter std::vector< **line_t**> dc_simple::dc_t::limiter [private]

4.82.4.6 maxgain std::vector< **mha_real_t**> dc_simple::dc_t::maxgain [private]

4.82.4.7 nbands unsigned int dc_simple::dc_t::nbands [private]

4.82.4.8 mon_l std::vector<float> dc_simple::dc_t::mon_l

4.82.4.9 mon_g std::vector<float> dc_simple::dc_t::mon_g

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

- **dc_simple.hh**
- **dc_simple.cpp**

4.83 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 m_)**
- **mha_real_t operator() (mha_real_t x)**

Private Attributes

- **mha_real_t m**
- **mha_real_t y0**

4.83.1 Constructor & Destructor Documentation

4.83.1.1 line_t() [1/2] dc_simple::dc_t::line_t::line_t (

```
mha_real_t x1,  
mha_real_t y1,  
mha_real_t x2,  
mha_real_t y2 )
```

4.83.1.2 line_t() [2/2] dc_simple::dc_t::line_t::line_t (

```
mha_real_t x1,  
mha_real_t y1,  
mha_real_t m_ )
```

4.83.2 Member Function Documentation

4.83.2.1 operator()() mha_real_t dc_simple::dc_t::line_t::operator() (

```
mha_real_t x ) [inline]
```

4.83.3 Member Data Documentation

4.83.3.1 m mha_real_t dc_simple::dc_t::line_t::m [private]

4.83.3.2 y0 mha_real_t dc_simple::dc_t::line_t::y0 [private]

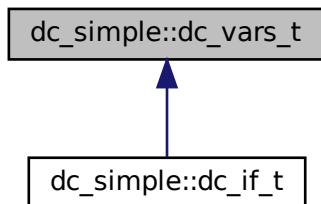
The documentation for this class was generated from the following files:

- dc_simple.hh
- dc_simple.cpp

4.84 dc_simple::dc_vars_t Class Reference

class for **dc_simple** (p. 86) plugin which registers variables to **MHAParser** (p. 122).

Inheritance diagram for dc_simple::dc_vars_t:



Public Member Functions

- **dc_vars_t (MHAParser::parser_t &p)**

Public Attributes

- **MHAParser::vfloat_t g50**
- **MHAParser::vfloat_t g80**
- **MHAParser::vfloat_t maxgain**
- **MHAParser::vfloat_t expansion_threshold**
- **MHAParser::vfloat_t expansion_slope**
- **MHAParser::vfloat_t limiter_threshold**
- **MHAParser::vfloat_t tauattack**
- **MHAParser::vfloat_t taudecay**
- **MHAParser::bool_t bypass**

4.84.1 Detailed Description

class for **dc_simple** (p. 86) plugin which registers variables to **MHAParser** (p. 122).

4.84.2 Constructor & Destructor Documentation

4.84.2.1 dc_vars_t() `dc_simple::dc_vars_t::dc_vars_t (`
`MHAParser::parser_t & p)`

4.84.3 Member Data Documentation

4.84.3.1 g50 `MHAParser::vfloat_t dc_simple::dc_vars_t::g50`

4.84.3.2 g80 `MHAParser::vfloat_t dc_simple::dc_vars_t::g80`

4.84.3.3 maxgain `MHAParser::vfloat_t dc_simple::dc_vars_t::maxgain`

4.84.3.4 expansion_threshold `MHAParser::vfloat_t dc_simple::dc_vars_t::expansion←_threshold`

4.84.3.5 expansion_slope `MHAParser::vfloat_t dc_simple::dc_vars_t::expansion←slope`

4.84.3.6 limiter_threshold `MHAParser::vfloat_t dc_simple::dc_vars_t::limiter←threshold`

4.84.3.7 tauattack `MHAParser::vfloat_t dc_simple::dc_vars_t::tauattack`

4.84.3.8 **taudecay** `MHAParser::vffloat_t dc_simple::dc_vars_t::taudecay`

4.84.3.9 **bypass** `MHAParser::bool_t dc_simple::dc_vars_t::bypass`

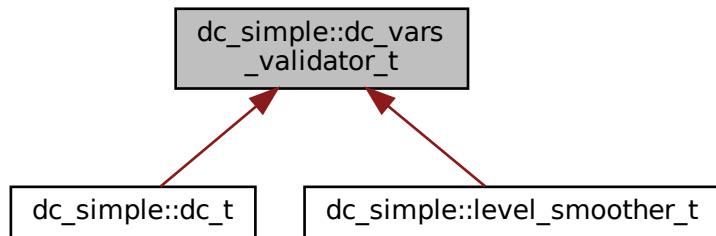
The documentation for this class was generated from the following files:

- `dc_simple.hh`
- `dc_simple.cpp`

4.85 `dc_simple::dc_vars_validator_t` Class Reference

Helper class to check sizes of configuration variable vectors.

Inheritance diagram for `dc_simple::dc_vars_validator_t`:



Public Member Functions

- `dc_vars_validator_t (const dc_vars_t &v, unsigned int s)`
Checks that all vectors in v have size s or size 1.

4.85.1 Detailed Description

Helper class to check sizes of configuration variable vectors.

4.85.2 Constructor & Destructor Documentation

```
4.85.2.1 dc_vars_validator_t() dc_simple::dc_vars_validator_t::dc_vars_validator_t
(
    const dc_vars_t & v,
    unsigned int s )
```

Checks that all vectors in `v` have size `s` or size 1.

Parameters

<i>v</i>	Aggregation of vectors to check the sizes of.
<i>s</i>	Desired vector size for all vectors

Exceptions

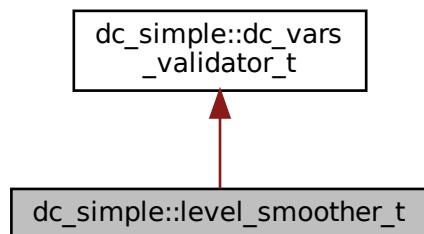
MHA_Error (p. 763)	if <i>s</i> == 0.
MHA_Error (p. 763)	if the size of any vector in <i>v</i> is neither <i>s</i> nor 1.

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

- **dc_simple.hh**
- **dc_simple.cpp**

4.86 dc_simple::level_smooother_t Class Reference

Inheritance diagram for dc_simple::level_smooother_t:



Public Member Functions

- **level_smooother_t** (const **dc_vars_t** &vars, **mha_real_t** filter_rate, **mhaconfig_t** buscfg)
- **mha_wave_t * process** (**mha_spec_t** *s)
- **mha_wave_t * process** (**mha_wave_t** *s)

Private Attributes

- **MHAFilter::o1flt_lowpass_t attack**
- **MHAFilter::o1flt_maxtrack_t decay**
- unsigned int **nbands**
- unsigned int **ffflen**
- **MHASignal::waveform_t level_wave**
- **MHASignal::waveform_t level_spec**

Additional Inherited Members

4.86.1 Constructor & Destructor Documentation

4.86.1.1 level_smoothen_t() dc_simple::level_smoothen_t::level_smoothen_t (const **dc_vars_t** & vars, **mha_real_t** filter_rate, **mhaconfig_t** buscfg)

4.86.2 Member Function Documentation

4.86.2.1 process() [1/2] **mha_wave_t** * dc_simple::level_smoothen_t::process (**mha_spec_t** * s)

4.86.2.2 process() [2/2] **mha_wave_t** * dc_simple::level_smoothen_t::process (**mha_wave_t** * s)

4.86.3 Member Data Documentation

4.86.3.1 attack `MHAFilter::olflt_lowpass_t` `dc_simple::level_smoothen_t::attack`
[private]

4.86.3.2 decay `MHAFilter::olflt_maxtrack_t` `dc_simple::level_smoothen_t::decay`
[private]

4.86.3.3 nbands `unsigned int` `dc_simple::level_smoothen_t::nbands` [private]

4.86.3.4 fftlen `unsigned int` `dc_simple::level_smoothen_t::fftlens` [private]

4.86.3.5 level_wave `MHASignal::waveform_t` `dc_simple::level_smoothen_t::level_wave`
[private]

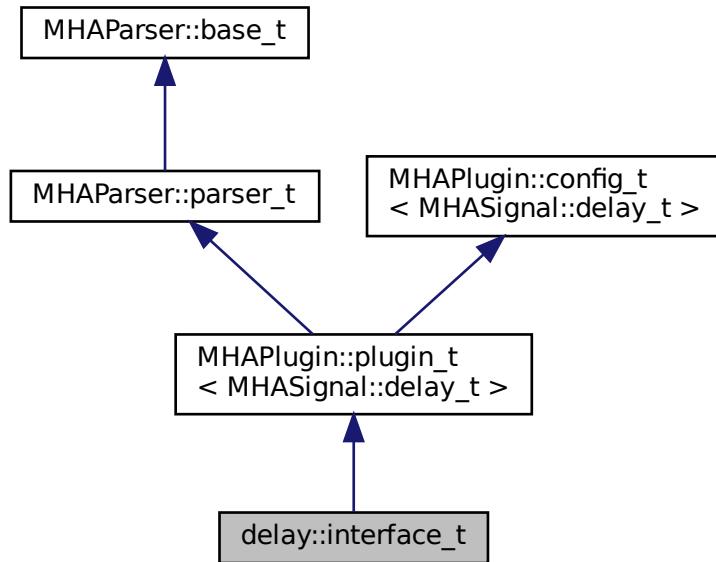
4.86.3.6 level_spec `MHASignal::waveform_t` `dc_simple::level_smoothen_t::level_spec`
[private]

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

- `dc_simple.hh`
- `dc_simple.cpp`

4.87 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

4.87.1 Constructor & Destructor Documentation

```
4.87.1.1 interface_t() delay::interface_t::interface_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.87.2 Member Function Documentation

```
4.87.2.1 prepare() void delay::interface_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAParser::plugin_t< MHASignal::delay_t >** (p. 1148).

```
4.87.2.2 process() mha_wave_t * delay::interface_t::process (
    mha_wave_t * s )
```

```
4.87.2.3 update() void delay::interface_t::update ( ) [private]
```

4.87.3 Member Data Documentation

```
4.87.3.1 delays MHParse::vint_t delay::interface_t::delays [private]
```

4.87.3.2 patchbay `MHAEVENTS::patchbay_t< interface_t>` `delay::interface_t::patchbay`
[private]

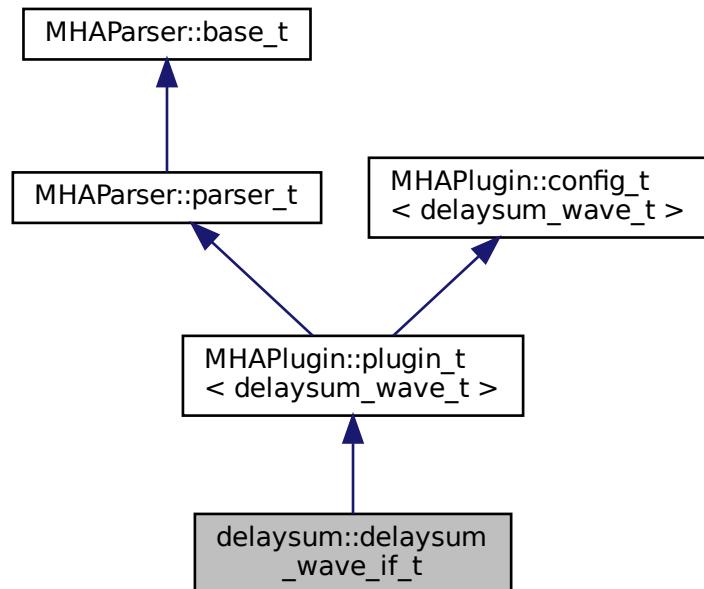
The documentation for this class was generated from the following files:

- `delay.hh`
- `delay.cpp`

4.88 delaysum::delaysum_wave_if_t Class Reference

Interface class for the delaysum plugin.

Inheritance diagram for delaysum::delaysum_wave_if_t:



Public Member Functions

- `delaysum_wave_if_t (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_wave_if_t > patchbay**
The patchbay to react to config changes.

Additional Inherited Members

4.88.1 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.

4.88.2 Constructor & Destructor Documentation

4.88.2.1 **delaysum_wave_if_t()** delaysum::delaysum_wave_if_t::delaysum_wave_if_t (

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

4.88.3 Member Function Documentation

4.88.3.1 **process()** mha_wave_t * delaysum::delaysum_wave_if_t::process (

```
mha_wave_t * wave )
```

4.88.3.2 `prepare()` `void delaysum::delaysum_wave_if_t::prepare (mhaconfig_t & tfcfg) [virtual]`

Implements **MHAPlugIn::plugin_t< delaysum_wave_t >** (p. 1148).

4.88.3.3 `release()` `void delaysum::delaysum_wave_if_t::release () [virtual]`

Reimplemented from **MHAPlugIn::plugin_t< delaysum_wave_t >** (p. 1149).

4.88.3.4 `update_cfg()` `void delaysum::delaysum_wave_if_t::update_cfg () [private]`

4.88.4 Member Data Documentation

4.88.4.1 `weights` `MHAParser::vfloat_t delaysum::delaysum_wave_if_t::weights [private]`

Linear weights to be multiplied with the audio signal, one factor for each channel.

Order is [chan0, chan1, ...]

4.88.4.2 `delay` `MHAParser::vint_t delaysum::delaysum_wave_if_t::delay [private]`

vector of channel-specific delays, in samples.

4.88.4.3 `patchbay` `MHAEvents::patchbay_t< delaysum_wave_if_t > delaysum::delaysum<-wave_if_t::patchbay [private]`

The patchbay to react to config changes.

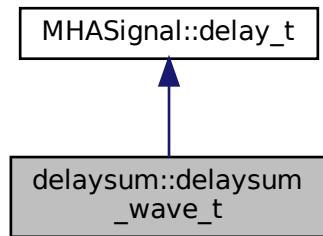
The documentation for this class was generated from the following file:

- **delaysum_wave.cpp**

4.89 delaysum::delaysum_wave_t Class Reference

Runtime configuration of the delaysum_wave plugin.

Inheritance diagram for delaysum::delaysum_wave_t:



Public Member Functions

- **delaysum_wave_t** (unsigned int nch, unsigned int fragsize, const std::vector< **mha_real_t** > &weights_, const std::vector< int > &delays_)

Constructor of the runtime configuration.
- **mha_wave_t * process (mha_wave_t *)**

Private Attributes

- std::vector< **mha_real_t** > **weights**

Relative weights for each channel. Order is [chan0, chan1, ...].
- **MHASignal::waveform_t out**

Output waveform.

4.89.1 Detailed Description

Runtime configuration of the delaysum_wave plugin.

Inherits from the already present delay_t class. The constructor initializes and validates the runtime configuration and forwards the delay vector to the delay_t class. The process function first calls delay_t::process and then multiplies every output channel with its weight and adds them into the output channel.

4.89.2 Constructor & Destructor Documentation

4.89.2.1 delaysum_wave_t()

```
delaysum::delaysum_wave_t::delaysum_wave_t (
    unsigned int nch,
    unsigned int fragsize,
    const std::vector< mha_real_t > & weights_,
    const std::vector< int > & delays_ )
```

Constructor of the runtime configuration.

Parameters

<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.

4.89.3 Member Function Documentation

4.89.3.1 process()

```
mha_wave_t * delaysum::delaysum_wave_t::process (
```

```
        mha_wave_t * signal )
```

4.89.4 Member Data Documentation

4.89.4.1 weights

```
std::vector< mha_real_t > delaysum::delaysum_wave_t::weights [private]
```

Relative weights for each channel. Order is [chan0, chan1, ...].

4.89.4.2 **out** `MHASignal::waveform_t delaysum::delaysum_wave_t::out` [private]

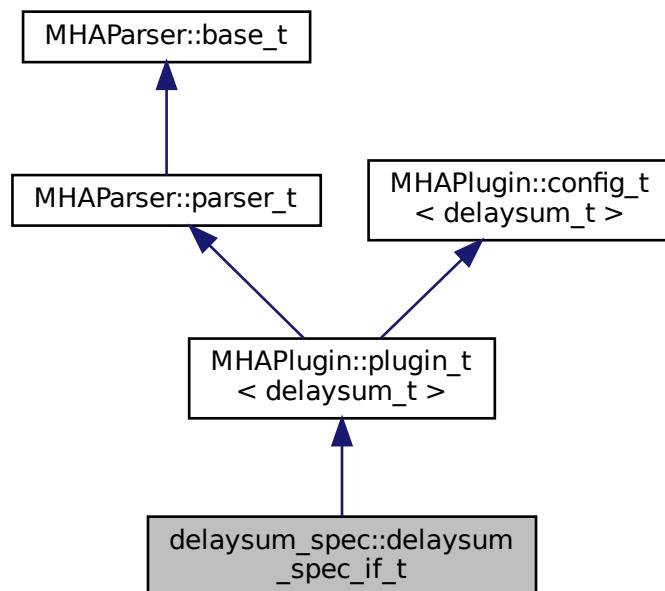
Output waveform.

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

- `delaysum_wave.cpp`

4.90 `delaysum_spec::delaysum_spec_if_t` Class Reference

Inheritance diagram for `delaysum_spec::delaysum_spec_if_t`:



Public Member Functions

- `delaysum_spec_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

- `MHAParser::vfloat_t groupdelay`
- `MHAParser::vfloat_t gain`
- `MHAEvents::patchbay_t< delaysum_spec_if_t > patchbay`

Additional Inherited Members

4.90.1 Constructor & Destructor Documentation

4.90.1.1 `delaysum_spec_if_t()` `delaysum_spec::delaysum_spec_if_t::delaysum_spec_if_t (`
`const algo_comm_t & iac,`
`const std::string & ,`
`const std::string &)`

4.90.2 Member Function Documentation

4.90.2.1 `process()` `mha_spec_t * delaysum_spec::delaysum_spec_if_t::process (`
`mha_spec_t * spec)`

4.90.2.2 `prepare()` `void delaysum_spec::delaysum_spec_if_t::prepare (`
`mhacconfig_t & signal_info) [virtual]`

Implements `MHAPlugin::plugin_t< delaysum_t >` (p. [1148](#)).

4.90.2.3 `update_cfg()` `void delaysum_spec::delaysum_spec_if_t::update_cfg () [private]`

4.90.3 Member Data Documentation

4.90.3.1 groupdelay `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::groupdelay` [private]

4.90.3.2 gain `MHAParser::vfloat_t delaysum_spec::delaysum_spec_if_t::gain` [private]

4.90.3.3 patchbay `MHAEvents::patchbay_t< delaysum_spec_if_t> delaysum_spec<::delaysum_spec_if_t::patchbay` [private]

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

- `delaysum_spec.cpp`

4.91 delaysum_spec::delaysum_t Class Reference

Public Member Functions

- `delaysum_t` (`std::vector< float > groupdelay, std::vector< float > gain, unsigned int n← Channels, unsigned int nFFT, float fs)`
- `mha_spec_t * process (mha_spec_t *)`

Private Attributes

- `MHASignal::spectrum_t scale`
- `MHASignal::spectrum_t output`

4.91.1 Constructor & Destructor Documentation

4.91.1.1 delaysum_t() `delaysum_spec::delaysum_t::delaysum_t (`
 `std::vector< float > groupdelay,`
 `std::vector< float > gain,`
 `unsigned int nChannels,`
 `unsigned int nFFT,`
 `float fs)`

4.91.2 Member Function Documentation

4.91.2.1 process() `mha_spec_t * delaysum_spec::delaysum_t::process (`
 `mha_spec_t * spec)`

4.91.3 Member Data Documentation

4.91.3.1 scale `MHASignal::spectrum_t delaysum_spec::delaysum_t::scale [private]`

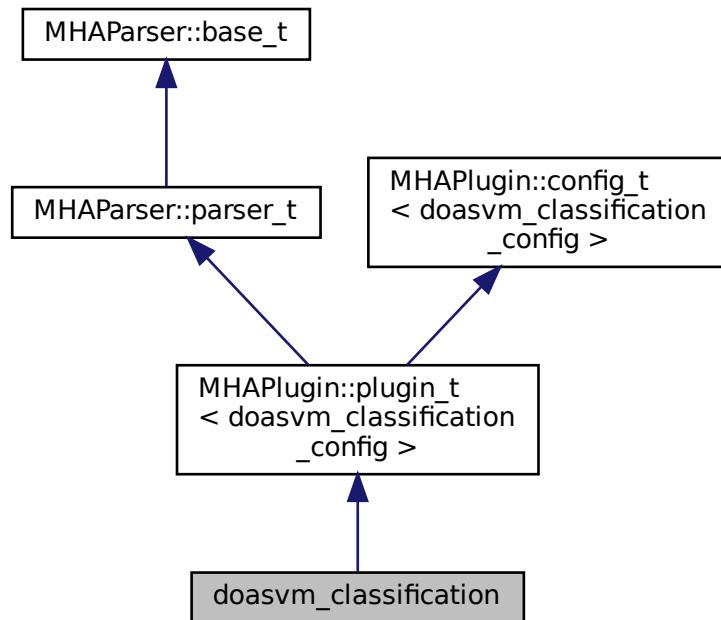
4.91.3.2 output `MHASignal::spectrum_t delaysum_spec::delaysum_t::output [private]`

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

- `delaysum_spec.cpp`

4.92 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

4.92.1 Constructor & Destructor Documentation

```
4.92.1.1 doasvm_classification() doasvm_classification::doasvm_classification (
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs our plugin.

```
4.92.1.2 ~doasvm_classification() doasvm_classification::~doasvm_classification (
)
```

4.92.2 Member Function Documentation

```
4.92.2.1 process() mha_wave_t * doasvm_classification::process (
    mha_wave_t * signal )
```

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

```
4.92.2.2 prepare() void doasvm_classification::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<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. 1148).

4.92.2.3 release() void doasvm_classification::release (void) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin_t< doasvm_classification_config >** (p. 1149).

4.92.2.4 update_cfg() void doasvm_classification::update_cfg () [private]

4.92.3 Member Data Documentation

4.92.3.1 angles **MHAParser::vfloat_t** doasvm_classification::angles

4.92.3.2 w **MHAParser::mfloat_t** doasvm_classification::w

4.92.3.3 b **MHAParser::vfloat_t** doasvm_classification::b

4.92.3.4 x **MHAParser::vfloat_t** doasvm_classification::x

4.92.3.5 `y` `MHAParser::vfloat_t` `doasvm_classification::y`

4.92.3.6 `p_name` `MHAParser::string_t` `doasvm_classification::p_name`

4.92.3.7 `max_p_ind_name` `MHAParser::string_t` `doasvm_classification::max_p_ind_name`

4.92.3.8 `vGCC_name` `MHAParser::string_t` `doasvm_classification::vGCC_name`

4.92.3.9 `patchbay` `MHAEvents::patchbay_t< doasvm_classification>` `doasvm_classification::patchbay` [private]

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

- `doasvm_classification.h`
- `doasvm_classification.cpp`

4.93 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`

4.93.1 Constructor & Destructor Documentation

4.93.1.1 doasvm_classification_config() doasvm_classification_config::doasvm_classification_config (

```
    algo_comm_t & ac,
    const mhaconfig_t in_cfg,
    doasvm_classification * _doasvm )
```

4.93.1.2 ~doasvm_classification_config() doasvm_classification_config::~doasvm_classification_config ()

4.93.2 Member Function Documentation

4.93.2.1 process() mha_wave_t * doasvm_classification_config::process (

```
    mha_wave_t * wave )
```

4.93.3 Member Data Documentation

4.93.3.1 ac algo_comm_t& doasvm_classification_config::ac

4.93.3.2 doasvm doasvm_classification* doasvm_classification_config::doasvm

4.93.3.3 p MHA_AC::waveform_t doasvm_classification_config::p

4.93.3.4 p_max MHA_AC::int_t doasvm_classification_config::p_max

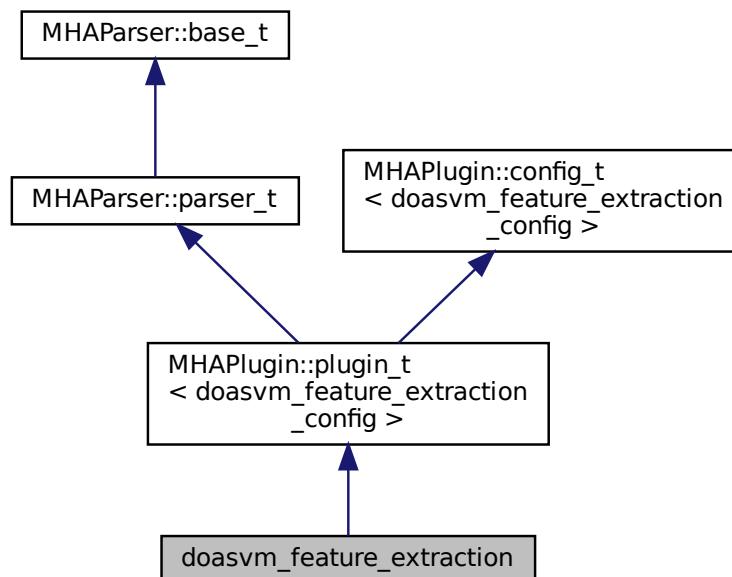
4.93.3.5 c mha_wave_t doasvm_classification_config::c

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

- **doasvm_classification.h**
- **doasvm_classification.cpp**

4.94 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

4.94.1 Constructor & Destructor Documentation

```
4.94.1.1 doasvm_feature_extraction() doasvm_feature_extraction::doasvm_feature_←  
extraction ( ←  
            algo_comm_t & ac,  
            const std::string & chain_name,  
            const std::string & algo_name )
```

Constructs our plugin.

```
4.94.1.2 ~doasvm_feature_extraction() doasvm_feature_extraction::~doasvm_feature_←  
_extraction ( )
```

4.94.2 Member Function Documentation

```
4.94.2.1 process() mha_wave_t * doasvm_feature_extraction::process (
    mha_wave_t * signal )
```

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

```
4.94.2.2 prepare() void doasvm_feature_extraction::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<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< doasvm_feature_extraction_config >** (p. [1148](#)).

4.94.2.3 release() `void doasvm_feature_extraction::release (void) [inline], [virtual]`

Reimplemented from **MHAParser::plugin_t< doasvm_feature_extraction_config >** (p. [1149](#)).

4.94.2.4 update_cfg() `void doasvm_feature_extraction::update_cfg () [private]`

4.94.3 Member Data Documentation

4.94.3.1 fftlen `MHAParser::int_t doasvm_feature_extraction::fftlen`

4.94.3.2 max_lag `MHAParser::int_t doasvm_feature_extraction::max_lag`

4.94.3.3 nupsample `MHAParser::int_t doasvm_feature_extraction::nupsample`

4.94.3.4 vGCC_name `MHAParser::string_t doasvm_feature_extraction::vGCC_name`

4.94.3.5 patchbay `MHAEVENTS::patchbay_t< doasvm_feature_extraction> doasvm_<→`
`feature_extraction::patchbay [private]`

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

- `doasvm_feature_extraction.h`
- `doasvm_feature_extraction.cpp`

4.95 doasvm_feature_extraction_config Class Reference

Public Member Functions

- `doasvm_feature_extraction_config (algo_comm_t &ac, const mhaconfig_t in_cfg, doasvm_feature_extraction *_doagcc)`
- `~doasvm_feature_extraction_config ()`
- `mha_wave_t * process (mha_wave_t *)`

Public Attributes

- `doasvm_feature_extraction * doagcc`
- `unsigned int wndlen`
- `unsigned int fftlen`
- `unsigned int G_length`
- `unsigned int GCC_start`
- `unsigned int GCC_end`
- `MHA_AC::waveform_t vGCC_ac`
- `mha_fft_t fft`
- `mha_fft_t ifft`
- `double hifftwin_sum`
- `MHASignal::waveform_t proc_wave`
- `MHASignal::waveform_t hwin`
- `MHASignal::waveform_t hifftwin`
- `MHASignal::waveform_t vGCC`
- `MHASignal::spectrum_t in_spec`
- `MHASignal::spectrum_t G`

4.95.1 Constructor & Destructor Documentation

4.95.1.1 doasvm_feature_extraction_config() doasvm_feature_extraction_config→
::doasvm_feature_extraction_config (
 algo_comm_t & ac,
 const mhaconfig_t in_cfg,
 doasvm_feature_extraction * _doagcc)

4.95.1.2 ~doasvm_feature_extraction_config() doasvm_feature_extraction_config→
::~doasvm_feature_extraction_config ()

4.95.2 Member Function Documentation

4.95.2.1 process() mha_wave_t * doasvm_feature_extraction_config::process (
 mha_wave_t * wave)

4.95.3 Member Data Documentation

4.95.3.1 doagcc doasvm_feature_extraction* doasvm_feature_extraction_config→
::doagcc

4.95.3.2 wndlen unsigned int doasvm_feature_extraction_config::wndlen

4.95.3.3 fftlen unsigned int doasvm_feature_extraction_config::fftlens

4.95.3.4 G_length unsigned int doasvm_feature_extraction_config::G_length

4.95.3.5 GCC_start unsigned int doasvm_feature_extraction_config::GCC_start

4.95.3.6 GCC_end unsigned int doasvm_feature_extraction_config::GCC_end

4.95.3.7 vGCC_ac MHA_AC::waveform_t doasvm_feature_extraction_config::vGCC_ac

4.95.3.8 fft mha_fft_t doasvm_feature_extraction_config::fft

4.95.3.9 ifft mha_fft_t doasvm_feature_extraction_config::ifft

4.95.3.10 hifftwin_sum double doasvm_feature_extraction_config::hifftwin_sum

4.95.3.11 proc_wave MHASignal::waveform_t doasvm_feature_extraction_config::proc_wave

4.95.3.12 hwin MHASignal::waveform_t doasvm_feature_extraction_config::hwin

4.95.3.13 hifftwin MHASignal::waveform_t doasvm_feature_extraction_config::hifftwin

4.95.3.14 vGCC `MHASignal::waveform_t` doasvm_feature_extraction_config::vGCC

4.95.3.15 in_spec `MHASignal::spectrum_t` doasvm_feature_extraction_config::in_spec

4.95.3.16 G `MHASignal::spectrum_t` doasvm_feature_extraction_config::G

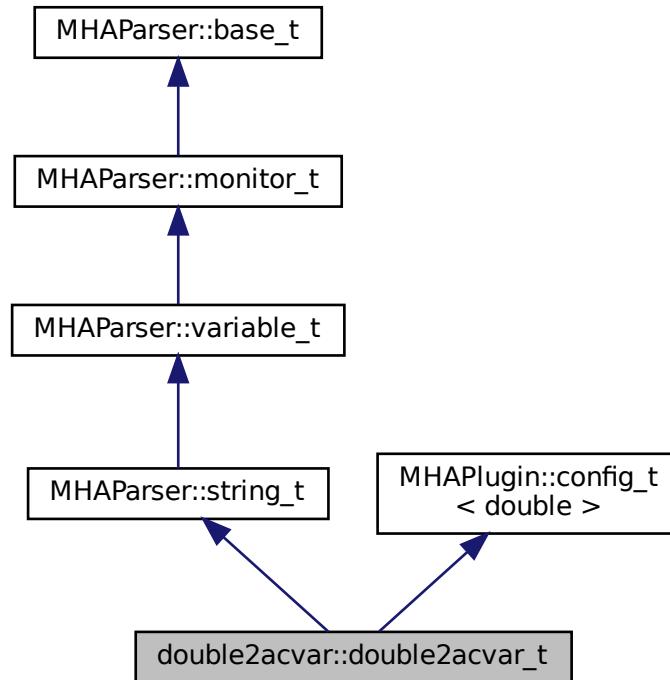
The documentation for this class was generated from the following files:

- `doasvm_feature_extraction.h`
- `doasvm_feature_extraction.cpp`

4.96 double2acvar::double2acvar_t Class Reference

Plugin interface class for **double2acvar** (p. 90).

Inheritance diagram for double2acvar::double2acvar_t:



Public Member Functions

- **double2acvar_t (algo_comm_t ac, const std::string &chain, const std::string &algo)**
Standard plugin constructor.
- **~double2acvar_t ()=default**
- template<class T >
T * process (T *s)
process() (p. 434) does not alter the signal and has same implementation regardless of signal domain.
- **void poll_latest_value_and_reinsert ()**
Called from process() (p. 434) and, when allowed, also from on_configuration_update() (p. 435).
- **void prepare_ (mhaconfig_t &)**
Prepare method as expected by the plugin interface macros.
- **void release_ ()**
Release method as expected by the plugin interface macros.
- **void on_configuration_update ()**
Callback function on write access to the string configuration value.

Private Attributes

- **MHA_AC::double_t ac_double**
AC variable inserted by this plugin.
- **MHAEvents::patchbay_t< double2acvar_t > patchbay**
Callback router.
- **bool is_prepared**
Flag to keep track if we are currently prepared.

Additional Inherited Members

4.96.1 Detailed Description

Plugin interface class for **double2acvar** (p. 90).

4.96.2 Constructor & Destructor Documentation

4.96.2.1 double2acvar_t()

```
double2acvar::double2acvar_t::double2acvar_t (
    algo_comm_t ac,
    const std::string & chain,
    const std::string & algo )
```

Standard plugin constructor.

Parameters

<i>ac</i>	Algorithm communication variable space.
<i>chain</i>	Unused parameter.
<i>algo</i>	Configured name of this plugin, also used as name of the AC variable.

4.96.2.2 ~double2acvar_t() `double2acvar::double2acvar_t::~double2acvar_t () [default]`

4.96.3 Member Function Documentation

4.96.3.1 process()

```
template<class T >
T * double2acvar::double2acvar_t::process (
    T * s )
```

process() (p. 434) does not alter the signal and has same implementation regardless of signal domain.

Parameters

<i>s</i>	Pointer to input signal structure, mha_wave_t (p. 839) or mha_spec_t (p. 793).
----------	--

Returns

s, unaltered.

4.96.3.2 poll_latest_value_and_reinsert()

Called from **process()** (p. 434) and, when allowed, also from **on_configuration_update()** (p. 435).

poll_latest_value_and_reinsert() (p. 434) retrieves the latest configured value and reinserts the AC variable into the AC space.

```
4.96.3.3 prepare_() void double2acvar::double2acvar_t::prepare_ (
    mhaconfig_t & )
```

Prepare method as expected by the plugin interface macros.

Parameter is not used nor altered. Sets is_prepared flag.

```
4.96.3.4 release_() void double2acvar::double2acvar_t::release_ ( )
```

Release method as expected by the plugin interface macros.

Resets is_prepared flag.

```
4.96.3.5 on_configuration_update() void double2acvar::double2acvar_t::on_configuration_update ( )
```

Callback function on write access to the string configuration value.

4.96.4 Member Data Documentation

```
4.96.4.1 ac_double MHA_AC::double_t double2acvar::double2acvar_t::ac_double [private]
```

AC variable inserted by this plugin.

```
4.96.4.2 patchbay MHAEvents::patchbay_t< double2acvar_t> double2acvar::double2acvar_t::patchbay [private]
```

Callback router.

4.96.4.3 `is_prepared` `bool double2acvar::double2acvar_t::is_prepared [private]`

Flag to keep track if we are currently prepared.

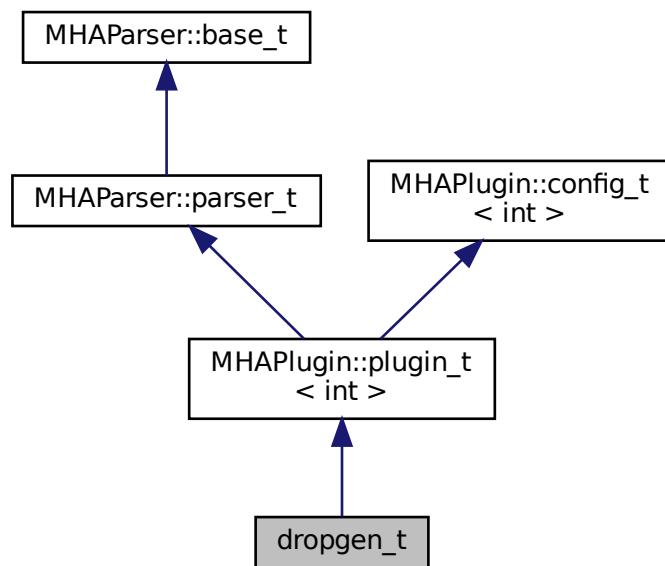
If we are, then signal processing is active and AC variables may only be accessed when MHA is currently executing out `process()` (p. 434) method.

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

- `double2acvar.cpp`

4.97 `dropgen_t` Class Reference

Inheritance diagram for `dropgen_t`:



Public Member Functions

- `dropgen_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 ()`

Public Attributes

- `MHAParser::float_t min_sleep_time`
- `MHAParser::float_t max_sleep_time`
- `MHAParser::float_t chance`
- `MHAEvents::patchbay_t< dropgen_t > patchbay`
- `std::random_device r`
- `std::mt19937 random_engine`
- `std::uniform_real_distribution dis`

Additional Inherited Members

4.97.1 Constructor & Destructor Documentation

```
4.97.1.1 dropgen_t() dropgen_t::dropgen_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.97.2 Member Function Documentation

```
4.97.2.1 process() [1/2] mha_wave_t * dropgen_t::process (
    mha_wave_t * s )
```

```
4.97.2.2 process() [2/2] mha_spec_t * dropgen_t::process (
    mha_spec_t * s )
```

```
4.97.2.3 prepare() void dropgen_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements `MHAPlugin::plugin_t< int >` (p. [1148](#)).

4.97.2.4 `release()` `void dropgen_t::release () [virtual]`

Reimplemented from **MHAParser::plugin_t< int >** (p. 1149).

4.97.3 Member Data Documentation

4.97.3.1 `min_sleep_time` `MHAParser::float_t dropgen_t::min_sleep_time`

4.97.3.2 `max_sleep_time` `MHAParser::float_t dropgen_t::max_sleep_time`

4.97.3.3 `chance` `MHAParser::float_t dropgen_t::chance`

4.97.3.4 `patchbay` `MHAEvents::patchbay_t< dropgen_t > dropgen_t::patchbay`

4.97.3.5 `r` `std::random_device dropgen_t::r`

4.97.3.6 `random_engine` `std::mt19937 dropgen_t::random_engine`

4.97.3.7 `dis` `std::uniform_real_distribution dropgen_t::dis`

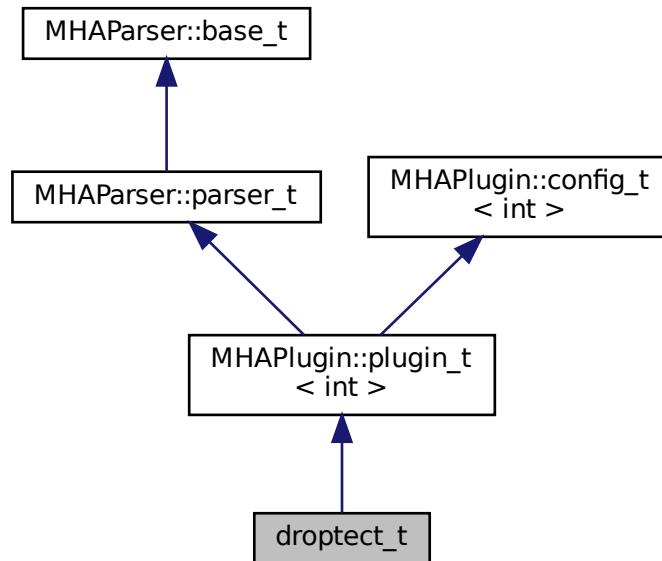
The documentation for this class was generated from the following file:

- **dropgen.cpp**

4.98 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)`**

Allocates and initializes storage for this algorithm.
- **`void release (void)`**

Deallocates storage.
- **`mha_spec_t * process (mha_spec_t *signal)`**

Compares current spectrum against history.

Private Attributes

- **MHAParser::vint_mon_t dropouts**
- **MHAParser::vint_mon_t consecutive_dropouts**
- **MHAParser::int_mon_t blocks**
- **MHAParser::bool_t reset**
- **MHAParser::float_t threshold**
- **MHASignal::waveform_t * current_powspec**
- **MHASignal::waveform_t * filtered_powspec**
- **MHAParser::float_t tau**
- **std::vector< bool > filter_activated**
- **float period**
The period of the process callback (duration of fragsize in seconds)
- **MHAParser::mfloat_mon_t filtered_powspec_mon**
User access to filtered spectrum.
- **MHAParser::vfloat_mon_t level_mon**
User access to broadband levels.

Additional Inherited Members

4.98.1 Detailed Description

Detect dropouts in a signal with a constant spectrum.

4.98.2 Constructor & Destructor Documentation

```
4.98.2.1 droptect_t() 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.

4.98.3 Member Function Documentation

```
4.98.3.1 prepare() void droptect_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Allocates and initializes storage for this algorithm.

Parameters

<i>signal_info</i>	contains fft length, number of channels, fft length and hop size.
--------------------	---

Implements **MHAParser::plugin_t< int >** (p. 1148).

4.98.3.2 release() `void droptect_t::release (void) [virtual]`

Deallocates storage.

Reimplemented from **MHAParser::plugin_t< int >** (p. 1149).

4.98.3.3 process() `mha_spec_t * droptect_t::process (mha_spec_t * signal)`

Compares current spectrum against history.

If spectral power has changed or is below threshold, this is interpreted as dropout occurrence.

4.98.4 Member Data Documentation

4.98.4.1 dropouts `MHAParser::vint_mon_t droptect_t::dropouts [private]`

4.98.4.2 consecutive_dropouts `MHAParser::vint_mon_t droptect_t::consecutive_← dropouts [private]`

4.98.4.3 blocks `MHAParser::int_mon_t droptect_t::blocks [private]`

4.98.4.4 reset `MHAParser::bool_t droptect_t::reset` [private]

4.98.4.5 threshold `MHAParser::float_t droptect_t::threshold` [private]

4.98.4.6 current_powspec `MHASignal::waveform_t* droptect_t::current_powspec` [private]

4.98.4.7 filtered_powspec `MHASignal::waveform_t* droptect_t::filtered_powspec` [private]

4.98.4.8 tau `MHAParser::float_t droptect_t::tau` [private]

4.98.4.9 filter_activated `std::vector<bool> droptect_t::filter_activated` [private]

4.98.4.10 period `float droptect_t::period` [private]

The period of the process callback (duration of fragsize in seconds)

4.98.4.11 filtered_powspec_mon `MHAParser::mfloat_mon_t droptect_t::filtered_<→powspec_mon` [private]

User access to filtered spectrum.

4.98.4.12 level_mon MHParse::vfloat_mon_t droptect_t::level_mon [private]

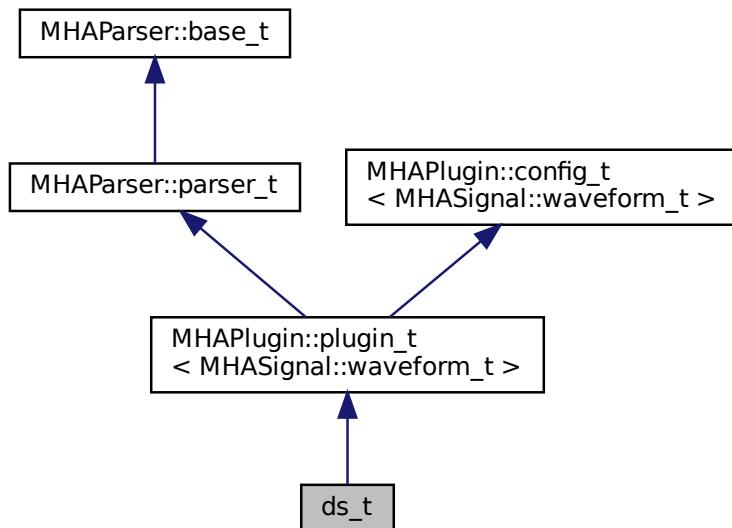
User access to broadband levels.

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

- **droptect.cpp**

4.99 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

4.99.1 Constructor & Destructor Documentation

4.99.1.1 `ds_t()` `ds_t::ds_t (`
 `algo_comm_t iac,`
 `std::string ,`
 `std::string)`

4.99.2 Member Function Documentation

4.99.2.1 `process()` `mha_wave_t * ds_t::process (`
 `mha_wave_t * s)`

4.99.2.2 `prepare()` `void ds_t::prepare (`
 `mhaconfig_t & cf) [virtual]`

Implements `MHAParser::parser_t< MHASignal::waveform_t >` (p. [1148](#)).

4.99.2.3 `release()` `void ds_t::release () [virtual]`

Reimplemented from `MHAParser::parser_t< MHASignal::waveform_t >` (p. [1149](#)).

4.99.3 Member Data Documentation

4.99.3.1 `ratio` `MHAParser::int_t ds_t::ratio [private]`

4.99.3.2 antialias `MHAFilter::iir_filter_t ds_t::antialias [private]`

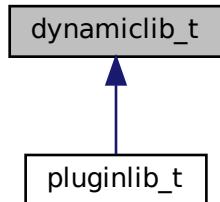
The documentation for this class was generated from the following file:

- `downsample.cpp`

4.100 dynamiclib_t Class Reference

Wrapper class around a shared library.

Inheritance diagram for dynamiclib_t:



Public Member Functions

- **dynamiclib_t** (const std::string &name_)

C'tor of the wrapper class.
- virtual void * **resolve** (const std::string &name_)

Resolves the function specified by name_ and returns a pointer to it or a nullptr if the function was not found in the wrapped library.
- virtual void * **resolve_checked** (const std::string &name_)

Resolves the function specified by name_ and returns a pointer to it or throws an exception if the function was not found.
- virtual ~**dynamiclib_t** ()

D'tor.
- virtual const std::string & **getmodulename** () const

Returns unqualified filename of the wrapped library sans file suffix.
- virtual const std::string & **getname** () const

Protected Member Functions

- **dynamiclib_t ()**
Default constructor.
- **void load_lib (const std::string &name_)**
Loads the library specified in name_ and saves a handle in h.

Protected Attributes

- **std::string fullname**
Fully qualified file name of the library.
- **std::string modulename**
Unqualified file name of the library.
- **mha_libhandle_t h**
Handle to the shared library.

4.100.1 Detailed Description

Wrapper class around a shared library.

Encapsulates the OS-specific stuff of loading the shared library, resolving functions, etc... Uses the dload API on Linux/macOS and the win32 API on Windows

4.100.2 Constructor & Destructor Documentation

4.100.2.1 dynamiclib_t() [1/2] `dynamiclib_t::dynamiclib_t (const std::string & name_)`

C'tor of the wrapper class.

Takes the the file name of a shared libary w/o the suffix as argument, searches for the library in the system-dependent standard paths for libraries and in MHA_LIBRARY_PATH. Calls load_lib for the actual work.

Parameters

<code>name_</code>	File name of the shared library, without suffix
--------------------	---

Exceptions

MHA_Error (p. 763)	if the library can not be found or can not be loaded
---------------------------	--

4.100.2.2 ~dynamiclib_t() `dynamiclib_t::~dynamiclib_t () [virtual]`

D'tor.

Closes the library handle.

4.100.2.3 dynamiclib_t() [2/2] `dynamiclib_t::dynamiclib_t () [protected]`

Default constructor.

4.100.3 Member Function Documentation

4.100.3.1 resolve() `void * dynamiclib_t::resolve (const std::string & name_) [virtual]`

Resolves the function specified by `name_` and returns a pointer to it or a `nullptr` if the function was not found in the wrapped library.

Parameters

<code>name_</code>	Name of the function to be resolved
--------------------	-------------------------------------

Returns

Pointer to the function

Reimplemented in **pluginlib_t** (p. 1359).

4.100.3.2 resolve_checked() void * dynamiclib_t::resolve_checked (const std::string & name_) [virtual]

Resolves the function specified by name_ and returns a pointer to it or throws an exception if the function was not found.

Parameters

<i>name_</i>	Name of the function to be resolved
--------------	-------------------------------------

Returns

Pointer to the function

4.100.3.3 getmodulename() virtual const std::string& dynamiclib_t::getmodulename () const [inline], [virtual]

Returns unqualified filename of the wrapped library sans file suffix.

Returns

Unqualified filename of the wrapped library

4.100.3.4 getname() virtual const std::string& dynamiclib_t::getname () const [inline], [virtual]

4.100.3.5 load_lib() void dynamiclib_t::load_lib (const std::string & name_) [protected]

Loads the library specified in name_ and saves a handle in h.

Parameters

<code>name</code>	unqualified file name of the shared library w/o suffix
<code>_</code>	

4.100.4 Member Data Documentation**4.100.4.1 fullname** std::string dynamiclib_t::fullname [protected]

Fully qualified file name of the library.

4.100.4.2 modulename std::string dynamiclib_t::modulename [protected]

Unqualified file name of the library.

4.100.4.3 h mha_libhandle_t dynamiclib_t::h [protected]

Handle to the shared library.

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

- **mha_os.h**
- **mha_os.cpp**

4.101 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 srat, 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**

4.101.1 Detailed Description

Real-time class for after burn effect.

The constructor processes the parameters and creates pre-processed variables for efficient realtime processing.

4.101.2 Constructor & Destructor Documentation

4.101.2.1 dc_afterburn_rt_t() `DynComp::dc_afterburn_rt_t::dc_afterburn_rt_t (`
`const std::vector< float > & cf,`
`unsigned int channels,`
`float srate,`
`const dc_afterburn_vars_t & vars)`

4.101.3 Member Function Documentation

4.101.3.1 burn() `void DynComp::dc_afterburn_rt_t::burn (`
`float & Gin,`
`float Lin,`
`unsigned int band,`
`unsigned int channel) [inline]`

gain modifier method (afterburn).

Parameters

<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 $\text{Gin} * \text{Lin}$.

4.101.4 Member Data Documentation

4.101.4.1 drain_inv std::vector<float> DynComp::dc_afterburn_rt_t::drain_inv [private]

4.101.4.2 conflux std::vector<float> DynComp::dc_afterburn_rt_t::conflux [private]

4.101.4.3 maxgain std::vector<float> DynComp::dc_afterburn_rt_t::maxgain [private]

4.101.4.4 mpo_inv std::vector<float> DynComp::dc_afterburn_rt_t::mpo_inv [private]

4.101.4.5 lp std::vector< **MHAFilter::olflt_lowpass_t**> DynComp::dc_afterburn_rt_t::lp [private]

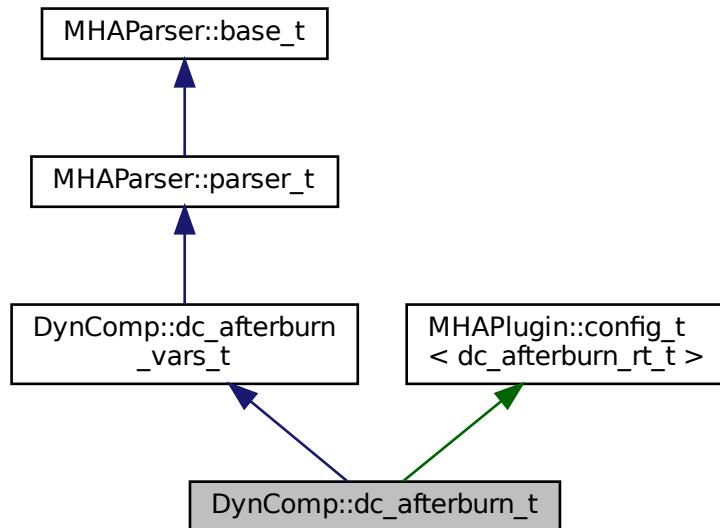
The documentation for this class was generated from the following files:

- **dc_afterburn.h**
- **dc_afterburn.cpp**

4.102 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

4.102.1 Detailed Description

Afterburn class, to be defined as a member of compressors.

4.102.2 Constructor & Destructor Documentation

4.102.2.1 dc_afterburn_t() DynComp::dc_afterburn_t::dc_afterburn_t ()

4.102.3 Member Function Documentation

4.102.3.1 set_fb_pars() void DynComp::dc_afterburn_t::set_fb_pars (const std::vector< float > & cf, unsigned int channels, float srate)

4.102.3.2 unset_fb_pars() void DynComp::dc_afterburn_t::unset_fb_pars ()

4.102.3.3 update_burner() void DynComp::dc_afterburn_t::update_burner () [inline]

4.102.3.4 burn() void DynComp::dc_afterburn_t::burn (float & Gin, float Lin, unsigned int band, unsigned int channel) [inline]

4.102.3.5 update() void DynComp::dc_afterburn_t::update () [private]

4.102.4 Member Data Documentation

4.102.4.1 patchbay `MHAEvents::patchbay_t< dc_afterburn_t>` DynComp::dc_afterburn_t::patchbay [private]

4.102.4.2 _cf std::vector<float> DynComp::dc_afterburn_t::_cf [private]

4.102.4.3 _channels unsigned int DynComp::dc_afterburn_t::_channels [private]

4.102.4.4 _srate float DynComp::dc_afterburn_t::_srate [private]

4.102.4.5 commit_pending bool DynComp::dc_afterburn_t::commit_pending [private]

4.102.4.6 fb_pars_configured bool DynComp::dc_afterburn_t::fb_pars_configured [private]

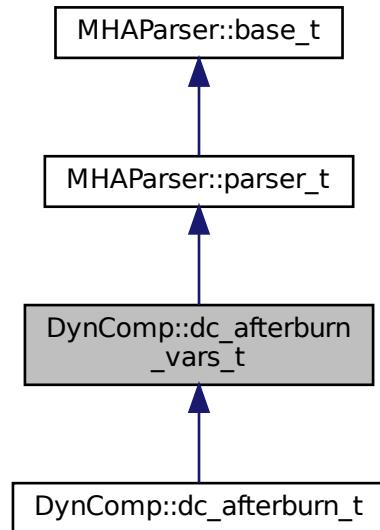
The documentation for this class was generated from the following files:

- `dc_afterburn.h`
- `dc_afterburn.cpp`

4.103 DynComp::dc_afterburn_vars_t Class Reference

Variables for `dc_afterburn_t` (p. 452) class.

Inheritance diagram for DynComp::dc_afterburn_vars_t:



Public Member Functions

- `dc_afterburn_vars_t()`

Public Attributes

- `MHParse::vfloat_t f`
- `MHParse::vfloat_t drain`
- `MHParse::vfloat_t conflux`
- `MHParse::vfloat_t maxgain`
- `MHParse::vfloat_t mpo`
- `MHParse::float_t taugain`
- `MHParse::kw_t commit`
- `MHParse::bool_t bypass`

Additional Inherited Members

4.103.1 Detailed Description

Variables for **dc_afterburn_t** (p. 452) class.

4.103.2 Constructor & Destructor Documentation

4.103.2.1 dc_afterburn_vars_t() `DynComp::dc_afterburn_vars_t::dc_afterburn_vars_t ()`

4.103.3 Member Data Documentation

4.103.3.1 f MHPARSER::vfloat_t `DynComp::dc_afterburn_vars_t::f`

4.103.3.2 drain MHPARSER::vfloat_t `DynComp::dc_afterburn_vars_t::drain`

4.103.3.3 conflux MHPARSER::vfloat_t `DynComp::dc_afterburn_vars_t::conflux`

4.103.3.4 maxgain MHPARSER::vfloat_t `DynComp::dc_afterburn_vars_t::maxgain`

4.103.3.5 mpo MHPARSER::vfloat_t `DynComp::dc_afterburn_vars_t::mpo`

4.103.3.6 taugain `MHAParser::float_t` `DynComp::dc_afterburn_vars_t::taugain`

4.103.3.7 commit `MHAParser::kw_t` `DynComp::dc_afterburn_vars_t::commit`

4.103.3.8 bypass `MHAParser::bool_t` `DynComp::dc_afterburn_vars_t::bypass`

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

- `dc_afterburn.h`
- `dc_afterburn.cpp`

4.104 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**

4.104.1 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. 459) member function via a gain prescription rule from an auditory profile.

4.104.2 Constructor & Destructor Documentation

```
4.104.2.1 gaintable_t() gaintable_t::gaintable_t (
    const std::vector< mha_real_t > & LInput,
    const std::vector< mha_real_t > & FCenter,
    unsigned int channels )
```

Constructor.

Parameters

<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).

```
4.104.2.2 ~gaintable_t() gaintable_t::~gaintable_t ( )
```

4.104.3 Member Function Documentation

4.104.3.1 update() 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.

4.104.3.2 get_gain() [1/3] mha_real_t gaintable_t::get_gain (mha_real_t *Lin*, mha_real_t *Fin*, unsigned int *channel*)

Read Gain from gain table.

Parameters

<i>Lin</i>	Input level
<i>Fin</i>	Input frequency (no match required)
<i>channel</i>	Audio channel

4.104.3.3 get_gain() [2/3] mha_real_t gaintable_t::get_gain (mha_real_t *Lin*, unsigned int *band*, unsigned int *channel*)

Read Gain from gain table.

Parameters

<i>Lin</i>	Input level
<i>band</i>	Input frequency band
<i>channel</i>	Audio channel

4.104.3.4 `get_gain()` [3/3] `void gaintable_t::get_gain (`
`const mha_wave_t & Lin,`
`mha_wave_t & Gain)`

Read Gains from gain table.

Parameters

<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.

4.104.3.5 `nbands()` `unsigned int DynComp::gaintable_t::nbands () const [inline]`

Return number of frequency bands.

4.104.3.6 `nchannels()` `unsigned int DynComp::gaintable_t::nchannels () const [inline]`

Return number of audio channels.

4.104.3.7 `get_iofun()` `std::vector< std::vector< mha_real_t > > gaintable_t::get_iofun () const`

Return current input-output function.

4.104.3.8 `get_vL()` `std::vector< mha_real_t> DynComp::gaintable_t::get_vL () const [inline]`

4.104.3.9 get_vF() std::vector< **mha_real_t**> DynComp::gaintable_t::get_vF () const [inline]

4.104.4 Member Data Documentation

4.104.4.1 num_L unsigned int DynComp::gaintable_t::num_L [private]

4.104.4.2 num_F unsigned int DynComp::gaintable_t::num_F [private]

4.104.4.3 num_channels unsigned int DynComp::gaintable_t::num_channels [private]

4.104.4.4 vL std::vector< **mha_real_t**> DynComp::gaintable_t::vL [private]

4.104.4.5 vF std::vector< **mha_real_t**> DynComp::gaintable_t::vF [private]

4.104.4.6 vFlog std::vector< **mha_real_t**> DynComp::gaintable_t::vFlog [private]

4.104.4.7 data std::vector<std::vector<std::vector< **mha_real_t**>>> DynComp::gaintable_t::data [private]

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

- **gaintable.h**
- **gaintable.cpp**

4.105 equalize::cfg_t Class Reference

Public Member Functions

- **cfg_t** (int infft, int inchannels, std::vector< std::vector< float > > ifgains)
- **cfg_t** (const **cfg_t** &) = delete
- **cfg_t** & **operator=** (const **cfg_t** &) = delete
- **~cfg_t** ()

Public Attributes

- int **num_bins**
- int **nchannels**
- **mha_real_t** * **fftgains**

4.105.1 Constructor & Destructor Documentation

4.105.1.1 cfg_t() [1/2] `cfg_t::cfg_t (`
 `int infft,`
 `int inchannels,`
 `std::vector< std::vector< float > > ifgains)`

4.105.1.2 cfg_t() [2/2] `equalize::cfg_t::cfg_t (`
 `const cfg_t &)` [delete]

4.105.1.3 ~cfg_t() `cfg_t::~cfg_t ()`

4.105.2 Member Function Documentation

```
4.105.2.1 operator=()   cfg_t& equalize::cfg_t::operator= (
    const  cfg_t & )  [delete]
```

4.105.3 Member Data Documentation

4.105.3.1 **num_bins** int equalize::cfg_t::num_bins

4.105.3.2 **nchannels** int equalize::cfg_t::nchannels

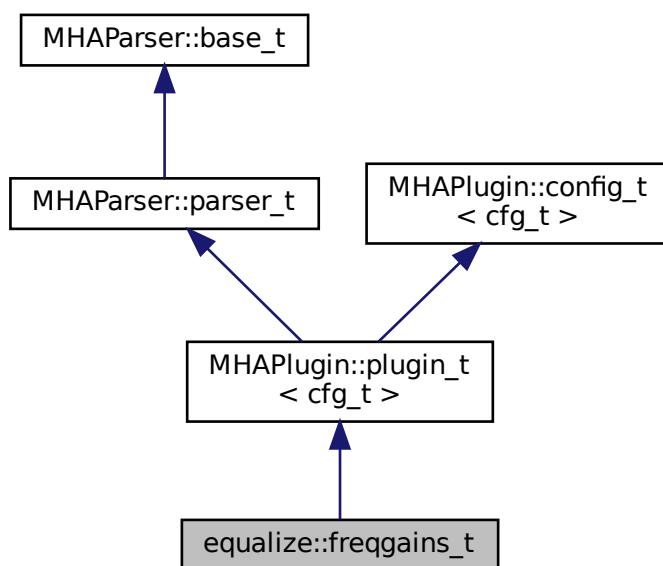
4.105.3.3 **fftgains** mha_real_t* equalize::cfg_t::fftgains

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

- **equalize.cpp**

4.106 equalize::freqgains_t Class Reference

Inheritance diagram for equalize::freqgains_t:



Public Member Functions

- `freqgains_t (const algo_comm_t &iac, const std::string &, const std::string &)`
- `mha_spec_t * process (mha_spec_t *)`
- `void prepare (mhaconfig_t &)`

Private Member Functions

- `void update_gains ()`
- `void update_id ()`

Private Attributes

- `MHAParser::mfloat_t fftgains`
- `MHAParser::string_t id`
- `MHAEvents::patchbay_t< freqgains_t > patchbay`

Additional Inherited Members

4.106.1 Constructor & Destructor Documentation

4.106.1.1 freqgains_t() `equalize::freqgains_t::freqgains_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

4.106.2 Member Function Documentation

4.106.2.1 process() `mha_spec_t * equalize::freqgains_t::process (`
 `mha_spec_t * s)`

```
4.106.2.2 prepare() void equalize::freqgains_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugIn::plugin_t< cfg_t >** (p. 1148).

```
4.106.2.3 update_gains() void equalize::freqgains_t::update_gains () [private]
```

```
4.106.2.4 update_id() void equalize::freqgains_t::update_id () [private]
```

4.106.3 Member Data Documentation

```
4.106.3.1 fftgains MHAParser::mfloat_t equalize::freqgains_t::fftgains [private]
```

```
4.106.3.2 id MHAParser::string_t equalize::freqgains_t::id [private]
```

```
4.106.3.3 patchbay MHAEEvents::patchbay_t< freqgains_t > equalize::freqgains_t::patchbay [private]
```

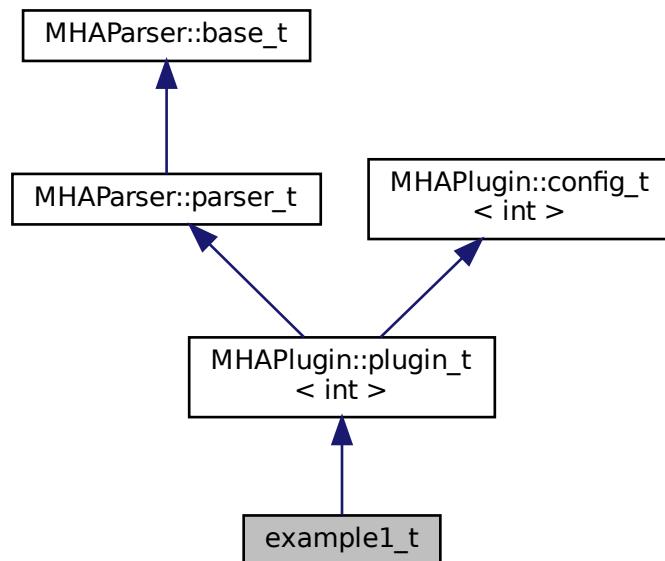
The documentation for this class was generated from the following file:

- **equalize.cpp**

4.107 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

4.107.1 Detailed Description

This C++ class implements the simplest example plugin for the step-by-step tutorial.

It inherits from **MHAPlugin::plugin_t** (p. 1146) for correct integration in the configuration language interface.

4.107.2 Constructor & Destructor Documentation

4.107.2.1 example1_t() `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.

4.107.3 Member Function Documentation

4.107.3.1 release() `void example1_t::release (void) [inline], [virtual]`

Release may be empty.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1149).

4.107.3.2 prepare() `void example1_t::prepare (mhaconfig_t & signal_info) [inline], [virtual]`

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains at least one channel

Parameters

<code>signal_info</code>	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. 1148).

4.107.3.3 process() `mha_wave_t* example1_t::process (`
 `mha_wave_t * signal) [inline]`

Signal processing performed by the plugin.

This plugin multiplies the signal in the first audio channel by a factor 0.1.

Parameters

<code>signal</code>	Pointer to the input signal structure.
---------------------	--

Returns

Returns a pointer to the input signal structure, with the signal modified by this plugin.
(In-place processing)

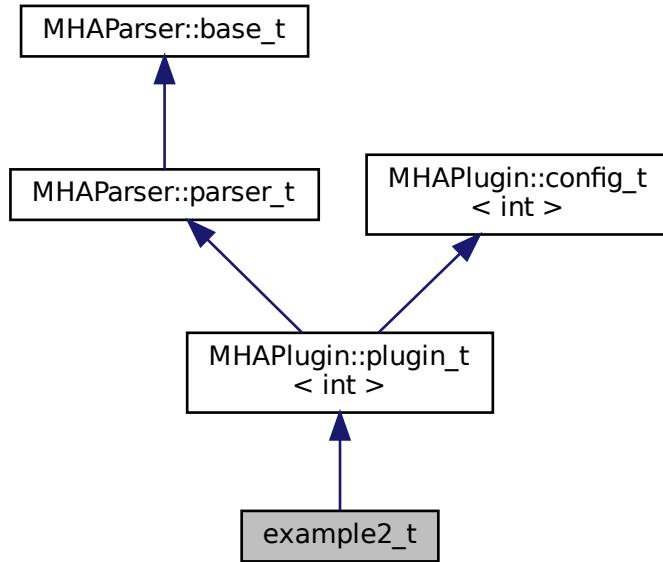
The documentation for this class was generated from the following file:

- `example1.cpp`

4.108 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

4.108.1 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.

4.108.2 Constructor & Destructor Documentation

```
4.108.2.1 example2_t() 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.

4.108.3 Member Function Documentation

```
4.108.3.1 prepare() void example2_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

<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. [1148](#)).

4.108.3.2 release() void example2_t::release (void) [virtual]

Undo restrictions posed in prepare.

Reimplemented from **MHAPlugIn::plugin_t< int >** (p. 1149).

4.108.3.3 process() mha_wave_t * example2_t::process (mha_wave_t * signal)

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

<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)

4.108.4 Member Data Documentation

4.108.4.1 scale_ch MHAParser::int_t example2_t::scale_ch [private]

Index of audio channel to scale.

4.108.4.2 factor MHAParser::float_t example2_t::factor [private]

The scaling factor applied to the selected channel.

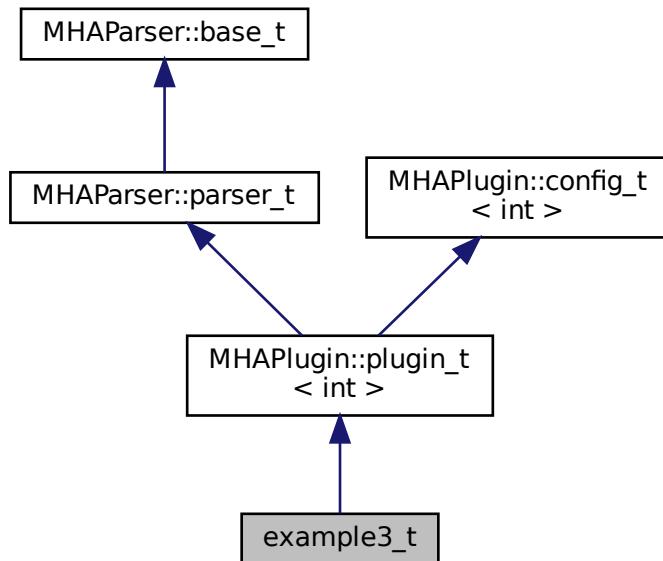
The documentation for this class was generated from the following file:

- **example2.cpp**

4.109 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

4.109.1 Detailed Description

A Plugin class using the openMHA Event mechanism.

This is the third example plugin for the step-by-step tutorial.

4.109.2 Constructor & Destructor Documentation

```
4.109.2.1 example3_t() 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.

4.109.3 Member Function Documentation

4.109.3.1 `on_scale_ch_writeaccess()` void example3_t::on_scale_ch_writeaccess ()
[private]

4.109.3.2 `on_scale_ch_valuechanged()` void example3_t::on_scale_ch_valuechanged () [private]

4.109.3.3 `on_scale_ch_readaccess()` void example3_t::on_scale_ch_readaccess ()
[private]

4.109.3.4 `on_prereadaccess()` void example3_t::on_prereadaccess () [private]

4.109.3.5 `prepare()` void example3_t::prepare (
 mhaconfig_t & signal_info) [virtual]

Plugin preparation.

This plugin checks that the input signal has the waveform domain and contains enough channels.

Parameters

<code>signal_info</code>	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. [1148](#)).

4.109.3.6 `release()` void example3_t::release (
 void) [virtual]

Bookkeeping only.

Reimplemented from **MHAPlugin::plugin_t< int >** (p. [1149](#)).

```
4.109.3.7 process() mha_wave_t * example3_t::process (
    mha_wave_t * signal )
```

Signal processing performed by the plugin.

This plugin multiplies the signal in the selected audio channel by the configured factor.

Parameters

<i>signal</i>	Pointer to the input signal structure.
---------------	--

Returns

Returns a pointer to the input signal structure, with the signal modified by this plugin.
(In-place processing)

4.109.4 Member Data Documentation

4.109.4.1 scale_ch `MHAParser::int_t` `example3_t::scale_ch` [private]

Index of audio channel to scale.

4.109.4.2 factor `MHAParser::float_t` `example3_t::factor` [private]

The scaling factor applied to the selected channel.

4.109.4.3 prepared `MHAParser::int_mon_t` `example3_t::prepared` [private]

Keep Track of the prepare/release calls.

4.109.4.4 patchbay MHAEvents::patchbay_t< example3_t> example3_t::patchbay [private]

The Event connector.

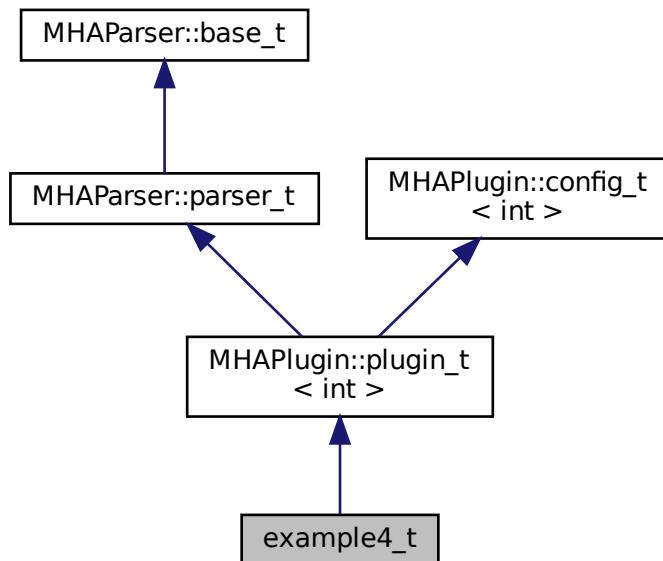
The documentation for this class was generated from the following file:

- **example3.cpp**

4.110 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

4.110.1 Detailed Description

A Plugin class using the spectral signal.

This is the fourth example plugin for the step-by-step tutorial.

4.110.2 Constructor & Destructor Documentation

4.110.2.1 **example4_t()** 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.

4.110.3 Member Function Documentation

4.110.3.1 `on_scale_ch_writeaccess()` void example4_t::on_scale_ch_writeaccess ()
[private]

4.110.3.2 `on_scale_ch_valuechanged()` void example4_t::on_scale_ch_valuechanged ()
[private]

4.110.3.3 `on_scale_ch_readaccess()` void example4_t::on_scale_ch_readaccess ()
[private]

4.110.3.4 `on_prereadaccess()` void example4_t::on_prereadaccess () [private]

4.110.3.5 `prepare()` void example4_t::prepare (
 mhaconfig_t & signal_info) [virtual]

Plugin preparation.

This plugin checks that the input signal has the spectral domain and contains enough channels.

Parameters

<code>signal_info</code>	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. 1148).

```
4.110.3.6 release() void example4_t::release (
    void ) [virtual]
```

Bookkeeping only.

Reimplemented from **MHAPlugIn::plugin_t< int >** (p. 1149).

```
4.110.3.7 process() mha_spec_t * example4_t::process (
    mha_spec_t * signal )
```

Signal processing performed by the plugin.

This plugin multiplies the spectral signal in the selected audio channel by the configured factor.

Parameters

<i>signal</i>	Pointer to the input signal structure.
---------------	--

Returns

Returns a pointer to the input signal structure, with the signal modified by this plugin.
(In-place processing)

4.110.4 Member Data Documentation

```
4.110.4.1 scale_ch MHAParser::int_t example4_t::scale_ch [private]
```

Index of audio channel to scale.

```
4.110.4.2 factor MHAParser::float_t example4_t::factor [private]
```

The scaling factor applied to the selected channel.

4.110.4.3 prepared `MHAParser::int_mon_t example4_t::prepared [private]`

Keep Track of the prepare/release calls.

4.110.4.4 patchbay `MHAEvents::patchbay_t< example4_t> example4_t::patchbay [private]`

The Event connector.

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

- `example4.cpp`

4.111 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`

4.111.1 Constructor & Destructor Documentation**4.111.1.1 example5_t()** `example5_t::example5_t (`
`unsigned int ichannel,`
`unsigned int numchannels,`
`mha_real_t iscale)`**4.111.2 Member Function Documentation**

```
4.111.2.1 process() mha_spec_t * example5_t::process (
    mha_spec_t * spec )
```

4.111.3 Member Data Documentation

4.111.3.1 **channel** unsigned int example5_t::channel [private]

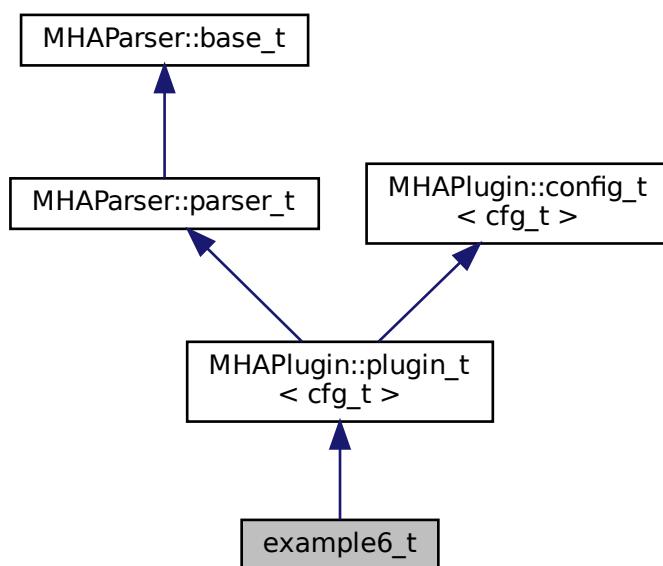
4.111.3.2 **scale** mha_real_t example5_t::scale [private]

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

- **example5.cpp**

4.112 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

4.112.1 Constructor & Destructor Documentation

```
4.112.1.1 example6_t() example6_t::example6_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.112.2 Member Function Documentation

```
4.112.2.1 process() mha_wave_t * example6_t::process (
    mha_wave_t * wave )
```

4.112.2.2 `prepare()` void example6_t::prepare (
 mhaconfig_t & tfcfg) [virtual]

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1148).

4.112.2.3 `update_cfg()` void example6_t::update_cfg () [private]

4.112.3 Member Data Documentation

4.112.3.1 `channel_no` MHAParser::int_t example6_t::channel_no [private]

4.112.3.2 `rmsdb` float example6_t::rmsdb [private]

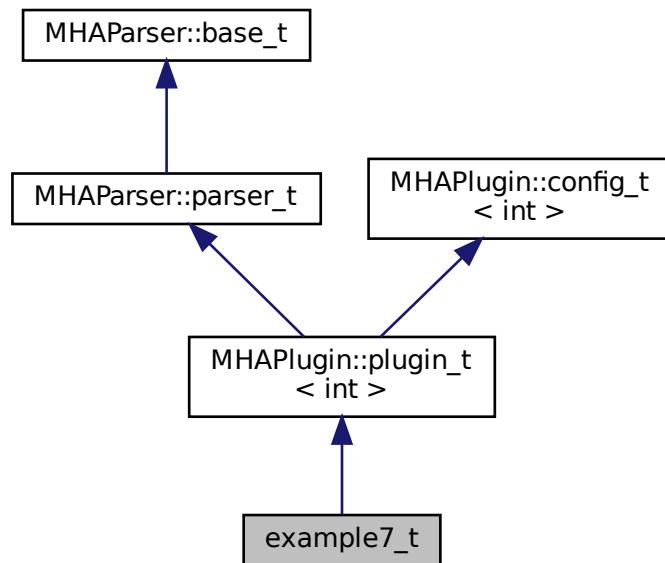
4.112.3.3 `patchbay` MHAEvents::patchbay_t< example6_t > example6_t::patchbay [private]

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

- **example6.cpp**

4.113 example7_t Class Reference

Inheritance diagram for example7_t:



Public Member Functions

- **example7_t (algo_comm_t &, const std::string &, const std::string &)**
- **void release (void)**
- **void prepare (mhaconfig_t &)**
- **mha_wave_t * process (mha_wave_t *)**

Additional Inherited Members

4.113.1 Constructor & Destructor Documentation

4.113.1.1 example7_t() example7_t::example7_t (

```

algo_comm_t & ac,
const std::string & chain_name,
const std::string & algo_name )
  
```

4.113.2 Member Function Documentation

4.113.2.1 release() void example7_t::release (void) [virtual]

Reimplemented from **MHAPlugIn::plugin_t< int >** (p. 1149).

4.113.2.2 prepare() void example7_t::prepare (mhaconfig_t & signal_info) [virtual]

Implements **MHAPlugIn::plugin_t< int >** (p. 1148).

4.113.2.3 process() mha_wave_t * example7_t::process (mha_wave_t * signal)

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

- **example7.hh**
- **example7.cpp**

4.114 expression_t Class Reference

Class for separating a string into a left hand value and a right hand value.

4.114.1 Detailed Description

Class for separating a string into a left hand value and a right hand value.

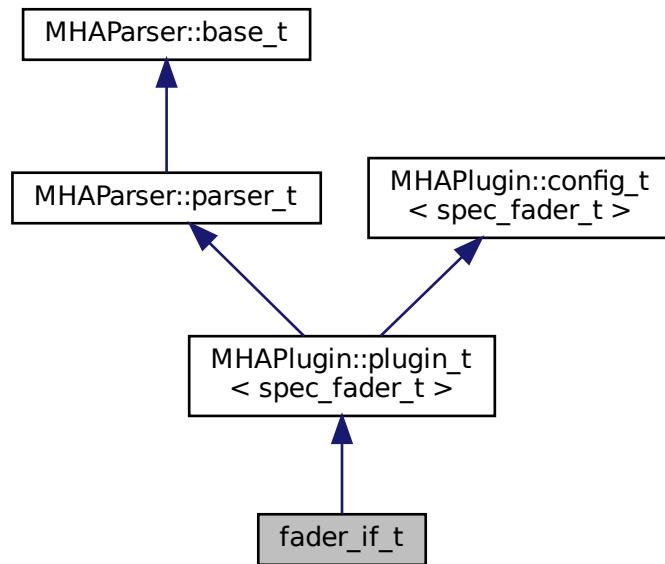
A list of valid operators can be provided. After construction, the class members lval, rval and op contain the appropriate contents.

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

- **mha_parser.cpp**

4.115 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`
- `MHParse::float_t tau`
- `MHParse::vfloat_t newgains`
- `mha_real_t * actgains`

Additional Inherited Members

4.115.1 Constructor & Destructor Documentation

```
4.115.1.1 fader_if_t() fader_if_t::fader_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.115.2 Member Function Documentation

```
4.115.2.1 process() mha_spec_t * fader_if_t::process (
    mha_spec_t * s )
```

```
4.115.2.2 prepare() void fader_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< spec_fader_t >** (p. 1148).

```
4.115.2.3 update_cfg() void fader_if_t::update_cfg ( ) [private]
```

4.115.3 Member Data Documentation

```
4.115.3.1 patchbay MHAEvents::patchbay_t< fader_if_t > fader_if_t::patchbay [private]
```

4.115.3.2 tau `MHAParser::float_t fader_if_t::tau` [private]

4.115.3.3 newgains `MHAParser::vfloat_t fader_if_t::newgains` [private]

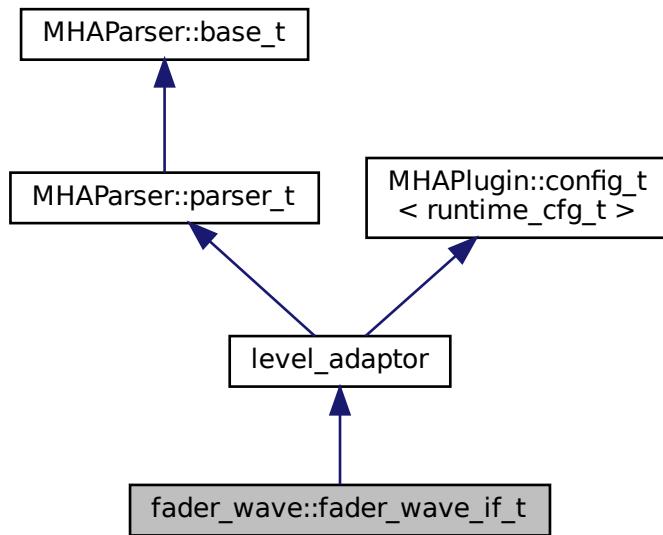
4.115.3.4 actgains `mha_real_t* fader_if_t::actgains` [private]

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

- `fader_spec.cpp`

4.116 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

4.116.1 Constructor & Destructor Documentation

```
4.116.1.1 fader_wave_if_t() fader_wave::fader_wave_if_t::fader_wave_if_t (
    algo_comm_t iac,
    const char * ,
    const char * )
```

4.116.2 Member Function Documentation

```
4.116.2.1 process() mha_wave_t * fader_wave::fader_wave_if_t::process (
    mha_wave_t * s )
```

```
4.116.2.2 prepare() void fader_wave::fader_wave_if_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements `MHAPlugin::plugin_t< runtime_cfg_t >` (p. [1148](#)).

4.116.2.3 `release()` void fader_wave::fader_wave_if_t::release () [virtual]

Reimplemented from **MHAPlugin::plugin_t< runtime_cfg_t >** (p. 1149).

4.116.2.4 `set_level()` void fader_wave::fader_wave_if_t::set_level () [private]

4.116.3 Member Data Documentation

4.116.3.1 `gain` MHAParser::vfloat_t fader_wave::fader_wave_if_t::gain [private]

4.116.3.2 `ramplen` MHAParser::float_t fader_wave::fader_wave_if_t::ramplen [private]

4.116.3.3 `patchbay` MHAEvents::patchbay_t< fader_wave_if_t > fader_wave::fader_wave_if_t::patchbay [private]

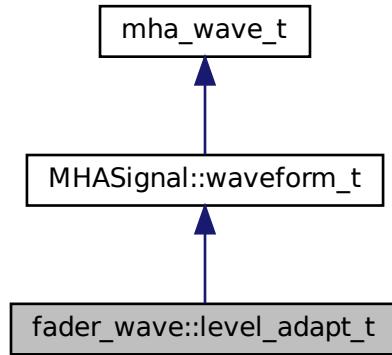
4.116.3.4 `prepared` bool fader_wave::fader_wave_if_t::prepared [private]

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

- **fader_wave.cpp**

4.117 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

4.117.1 Constructor & Destructor Documentation

4.117.1.1 level_adapt_t() fader_wave::level_adapt_t::level_adapt_t (

```
mhaconfig_t cf,
mha_real_t adapt_len,
std::vector< float > l_new,
std::vector< float > l_old )
```

4.117.2 Member Function Documentation

4.117.2.1 update_frame() void fader_wave::level_adapt_t::update_frame ()

4.117.2.2 get_level() std::vector<float> fader_wave::level_adapt_t::get_level ()
const [inline]

4.117.2.3 can_update() bool fader_wave::level_adapt_t::can_update () const [inline]

4.117.3 Member Data Documentation

4.117.3.1 ilen unsigned int fader_wave::level_adapt_t::ilen [private]

4.117.3.2 pos unsigned int fader_wave::level_adapt_t::pos [private]

4.117.3.3 wnd MHAWindow::fun_t fader_wave::level_adapt_t::wnd [private]

4.117.3.4 `l_new` std::vector<float> fader_wave::level_adapt_t::l_new [private]

4.117.3.5 `l_old` std::vector<float> fader_wave::level_adapt_t::l_old [private]

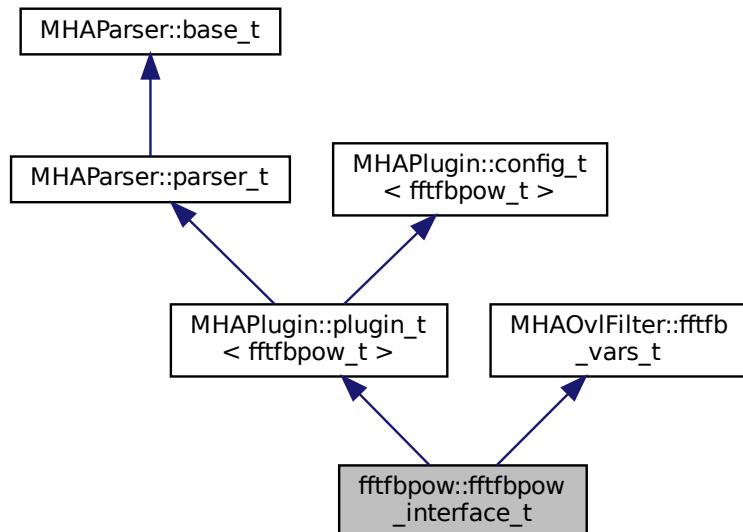
The documentation for this class was generated from the following file:

- `fader_wave.cpp`

4.118 fftfbpow::fftfbpow_interface_t Class Reference

Interface class for fftfbpow plugin.

Inheritance diagram for fftfbpow::fftfbpow_interface_t:



Public Member Functions

- `fftfbpow_interface_t (const algo_comm_t & ac, const std::string &, const std::string &algo_name)`
Constructor with standard MHA constructor parameters.
- `void prepare (mhaconfig_t &tf)`
Standard MHA plugin prepare function.
- `mha_spec_t * process (mha_spec_t *s)`
Standard MHA plugin process fct.

Private Member Functions

- void **update_cfg ()**
Constructs new runtime configuration in thread-safe manner.

Private Attributes

- std::string **name**
Configured name of this plugin instance.
- **MHAEvents::patchbay_t< fftfbpow_interface_t > patchbay**
Patchbay to connect to MHA configuration interface.

Additional Inherited Members

4.118.1 Detailed Description

Interface class for fftfbpow plugin.

4.118.2 Constructor & Destructor Documentation

```
4.118.2.1 fftfbpow_interface_t() fftfbpow::fftfbpow_interface_t::fftfbpow_interface_t(
    const algo_comm_t & ac,
    const std::string & ,
    const std::string & algo_name )
```

Constructor with standard MHA constructor parameters.

Parameters

<i>ac</i>	Handle to algorithm communication variable space
<i>algo_name</i>	Configured name of this plugin instance

4.118.3 Member Function Documentation

4.118.3.1 `prepare()` `void fftfbpow::fftfbpow_interface_t::prepare (mhaconfig_t & tf) [virtual]`

Standard MHA plugin prepare function.

Ensures that the input is in the frequency domain, calls **update_cfg()** (p. 495) and inserts fbpow into the AC space.

Parameters

<code>tf</code>	Incoming mha configuration structure, contains information about input signal
-----------------	---

Implements **MHAPlugin::plugin_t< fftfbpow_t >** (p. 1148).

4.118.3.2 `process()` `mha_spec_t * fftfbpow::fftfbpow_interface_t::process (mha_spec_t * s)`

Standard MHA plugin process fct.

Polls new config and calls **process()** (p. 495) of the runtime configuration.

Parameters

<code>s</code>	Input spectrum
----------------	----------------

Returns

Unchanged input spectrum

4.118.3.3 `update_cfg()` `void fftfbpow::fftfbpow_interface_t::update_cfg () [private]`

Constructs new runtime configuration in thread-safe manner.

4.118.4 Member Data Documentation

4.118.4.1 **name** std::string fftfbpow::fftfbpow_interface_t::name [private]

Configured name of this plugin instance.

4.118.4.2 **patchbay** MHAEvents::patchbay_t< fftfbpow_interface_t> fftfbpow::fftfbpow<->_interface_t::patchbay [private]

Patchbay to connect to MHA configuration interface.

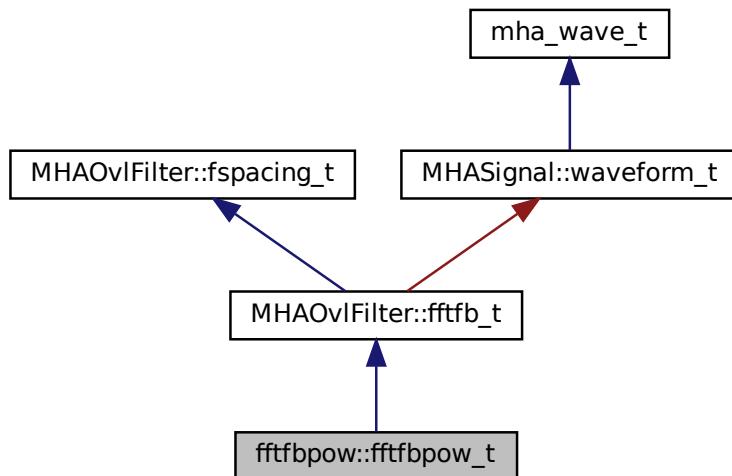
The documentation for this class was generated from the following file:

- **fftfbpow.cpp**

4.119 fftfbpow::fftfbpow_t Class Reference

Run time configuration for the fftfbpow plugin.

Inheritance diagram for fftfbpow::fftfbpow_t:



Public Member Functions

- **fftfbpow_t** (**MHAOvIFilter::fftfb_vars_t** &vars, unsigned int nch, unsigned int nfft, **mha_real_t** fs, **algo_comm_t** ac, std::string name)
Constructor of the run time configuration.

Public Attributes

- **MHA_AC::waveform_t fbpow**

AC variable containing the estimated power in each frequency band.

Additional Inherited Members

4.119.1 Detailed Description

Run time configuration for the fftfbpow plugin.

4.119.2 Constructor & Destructor Documentation

```
4.119.2.1 fftfbpow_t() fftfbpow::fftfbpow_t::fftfbpow_t (
    MHAOvlFilter::fftfb_vars_t & vars,
    unsigned int nch,
    unsigned int nfft,
    mha_real_t fs,
    algo_comm_t ac,
    std::string name )
```

Constructor of the run time configuration.

Parameters

<i>vars</i>	Set of configuration variables for FFT-based overlapping filters
<i>nch</i>	Number of audio input channels
<i>nfft</i>	Length of FFT
<i>fs</i>	Sampling rate
<i>ac</i>	AC space
<i>name</i>	Configured name of plugin interface, used as prefix for AC variable names

4.119.3 Member Data Documentation

4.119.3.1 fbpow MHA_AC::waveform_t fftfbpow::fftfbpow_t::fbpow

AC variable containing the estimated power in each frequency band.

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

- **fftfbpow.cpp**

4.120 fftfilter::fftfilter_t Class Reference**Public Member Functions**

- **fftfilter_t** (const **MHAParser::mfloat_t** &irs, const unsigned int & **fragsize**, const unsigned int & **channels**, const unsigned int & **fftlen**)
Initialization of new run-time configuration from channel-specific impulse responses.
- **mha_wave_t * process** (**mha_wave_t** *)

Private Attributes

- unsigned int **irslen**
Length of the longest impulse response applied.
- unsigned int **fragsize**
The block size (samples per channel) for waveform audio data.
- unsigned int **fftlen**
FFT length used for filtering.
- unsigned int **channels**
Number of prepared audio channels processed by this MHA plugin.
- **MHAFilter::fftfilter_t fftfilt**
The filter object.

4.120.1 Detailed Description

Run-time configuration class for the fftfilter MHA plugin.

4.120.2 Constructor & Destructor Documentation**4.120.2.1 fftfilter_t() fftfilter::fftfilter_t::fftfilter_t (**

```
const MHAParser::mfloat_t & irs,
const unsigned int & fragsize_,
const unsigned int & channels_,
const unsigned int & fftlen_ )
```

Initialization of new run-time configuration from channel-specific impulse responses.

Parameters

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

4.120.3 Member Function Documentation

4.120.3.1 process() `mha_wave_t * fftfilter::fftfilter_t::process (mha_wave_t * s)`

Let fftfiler object handle the filtering

4.120.4 Member Data Documentation

4.120.4.1 irslen `unsigned int fftfilter::fftfilter_t::irslen [private]`

Length of the longest impulse response applied.

4.120.4.2 fragsize `unsigned int fftfilter::fftfilter_t::fragsize [private]`

The block size (samples per channel) for waveform audio data.

4.120.4.3 fftlen `unsigned int fftfilter::fftfilter_t::fftlens [private]`

FFT length used for filtering.

4.120.4.4 channels `unsigned int fftfilter::fftfilter_t::channels [private]`

Number of prepared audio channels processed by this MHA plugin.

4.120.4.5 fftfilt `MHAFilter::fftfilter_t fftfilter::fftfilter_t::fftfilt [private]`

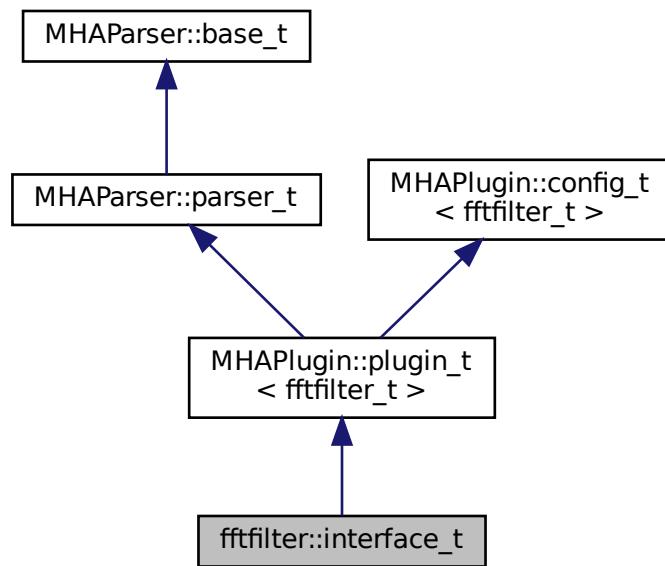
The filter object.

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

- `fftfilter.cpp`

4.121 fftfilter::interface_t Class Reference

Inheritance diagram for fftfilter::interface_t:



Public Member Functions

- `interface_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 ()**

Private Attributes

- MHAParser::mfloat_t **irs**
- MHAParser::int_t **fftlen**
- MHAParser::int_mon_t **fftlen_final**
- MHAEvents::patchbay_t< interface_t > **patchbay**

Additional Inherited Members

4.121.1 Detailed Description

Implements the MHA plugin interface for FFTFilter

4.121.2 Constructor & Destructor Documentation

```
4.121.2.1 interface_t() fftfilter::interface_t::interface_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.121.3 Member Function Documentation

```
4.121.3.1 process() mha_wave_t * fftfilter::interface_t::process (
    mha_wave_t * s )
```

4.121.3.2 `prepare()` void fftfilter::interface_t::prepare (mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< fftfilter_t >** (p. 1148).

4.121.3.3 `update()` void fftfilter::interface_t::update () [private]

4.121.4 Member Data Documentation

4.121.4.1 `irs` MHAParser::mfloat_t fftfilter::interface_t::irs [private]

4.121.4.2 `fftlen` MHAParser::int_t fftfilter::interface_t::fftlen [private]

4.121.4.3 `fftlen_final` MHAParser::int_mon_t fftfilter::interface_t::fftlen_final [private]

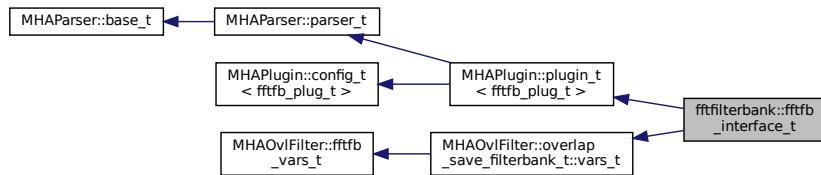
4.121.4.4 `patchbay` MHAEvents::patchbay_t< interface_t > fftfilter::interface_t::patchbay [private]

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

- **fftfilter.cpp**

4.122 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

4.122.1 Constructor & Destructor Documentation

4.122.1.1 fftfb_interface_t() 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

4.122.2 Member Function Documentation

4.122.2.1 `prepare()` `void fftfilterbank::fftfb_interface_t::prepare (mhaconfig_t & tf) [virtual]`

Prepare all variables for processing.

In this function, all variables are initialised and the filter shapes for each band are calculated. The filter shapes $W(f)$ are defined as

$$W(f) = W(T(S(f))) = W(x), \quad x = T(S(f)) = T(\hat{f}),$$

$W(x)$ 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. 1148).

4.122.2.2 `release()` `void fftfilterbank::fftfb_interface_t::release () [virtual]`

Reimplemented from **MHAPlugIn::plugin_t< fftfb_plug_t >** (p. 1149).

4.122.2.3 process() [1/2] `mha_spec_t * fftfilterbank::fftfb_interface_t::process (mha_spec_t * s)`

4.122.2.4 process() [2/2] `mha_wave_t * fftfilterbank::fftfb_interface_t::process (mha_wave_t * s)`

4.122.2.5 update_cfg() `void fftfilterbank::fftfb_interface_t::update_cfg ()` [private]

4.122.3 Member Data Documentation

4.122.3.1 return_imag `MHAParser::bool_t fftfilterbank::fftfb_interface_t::return_imag` [private]

4.122.3.2 patchbay `MHAEvents::patchbay_t< fftfb_interface_t> fftfilterbank::fftfb_interface_t::patchbay` [private]

4.122.3.3 nchannels `MHA_AC::int_t fftfilterbank::fftfb_interface_t::nchannels` [private]

4.122.3.4 algo `std::string fftfilterbank::fftfb_interface_t::algo` [private]

4.122.3.5 prepared `bool fftfilterbank::fftfb_interface_t::prepared` [private]

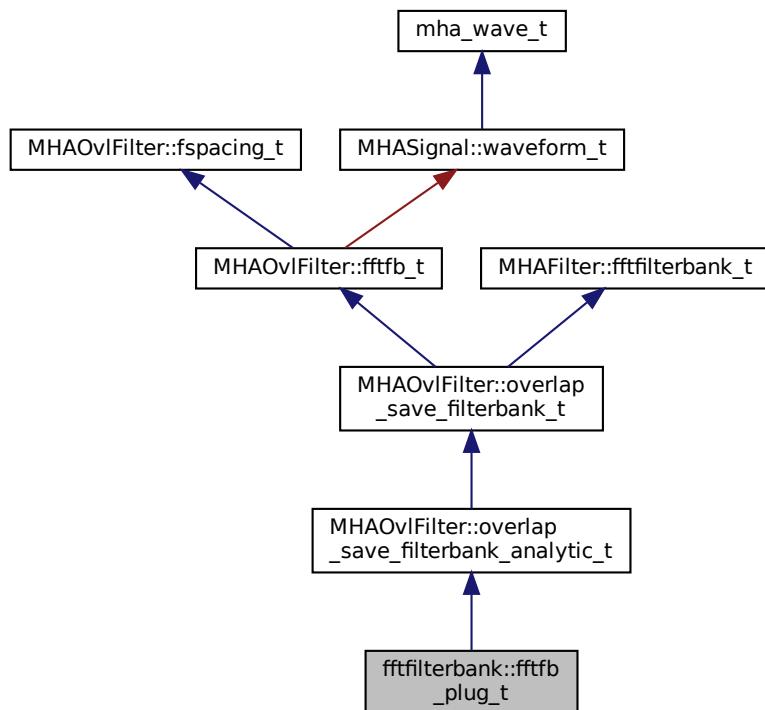
4.122.3.6 nbands `unsigned int fftfilterbank::fftfb_interface_t::nbands [private]`

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

- `fftfilterbank.cpp`

4.123 `fftfilterbank::fftfb_plug_t` Class Reference

Inheritance diagram for `fftfilterbank::fftfb_plug_t`:



Public Member Functions

- `fftfb_plug_t (MHAOvIFilter::overlap_save_filterbank_t::vars_t &, mhaconfig_t chcfg, algo_comm_t ac, std::string alg, bool return_imag)`
- `mha_spec_t * process (mha_spec_t *)`
- `mha_wave_t * process (mha_wave_t *)`
- `void insert ()`

Private Attributes

- `MHAOvlFilter::fftfb_ac_info_t fb_acinfo`
- `MHASignal::spectrum_t s_out`
- `MHA_AC::waveform_t imag`
- `bool return_imag_`

Additional Inherited Members

4.123.1 Constructor & Destructor Documentation

```
4.123.1.1 fftfb_plug_t() fftfilterbank::fftfb_plug_t::fftfb_plug_t (
    MHAOvlFilter::overlap_save_filterbank_t::vars_t & vars,
    mhaconfig_t chcfg,
    algo_comm_t ac,
    std::string alg,
    bool return_imag )
```

4.123.2 Member Function Documentation

```
4.123.2.1 process() [1/2] mha_spec_t * fftfilterbank::fftfb_plug_t::process (
    mha_spec_t * s )
```

```
4.123.2.2 process() [2/2] mha_wave_t * fftfilterbank::fftfb_plug_t::process (
    mha_wave_t * s )
```

```
4.123.2.3 insert() void fftfilterbank::fftfb_plug_t::insert ( )
```

4.123.3 Member Data Documentation

4.123.3.1 `fb_acinfo` `MHAOvlFilter::fftfb_ac_info_t fftfilterbank::fftfb_plug_t::fb_acinfo` [private]

4.123.3.2 `s_out` `MHASignal::spectrum_t fftfilterbank::fftfb_plug_t::s_out` [private]

4.123.3.3 `imag` `MHA_AC::waveform_t fftfilterbank::fftfb_plug_t::imag` [private]

4.123.3.4 `return_imag_` `bool fftfilterbank::fftfb_plug_t::return_imag_` [private]

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

- `fftfilterbank.cpp`

4.124 `fshift::fshift_config_t` Class Reference

`fshift` runtime config class

Public Member Functions

- **`fshift_config_t (fshift_t const *const plug)`**
C'tor of the `fshift` plugin runtime configuration class.
- **`~fshift_config_t ()=default`**
- **`mha_spec_t * process (mha_spec_t *)`**

Private Attributes

- const unsigned int **kmin**
FFT bin corresponding to fmin.
- const unsigned **kmax**
FFT bin corresponding to fmax.
- const int **df**
Frequency shift expressed in FFT bins.
- const **mha_complex_t delta_phi**
Phase advance per fft frame.
- **mha_complex_t delta_phi_total**
Sum of all phase advances.

4.124.1 Detailed Description

fshift runtime config class

4.124.2 Constructor & Destructor Documentation

4.124.2.1 **fshift_config_t()** fshift::fshift_config_t::fshift_config_t (

fshift_t const *const *plug*) [explicit]

C'tor of the fshift plugin runtime configuration class.

Parameters

<i>plug</i>	ptr to the plugin interface class. Configuration information is given this way to keep the argument list small.
-------------	---

4.124.2.2 ~**fshift_config_t()** fshift::fshift_config_t::~fshift_config_t () [default]

4.124.3 Member Function Documentation

4.124.3.1 process() `mha_spec_t * fshift::fshift_config_t::process (mha_spec_t * in)`

4.124.4 Member Data Documentation

4.124.4.1 kmin `const unsigned int fshift::fshift_config_t::kmin [private]`

FFT bin corresponding to fmin.

4.124.4.2 kmax `const unsigned fshift::fshift_config_t::kmax [private]`

FFT bin corresponding to fmax.

4.124.4.3 df `const int fshift::fshift_config_t::df [private]`

Frequency shift expressed in FFT bins.

4.124.4.4 delta_phi `const mha_complex_t fshift::fshift_config_t::delta_phi [private]`

Phase advance per fft frame.

4.124.4.5 delta_phi_total `mha_complex_t fshift::fshift_config_t::delta_phi_total [private]`

Sum of all phase advances.

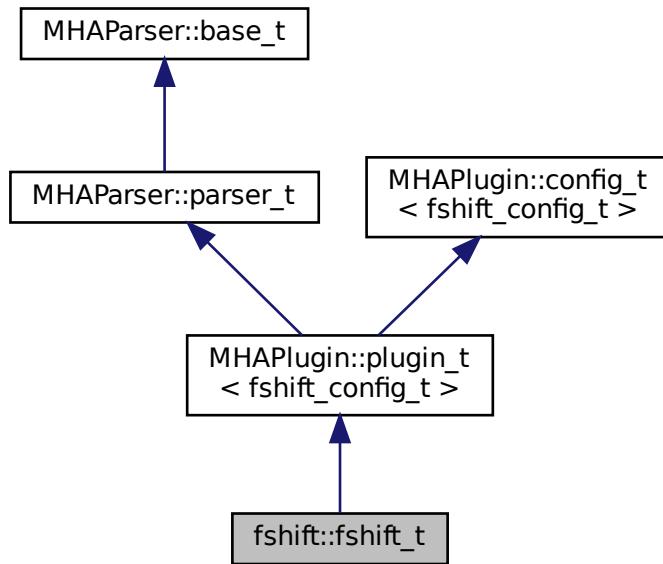
The documentation for this class was generated from the following files:

- **fshift.hh**
- **fshift.cpp**

4.125 fshift::fshift_t Class Reference

fshift plugin interface class

Inheritance diagram for fshift::fshift_t:



Public Member Functions

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

Private Member Functions

- **`void update_cfg ()`**

Private Attributes

- **MHAEvents::patchbay_t< fshift_t > patchbay**
patch bay for connecting configuration parser events with local member functions:
- **MHAParser::float_t m_fmin**
- **MHAParser::float_t m_fmax**
upper boundary for frequency shifter
- **MHAParser::float_t m_df**
Shift frequency in Hz.

Additional Inherited Members**4.125.1 Detailed Description**

fshift plugin interface class

4.125.2 Constructor & Destructor Documentation

4.125.2.1 fshift_t() `fshift::fshift_t::fshift_t (`
`algo_comm_t & ac,`
`const std::string & chain_name,`
`const std::string & algo_name)`

Constructs our plugin.

4.125.2.2 ~fshift_t() `fshift::fshift_t::~fshift_t ()`

4.125.3 Member Function Documentation

4.125.3.1 process() `mha_spec_t * fshift::fshift_t::process (`
`mha_spec_t * signal)`

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

4.125.3.2 prepare() `void fshift::fshift_t::prepare (`
`mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<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< fshift_config_t >** (p. [1148](#)).

4.125.3.3 release() `void fshift::fshift_t::release (void) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< fshift_config_t >** (p. [1149](#)).

4.125.3.4 fmin() `mha_real_t fshift::fshift_t::fmin () const [inline]`

4.125.3.5 fmax() `mha_real_t fshift::fshift_t::fmax () const [inline]`

4.125.3.6 df() `mha_real_t fshift::fshift_t::df () const [inline]`

4.125.3.7 update_cfg() `void fshift::fshift_t::update_cfg () [private]`

4.125.4 Member Data Documentation

4.125.4.1 patchbay `MHAEvents::patchbay_t< fshift_t > fshift::fshift_t::patchbay [private]`

patch bay for connecting configuration parser events with local member functions:

4.125.4.2 m_fmin `MHAParser::float_t fshift::fshift_t::m_fmin` [private]

4.125.4.3 m_fmax `MHAParser::float_t fshift::fshift_t::m_fmax` [private]

upper boundary for frequency shifter

4.125.4.4 m_df `MHAParser::float_t fshift::fshift_t::m_df` [private]

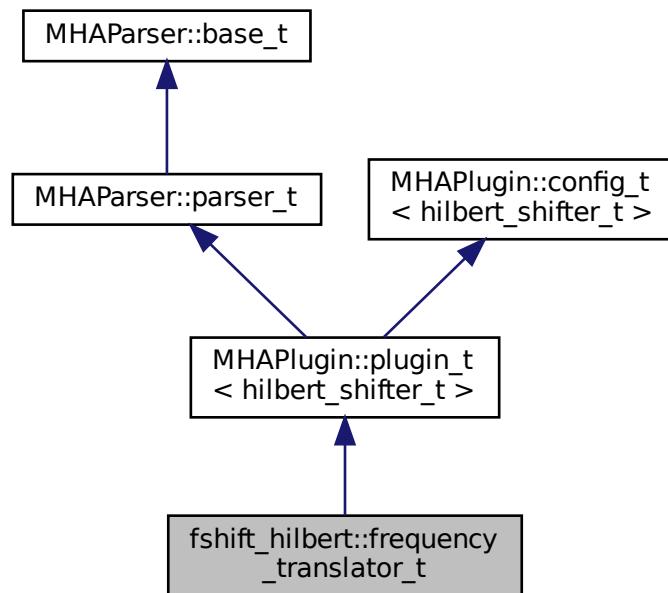
Shift frequency in Hz.

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

- `fshift.hh`
- `fshift.cpp`

4.126 fshift_hilbert::frequency_translator_t Class Reference

Inheritance diagram for `fshift_hilbert::frequency_translator_t`:



Public Member Functions

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

Private Member Functions

- `void update ()`

Private Attributes

- `MHAEvents::patchbay_t< frequency_translator_t > patchbay`
- `MHAParser::vfloat_t df`
Vector containing the shift frequencies in Hz.
- `MHAParser::float_t fmin`
Lower boundary for frequency shifter.
- `MHAParser::float_t fmax`
Upper boundary for frequency shifter.
- `MHAParser::int_t irslen`
Maximum length of cut off filter response.
- `MHAParser::kw_t phasemode`
Mode of gain smoothing.

Additional Inherited Members

4.126.1 Constructor & Destructor Documentation

4.126.1.1 frequency_translator_t() `fshift_hilbert::frequency_translator_t::frequency_translator_t (`

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

4.126.2 Member Function Documentation

4.126.2.1 process() `mha_spec_t * fshift_hilbert::frequency_translator_t::process (mha_spec_t * s)`

4.126.2.2 prepare() `void fshift_hilbert::frequency_translator_t::prepare (mhaconfig_t & tf) [virtual]`

Implements **MHAPlugIn::plugin_t< hilbert_shifter_t >** (p. 1148).

4.126.2.3 release() `void fshift_hilbert::frequency_translator_t::release () [virtual]`

Reimplemented from **MHAPlugIn::plugin_t< hilbert_shifter_t >** (p. 1149).

4.126.2.4 update() `void fshift_hilbert::frequency_translator_t::update () [private]`

4.126.3 Member Data Documentation

4.126.3.1 patchbay `MHAEvents::patchbay_t< frequency_translator_t > fshift_hilbert< ::frequency_translator_t::patchbay [private]`

4.126.3.2 df `MHAParser::vfloat_t fshift_hilbert::frequency_translator_t::df [private]`

Vector containing the shift frequencies in Hz.

4.126.3.3 fmin `MHAParser::float_t fshift_hilbert::frequency_translator_t::fmin [private]`

Lower boundary for frequency shifter.

4.126.3.4 fmax `MHAParser::float_t fshift_hilbert::frequency_translator_t::fmax`
[private]

Upper boundary for frequency shifter.

4.126.3.5 irslen `MHAParser::int_t fshift_hilbert::frequency_translator_t::irslen`
[private]

Maximum length of cut off filter response.

4.126.3.6 phasemode `MHAParser::kw_t fshift_hilbert::frequency_translator_t::phasemode`
[private]

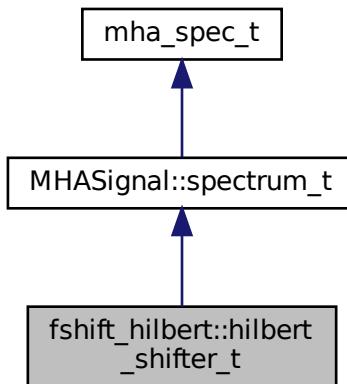
Mode of gain smoothing.

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

- `fshift_hilbert.cpp`

4.127 fshift_hilbert::hilbert_shifter_t Class Reference

Inheritance diagram for fshift_hilbert::hilbert_shifter_t:



Public Member Functions

- **hilbert_shifter_t** (unsigned int fftlen, unsigned int **channels**, **mha_real_t** srate, unsigned int **kmin**, unsigned int **kmax**, std::vector< **mha_real_t** > dphi, unsigned int **frameshift**, unsigned int maxirslen, unsigned int phasemode)
- **~hilbert_shifter_t ()**
- void **process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::spectrum_t fullspec**
Part of the spectrum to be frequency shifted.
- **MHASignal::spectrum_t analytic**
Analytic signal, defined as $a(t)=x(t)+i\cdot H(x(t))$
- **MHASignal::waveform_t shifted**
The frequency shifted signal in the time domain.
- **MHASignal::spectrum_t mixw_shift**
Helper variable containing the coefficients used to split the spectrum.
- **MHASignal::spectrum_t mixw_ref**
Helper variable containing the coefficients used to split the spectrum.
- fftw_plan **plan_spec2analytic**
FFT plan for the transformation of fullspec into the time domain.
- **mha_fft_t mhafft**
MHA wrapper object for fftw.
- **MHASignal::waveform_t df**
Vector holding one delta f value for every channel.
- unsigned int **kmin**
FFT frame that corresponds to f_min.
- unsigned int **kmax**
FFT frame that corresponds to f_max.
- unsigned int **frameshift**
Total phase advance within one fragment.
- std::vector< **mha_complex_t** > **delta_phi**
Phase advance per frame.
- std::vector< **mha_complex_t** > **delta_phi_total**
Sum of all phase advances.

Additional Inherited Members

4.127.1 Constructor & Destructor Documentation

```
4.127.1.1 hilbert_shifter_t() fshift_hilbert::hilbert_shifter_t::hilbert_shifter_t (
    unsigned int fftlen,
    unsigned int channels,
    mha_real_t srate,
    unsigned int kmin,
    unsigned int kmax,
    std::vector< mha_real_t > dphi,
    unsigned int frameshift,
    unsigned int maxirslen,
    unsigned int phasemode )
```

```
4.127.1.2 ~hilbert_shifter_t() fshift_hilbert::hilbert_shifter_t::~hilbert_shifter_t ( )
```

4.127.2 Member Function Documentation

```
4.127.2.1 process() void fshift_hilbert::hilbert_shifter_t::process (
    mha_spec_t * s )
```

4.127.3 Member Data Documentation

```
4.127.3.1 fullspec MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::fullspec
[private]
```

Part of the spectrum to be frequency shifted.

```
4.127.3.2 analytic MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::analytic
[private]
```

Analytic signal, defined as $a(t) = x(t) + i \cdot H(x(t))$

4.127.3.3 shifted `MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::shifted` [private]

The frequency shifted signal in the time domain.

4.127.3.4 mixw_shift `MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::mixw_shift` [private]

Helper variable containing the coefficients used to split the spectrum.

Contains 1 for every fft bin to be frequency shifted, 0 for all others

4.127.3.5 mixw_ref `MHASignal::spectrum_t fshift_hilbert::hilbert_shifter_t::mixw_ref` [private]

Helper variable containing the coefficients used to split the spectrum.

Contains 0 for every fft bin to be frequency shifted, 1 for all others

4.127.3.6 plan_spec2analytic `fftw_plan fshift_hilbert::hilbert_shifter_t::plan_spec2analytic` [private]

FFT plan for the transformation of fullspec into the time domain.

4.127.3.7 mhafft `mha_fft_t fshift_hilbert::hilbert_shifter_t::mhafft` [private]

MHA wrapper object for fftw.

4.127.3.8 df `MHASignal::waveform_t fshift_hilbert::hilbert_shifter_t::df` [private]

Vector holding one delta f value for every channel.

4.127.3.9 kmin unsigned int fshift_hilbert::hilbert_shifter_t::kmin [private]

FFT frame that corresponds to f_min.

4.127.3.10 kmax unsigned int fshift_hilbert::hilbert_shifter_t::kmax [private]

FFT frame that corresponds to f_max.

4.127.3.11 frameshift unsigned int fshift_hilbert::hilbert_shifter_t::frameshift [private]

Total phase advance within one fragment.

4.127.3.12 delta_phi std::vector< mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi [private]

Phase advance per frame.

4.127.3.13 delta_phi_total std::vector< mha_complex_t> fshift_hilbert::hilbert_shifter_t::delta_phi_total [private]

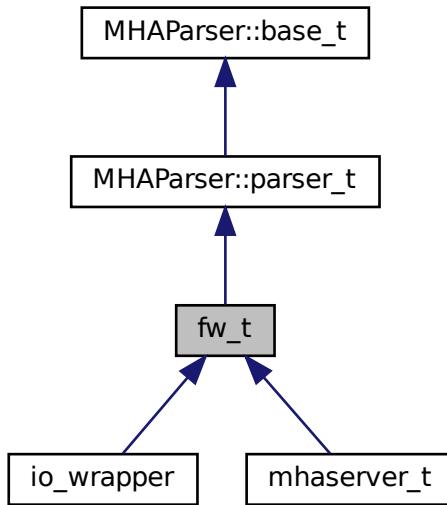
Sum of all phase advances.

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

- **fshift_hilbert.cpp**

4.128 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

- `enum state_t {
 fw_unprepared, fw_stopped, fw_starting, fw_running,
 fw_stopping, fw_exiting }`

Private Member Functions

- void **prepare** ()
preparation for processing
- void **start** ()
start of processing
- void **stop** ()
stop/pause of processing
- void **release** ()
release of IO device
- void **quit** ()
controlled quit
- void **stopped** (int, int)
- void **started** ()
- int **process** (mha_wave_t *, mha_wave_t **)
- void **exec_fw_command** ()
- void **load_proc_lib** ()
- void **load_io_lib** ()
- void **fw_sleep_cmd** ()
- void **fw_until_cmd** ()
- void **get_input_signal_dimension** ()
- void **async_read** ()
- void **async_poll_msg** ()
- void **get_parserstate** ()

Static Private Member Functions

- static void **stopped** (void *h, int proc_err, int io_err)
- static void **started** (void *h)
- static int **process** (void *h, mha_wave_t *sIn, mha_wave_t **sOut)

Private Attributes

- fw_vars_t **prepare_vars**
- MHParse::int_mon_t **nchannels_out**
- MHParse::string_t **proc_name**
- MHParse::string_t **io_name**
- MHParse::bool_t **exit_on_stop**
- MHParse::int_t **fw_sleep**
- MHParse::string_t **fw_until**
- MHParse::kw_t **fw_cmd**
- MHParse::string_mon_t **parserstate**
- MHParse::string_t **errorlog**
- MHParse::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::mhaplugloader_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

4.128.1 Member Enumeration Documentation

4.128.1.1 **state_t** enum **fw_t::state_t** [private]

Enumerator

fw_unprepared	
fw_stopped	
fw_starting	
fw_running	
fw_stopping	
fw_exiting	

4.128.2 Constructor & Destructor Documentation

4.128.2.1 **fw_t()** **fw_t::fw_t** ()

4.128.2.2 ~fw_t() fw_t::~fw_t ()**4.128.3 Member Function Documentation****4.128.3.1 exit_request()** bool fw_t::exit_request () const [inline]**4.128.3.2 prepare()** void fw_t::prepare () [private]

preparation for processing

4.128.3.3 start() void fw_t::start () [private]

start of processing

4.128.3.4 stop() void fw_t::stop () [private]

stop/pause of processing

4.128.3.5 release() void fw_t::release () [private]

release of IO device

4.128.3.6 quit() void fw_t::quit () [private]

controlled quit

4.128.3.7 stopped() [1/2] static void fw_t::stopped (void * *h*, int *proc_err*, int *io_err*) [inline], [static], [private]

4.128.3.8 started() [1/2] static void fw_t::started (void * *h*) [inline], [static], [private]

4.128.3.9 process() [1/2] static int fw_t::process (void * *h*, mha_wave_t * *sIn*, mha_wave_t ** *sOut*) [inline], [static], [private]

4.128.3.10 stopped() [2/2] void fw_t::stopped (int *proc_err*, int *io_err*) [private]

4.128.3.11 started() [2/2] void fw_t::started () [private]

4.128.3.12 process() [2/2] int fw_t::process (mha_wave_t * *s_in*, mha_wave_t ** *s_out*) [private]

4.128.3.13 exec_fw_command() void fw_t::exec_fw_command () [private]

4.128.3.14 load_proc_lib() void fw_t::load_proc_lib () [private]

4.128.3.15 load_io_lib() void fw_t::load_io_lib () [private]

4.128.3.16 fw_sleep_cmd() void fw_t::fw_sleep_cmd () [private]

4.128.3.17 fw_until_cmd() void fw_t::fw_until_cmd () [private]

4.128.3.18 get_input_signal_dimension() void fw_t::get_input_signal_dimension () [private]

4.128.3.19 async_read() void fw_t::async_read () [inline], [private]

4.128.3.20 async_poll_msg() void fw_t::async_poll_msg () [private]

4.128.3.21 get_parserstate() void fw_t::get_parserstate () [private]

4.128.4 Member Data Documentation

4.128.4.1 `prepare_vars` `MHAParser::fw_vars_t fw_t::prepare_vars` [private]

4.128.4.2 `nchannels_out` `MHAParser::int_mon_t fw_t::nchannels_out` [private]

4.128.4.3 `proc_name` `MHAParser::string_t fw_t::proc_name` [private]

4.128.4.4 `io_name` `MHAParser::string_t fw_t::io_name` [private]

4.128.4.5 `exit_on_stop` `MHAParser::bool_t fw_t::exit_on_stop` [private]

4.128.4.6 `fw_sleep` `MHAParser::int_t fw_t::fw_sleep` [private]

4.128.4.7 `fw_until` `MHAParser::string_t fw_t::fw_until` [private]

4.128.4.8 `fw_cmd` `MHAParser::kw_t fw_t::fw_cmd` [private]

4.128.4.9 `parserstate` `MHAParser::string_mon_t fw_t::parserstate` [private]

4.128.4.10 errorlog `MHAParser::string_t fw_t::errorlog [private]`

4.128.4.11 fatallog `MHAParser::string_t fw_t::fatallog [private]`

4.128.4.12 plugins `MHAParser::vstring_t fw_t::plugins [private]`

4.128.4.13 plugin_paths `MHAParser::vstring_t fw_t::plugin_paths [private]`

4.128.4.14 dump_mha `MHAParser::bool_t fw_t::dump_mha [private]`

4.128.4.15 inst_name `MHAParser::string_t fw_t::inst_name [private]`

A variable for naming MHA instances.

4.128.4.16 ac `MHAKernel::algo_comm_class_t fw_t::ac [private]`

4.128.4.17 proc_lib `PluginLoader::mhaplugloader_t* fw_t::proc_lib [private]`

4.128.4.18 io_lib `io_lib_t* fw_t::io_lib [private]`

4.128.4.19 cfin `mhaconfig_t fw_t::cfin` [private]

4.128.4.20 cfout `mhaconfig_t fw_t::cfout` [private]

4.128.4.21 state `state_t fw_t::state` [private]

4.128.4.22 b_exit_request `bool fw_t::b_exit_request` [private]

4.128.4.23 proc_error `int fw_t::proc_error` [protected]

4.128.4.24 io_error `int fw_t::io_error` [protected]

4.128.4.25 proc_error_string `MHAParser::string_mon_t fw_t::proc_error_string` [private]

4.128.4.26 patchbay `MHAEvents::patchbay_t< fw_t> fw_t::patchbay` [private]

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

- `mhafw_lib.h`
- `mhafw_lib.cpp`

4.129 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`

4.129.1 Constructor & Destructor Documentation

4.129.1.1 fw_vars_t() `fw_vars_t::fw_vars_t (`
`MHAParser::parser_t & p) [explicit]`

4.129.2 Member Function Documentation

4.129.2.1 lock_srate_fragsize() `void fw_vars_t::lock_srate_fragsize ()`

4.129.2.2 lock_channels() `void fw_vars_t::lock_channels ()`

4.129.2.3 unlock_srate_fragsize() `void fw_vars_t::unlock_srate_fragsize ()`

4.129.2.4 unlock_channels() `void fw_vars_t::unlock_channels ()`

4.129.3 Member Data Documentation

4.129.3.1 pinchannels `MHAParser::int_t fw_vars_t::pinchannels`

4.129.3.2 pfragmentsize `MHAParser::int_t fw_vars_t::pfragmentsize`

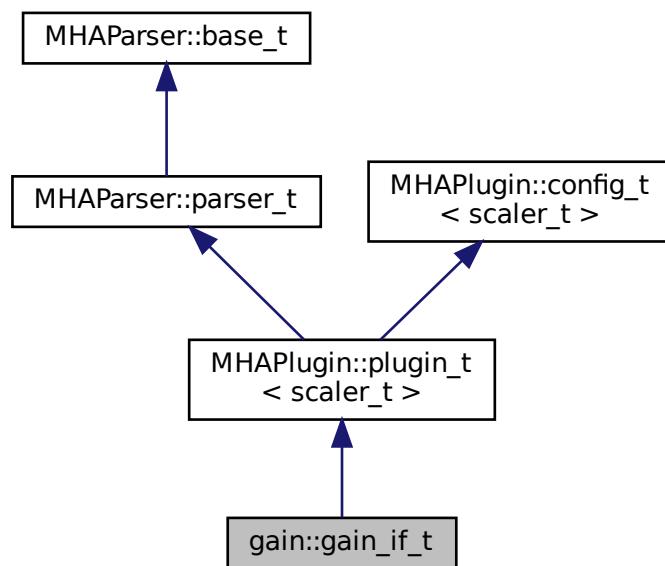
4.129.3.3 psrate `MHAParser::float_t fw_vars_t::psrate`

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

- `mhafw_lib.h`
- `mhafw_lib.cpp`

4.130 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_minmax ()`

Private Attributes

- `MHAEvents::patchbay_t< gain_if_t > patchbay`
- `MHAParser::vfloat_t gains`
- `MHAParser::float_t vmin`
- `MHAParser::float_t vmax`

Additional Inherited Members

4.130.1 Constructor & Destructor Documentation

```
4.130.1.1 gain_if_t() gain::gain_if_t::gain_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.130.2 Member Function Documentation

```
4.130.2.1 process() [1/2] mha_wave_t * gain::gain_if_t::process (
    mha_wave_t * s )
```

4.130.2.2 process() [2/2] `mha_spec_t * gain::gain_if_t::process (mha_spec_t * s)`

4.130.2.3 prepare() `void gain::gain_if_t::prepare (mhaconfig_t & tf) [virtual]`

Implements `MHAPlugIn::plugin_t<scaler_t>` (p. [1148](#)).

4.130.2.4 release() `void gain::gain_if_t::release () [virtual]`

Reimplemented from `MHAPlugIn::plugin_t<scaler_t>` (p. [1149](#)).

4.130.2.5 update_gain() `void gain::gain_if_t::update_gain () [private]`

4.130.2.6 update_minmax() `void gain::gain_if_t::update_minmax () [private]`

4.130.3 Member Data Documentation

4.130.3.1 patchbay `MHAEvents::patchbay_t< gain_if_t > gain::gain_if_t::patchbay` [private]

4.130.3.2 gains `MHAParser::vfloat_t gain::gain_if_t::gains` [private]

4.130.3.3 vmin `MHAParser::float_t gain::gain_if_t::vmin` [private]

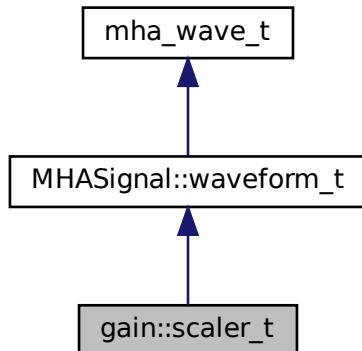
4.130.3.4 vmax `MHAParser::float_t gain::gain_if_t::vmax` [private]

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

- `gain.cpp`

4.131 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

4.131.1 Constructor & Destructor Documentation

```
4.131.1.1 scaler_t() gain::scaler_t::scaler_t (
    const unsigned int & channels,
    const MHAParser::vfloat_t & gains )
```

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

- `gain.cpp`

4.132 gsc_adaptive_stage::gsc_adaptive_stage Class Reference

Public Member Functions

- **gsc_adaptive_stage** (**algo_comm_t** & **ac**, const **mhaconfig_t**, int **lenOldSamps**, bool **doCircularComp**, float **mu**, float **alp**, bool **useVAD**, const std::string &**vadName**_)

Ctor of the rt processing class.
- **~gsc_adaptive_stage** ()=default
- **mha_wave_t * process** (**mha_wave_t** ***wavin**)

Processing callback.

Private Member Functions

- **void insert ()**

Re-insert all AC variables into the AC space.

Private Attributes

- **algo_comm_t ac**

Handle to AC space.
- **unsigned int lenOldSamps**

Number of old samples to buffer.
- **unsigned int lenNewSamps**

Number of new samples.
- **unsigned int bufSize**

Total buffer size.
- **float frac_old**

Fraction of new samples to total buffer size.
- **mha_fft_t mha_fft**

FFT handle.
- **unsigned int nfreq**

Number of frequency bins.
- **unsigned int nchan**

- **Number of channels in input signal.**
- **unsigned int desired_chan**
Channel index containing the desired response.
- **bool doCircularComp**
Whether to compensate for circular convolution.
- **float mu**
Linear coefficient for gradient.
- **float alp**
Autoregressive coefficient for estimating PSD.
- **bool useVAD**
Whether to use VAD.
- **std::string vadName**
Name of VAD AC variable.
- **MHA_AC::waveform_t x**
Buffered input signal.
- **MHA_AC::spectrum_t X**
FFT of the buffered input signal.
- **MHA_AC::spectrum_t W**
Time-varying filter.
- **MHA_AC::spectrum_t Y**
Filter output, frequency domain.
- **MHA_AC::waveform_t y**
Filter output, time domain.
- **MHA_AC::waveform_t d**
Desired response.
- **MHA_AC::waveform_t e**
Error signal, time domain.
- **MHA_AC::spectrum_t E**
Error signal, frequency domain.
- **MHA_AC::spectrum_t E2**
*Error spectrum multiplied by input spectrum: $E2=X*E$.*
- **MHA_AC::waveform_t grad**
Gradient.
- **MHA_AC::spectrum_t Grad**
FT of the gradient.
- **MHA_AC::waveform_t e_out**
Error signal.
- **MHA_AC::waveform_t P**
Signal power estimate.
- **MHA_AC::waveform_t Psum**
Signal power estimate, summed over all channels.

4.132.1 Constructor & Destructor Documentation

4.132.1.1 gsc_adaptive_stage() `gsc_adaptive_stage::gsc_adaptive_stage::gsc_adaptive_stage (algo_comm_t & ac, const mhaconfig_t in_cfg, int lenOldSamps, bool doCircularComp, float mu, float alp, bool useVAD, const std::string & vadName_)`

Ctor of the rt processing class.

Parameters

<code>ac</code>	Handle to AC space
<code>in_cfg</code>	Input signal configuration
<code>lenOldSamps</code>	How many old samples to buffer
<code>doCircularComp</code>	Compensate for circular convolution?
<code>mu</code>	Scalar coefficient for gradient
<code>alp</code>	Autoregressive coefficient for estimating PSD
<code>useVAD</code>	Use voice activity detection?
<code>vadName_</code>	Name of the VAD AC variable

4.132.1.2 ~gsc_adaptive_stage() `gsc_adaptive_stage::gsc_adaptive_stage::~gsc_adaptive_stage () [default]`

4.132.2 Member Function Documentation

4.132.2.1 process() `mha_wave_t * gsc_adaptive_stage::gsc_adaptive_stage::process (mha_wave_t * wavin)`

Processing callback.

Parameters

<code>wavin</code>	input signal
--------------------	--------------

Returns

Returns a pointer to the output signal

4.132.2.2 insert() void gsc_adaptive_stage::gsc_adaptive_stage::insert () [private]

Re-insert all AC variables into the AC space.

4.132.3 Member Data Documentation

4.132.3.1 ac algo_comm_t gsc_adaptive_stage::gsc_adaptive_stage::ac [private]

Handle to AC space.

4.132.3.2 lenOldSamps unsigned int gsc_adaptive_stage::gsc_adaptive_stage::lenOldSamps [private]

Number of old samples to buffer.

4.132.3.3 lenNewSamps unsigned int gsc_adaptive_stage::gsc_adaptive_stage::lenNewSamps [private]

Number of new samples.

4.132.3.4 bufSize unsigned int gsc_adaptive_stage::gsc_adaptive_stage::bufSize [private]

Total buffer size.

Must be lenOldSamps+lenNewSamps

4.132.3.5 `frac_old` float gsc_adaptive_stage::gsc_adaptive_stage::frac_old [private]

Fraction of new samples to total buffer size.

4.132.3.6 `mha_fft` mha_fft_t gsc_adaptive_stage::gsc_adaptive_stage::mha_fft [private]

FFT handle.

4.132.3.7 `nfreq` unsigned int gsc_adaptive_stage::gsc_adaptive_stage::nfreq [private]

Number of frequency bins.

4.132.3.8 `nchan` unsigned int gsc_adaptive_stage::gsc_adaptive_stage::nchan [private]

Number of channels in input signal.

4.132.3.9 `desired_chan` unsigned int gsc_adaptive_stage::gsc_adaptive_stage::desired←
_chan [private]

Channel index containing the desired response.

Always last channel by convention

4.132.3.10 `doCircularComp` bool gsc_adaptive_stage::gsc_adaptive_stage::doCircular←
Comp [private]

Whether to compensate for circular convolution.

4.132.3.11 mu float gsc_adaptive_stage::gsc_adaptive_stage::mu [private]

Linear coefficient for gradient.

4.132.3.12 alp float gsc_adaptive_stage::gsc_adaptive_stage::alp [private]

Autoregressive coefficient for estimating PSD.

4.132.3.13 useVAD bool gsc_adaptive_stage::gsc_adaptive_stage::useVAD [private]

Wether to use VAD.

4.132.3.14 vadName std::string gsc_adaptive_stage::gsc_adaptive_stage::vadName [private]

Name of VAD AC variable.

4.132.3.15 x MHA_AC::waveform_t gsc_adaptive_stage::gsc_adaptive_stage::x [private]

Buffered input signal.

4.132.3.16 X MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::X [private]

FFT of the buffered input signal.

4.132.3.17 W MHA_AC::spectrum_t gsc_adaptive_stage::gsc_adaptive_stage::W [private]

Time-varying filter.

4.132.3.18 Y **MHA_AC::spectrum_t** gsc_adaptive_stage::gsc_adaptive_stage::Y [private]

Filter output, frequency domain.

4.132.3.19 y **MHA_AC::waveform_t** gsc_adaptive_stage::gsc_adaptive_stage::y [private]

Filter output, time domain.

4.132.3.20 d **MHA_AC::waveform_t** gsc_adaptive_stage::gsc_adaptive_stage::d [private]

Desired response.

4.132.3.21 e **MHA_AC::waveform_t** gsc_adaptive_stage::gsc_adaptive_stage::e [private]

Error signal, time domain.

4.132.3.22 E **MHA_AC::spectrum_t** gsc_adaptive_stage::gsc_adaptive_stage::E [private]

Error signal, frequency domain.

4.132.3.23 E2 **MHA_AC::spectrum_t** gsc_adaptive_stage::gsc_adaptive_stage::E2 [private]

Error spectrum multiplied by input spectrum: $E2=X*E$.

4.132.3.24 grad **MHA_AC::waveform_t** gsc_adaptive_stage::gsc_adaptive_stage::grad [private]

Gradient.

4.132.3.25 Grad `MHA_AC::spectrum_t` `gsc_adaptive_stage::gsc_adaptive_stage::Grad`
[private]

FT of the gradient.

4.132.3.26 e_out `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::e_out`
[private]

Error signal.

4.132.3.27 P `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::P` [private]

Signal power estimate.

4.132.3.28 Psum `MHA_AC::waveform_t` `gsc_adaptive_stage::gsc_adaptive_stage::Psum`
[private]

Signal power estimate, summed over all channels.

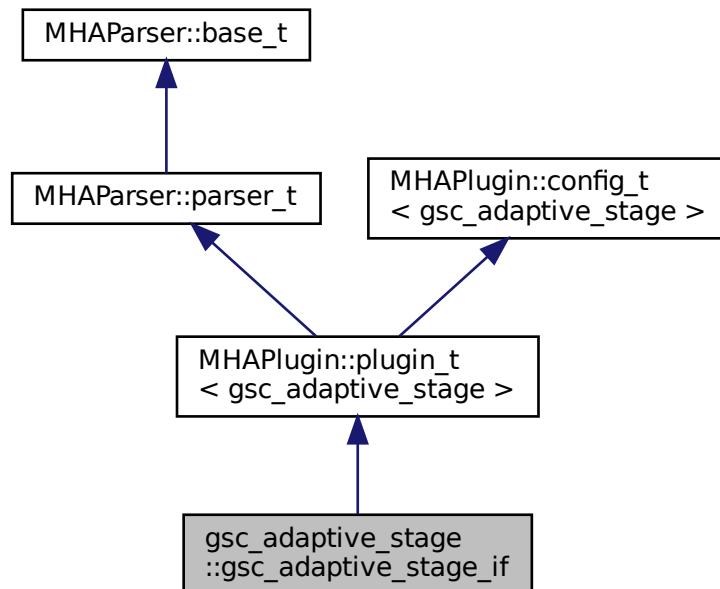
The documentation for this class was generated from the following files:

- `gsc_adaptive_stage.hh`
- `gsc_adaptive_stage.cpp`

4.133 gsc_adaptive_stage::gsc_adaptive_stage_if Class Reference

Plugin interface class.

Inheritance diagram for gsc_adaptive_stage::gsc_adaptive_stage_if:



Public Member Functions

- **gsc_adaptive_stage_if (algo_comm_t & ac, const std::string &, const std::string &)**
Constructs the interface to the adaptive filter plugin.
- **~gsc_adaptive_stage_if ()=default**
- **mha_wave_t * process (mha_wave_t *)**
This plugin implements noise reduction using spectral subtraction: By nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.
- **void prepare (mhaconfig_t &)**
Plugin preparation.
- **void release (void)**

Private Member Functions

- **void update_cfg ()**
Update the rt config.
- **void on_model_param_valuechanged ()**

Private Attributes

- **MHAEvents::patchbay_t < gsc_adaptive_stage_if > patchbay**
- **MHAParser::int_t lenOldSamps**
How many old samples should be buffered per filter block.
- **MHAParser::bool_t doCircularComp**
Whether to compensate for circular convolution.
- **MHAParser::float_t mu**
Linear coefficient for gradient used in filter adaption.
- **MHAParser::float_t alp**
Autoregressive coefficient for PSD estimation.
- **MHAParser::bool_t useVAD**
Whether to use VAD for conditional filter adaption.
- **MHAParser::string_t vadName**
Name of VAD AC variable.

Additional Inherited Members

4.133.1 Detailed Description

Plugin interface class.

4.133.2 Constructor & Destructor Documentation

4.133.2.1 gsc_adaptive_stage_if() `gsc_adaptive_stage::gsc_adaptive_stage_if::gsc_adaptive_stage_if (algo_comm_t & ac, const std::string & , const std::string &)`

Constructs the interface to the adaptive filter plugin.

Parameters

<code>ac</code>	Handle to the ac space
-----------------	------------------------

4.133.2.2 ~gsc_adaptive_stage_if() `gsc_adaptive_stage::gsc_adaptive_stage_if::~gsc_adaptive_stage_if() [default]`

4.133.3 Member Function Documentation

4.133.3.1 process() `mha_wave_t * gsc_adaptive_stage::gsc_adaptive_stage_if::process(mha_wave_t * signal)`

This plugin implements noise reduction using spectral subtraction: By nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.

Parameters

<code>signal</code>	Pointer to the input signal structure.
---------------------	--

Returns

Returns a pointer to the input signal structure, with the signal modified by this plugin.

4.133.3.2 prepare() `void gsc_adaptive_stage::gsc_adaptive_stage_if::prepare(mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the spectral domain and contains at least one channel

Parameters

<code>signal_info</code>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------------	---

Implements `MHAPlugin::plugin_t< gsc_adaptive_stage >` (p. [1148](#)).

4.133 gsc_adaptive_stage::gsc_adaptive_stage_if Class Reference

4.133.3.3 release() void gsc_adaptive_stage::gsc_adaptive_stage_if::release (void) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin_t< gsc_adaptive_stage >** (p. 1149).

4.133.3.4 update_cfg() void gsc_adaptive_stage::gsc_adaptive_stage_if::update_cfg () [private]

Update the rt config.

4.133.3.5 on_model_param_valuechanged() void gsc_adaptive_stage::gsc_adaptive_stage_if::on_model_param_valuechanged () [private]

4.133.4 Member Data Documentation

4.133.4.1 patchbay **MHAEvents::patchbay_t< gsc_adaptive_stage_if >** gsc_adaptive_stage::gsc_adaptive_stage_if::patchbay [private]

4.133.4.2 lenOldSamps **MHAParser::int_t** gsc_adaptive_stage::gsc_adaptive_stage_if::lenOldSamps [private]

How many old samples should be buffered per filter block.

4.133.4.3 doCircularComp **MHAParser::bool_t** gsc_adaptive_stage::gsc_adaptive_stage_if::doCircularComp [private]

Whether to compensate for circular convolution.

4.133.4.4 mu `MHAParser::float_t gsc_adaptive_stage::gsc_adaptive_stage_if::mu`
[private]

Linear coefficient for gradient used in filter adaption.

4.133.4.5 alp `MHAParser::float_t gsc_adaptive_stage::gsc_adaptive_stage_if::alp`
[private]

Autoregressive coefficient for PSD estimation.

4.133.4.6 useVAD `MHAParser::bool_t gsc_adaptive_stage::gsc_adaptive_stage_if::useVAD`
[private]

Wether to use VAD for conditional filter adaption.

4.133.4.7 vadName `MHAParser::string_t gsc_adaptive_stage::gsc_adaptive_stage_if::vadName`
[private]

Name of VAD AC variable.

Ignored if useVAD=no

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

- `gsc_adaptive_stage_if.hh`
- `gsc_adaptive_stage_if.cpp`

4.134 gtfb_analyzer::gtfb_analyzer_cfg_t Struct Reference

Configuration for Gammatone Filterbank Analyzer.

Public Member Functions

- unsigned **bands** () const
Each band is split into this number of bands.
- unsigned **channels** () const
The number of separate audio channels.
- unsigned **frames** () const
The number of frames in one chunk.
- **mha_complex_t * states** (unsigned channel, unsigned band)
Returns pointer to filter states for that band.
- **gtfb_analyzer_cfg_t** (unsigned ch, unsigned **frames**, unsigned ord, const std::vector<
mha_complex_t > &_coeff, const std::vector< **mha_complex_t** > &_norm_phase)
Create a configuration for Gammatone Filterbank Analyzer.
- **~gtfb_analyzer_cfg_t()**
- **mha_complex_t & cvalue** (unsigned frame, unsigned channel, unsigned band)

Public Attributes

- unsigned **order**
The order of the gammatone filters.
- std::vector< **mha_complex_t** > **coeff**
The complex coefficients of the gammatone filter bands.
- std::vector< **mha_complex_t** > **norm_phase**
Combination of normalization and phase correction factor.
- **mha_wave_t s_out**
Storage for the (complex) output signal.
- std::vector< **mha_complex_t** > **state**
Storage for Filter state.

4.134.1 Detailed Description

Configuration for Gammatone Filterbank Analyzer.

4.134.2 Constructor & Destructor Documentation

```
4.134.2.1 gtfb_analyzer_cfg_t() gtfb_analyzer::gtfb_analyzer_cfg_t::gtfb_analyzer_cfg_t (
    unsigned ch,
    unsigned frames,
    unsigned ord,
    const std::vector< mha_complex_t > & _coeff,
    const std::vector< mha_complex_t > & _norm_phase ) [inline]
```

Create a configuration for Gammatone Filterbank Analyzer.

Parameters

<i>ch</i>	Number of Audio channels.
<i>frames</i>	Number of Audio frames per chunk.
<i>ord</i>	The order of the gammatone filters.
<i>_coeff</i>	Complex gammatone filter coefficients.
<i>_norm_phase</i>	Normalization and phase correction factors.

4.134.2.2 ~gtfb_analyzer_cfg_t() `gtfb_analyzer::gtfb_analyzer_cfg_t::~gtfb_analyzer_cfg_t () [inline]`

4.134.3 Member Function Documentation

4.134.3.1 bands() `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::bands () const [inline]`

Each band is split into this number of bands.

4.134.3.2 channels() `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::channels () const [inline]`

The number of separate audio channels.

4.134.3.3 frames() `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::frames () const [inline]`

The number of frames in one chunk.

4.134.3.4 states() `mha_complex_t* gtfb_analyzer::gtfb_analyzer_cfg_t::states (`
 `unsigned channel,`
 `unsigned band) [inline]`

Returns pointer to filter states for that band.

4.134.3.5 cvalue() `mha_complex_t& gtfb_analyzer::gtfb_analyzer_cfg_t::cvalue (`
 `unsigned frame,`
 `unsigned channel,`
 `unsigned band) [inline]`

4.134.4 Member Data Documentation

4.134.4.1 order `unsigned gtfb_analyzer::gtfb_analyzer_cfg_t::order`

The order of the gammatone filters.

4.134.4.2 coeff `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::coeff`

The complex coefficients of the gammatone filter bands.

4.134.4.3 norm_phase `std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::norm_phase`

Combination of normalization and phase correction factor.

4.134.4.4 s_out mha_wave_t gtfb_analyzer::gtfb_analyzer_cfg_t::s_out

Storage for the (complex) output signal.

Each of the real input audio channels is split into frequency bands with complex time signal output. The split complex time signal is again stored in a **mha_wave_t** (p. 839) buffer. Each complex time signal is stored as adjacent real and imaginary channels. Complex output from one source channel is stored in adjacent complex output channels.

Example: If the input has 2 channels ch0 ch1, and **gtfb_analyzer** (p. 96) splits into 3 bands b0 b1 b2, then the order of output channels in s_out is: ch0_b0_real ch0_b0_imag ch0_b1_real ch0_b1_imag ch0_b2_real ch0_b2_imag ch1_b0_real ch1_b1_imag ch1_b1_real ch1_b1_imag ch1_b2_real ch1_b2_imag

4.134.4.5 state std::vector< mha_complex_t> gtfb_analyzer::gtfb_analyzer_cfg_t::state

Storage for Filter state.

Holds **channels()** (p. 550) * **bands()** (p. 550) * order complex filter states. Layout: state[(**bands()** (p. 550)*channel+band)*order+stage]

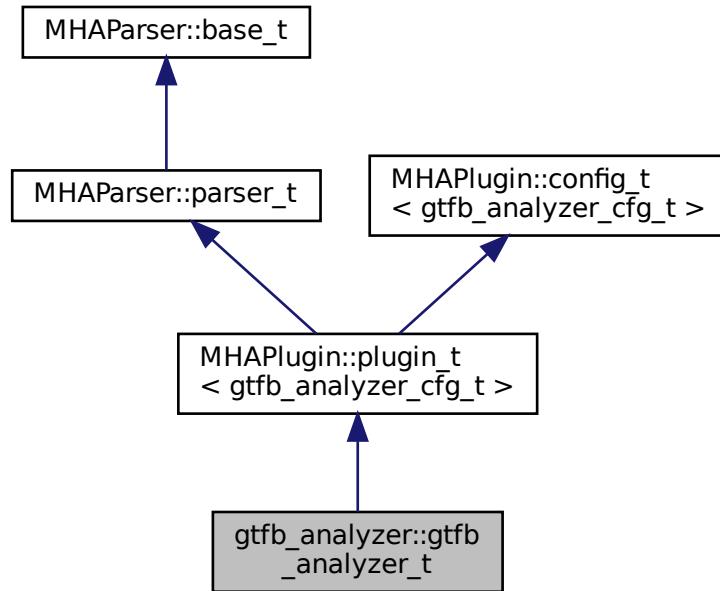
The documentation for this struct was generated from the following file:

- **gtfb_analyzer.cpp**

4.135 gtfb_analyzer::gtfb_analyzer_t Class Reference

Gammatone Filterbank Analyzer Plugin.

Inheritance diagram for gtfb_analyzer::gtfb_analyzer_t:



Public Member Functions

- **gtfb_analyzer_t** (const **algo_comm_t** &, const std::string &thread_name, const std::string &algo_name)
- **mha_wave_t * process (mha_wave_t *)**
- void **prepare (mhaconfig_t &)**

Private Member Functions

- void **update_cfg ()**

Private Attributes

- **MHAEvents::patchbay_t< gtfb_analyzer_t > patchbay**
- bool **prepared**
- **MHParse::int_t order**
- **MHParse::vcomplex_t coeff**
- **MHParse::vcomplex_t norm_phase**

Additional Inherited Members

4.135.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

4.135.2 Constructor & Destructor Documentation

```
4.135.2.1 gtfb_analyzer_t() gtfb_analyzer::gtfb_analyzer_t::gtfb_analyzer_t (
    const algo_comm_t & iac,
    const std::string & thread_name,
    const std::string & algo_name )
```

4.135.3 Member Function Documentation

```
4.135.3.1 process() mha_wave_t * gtfb_analyzer::gtfb_analyzer_t::process (
    mha_wave_t * s )
```

```
4.135.3.2 prepare() void gtfb_analyzer::gtfb_analyzer_t::prepare (
    mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< gtfb_analyzer_cfg_t >** (p. [1148](#)).

```
4.135.3.3 update_cfg() void gtfb_analyzer::gtfb_analyzer_t::update_cfg ( ) [private]
```

4.135.4 Member Data Documentation

4.135.4.1 patchbay `MHAEvents::patchbay_t< gtfb_analyzer_t> gtfb_analyzer::gtfb->`
`_analyzer_t::patchbay [private]`

4.135.4.2 prepared `bool gtfb_analyzer::gtfb_analyzer_t::prepared [private]`

4.135.4.3 order `MHAParser::int_t gtfb_analyzer::gtfb_analyzer_t::order [private]`

4.135.4.4 coeff `MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t::coeff [private]`

4.135.4.5 norm_phase `MHAParser::vcomplex_t gtfb_analyzer::gtfb_analyzer_t->`
`::norm_phase [private]`

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

- `gtfb_analyzer.cpp`

4.136 gtfb_simd_cfg_t Class Reference

Public Member Functions

- `unsigned get_bands () const`
Each band is split into this number of bands.
- `unsigned get_channels () const`
The number of separate audio channels.
- `unsigned get_frames () const`
The number of frames in one chunk.
- `gtfb_simd_cfg_t (gtfb_simd_cfg_t &&)=delete`
- `gtfb_simd_cfg_t & operator= (gtfb_simd_cfg_t &&)=delete`
- `gtfb_simd_cfg_t (const gtfb_simd_cfg_t &)=delete`
- `gtfb_simd_cfg_t & operator= (const gtfb_simd_cfg_t &)=delete`
- `gtfb_simd_cfg_t (unsigned ch, unsigned frames, unsigned ord, const std::vector< mha_complex_t > &_coeff, const std::vector< mha_complex_t > &_norm_phase)`
- `~gtfb_simd_cfg_t ()`
- `mha_wave_t * process (mha_wave_t *s_in)`

Private Attributes

- **unsigned order**
The order of the gammatone filters.
- **unsigned bands**
Number of frequency bands per channel.
- **unsigned channels**
Number of input audio channels.
- **unsigned bandsXchannels**
Product of bands and channels.
- **std::vector< mha_complex_t > norm_phase**
Combination of normalization and phase correction factor.
- **float * rinputs**
input signal (1 sample) multiplied with norm_phase
- **float * iinputs**
- **float * rcoefficients**
The complex coefficients of the gammatone filter bands.
- **float * icoefficients**
- **float * rstates**
- **float * istates**
- **float * sout_buf**
output signal buffer, used by s_out.
- **float * large_array**
Large float array.
- **mha_wave_t s_out**

4.136.1 Detailed Description

Configuration for Gammatone Filterbank SIMD Analyzer.

4.136.2 Constructor & Destructor Documentation

4.136.2.1 gtfb_simd_cfg_t() [1/3] `gtfb_simd_cfg_t::gtfb_simd_cfg_t (` `gtfb_simd_cfg_t &&) [delete]`

4.136.2.2 gtfb_simd_cfg_t() [2/3] `gtfb_simd_cfg_t::gtfb_simd_cfg_t (const gtfb_simd_cfg_t &) [delete]`

4.136.2.3 gtfb_simd_cfg_t() [3/3] `gtfb_simd_cfg_t::gtfb_simd_cfg_t (unsigned ch, unsigned frames, unsigned ord, const std::vector< mha_complex_t > & _coeff, const std::vector< mha_complex_t > & _norm_phase) [inline]`

Create a configuration for Gammatone Filterbank Analyzer.

Parameters

<i>ch</i>	Number of Audio channels.
<i>frames</i>	Number of Audio frames per chunk.
<i>ord</i>	The order of the gammatone filters.
<i>_coeff</i>	Complex gammatone filter coefficients.
<i>_norm_phase</i>	Normalization and phase correction factors.

4.136.2.4 ~gtfb_simd_cfg_t() `gtfb_simd_cfg_t::~gtfb_simd_cfg_t () [inline]`

4.136.3 Member Function Documentation

4.136.3.1 get_bands() `unsigned gtfb_simd_cfg_t::get_bands () const [inline]`

Each band is split into this number of bands.

4.136.3.2 get_channels() `unsigned gtfb_simd_cfg_t::get_channels () const [inline]`

The number of separate audio channels.

4.136.3.3 `get_frames()` `unsigned gtfb_simd_cfg_t::get_frames () const [inline]`

The number of frames in one chunk.

4.136.3.4 `operator=()` [1/2] `gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= (gtfb_simd_cfg_t &&) [delete]`

4.136.3.5 `operator=()` [2/2] `gtfb_simd_cfg_t& gtfb_simd_cfg_t::operator= (const gtfb_simd_cfg_t &) [delete]`

4.136.3.6 `process()` `mha_wave_t* gtfb_simd_cfg_t::process (mha_wave_t * s_in) [inline]`

4.136.4 Member Data Documentation

4.136.4.1 `order` `unsigned gtfb_simd_cfg_t::order [private]`

The order of the gammatone filters.

4.136.4.2 `bands` `unsigned gtfb_simd_cfg_t::bands [private]`

Number of frequency bands per channel.

4.136.4.3 `channels` `unsigned gtfb_simd_cfg_t::channels [private]`

Number of input audio channels.

4.136.4.4 bandsXchannels unsigned gtfb_simd_cfg_t::bandsXchannels [private]

Product of bands and channels.

4.136.4.5 norm_phase std::vector< mha_complex_t> gtfb_simd_cfg_t::norm_phase [private]

Combination of normalization and phase correction factor.

4.136.4.6 rinputs float* gtfb_simd_cfg_t::rinputs [private]

input signal (1 sample) multiplied with norm_phase

4.136.4.7 iinputs float* gtfb_simd_cfg_t::iinputs [private]**4.136.4.8 rcoefficients** float* gtfb_simd_cfg_t::rcoefficients [private]

The complex coefficients of the gammatone filter bands.

4.136.4.9 icoefficients float* gtfb_simd_cfg_t::icoefficients [private]**4.136.4.10 rstates** float* gtfb_simd_cfg_t::rstates [private]

Storage for Filter state. Holds **channels()** (p. 558) * **bands()** (p. 558) * order complex filter states. Layout: state[stage * bandsXchannels + **bands()** (p. 558)*channel+band]

4.136.4.11 `istates` float* gtfb_simd_cfg_t::istates [private]**4.136.4.12 `sout_buf`** float* gtfb_simd_cfg_t::sout_buf [private]

output signal buffer, used by `s_out`.

Contains bandsXchannels * fragsize *2 entries. all real parts come before all complex parts. Order is: channel 0 band 0 real, channel 0 band 1 real, ..., channel 1 band 0 real channel 1 band 1 real, ... channel 0 band 0 imag ... then next sample

4.136.4.13 `large_array` float* gtfb_simd_cfg_t::large_array [private]

Large float array.

All previous pointers point into this array. Contains 3 unused entries to be able to adjust for alignment.

4.136.4.14 `s_out` mha_wave_t gtfb_simd_cfg_t::s_out [private]

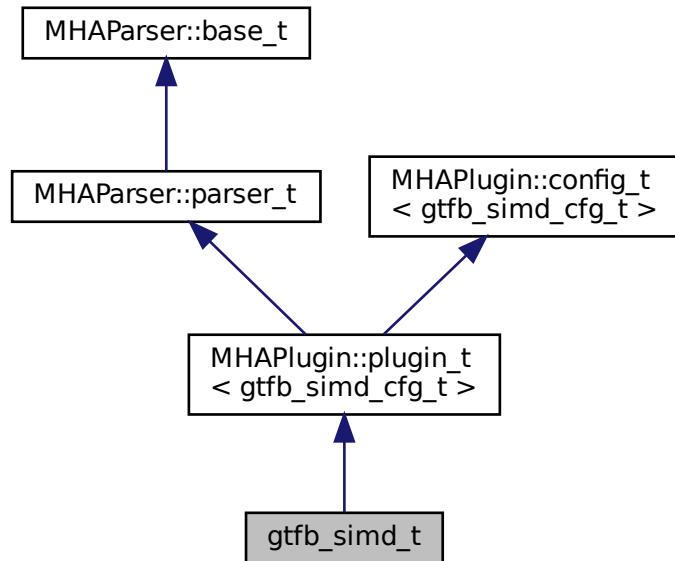
Storage for the (complex) output signal. Each of the real input audio channels is split into frequency bands with complex time signal output. The split complex time signal is again stored in a `mha_wave_t` (p. 839) buffer. Each complex time signal is stored as adjacent real and imaginary channels. Complex output from one source channel is stored in adjacent complex output channels.

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

- `gtfb_simd.cpp`

4.137 gtfb_simd_t Class Reference

Inheritance diagram for gtfb_simd_t:



Public Member Functions

- `gtfb_simd_t (const algo_comm_t &, const std::string &thread_name, const std::string &algo_name)`
- `mha_wave_t * process (mha_wave_t *)`
- `void prepare (mhaconfig_t &)`

Private Member Functions

- `void update_cfg ()`

Private Attributes

- `MHAEvents::patchbay_t < gtfb_simd_t > patchbay`
- `bool prepared`
- `MHAParser::int_t order`
- `MHAParser::vcomplex_t coeff`
- `MHAParser::vcomplex_t norm_phase`

Additional Inherited Members

4.137.1 Constructor & Destructor Documentation

4.137.1.1 `gtfb_simd_t()` `gtfb_simd_t::gtfb_simd_t (`
 `const algo_comm_t & iac,`
 `const std::string & thread_name,`
 `const std::string & algo_name)`

4.137.2 Member Function Documentation

4.137.2.1 `process()` `mha_wave_t * gtfb_simd_t::process (`
 `mha_wave_t * s)`

4.137.2.2 `prepare()` `void gtfb_simd_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements `MHAPlugin::plugin_t< gtfb_simd_cfg_t >` (p. [1148](#)).

4.137.2.3 `update_cfg()` `void gtfb_simd_t::update_cfg () [private]`

4.137.3 Member Data Documentation

4.137.3.1 `patchbay` `MHAEvents::patchbay_t< gtfb_simd_t > gtfb_simd_t::patchbay`
[private]

4.137.3.2 prepared `bool gtfb_simd_t::prepared [private]`

4.137.3.3 order `MHAParser::int_t gtfb_simd_t::order [private]`

4.137.3.4 coeff `MHAParser::vcomplex_t gtfb_simd_t::coeff [private]`

4.137.3.5 norm_phase `MHAParser::vcomplex_t gtfb_simd_t::norm_phase [private]`

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

- `gtfb_simd.cpp`

4.138 gtfb_simple_rt_t Class Reference

Runtime configuration class of gtfb_simple_bridge plugin.

Public Member Functions

- **gtfb_simple_rt_t** (`algo_comm_t ac, const std::string &name, mhaconfig_t &chcfg, std::vector< mha_real_t > cf, std::vector< mha_real_t > bw, unsigned int order, unsigned int pre_stages, unsigned int desired_delay, std::vector< mha_real_t > &vclass, std::vector< mha_real_t > &resynthesis_gain, const std::string &element_gain_name)`)
- **mha_wave_t * pre_plugin (mha_wave_t *s)**
Filter real input signal s with the pre_stages filter orders in each gammatone filter band.
- **mha_wave_t * post_plugin (mha_wave_t *s)**
Post-filter the complex-valued filter-bank signal s after it has been processed by the loaded plugin.
- **const MHAFilter::gamma_filt_t & get_gf () const**
Const-accessor to contained gammatone filterbank object.

Static Private Member Functions

- **static std::vector< mha_real_t > duplicate_vector (const std::vector< mha_real_t > &src, unsigned int nchannels)**
Helper function to repeat the elements in a vector.

Private Attributes

- **unsigned int _order**
Total number of gammatone filter orders.
- **unsigned int _pre_stages**
Number of filter orders applied before the loaded plugin is invoked.
- **unsigned int nbands**
Number of frequency bands to produce = number of gammatone filters.
- **MHA_AC::waveform_t imag**
Storage for the imaginary part of the filterbank signal.
- **MHA_AC::waveform_t accf**
AC variable to publish the center frequencies of the gammatone filters.
- **MHA_AC::waveform_t acbw**
AC variable to publish the bandwidths of the gammatone filters.
- **MHASignal::waveform_t input**
Real part of the filterbank signal.
- **MHASignal::waveform_t output**
Resynthesized broadband signal, used as the output signal of this plugin.
- **MHAFilter::gamma_flt_t gf**
The gammatone filter bank implementation.
- **MHA_AC::waveform_t cLTASS**
AC variable to publish band-specific LTASS level correction values.
- **MHA_AC::waveform_t ac_resynthesis_gain**
AC variable to publish the configured per-frequency resynthesis gains.
- **std::string element_gain_name_**
Either an empty string, or the name of an AC variable from which element-wise linear factors are read.
- **algo_comm_t _ac**
Algorithm Communication Variable space.

4.138.1 Detailed Description

Runtime configuration class of gtfb_simple_bridge plugin.

4.138.2 Constructor & Destructor Documentation

```
4.138.2.1 gtfb_simple_rt_t() gtfb_simple_rt_t::gtfb_simple_rt_t (
    algo_comm_t ac,
    const std::string & name,
    mhaconfig_t & chcfg,
    std::vector< mha_real_t > cf,
    std::vector< mha_real_t > bw,
    unsigned int order,
    unsigned int pre_stages,
    unsigned int desired_delay,
    std::vector< mha_real_t > & voltass,
    std::vector< mha_real_t > & resynthesis_gain,
    const std::string & element_gain_name )
```

4.138.3 Member Function Documentation

4.138.3.1 pre_plugin() mha_wave_t * gtfb_simple_rt_t::pre_plugin (mha_wave_t * s)

Filter real input signal s with the pre_stages filter orders in each gammatone filter band.

The real part of the complex output is returned in the return value of the method, the imaginary part is stored into the AC variable.

Parameters

s	real-valued, broad-band input signal
---	--------------------------------------

Returns

real part of complex-valued output signal after pre_stages gammatone filter orders have been applied in each band. Order of output bands in real and imaginary output are: (channel0,band0), (channel0,band1), ..., (cannel1,band0), ...

4.138.3.2 post_plugin() mha_wave_t * gtfb_simple_rt_t::post_plugin (mha_wave_t * s)

Post-filter the complex-valued filter-bank signal s after it has been processed by the loaded plugin.

The remaining gammatone filter orders are applied to restrict the loaded plugin's output signal to the respective bands. After

Parameters

<i>s</i>	complex-valued, filter-bank signal. This signal is produced by letting the loaded plugin process the output signal of the pre_plugin method.
----------	--

Returns

real part of complex-valued output signal after pre_stages gammatone filter orders have been applied in each band. Order of output bands in real and imaginary output are: (channel0,band0), (channel0,band1), ..., (cannel1,band0), ...

4.138.3.3 `get_gf()` const `MHAFilter::gamma_flt_t& gtfb_simple_rt_t::get_gf () const [inline]`

Const-accessor to contained gammatone filterbank object.

4.138.3.4 `duplicate_vector()` `std::vector< mha_real_t > gtfb_simple_rt_t::duplicate←_vector (`
`const std::vector< mha_real_t > & src,`
`unsigned int nchannels) [static], [private]`

Helper function to repeat the elements in a vector.

Parameters

<i>src</i>	vector to repeat
<i>nchannels</i>	number of times to repeat input vector

Returns

a vector that returns *nchannels* repetitions of input vector.

4.138.4 Member Data Documentation

4.138.4.1 _order unsigned int gtfb_simple_rt_t::_order [private]

Total number of gammatone filter orders.

4.138.4.2 _pre_stages unsigned int gtfb_simple_rt_t::_pre_stages [private]

Number of filter orders applied before the loaded plugin is invoked.

4.138.4.3 nbands unsigned int gtfb_simple_rt_t::nbands [private]

Number of frequency bands to produce = number of gammatone filters.

4.138.4.4 imag MHA_AC::waveform_t gtfb_simple_rt_t::imag [private]

Storage for the imaginary part of the filterbank signal.

It is used as the imaginary input signal for the loaded plugin. Furthermore, it is expected that the loaded plugin processes the imaginary part of the data in place.

4.138.4.5 accf MHA_AC::waveform_t gtfb_simple_rt_t::accf [private]

AC variable to publish the center frequencies of the gammatone filters.

4.138.4.6 acbw MHA_AC::waveform_t gtfb_simple_rt_t::acb_w [private]

AC variable to publish the bandwidths of the gammatone filters.

4.138.4.7 input MHASignal::waveform_t gtfb_simple_rt_t::input [private]

Real part of the filterbank signal.

It is used as the real input signal to the loaded plugin.

4.138.4.8 output `MHASignal::waveform_t` `gtfb_simple_rt_t::output` [private]

Resynthesized broadband signal, used as the output signal of this plugin.

4.138.4.9 gf `MHAFilter::gamma_flt_t` `gtfb_simple_rt_t::gf` [private]

The gammatone filter bank implementation.

4.138.4.10 cLTASS `MHA_AC::waveform_t` `gtfb_simple_rt_t::cLTASS` [private]

AC variable to publish band-specific LTASS level correction values.

4.138.4.11 ac_resynthesis_gain `MHA_AC::waveform_t` `gtfb_simple_rt_t::ac_resynthesis←_gain` [private]

AC variable to publish the configured per-frequency resynthesis gains.

4.138.4.12 element_gain_name_ `std::string` `gtfb_simple_rt_t::element_gain_name_` [private]

Either an empty string, or the name of an AC variable from which element-wise linear factors are read.

4.138.4.13 _ac `algo_comm_t` `gtfb_simple_rt_t::_ac` [private]

Algorithm Communication Variable space.

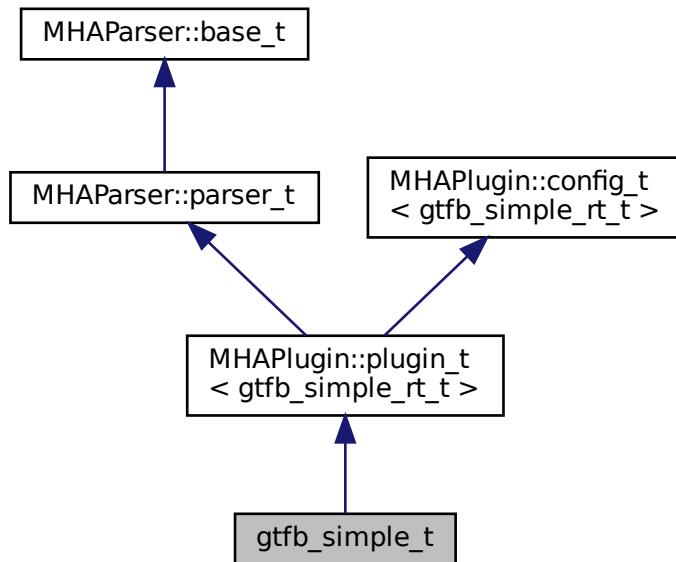
The documentation for this class was generated from the following file:

- `gtfb_simple_bridge.cpp`

4.139 gtfb_simple_t Class Reference

Interface class of gtfb_simple_bridge plugin.

Inheritance diagram for gtfb_simple_t:



Public Member Functions

- **`gtfb_simple_t (algo_comm_t ac, const std::string &chain, const std::string &algo)`**
Constructor.
- **`void prepare (mhaconfig_t &chcfg)`**
Prepare contained plugin for signal processing.
- **`void release ()`**
Releases contained plugin.
- **`mha_wave_t * process (mha_wave_t *sln)`**
Process the input signal: Computes the filterbank with the pre-stages, calls the loaded plugin for processing of the filterbank signal, and resynthesizes the result to a modified broadband signal.
- **`void setlock (bool b)`**
Locks / unlocks all configuration variables before / after signal processing.

Private Attributes

- **MHAParser::mhapluginloader_t plug**
Handle to the loaded plugin.
- **MHAOvIFilter::fscale_bw_t fscale**
Object determines the filterbank frequencies.
- **MHAParser::int_t order**
total order of gammatone filter
- **MHAParser::int_t prestages**
Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.
- **MHAParser::int_t desired_delay**
Desired group delay in audio samples.
- **MHAParser::string_t element_gain_name**
Number of AC variable to take element-wise gain factors from for resynthesis.
- **MHAParser::vfloat_mon_t cLTASS**
Monitoring of LTASS correction values / dB.
- **MHAParser::vfloat_mon_t resynthesis_gain**
configured frequency-specific resynthesis gains
- **MHAParser::string_mon_t gf_internals**
For tests and debugging: a serialization of the gammatone filter internals.
- std::string **name_**
Configured algorithm name, used to name the AC variables.

Additional Inherited Members

4.139.1 Detailed Description

Interface class of gtfb_simple_bridge plugin.

4.139.2 Constructor & Destructor Documentation

4.139.2.1 **gtfb_simple_t()** `gtfb_simple_t::gtfb_simple_t (` `algo_comm_t ac,` `const std::string & chain,` `const std::string & algo)`

Constructor.

Registers parser variables.

Parameters

<i>ac</i>	Algorithm Communication Variable space
<i>chain</i>	chain name
<i>algo</i>	configured name of this plugin instance

4.139.3 Member Function Documentation

4.139.3.1 `prepare()` `void gtfb_simple_t::prepare (mhaconfig_t & chcfg) [virtual]`

Prepare contained plugin for signal processing.

Allocates the runtime configuration instance. Locks all variables.

Implements `MHAPlugin::plugin_t< gtfb_simple_rt_t >` (p. [1148](#)).

4.139.3.2 `release()` `void gtfb_simple_t::release () [virtual]`

Releases contained plugin.

Unlocks all variables.

Reimplemented from `MHAPlugin::plugin_t< gtfb_simple_rt_t >` (p. [1149](#)).

4.139.3.3 `process()` `mha_wave_t * gtfb_simple_t::process (mha_wave_t * sIn)`

Process the input signal: Computes the filterbank with the pre-stages, calls the loaded plugin for processing of the filterbank signal, and resynthesizes the result to a modified broadband signal.

4.139.3.4 setlock() void gtfb_simple_t::setlock (bool b) [inline]

Locks / unlocks all configuration variables before / after signal processing.

4.139.4 Member Data Documentation

4.139.4.1 plug MHAParser::mhapluginloader_t gtfb_simple_t::plug [private]

Handle to the loaded plugin.

4.139.4.2 fscale MHAOv1Filter::fscale_bw_t gtfb_simple_t::fscale [private]

Object determines the filterbank frequencies.

4.139.4.3 order MHAParser::int_t gtfb_simple_t::order [private]

total order of gammatone filter

4.139.4.4 prestages MHAParser::int_t gtfb_simple_t::prestages [private]

Number of gammatone filter order to process before the loaded plugin processes the filterbank signal.

4.139.4.5 desired_delay MHAParser::int_t gtfb_simple_t::desired_delay [private]

Desired group delay in audio samples.

4.139.4.6 element_gain_name `MHAParser::string_t gtfb_simple_t::element_gain_` ↵
name [private]

Number of AC variable to take element-wise gain factors from for resynthesis.

4.139.4.7 cLTASS `MHAParser::vfloat_mon_t gtfb_simple_t::cLTASS` [private]

Monitoring of LTASS correction values / dB.

4.139.4.8 resynthesis_gain `MHAParser::vfloat_mon_t gtfb_simple_t::resynthesis_` ↵
gain [private]

configured frequency-specific resynthesis gains

4.139.4.9 gf_internals `MHAParser::string_mon_t gtfb_simple_t::gf_internals` [private]

For tests and debugging: a serialization of the gammatone filter internals.

4.139.4.10 name_ `std::string gtfb_simple_t::name_` [private]

Configured algorithm name, used to name the AC variables.

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

- `gtfb_simple_bridge.cpp`

4.140 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**

4.140.1 Constructor & Destructor Documentation

4.140.1.1 hanning_ramps_t() `hanning_ramps_t::hanning_ramps_t (`
 `unsigned int la,`
 `unsigned int lb)`

4.140.1.2 ~hanning_ramps_t() `hanning_ramps_t::~hanning_ramps_t ()`

4.140.2 Member Function Documentation

4.140.2.1 operator()() `void hanning_ramps_t::operator() (`
 `MHASignal::waveform_t & b)`

4.140.3 Member Data Documentation

4.140.3.1 len_a `unsigned int hanning_ramps_t::len_a [private]`

4.140.3.2 len_b `unsigned int hanning_ramps_t::len_b [private]`

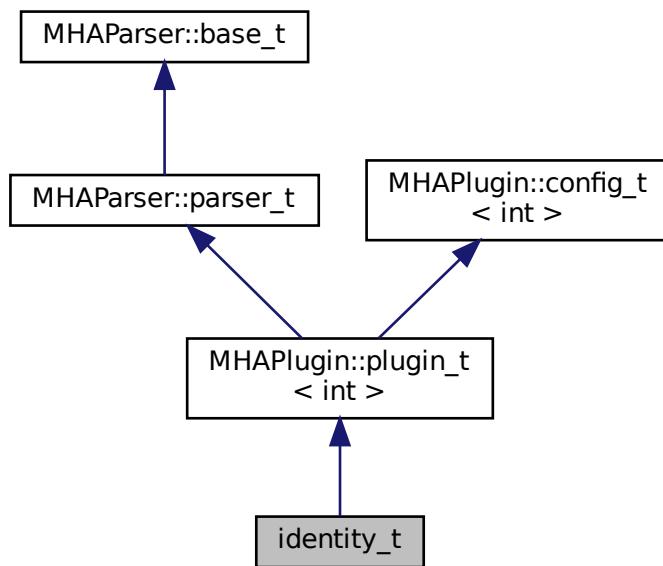
4.140.3.3 ramp_a mha_real_t* hanning_ramps_t::ramp_a [private]**4.140.3.4 ramp_b mha_real_t* hanning_ramps_t::ramp_b [private]**

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

- **spec2wave.cpp**

4.141 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

4.141.1 Constructor & Destructor Documentation

4.141.1.1 `identity_t()` `identity_t::identity_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

4.141.2 Member Function Documentation

4.141.2.1 `process()` [1/2] `mha_wave_t * identity_t::process (`
 `mha_wave_t * s)`

4.141.2.2 `process()` [2/2] `mha_spec_t * identity_t::process (`
 `mha_spec_t * s)`

4.141.2.3 `prepare()` `void identity_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPlugin::plugin_t< int >** (p. 1148).

4.141.2.4 `release()` `void identity_t::release () [virtual]`

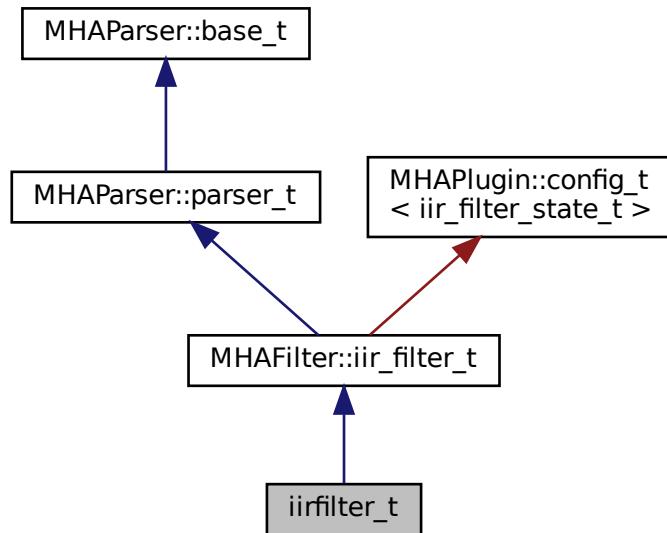
Reimplemented from **MHAPlugin::plugin_t< int >** (p. 1149).

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

- **identity.cpp**

4.142 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

4.142.1 Constructor & Destructor Documentation

4.142.1.1 `iirfilter_t()` `iirfilter_t::iirfilter_t (`
 `const algo_comm_t & ac,`
 `const std::string & th,`
 `const std::string & al)`

4.142.2 Member Function Documentation

4.142.2.1 `prepare_()` `void iirfilter_t::prepare_ (`
`mhaconfig_t & tf)`

4.142.2.2 `release_()` `void iirfilter_t::release_ () [inline]`

4.142.2.3 `process()` `mha_wave_t * iirfilter_t::process (`
`mha_wave_t * s)`

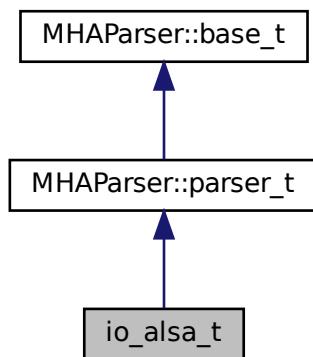
The documentation for this class was generated from the following file:

- `iirfilter.cpp`

4.143 `io_alsa_t` Class Reference

MHA IO interface class for ALSA IO.

Inheritance diagram for `io_alsa_t`:



Public Member Functions

- **io_alsa_t** (unsigned int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void * proc_handle, **IOStartedEvent_t** start_event, void * start_handle, **IOStoppedEvent_t** stop_event, void * stop_handle)
Constructor, receives the callback handles to interact with the MHA framework.
- template<typename T = void>
void **prepare** (int, int)
Called after the framework has prepared the processing plugins and the number of input and output channels are fixed.
- void **release** ()
MHA framework leaves prepared state.
- void **start** ()
MHA framework calls this function when signal processing should start.
- void **stop** ()
MHA framework calls this function when signal processing should stop.
- template<> void **prepare** (int nch_in, int nch_out)

Static Public Member Functions

- static void * **thread_start** (void *h)
MHAIOAIsa uses a separate thread that calls the alsa read and write functions to read and write audio samples, these functions are blocking until samples can be read or written.

Private Member Functions

- void **process** ()

Private Attributes

- bool **b_process**
- unsigned int **fw_fragsize**
- unsigned int **fw_samplerate**
- **IOProcessEvent_t** proc_event
- void * proc_handle
- **IOStartedEvent_t** start_event
- void * start_handle
- **IOStoppedEvent_t** stop_event
- void * stop_handle
- **alsa_base_t** * dev_in
- **alsa_base_t** * dev_out
- pthread_t proc_thread
- **alsa_dev_par_parser_t** p_in
- **alsa_dev_par_parser_t** p_out
- **MHAParser::bool_t** pcmlink
- **MHAParser::int_t** priority
- **MHAParser::kw_t** format
- **MHAParser::int_mon_t** alsas_start_counter
- **MHAEvents::patchbay_t**< **io_alsa_t** > patchbay

Additional Inherited Members

4.143.1 Detailed Description

MHA IO interface class for ALSA IO.

4.143.2 Constructor & Destructor Documentation

```
4.143.2.1 io_alsa_t() io_alsa_t::io_alsa_t (
    unsigned int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

Constructor, receives the callback handles to interact with the MHA framework.

4.143.3 Member Function Documentation

```
4.143.3.1 prepare() [1/2] template<typename T >
void io_alsa_t::prepare (
    int nch_in,
    int nch_out )
```

Called after the framework has perpared the processing plugins and the number of input and output channels are fixed.

open pcm streams

```
4.143.3.2 release() void io_alsa_t::release ( )
```

MHA framework leaves prepared state.

4.143.3.3 start() void io_alsa_t::start ()

MHA framework calls this function when signal processing should start.

4.143.3.4 stop() void io_alsa_t::stop ()

MHA framework calls this function when signal processing should stop.

4.143.3.5 thread_start() void * io_alsa_t::thread_start (void * h) [static]

MHAIOAlsa uses a separate thread that calls the alsa read and write functions to read and write audio samples, these functions are blocking until samples can be read or written.

This is the start function of that thread.

4.143.3.6 process() void io_alsa_t::process () [private]**4.143.3.7 prepare() [2/2]** template<>
void io_alsa_t::prepare (int nch_in, int nch_out)**4.143.4 Member Data Documentation****4.143.4.1 b_process** bool io_alsa_t::b_process [private]**4.143.4.2 fw fragsize** unsigned int io_alsa_t::fw fragsize [private]

4.143.4.3 fw_samplerate unsigned int io_alsa_t::fw_samplerate [private]

4.143.4.4 proc_event IOProcessEvent_t io_alsa_t::proc_event [private]

4.143.4.5 proc_handle void* io_alsa_t::proc_handle [private]

4.143.4.6 start_event IOStartedEvent_t io_alsa_t::start_event [private]

4.143.4.7 start_handle void* io_alsa_t::start_handle [private]

4.143.4.8 stop_event IOStoppedEvent_t io_alsa_t::stop_event [private]

4.143.4.9 stop_handle void* io_alsa_t::stop_handle [private]

4.143.4.10 dev_in alsa_base_t* io_alsa_t::dev_in [private]

4.143.4.11 dev_out alsa_base_t* io_alsa_t::dev_out [private]

4.143.4.12 proc_thread pthread_t io_alsa_t::proc_thread [private]

4.143.4.13 p_in alsa_dev_par_parser_t io_alsa_t::p_in [private]

4.143.4.14 p_out alsa_dev_par_parser_t io_alsa_t::p_out [private]

4.143.4.15 pcmlink MHAParser::bool_t io_alsa_t::pcmlink [private]

4.143.4.16 priority MHAParser::int_t io_alsa_t::priority [private]

4.143.4.17 format MHAParser::kw_t io_alsa_t::format [private]

4.143.4.18 alsa_start_counter MHAParser::int_mon_t io_alsa_t::alsa_start_counter [private]

4.143.4.19 patchbay MHAEvents::patchbay_t< io_alsa_t> io_alsa_t::patchbay [private]

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

- **MHAIoalsa.cpp**

4.144 **io_asterisk_fwcb_t** Class Reference

TCP sound-io library's interface to the framework callbacks.

Public Member Functions

- **io_asterisk_fwcb_t (IOProcessEvent_t proc_event, void * proc_handle, IOStartedEvent_t start_event, void * start_handle, IOStoppedEvent_t stop_event, void * stop_handle)**
Constructor stores framework handles and initializes error numbers to 0.
- virtual ~**io_asterisk_fwcb_t ()**
Do-nothing destructor.
- virtual void **start ()**
Call the framework's start callback.
- virtual int **process (mha_wave_t *sIn, mha_wave_t *&sOut)**
Call the frameworks processing callback.
- virtual void **set_errnos (int proc_err, int io_err)**
Save error numbers to use during.
- virtual void **stop ()**
Call the frameworks stop callback.

Private Attributes

- **IOProcessEvent_t proc_event**
Pointer to signal processing callback function.
- **IOStartedEvent_t start_event**
Pointer to start notification callback function.
- **IOStoppedEvent_t stop_event**
Pointer to stop notification callback function.
- void * **proc_handle**
Handles belonging to framework.
- void * **start_handle**
- void * **stop_handle**
- int **proc_err**
Errors from the processing callback and from the TCP IO itself are stored here before closing Network handles.
- int **io_err**

4.144.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

4.144.2 Constructor & Destructor Documentation

```
4.144.2.1 io_asterisk_fwcb_t() io_asterisk_fwcb_t::io_asterisk_fwcb_t (
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

Constructor stores framework handles and initializes error numbers to 0.

```
4.144.2.2 ~io_asterisk_fwcb_t() virtual io_asterisk_fwcb_t::~io_asterisk_fwcb_t (
) [inline], [virtual]
```

Do-nothing destructor.

4.144.3 Member Function Documentation

```
4.144.3.1 start() void io_asterisk_fwcb_t::start ( ) [virtual]
```

Call the framework's start callback.

```
4.144.3.2 process() int io_asterisk_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.

4.144.3.3 set_errnos() void io_asterisk_fwcbt::set_errnos (int *proc_err*, int *io_err*) [virtual]

Save error numbers to use during.

See also

[stop](#) (p. 586)

Parameters

<i>proc_err</i>	The error number from the
-----------------	---------------------------

See also

[process](#) (p. 585) callback.

Parameters

<i>io_err</i>	The error number from the io library itself.
---------------	--

4.144.3.4 stop() void io_asterisk_fwcbt::stop () [virtual]

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

[set_errnos](#) (p. 586).

4.144.4 Member Data Documentation

4.144.4.1 proc_event IOProcessEvent_t io_asterisk_fwcb_t::proc_event [private]

Pointer to signal processing callback function.

4.144.4.2 start_event IOStartedEvent_t io_asterisk_fwcb_t::start_event [private]

Pointer to start notification callback function.

Called when a new TCP connection is established or the user issues the start command while there is a connection.

4.144.4.3 stop_event IOSToppedEvent_t io_asterisk_fwcb_t::stop_event [private]

Pointer to stop notification callback function.

Called when the connection is closed.

4.144.4.4 proc_handle void* io_asterisk_fwcb_t::proc_handle [private]

Handles belonging to framework.

4.144.4.5 start_handle void * io_asterisk_fwcb_t::start_handle [private]

4.144.4.6 stop_handle void * io_asterisk_fwcb_t::stop_handle [private]

4.144.4.7 proc_err int io_asterisk_fwcbs_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. 586) from that callback. Within **stop** (p. 586), these error numbers are read again and transmitted to the framework.

4.144.4.8 io_err int io_asterisk_fwcbs_t::io_err [private]

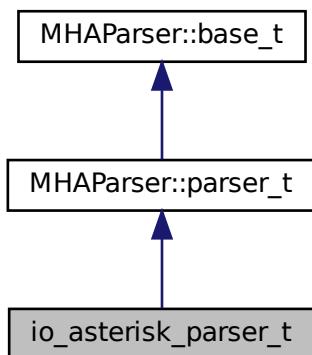
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

4.145 io_asterisk_parser_t Class Reference

The parser interface of the IOAsterisk library.

Inheritance diagram for io_asterisk_parser_t:



Public Member Functions

- virtual const std::string & **get_local_address** () const
Read parser variable local_address, this is the address of the network interface that should listen for incoming connections.
- virtual unsigned short **get_local_port** () const
Read parser variable local_port, this is the TCP port that should be used for incoming connections.
- virtual void **set_local_port** (unsigned short port)
Set parser variable local_port.
- virtual bool **get_server_port_open** () const
Return the status of the server port as it is known to the parser.
- virtual void **set_server_port_open** (bool open)
Inform the parser of the current status of the server socket.
- virtual bool **get_connected** () const
Return the parser's knowledge concerning 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_asterisk_parser_t** ()
Constructor initializes parser variables.
- virtual ~**io_asterisk_parser_t** ()
Do-nothing destructor.
- virtual void **debug** (const std::string &message)

Private Attributes

- **MHAParser::string_t local_address**
Lets the user set the local network interface to listen on.
- **MHAParser::int_t local_port**
Lets the user choose the local tcp port to listen on.
- **MHAParser::int_mon_t server_port_open**
Indicates 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

4.145.1 Detailed Description

The parser interface of the IOAsterisk library.

4.145.2 Constructor & Destructor Documentation

4.145.2.1 `io_asterisk_parser_t()` `io_asterisk_parser_t::io_asterisk_parser_t ()`

Constructor initializes parser variables.

4.145.2.2 `~io_asterisk_parser_t()` `virtual io_asterisk_parser_t::~io_asterisk_parser_t () [inline], [virtual]`

Do-nothing destructor.

4.145.3 Member Function Documentation

4.145.3.1 `get_local_address()` `virtual const std::string& io_asterisk_parser_t::get_local_address () const [inline], [virtual]`

Read parser variable `local_address`, this is the address of the network interface that should listen for incoming connections.

Returns

A string containing the address of the local interface as it was set by the user.

4.145.3.2 get_local_port() `unsigned short io_asterisk_parser_t::get_local_port () const [virtual]`

Read parser variable local_port, this is the TCP port that should be used for incoming connections.

Returns

The local tcp port to listen on as it was chosen by the user. The port number is between MIN_TCP_PORT and MAX_TCP_PORT.

4.145.3.3 set_local_port() `void io_asterisk_parser_t::set_local_port (unsigned short port) [virtual]`

Set parser variable local_port.

This is needed when it was set to 0 before: In this case, the OS chooses a free port for the TCP server socket, and the port that it chose has to be published to the user over the parser interface.

Parameters

<code>port</code>	The TCP port number that is currently used. In the range [MIN_TCP_PORT, MAX_TCP_PORT], excluding 0.
-------------------	---

Precondition

`get_local_port()` (p. 590) currently returns 0.

4.145.3.4 get_server_port_open() `bool io_asterisk_parser_t::get_server_port_open () const [virtual]`

Return the status of the server port as it is known to the parser.

Returns

false after initialization, or the value most recently set via

See also

`set_server_port_open` (p. 591).

4.145.3.5 `set_server_port_open()` void io_asterisk_parser_t::set_server_port_open (bool open) [virtual]

Inform the parser of the current status of the server socket.

Parameters

<code>open</code>	Indicates whether the server socket has just been opened or closed.
-------------------	---

Precondition

`open` may only have the value true if [get_server_port_open\(\) \(p. 591\)](#) currently returns false.

Postcondition

See also

[get_server_port_open \(p. 591\)](#) returns the **value** (p. 49) of `open`.

4.145.3.6 `get_connected()` bool io_asterisk_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 \(p. 592\)](#).

4.145.3.7 `set_connected()` void io_asterisk_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. 592).

Postcondition**See also**

get_connected (p. 592) returns the **value** (p. 49) of open.

4.145.3.8 set_new_peer() `void io_asterisk_parser_t::set_new_peer (`
`unsigned short port,`
`const std::string & host) [virtual]`

Set parser monitor variables `peer_port` and `peer_address`, and calls `set_connected(true)`.

This method should be called when a new connection is established.

Parameters

<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. 592) currently returns false.

Postcondition**See also**

get_connected (p. 592) returns true.

4.145.3.9 debug() `virtual void io_asterisk_parser_t::debug (const std::string & message) [inline], [virtual]`

4.145.4 Member Data Documentation

4.145.4.1 local_address `MHAParser::string_t io_asterisk_parser_t::local_address [private]`

Lets the user set the local network interface to listen on.

4.145.4.2 local_port `MHAParser::int_t io_asterisk_parser_t::local_port [private]`

Lets the user choose the local tcp port to listen on.

4.145.4.3 server_port_open `MHAParser::int_mon_t io_asterisk_parser_t::server_port_open [private]`

Indicates whether the TCP server socket is currently open.

4.145.4.4 connected `MHAParser::int_mon_t io_asterisk_parser_t::connected [private]`

Indicator if there currently is a sound data connection over TCP.

4.145.4.5 peer_address `MHAParser::string_mon_t io_asterisk_parser_t::peer_address [private]`

Display the ip address of the currently connected sound data client.

4.145.4.6 peer_port `MHAParser::int_mon_t io_asterisk_parser_t::peer_port [private]`

Display the tcp port used by the current sound data client.

4.145.4.7 debug_filename `MHAParser::string_t io_asterisk_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

4.145.4.8 debug_file `FILE* io_asterisk_parser_t::debug_file [private]`

file handle to write debugging info to

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

- **MHAIOAsterisk.cpp**

4.146 **io_asterisk_sound_t Class Reference**

Sound data handling of io tcp library.

Public Member Functions

- **io_asterisk_sound_t** (int **fragsize**, float **samplerate**)

Initialize sound data handling.
- virtual ~**io_asterisk_sound_t** ()

Do-nothing destructor.
- virtual void **prepare** (int **num_inchannels**, int **num_outchannels**)

Called during prepare, sets number of audio channels and allocates sound data storage.
- virtual void **release** ()

Called during release.
- virtual int **chunkbytes_in** () const

Number of bytes that constitute one input sound chunk.
- virtual std::string **header** () const

Create the tcp sound header lines.
- std::string & **hton** (const **mha_wave_t** ***s_out**)

Serialize data for network transfer.
- **mha_wave_t** * **ntoh** (const std::string &**data**)

Deserialize data from network.

Private Attributes

- int **fragsize**

Number of sound samples in each channel expected and returned from processing callback.
- float **samplerate**

Sampling rate.
- int **num_inchannels**

Number of input channels.
- int **num_outchannels**
- **MHASignal::waveform_t** * **s_in**

Storage for input signal.
- std::string **output_data**

Serialized data for network transfer.

4.146.1 Detailed Description

Sound data handling of io tcp library.

4.146.2 Constructor & Destructor Documentation

4.146.2.1 **io_asterisk_sound_t()** `io_asterisk_sound_t::io_asterisk_sound_t (`

```
    int fragsize,
    float samplerate )
```

Initialize sound data handling.

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.

4.146.2.2 ~io_asterisk_sound_t() virtual io_asterisk_sound_t::~io_asterisk_sound_t () [inline], [virtual]

Do-nothing destructor.

4.146.3 Member Function Documentation

4.146.3.1 prepare() void io_asterisk_sound_t::prepare (int *num_inchannels*, int *num_outchannels*) [virtual]

Called during prepare, sets number of audio channels and allocates sound data storage.

Parameters

<i>num_inchannels</i>	Number of input audio channels.
<i>num_outchannels</i>	Number of output audio channels.

4.146.3.2 release() void io_asterisk_sound_t::release () [virtual]

Called during release.

Deletes sound data storage.

4.146.3.3 chunkbytes_in() `int io_asterisk_sound_t::chunkbytes_in () const [virtual]`

Number of bytes that constitute one input sound chunk.

Returns

Number of bytes to read from TCP connection before invoking signal processing.

4.146.3.4 header() `std::string io_asterisk_sound_t::header () const [virtual]`

Create the tcp sound header lines.

4.146.3.5 hton() `std::string & io_asterisk_sound_t::hton (const mha_wave_t * s_out)`

Serialize data for network transfer.

4.146.3.6 ntoh() `mha_wave_t * io_asterisk_sound_t::ntoh (const std::string & data)`

Deserialize data from network.

4.146.4 Member Data Documentation**4.146.4.1 fragsize** `int io_asterisk_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

4.146.4.2 samplerate float io_asterisk_sound_t::samplerate [private]

Sampling rate.

Number of samples per second in each channel.

4.146.4.3 num_inchannels int io_asterisk_sound_t::num_inchannels [private]

Number of input channels.

Number of channels expected from and returned by signal processing callback.

4.146.4.4 num_outchannels int io_asterisk_sound_t::num_outchannels [private]**4.146.4.5 s_in** MHASignal::waveform_t* io_asterisk_sound_t::s_in [private]

Storage for input signal.

4.146.4.6 output_data std::string io_asterisk_sound_t::output_data [private]

Serialized data for network transfer.

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

- **MHAIOAsterisk.cpp**

4.147 io_asterisk_t Class Reference

The tcp sound io library.

Public Member Functions

- **io_asterisk_t** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle)
- void **prepare** (int num_inchannels, int num_outchannels)

Allocate server socket and start thread waiting for sound data exchange.
- void **start** ()

Call frameworks start callback if there is a sound data connection at the moment.
- void **stop** ()

Close the current connection if there is one.
- void **release** ()

Close the current connection and close the server socket.
- virtual void **accept_loop** ()

IO thread executes this method.
- virtual void **connection_loop** (**MHA_TCP::Connection** *c)

IO thread executes this method for each connection.
- virtual void **parse** (const char *cmd, char *retval, unsigned int len)

Parser interface.
- virtual ~**io_asterisk_t** ()

Private Attributes

- **io_asterisk_parser_t** parser
- **io_asterisk_sound_t** sound
- **io_asterisk_fwcb_t** fwcb
- **MHA_TCP::Server** * server
- **MHA_TCP::Thread** * thread
- **MHA_TCP::Async_Notify** notify_start
- **MHA_TCP::Async_Notify** notify_stop
- **MHA_TCP::Async_Notify** notify_release

4.147.1 Detailed Description

The tcp sound io library.

4.147.2 Constructor & Destructor Documentation

```
4.147.2.1 io_asterisk_t() io_asterisk_t::io_asterisk_t (
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

```
4.147.2.2 ~io_asterisk_t() virtual io_asterisk_t::~io_asterisk_t ( ) [inline],
[virtual]
```

4.147.3 Member Function Documentation

```
4.147.3.1 prepare() void io_asterisk_t::prepare (
    int num_inchannels,
    int num_outchannels )
```

Allocate server socket and start thread waiting for sound data exchange.

prepare opens the tcp server socket and starts the io thread that listens for audio data on the tcp socket after doing some sanity checks

```
4.147.3.2 start() void io_asterisk_t::start ( )
```

Call frameworks start callback if there is a sound data connection at the moment.

```
4.147.3.3 stop() void io_asterisk_t::stop ( )
```

Close the current connection if there is one.

stop IO thread

4.147.3.4 release() void io_asterisk_t::release ()

Close the current connection and close the server socket.

Stop IO thread and close server socket.

4.147.3.5 accept_loop() void io_asterisk_t::accept_loop () [virtual]

IO thread executes this method.

4.147.3.6 connection_loop() void io_asterisk_t::connection_loop (
MHA_TCP::Connection * c) [virtual]

IO thread executes this method for each connection.

Parameters

c	pointer to connection. connection_loop deletes connection before exiting.
----------	---

4.147.3.7 parse() virtual void io_asterisk_t::parse (
const char * cmd,
char * retval,
unsigned int len) [inline], [virtual]

Parser interface.

4.147.4 Member Data Documentation**4.147.4.1 parser** io_asterisk_parser_t io_asterisk_t::parser [private]

4.147.4.2 sound `io_asterisk_sound_t` `io_asterisk_t::sound` [private]

4.147.4.3 fwcb `io_asterisk_fwcb_t` `io_asterisk_t::fwcb` [private]

4.147.4.4 server `MHA_TCP::Server*` `io_asterisk_t::server` [private]

4.147.4.5 thread `MHA_TCP::Thread*` `io_asterisk_t::thread` [private]

4.147.4.6 notify_start `MHA_TCP::Async_Notify` `io_asterisk_t::notify_start` [private]

4.147.4.7 notify_stop `MHA_TCP::Async_Notify` `io_asterisk_t::notify_stop` [private]

4.147.4.8 notify_release `MHA_TCP::Async_Notify` `io_asterisk_t::notify_release` [private]

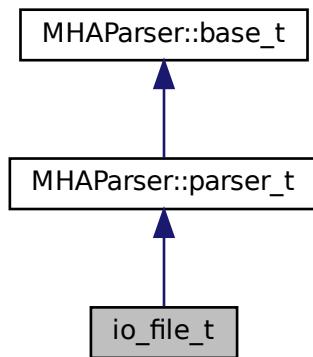
The documentation for this class was generated from the following file:

- **MHAIOAsterisk.cpp**

4.148 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**
- MHParse::string_t **filename_input**
- MHParse::string_t **filename_output**
- MHParse::kw_t **output_sample_format**
- MHParse::int_t **startsample**
- MHParse::int_t **length**
- MHParse::bool_t **strict_channel_match**
- MHParse::bool_t **strict_srstate_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 **sfinfo_in**
- SF_INFO **sfinfo_out**
- sf_count_t **total_read**

Additional Inherited Members

4.148.1 Detailed Description

File IO.

4.148.2 Constructor & Destructor Documentation

```
4.148.2.1 io_file_t() io_file_t::io_file_t (
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

4.148.2.2 ~io_file_t() io_file_t::~io_file_t ()

4.148.3 Member Function Documentation

```
4.148.3.1 prepare() void io_file_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate FILE client and install internal ports.

4.148.3.2 start() void io_file_t::start ()

4.148.3.3 stop() void io_file_t::stop ()

4.148.3.4 release() void io_file_t::release ()

Remove FILE client and deallocate internal ports and buffers.

4.148.3.5 stopped() void io_file_t::stopped (int proc_err, int io_err) [private]

4.148.3.6 setlock() 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.
---------------	--

4.148.4 Member Data Documentation

4.148.4.1 fragsize int io_file_t::fragsize [private]

4.148.4.2 samplerate float io_file_t::samplerate [private]

4.148.4.3 nchannels_in int io_file_t::nchannels_in [private]

4.148.4.4 nchannels_file_in int io_file_t::nchannels_file_in [private]

4.148.4.5 nchannels_out int io_file_t::nchannels_out [private]

4.148.4.6 proc_event IOProcessEvent_t io_file_t::proc_event [private]

4.148.4.7 proc_handle void* io_file_t::proc_handle [private]

4.148.4.8 start_event IOStartedEvent_t io_file_t::start_event [private]

4.148.4.9 start_handle void* io_file_t::start_handle [private]

4.148.4.10 stop_event IOStoppedEvent_t io_file_t::stop_event [private]

4.148.4.11 stop_handle void* io_file_t::stop_handle [private]

4.148.4.12 filename_input MHAParser::string_t io_file_t::filename_input [private]

4.148.4.13 filename_output MHAParser::string_t io_file_t::filename_output [private]

4.148.4.14 output_sample_format `MHAParser::kw_t io_file_t::output_sample_format` [private]

4.148.4.15 startsample `MHAParser::int_t io_file_t::startsample` [private]

4.148.4.16 length `MHAParser::int_t io_file_t::length` [private]

4.148.4.17 strict_channel_match `MHAParser::bool_t io_file_t::strict_channel_match` [private]

4.148.4.18 strict_srate_match `MHAParser::bool_t io_file_t::strict_srate_match` [private]

4.148.4.19 s_in `MHASignal::waveform_t* io_file_t::s_in` [private]

4.148.4.20 s_file_in `MHASignal::waveform_t* io_file_t::s_file_in` [private]

4.148.4.21 s_out `mha_wave_t* io_file_t::s_out` [private]

4.148.4.22 b_prepared `bool io_file_t::b_prepared` [private]

4.148.4.23 sf_in SNDFILE* io_file_t::sf_in [private]

4.148.4.24 sf_out SNDFILE* io_file_t::sf_out [private]

4.148.4.25 sfinf_in SF_INFO io_file_t::sfinf_in [private]

4.148.4.26 sfinf_out SF_INFO io_file_t::sfinf_out [private]

4.148.4.27 total_read sf_count_t io_file_t::total_read [private]

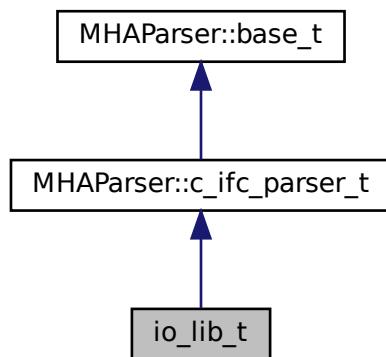
The documentation for this class was generated from the following file:

- **MHAIOFile.cpp**

4.149 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**
- **pluginlib_t lib_handle**
- void * **lib_data**
- **IOInit_t IOInit_cb**
- **IOPrepare_t IOPrepare_cb**
- **IOStart_t IOStart_cb**
- **IOStop_t IOStop_cb**
- **IOResume_t IOResume_cb**
- **IOResume_t IOResume_cb**
- **IOSetVar_t IOSetVar_cb**
- **IOStrError_t IOStrError_cb**
- **IODestroy_t IODestroy_cb**

Additional Inherited Members

4.149.1 Detailed Description

Class for loading MHA sound IO module.

4.149.2 Constructor & Destructor Documentation

4.149.2.1 `io_lib_t()` `io_lib_t::io_lib_t (`

```
    int fragsize,
    float samplerate,
    IOPProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOSoppedEvent_t stop_event,
    void * stop_handle,
    std::string libname )
```

load and initialize MHA sound io module.

4.149.2.2 `~io_lib_t()` `io_lib_t::~io_lib_t ()`

Deinitialize and unload this MHA sound io module.

4.149.3 Member Function Documentation

4.149.3.1 `prepare()` `void io_lib_t::prepare (`

```
    unsigned int inch,
    unsigned int outch )
```

Prepare the sound io module.

After preparation, the sound io module may start the sound processing at any time (external trigger). When the sound processing is started, the sound io module will call the `start_event` callback.

Parameters

<i>inch</i>	number of input channels
<i>outch</i>	number of output channels

4.149.3.2 start() void io_lib_t::start ()

Tell the sound io module to start sound processing.

Some io modules need this, for others that wait for external events this method might do nothing.

4.149.3.3 stop() void io_lib_t::stop ()**4.149.3.4 release()** void io_lib_t::release ()**4.149.3.5 lib_str_error()** std::string io_lib_t::lib_str_error (int err)**4.149.3.6 test_error()** void io_lib_t::test_error () [protected]**4.149.4 Member Data Documentation****4.149.4.1 lib_err** int io_lib_t::lib_err [protected]**4.149.4.2 lib_handle** pluginlib_t io_lib_t::lib_handle [protected]

4.149.4.3 lib_data void* io_lib_t::lib_data [protected]

4.149.4.4 IOInit_cb IOInit_t io_lib_t::IOInit_cb [protected]

4.149.4.5 IOPrepare_cb IOPrepare_t io_lib_t::IOPrepare_cb [protected]

4.149.4.6 IOStart_cb IOStart_t io_lib_t::IOStart_cb [protected]

4.149.4.7 IOStop_cb IOStop_t io_lib_t::IOStop_cb [protected]

4.149.4.8 IOResume_cb IOResume_t io_lib_t::IOResume_cb [protected]

4.149.4.9 IOSetVar_cb IOSetVar_t io_lib_t::IOSetVar_cb [protected]

4.149.4.10 IOStrError_cb IOStrError_t io_lib_t::IOStrError_cb [protected]

4.149.4.11 IODestroy_cb IODestroy_t io_lib_t::IODestroy_cb [protected]

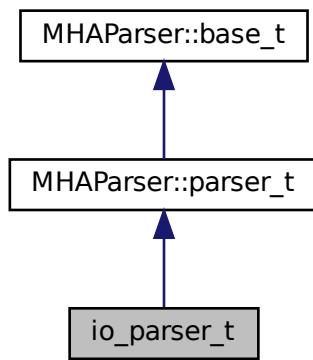
The documentation for this class was generated from the following files:

- **mhafw_lib.h**
- **mhafw_lib.cpp**

4.150 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

4.150.1 Detailed Description

Main class for Parser IO.

4.150.2 Constructor & Destructor Documentation

4.150.2.1 **io_parser_t()** `io_parser_t::io_parser_t (`

```
        unsigned int fragsize,
        IOProcessEvent_t proc_event,
        void * proc_handle,
        IOStartedEvent_t start_event,
        void * start_handle,
        IOStoppedEvent_t stop_event,
        void * stop_handle )
```

4.150.2.2 ~io_parser_t() `io_parser_t::~io_parser_t ()`**4.150.3 Member Function Documentation****4.150.3.1 prepare()** `void io_parser_t::prepare (`
 `int nch_in,`
 `int nch_out)`

Allocate buffers, activate JACK client and install internal ports.

4.150.3.2 start() `void io_parser_t::start ()`**4.150.3.3 stop()** `void io_parser_t::stop ()`**4.150.3.4 release()** `void io_parser_t::release ()`

Remove JACK client and deallocate internal ports and buffers.

4.150.3.5 stopped() `void io_parser_t::stopped (`
 `int proc_err,`
 `int io_err) [private]`**4.150.3.6 started()** `void io_parser_t::started () [private]`

4.150.3.7 process_frame() void io_parser_t::process_frame () [private]

4.150.4 Member Data Documentation

4.150.4.1 fragsize unsigned int io_parser_t::fragsize [private]

4.150.4.2 nchannels_in unsigned int io_parser_t::nchannels_in [private]

4.150.4.3 nchannels_out unsigned int io_parser_t::nchannels_out [private]

4.150.4.4 proc_event IOProcessEvent_t io_parser_t::proc_event [private]

4.150.4.5 proc_handle void* io_parser_t::proc_handle [private]

4.150.4.6 start_event IOStartedEvent_t io_parser_t::start_event [private]

4.150.4.7 start_handle void* io_parser_t::start_handle [private]

4.150.4.8 stop_event IOStoppedEvent_t io_parser_t::stop_event [private]

4.150.4.9 stop_handle void* io_parser_t::stop_handle [private]

4.150.4.10 input MHAParser::mfloat_t io_parser_t::input [private]

4.150.4.11 output MHAParser::mfloat_mon_t io_parser_t::output [private]

4.150.4.12 s_in MHASignal::waveform_t* io_parser_t::s_in [private]

4.150.4.13 s_out mha_wave_t* io_parser_t::s_out [private]

4.150.4.14 b_fw_started bool io_parser_t::b_fw_started [private]

4.150.4.15 b_stopped bool io_parser_t::b_stopped [private]

4.150.4.16 b_prepared bool io_parser_t::b_prepared [private]

4.150.4.17 b_starting bool io_parser_t::b_starting [private]

4.150.4.18 patchbay `MHAEEvents::patchbay_t< io_parser_t> io_parser_t::patchbay`
[private]

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

- `MHAIOParser.cpp`

4.151 `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`

4.151.1 Detailed Description

TCP sound-io library's interface to the framework callbacks.

4.151.2 Constructor & Destructor Documentation

```
4.151.2.1 io_tcp_fwcb_t() io_tcp_fwcb_t::io_tcp_fwcb_t (
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

Constructor stores framework handles and initializes error numbers to 0.

```
4.151.2.2 ~io_tcp_fwcb_t() virtual io_tcp_fwcb_t::~io_tcp_fwcb_t ( ) [inline],
[virtual]
```

Do-nothing destructor.

4.151.3 Member Function Documentation

```
4.151.3.1 start() void io_tcp_fwcb_t::start ( ) [virtual]
```

Call the framework's start callback.

```
4.151.3.2 process() 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.

4.151.3.3 set_errnos() `void io_tcp_fwcb_t::set_errnos (int proc_err, int io_err) [virtual]`

Save error numbers to use during.

See also

stop (p. 622)

Parameters

<i>proc_err</i>	The error number from the
-----------------	---------------------------

See also

process (p. 621) callback.

Parameters

<i>io_err</i>	The error number from the io library itself.
---------------	--

4.151.3.4 stop() `void io_tcp_fwcb_t::stop () [virtual]`

Call the frameworks stop callback.

Uses the error numbers set previously with

See also

[set_errnos](#) (p. 622).

4.151.4 Member Data Documentation**4.151.4.1 proc_event IOProcessEvent_t io_tcp_fwcb_t::proc_event [private]**

Pointer to signal processing callback function.

4.151.4.2 start_event IOStartedEvent_t io_tcp_fwcb_t::start_event [private]

Pointer to start notification callback function.

Called when a new TCP connection is established or the user issues the start command while there is a connection.

4.151.4.3 stop_event IOStoppedEvent_t io_tcp_fwcb_t::stop_event [private]

Pointer to stop notification callback function.

Called when the connection is closed.

4.151.4.4 proc_handle void* io_tcp_fwcb_t::proc_handle [private]

Handles belonging to framework.

4.151.4.5 start_handle void * io_tcp_fwcb_t::start_handle [private]**4.151.4.6 stop_handle void * io_tcp_fwcb_t::stop_handle [private]**

4.151.4.7 proc_err int io_tcp_fwcbs_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. 622) from that callback. Within **stop** (p. 622), these error numbers are read again and transmitted to the framework.

4.151.4.8 io_err int io_tcp_fwcbs_t::io_err [private]

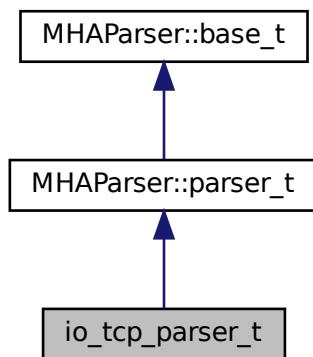
The documentation for this class was generated from the following file:

- **MHAIOTCP.cpp**

4.152 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

4.152.1 Detailed Description

The parser interface of the IOTCP library.

4.152.2 Constructor & Destructor Documentation

4.152.2.1 `io_tcp_parser_t()` `io_tcp_parser_t::io_tcp_parser_t ()`

Constructor initializes parser variables.

4.152.2.2 `~io_tcp_parser_t()` `virtual io_tcp_parser_t::~io_tcp_parser_t () [inline], [virtual]`

Do-nothing destructor.

4.152.3 Member Function Documentation

4.152.3.1 `get_local_address()` `virtual const std::string& io_tcp_parser_t::get_local_address () const [inline], [virtual]`

Read parser variable `local_address`, this is the address of the network interface that should listen for incoming connections.

Returns

A string containing the address of the local interface as it was set by the user.

4.152.3.2 get_local_port() `unsigned short io_tcp_parser_t::get_local_port () const [virtual]`

Read parser variable local_port, this is the TCP port that should be used for incoming connections.

Returns

The local tcp port to listen on as it was chosen by the user. The port number is between MIN_TCP_PORT and MAX_TCP_PORT.

4.152.3.3 set_local_port() `void io_tcp_parser_t::set_local_port (unsigned short port) [virtual]`

Set parser variable local_port.

This is needed when it was set to 0 before: In this case, the OS chooses a free port for the TCP server socket, and the port that it chose has to be published to the user via the parser interface.

Parameters

<code>port</code>	The TCP port number that is currently used. In the range [MIN_TCP_PORT, MAX_TCP_PORT], excluding 0.
-------------------	---

Precondition

`get_local_port()` (p. 626) currently returns 0.

4.152.3.4 get_server_port_open() `bool io_tcp_parser_t::get_server_port_open () const [virtual]`

Return the status of the server port as it is known to the parser.

Returns

false after initialization, or the value most recently set via

See also

`set_server_port_open` (p. 627).

4.152.3.5 `set_server_port_open()` `void io_tcp_parser_t::set_server_port_open (bool open) [virtual]`

Inform the parser of the current status of the server socket.

Parameters

<code>open</code>	Indicates whether the server socket has just been opened or closed.
-------------------	---

Precondition

`open` may only have the value true if [get_server_port_open\(\) \(p. 627\)](#) currently returns false.

Postcondition

See also

[get_server_port_open \(p. 627\)](#) returns the **value** (p. 49) of `open`.

4.152.3.6 `get_connected()` `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 \(p. 628\)](#).

4.152.3.7 `set_connected()` `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. 628).

Postcondition**See also**

get_connected (p. 628) returns the **value** (p. 49) of open.

4.152.3.8 set_new_peer() void io_tcp_parser_t::set_new_peer (unsigned short *port*, const std::string & *host*) [virtual]

Set parser monitor variables *peer_port* and *peer_address*, and calls *set_connected(true)*.

This method should be called when a new connection is established.

Parameters

<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. 628) currently returns false.

Postcondition**See also**

get_connected (p. 628) returns true.

4.152.3.9 debug() `virtual void io_tcp_parser_t::debug (const std::string & message) [inline], [virtual]`

4.152.4 Member Data Documentation

4.152.4.1 local_address `MHAParser::string_t io_tcp_parser_t::local_address [private]`

Lets the user set the local network interface to listen on.

4.152.4.2 local_port `MHAParser::int_t io_tcp_parser_t::local_port [private]`

Lets the user choose the local tcp port to listen on.

4.152.4.3 server_port_open `MHAParser::int_mon_t io_tcp_parser_t::server_port_open [private]`

Indicates whether the TCP server socket is currently open.

4.152.4.4 connected `MHAParser::int_mon_t io_tcp_parser_t::connected [private]`

Indicator if there currently is a sound data connection over TCP.

4.152.4.5 peer_address `MHAParser::string_mon_t io_tcp_parser_t::peer_address [private]`

Display the ip address of the currently connected sound data client.

4.152.4.6 peer_port `MHAParser::int_mon_t io_tcp_parser_t::peer_port [private]`

Display the tcp port used by the current sound data client.

4.152.4.7 debug_filename `MHAParser::string_t io_tcp_parser_t::debug_filename [private]`

filename to write debugging info to (if non-empty)

4.152.4.8 debug_file `FILE* io_tcp_parser_t::debug_file [private]`

file handle to write debugging info to

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

- **MHAIOTCP.cpp**

4.153 **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. 839) 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.

4.153.1 Detailed Description

Sound data handling of io tcp library.

4.153.2 Constructor & Destructor Documentation

4.153.2.1 `io_tcp_sound_t()` `io_tcp_sound_t::io_tcp_sound_t (`
`int fragsize,`
`float samplerate)`

Initialize sound data handling.

Checks sound data type by calling

See also

check_sound_data_type (p. 633).

Parameters

<code><i>fragsize</i></code>	Number of sound samples in each channel expected and returned from processing callback.
<code><i>samplerate</i></code>	Number of samples per second in each channel.

4.153.2.2 `~io_tcp_sound_t()` `virtual io_tcp_sound_t::~io_tcp_sound_t () [inline],`
`[virtual]`

Do-nothing destructor.

4.153.3 Member Function Documentation

4.153.3.1 `check_sound_data_type()` `void io_tcp_sound_t::check_sound_data_type ()`
`[static], [private]`

Check if mha_real_t is a usable 32-bit floating point type.

Exceptions

MHA_Error (p. 763)	if mha_real_t is not compatible to 32-bit float.
---------------------------	--

4.153.3.2 prepare() void io_tcp_sound_t::prepare (int num_inchannels, int num_outchannels) [virtual]

Called during prepare, sets number of audio channels and allocates sound data storage.

Parameters

<i>num_inchannels</i>	Number of input audio channels.
<i>num_outchannels</i>	Number of output audio channels.

4.153.3.3 release() void io_tcp_sound_t::release () [virtual]

Called during release.

Deletes sound data storage.

4.153.3.4 chunkbytes_in() int io_tcp_sound_t::chunkbytes_in () const [virtual]

Number of bytes that constitute one input sound chunk.

Returns

Number of bytes to read from TCP connection before invoking signal processing.

4.153.3.5 header() std::string io_tcp_sound_t::header () const [virtual]

Create the tcp sound header lines.

4.153.3.6 ntoh() mha_wave_t * io_tcp_sound_t::ntoh (const std::string & data) [virtual]

Copy data received from tcp into **mha_wave_t** (p. 839) structure.

Doing network-to-host byte order swapping in the process.

Parameters

<i>data</i>	One chunk (
-------------	-------------

See also

chunkbytes_in (p. 634)) of sound data to process.

Returns

Pointer to the sound data storage.

4.153.3.7 hton() `std::string io_tcp_sound_t::hton (`
`const mha_wave_t * s_out) [virtual]`

Copy sound data from the output sound structure to a string.

Doing host-to-network byte order swapping while at it.

Parameters

<i>s_out</i>	Pointer to the storage of the sound to put out.
--------------	---

Returns

The sound data in network byte order.

4.153.4 Member Data Documentation

4.153.4.1 fragsize `int io_tcp_sound_t::fragsize [private]`

Number of sound samples in each channel expected and returned from processing callback.

4.153.4.2 samplerate float io_tcp_sound_t::samplerate [private]

Sampling rate.

Number of samples per second in each channel.

4.153.4.3 num_inchannels int io_tcp_sound_t::num_inchannels [private]

Number of input channels.

Number of channels expected from and returned by signal processing callback.

4.153.4.4 num_outchannels int io_tcp_sound_t::num_outchannels [private]**4.153.4.5 s_in** MHASignal::waveform_t* io_tcp_sound_t::s_in [private]

Storage for input signal.

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

- **MHAIOTCP.cpp**

4.154 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]

4.154.1 Detailed Description

This union helps in conversion of floats from host byte order to network byte order and back again.

4.154.2 Member Data Documentation

4.154.2.1 f float io_tcp_sound_t::float_union::f

4.154.2.2 i unsigned int io_tcp_sound_t::float_union::i

4.154.2.3 c char io_tcp_sound_t::float_union::c[4]

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

- **MHAIOTCP.cpp**

4.155 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`

4.155.1 Detailed Description

The tcp sound io library.

4.155.2 Constructor & Destructor Documentation

4.155.2.1 `io_tcp_t()` `io_tcp_t::io_tcp_t (`
 `int fragsize,`
 `float samplerate,`
 `IOPProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle)`

4.155.2.2 `~io_tcp_t()` `virtual io_tcp_t::~io_tcp_t () [inline], [virtual]`

4.155.3 Member Function Documentation

```
4.155.3.1 prepare() void io_tcp_t::prepare (
    int num_inchannels,
    int num_outchannels )
```

Allocate server socket and start thread waiting for sound data exchange.

prepare opens the tcp server socket and starts the io thread that listens for audio data on the tcp socket after doing some sanity checks

```
4.155.3.2 start() void io_tcp_t::start ( )
```

Call frameworks start callback if there is a sound data connection at the moment.

```
4.155.3.3 stop() void io_tcp_t::stop ( )
```

Close the current connection if there is one.

stop IO thread

```
4.155.3.4 release() void io_tcp_t::release ( )
```

Close the current connection and close the server socket.

Stop IO thread and close server socket.

```
4.155.3.5 accept_loop() void io_tcp_t::accept_loop ( ) [virtual]
```

IO thread executes this method.

```
4.155.3.6 connection_loop() void io_tcp_t::connection_loop (
    MHA_TCP::Connection * c ) [virtual]
```

IO thread executes this method for each connection.

Parameters

<i>c</i>	pointer to connection. connection_loop deletes connection before exiting.
----------	---

```
4.155.3.7 parse() virtual void io_tcp_t::parse (
    const char * cmd,
    char * retval,
    unsigned int len ) [inline], [virtual]
```

Parser interface.

4.155.4 Member Data Documentation

```
4.155.4.1 parser io_tcp_parser_t io_tcp_t::parser [private]
```

```
4.155.4.2 sound io_tcp_sound_t io_tcp_t::sound [private]
```

```
4.155.4.3 fwcb io_tcp_fwcb_t io_tcp_t::fwcb [private]
```

```
4.155.4.4 server MHA_TCP::Server* io_tcp_t::server [private]
```

```
4.155.4.5 thread MHA_TCP::Thread* io_tcp_t::thread [private]
```

4.155.4.6 notify_start `MHA_TCP::Async_Notify` `io_tcp_t::notify_start` [private]

4.155.4.7 notify_stop `MHA_TCP::Async_Notify` `io_tcp_t::notify_stop` [private]

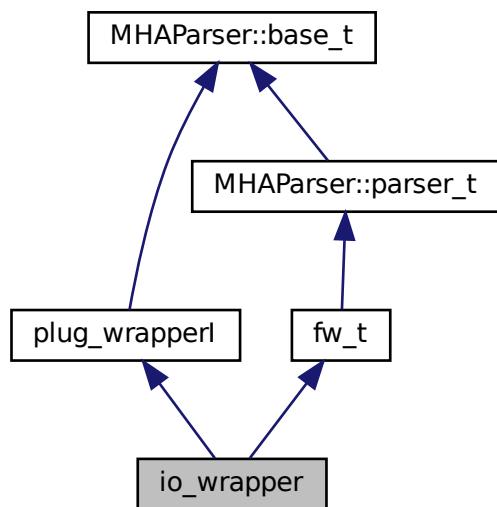
4.155.4.8 notify_release `MHA_TCP::Async_Notify` `io_tcp_t::notify_release` [private]

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

- `MHAIOTCP.cpp`

4.156 io_wrapper Class Reference

Inheritance diagram for `io_wrapper`:



Public Member Functions

- **io_wrapper (algo_comm_t iac, const std::string &libname)**
- virtual ~**io_wrapper ()**=default
- virtual std::vector< std::string > **get_categories ()**
- virtual std::string **parse (const std::string &str)**
- virtual bool **has_parser ()**
- virtual std::string **get_documentation ()**
- virtual bool **has_process (mha_domain_t in, mha_domain_t out)**

Additional Inherited Members

4.156.1 Constructor & Destructor Documentation

4.156.1.1 io_wrapper() `io_wrapper::io_wrapper (algo_comm_t iac, const std::string & libname) [inline]`

4.156.1.2 ~io_wrapper() `virtual io_wrapper::~io_wrapper () [virtual], [default]`

4.156.2 Member Function Documentation

4.156.2.1 get_categories() `virtual std::vector<std::string> io_wrapper::get_categories () [inline], [virtual]`

Implements **plug_wrapperl** (p. [1350](#)).

4.156.2.2 parse() `virtual std::string io_wrapper::parse (const std::string & str) [inline], [virtual]`

Implements **plug_wrapperl** (p. [1350](#)).

4.156.2.3 has_parser() virtual bool io_wrapper::has_parser () [inline], [virtual]

Implements **plug_wrapperl** (p. 1350).

4.156.2.4 get_documentation() virtual std::string io_wrapper::get_documentation () [inline], [virtual]

Implements **plug_wrapperl** (p. 1350).

4.156.2.5 has_process() virtual bool io_wrapper::has_process (mha_domain_t in, mha_domain_t out) [inline], [virtual]

Implements **plug_wrapperl** (p. 1351).

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

- **generatemhaplugindoc.cpp**

4.157 **latex_doc_t** Class Reference

Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUMENTATION macro.

Public Member Functions

- **latex_doc_t** (const std::string & **pluginname**, const std::string & **plugin_macro**)
Constructor loads the plugin into this process.
- std::string **get_latex_doc** ()
This method accesses the compiled-in contents of the MHAPLUGIN_DOCUMENTATION macro and the exported interface functions of the loaded plugin to produce latex documentation for the plugin.
- std::string **get_main_category** () const
- std::vector< std::string > **get_categories** () const

Private Member Functions

- std::string **strdom** (mha_domain_t d) const
- std::string **get_ac** (MHAKernel::algo_comm_class_t & ac, std::string txt) const
- std::string **parsername** (std::string s) const
- std::string **get_parser_var** (MHAParser::base_t *p, std::string name) const
- std::string **get_parser_tab** (MHAParser::base_t *p, const std::string &prefix, const std::string &latex_macro) const

Private Attributes

- const std::string **plugname**
- const std::string **latex_plugname**
- **MHAKernel::algo_comm_class_t ac**
- std::unique_ptr< **plug_wrapperl** > **loader**
- const std::string **plugin_macro**

4.157.1 Detailed Description

Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUMENTATION macro.

4.157.2 Constructor & Destructor Documentation

4.157.2.1 **latex_doc_t()** `latex_doc_t::latex_doc_t (` `const std::string & plugname,` `const std::string & plugin_macro)`

Constructor loads the plugin into this process.

Parameters

<i>plugname</i>	Name of the MHA plugin to process
<i>plugin_macro</i>	name of the LaTeX section macro that documents a single plugin (e.g. "section", "subsection", "subsubsection", ...)

4.157.3 Member Function Documentation

4.157.3.1 **get_latex_doc()** std::string latex_doc_t::get_latex_doc ()

This method accesses the compiled-in contents of the MHAPLUGIN_DOCUMENTATION macro and the exported interface functions of the loaded plugin to produce latex documentation for the plugin.

It tentatively prepares the plugin for processing and checks the AC variables registered by the plugin.

Returns

the complete latex documentation for this plugin

4.157.3.2 **get_main_category()** std::string latex_doc_t::get_main_category () const

Returns

the first word of the categories string in the MHAPLUGIN_DOCUMENTATION macro

4.157.3.3 **get_categories()** std::vector< std::string > latex_doc_t::get_categories () const

Returns

a vector of all words in the categories string in the MHAPLUGIN_DOCUMENTATION macro

4.157.3.4 **strdom()** std::string latex_doc_t::strdom (mha_domain_t d) const [private]

4.157.3.5 `get_ac()` std::string latex_doc_t::get_ac (MHAKernel::algo_comm_class_t & ac, std::string txt) const [private]

4.157.3.6 `parsername()` std::string latex_doc_t::parsername (std::string s) const [private]

4.157.3.7 `get_parser_var()` std::string latex_doc_t::get_parser_var (MHAParser::base_t * p, std::string name) const [private]

4.157.3.8 `get_parser_tab()` std::string latex_doc_t::get_parser_tab (MHAParser::base_t * p, const std::string & prefix, const std::string & latex_macro) const [private]

4.157.4 Member Data Documentation

4.157.4.1 `plugname` const std::string latex_doc_t::plugname [private]

4.157.4.2 `latex_plugname` const std::string latex_doc_t::latex_plugname [private]

4.157.4.3 `ac` MHAKernel::algo_comm_class_t latex_doc_t::ac [private]

4.157.4.4 loader std::unique_ptr< **plug_wrapperI**> latex_doc_t::loader [private]

4.157.4.5 plugin_macro const std::string latex_doc_t::plugin_macro [private]

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

- **generatemhaplugindoc.cpp**

4.158 level_matching::channel_pair Class Reference

Public Member Functions

- **channel_pair** (const std::pair< int, int > &idx_, **mha_real_t** filter_rate_, **mha_real_t** mismatch_time_constant_)
Channel pair ctor.
- **channel_pair** (int idx1_, int idx2_, **mha_real_t** filter_rate_, **mha_real_t** mismatch_time_constant_)
Channel pair ctor.
- **mha_real_t update_mismatch** (const **mha_wave_t** &signal_)
Calculates the filtered rms level mismatch.
- **mha_real_t update_mismatch** (const **mha_spec_t** &signal_, unsigned fftlen_)
Calculates the filtered rms level mismatch.
- **mha_real_t get_mismatch** () const
Get last filter result.
- const std::pair< int, int > & **get_idx** () const

Private Attributes

- std::pair< int, int > **idx**
Indices of channels.
- **MHAFilter::o1flt_lowpass_t mismatch**
Low-pass filtered level mismatch.

4.158.1 Constructor & Destructor Documentation

4.158.1.1 channel_pair() [1/2] level_matching::channel_pair::channel_pair (const std::pair< int, int > & idx_, **mha_real_t** filter_rate_, **mha_real_t** mismatch_time_constant_)

Channel pair ctor.

Parameters

<i>idx_</i>	Pair of channel indices
<i>filter_rate_</i>	Filter rate of low pass filter
<i>mismatch_time_</i> ↵ <i>constant_</i>	Time constant of low pass filter

4.158.1.2 channel_pair() [2/2] `level_matching::channel_pair::channel_pair (`
 `int idx1_,`
 `int idx2_,`
 `mha_real_t filter_rate_,`
 `mha_real_t mismatch_time_constant_)`

Channel pair ctor.

Parameters

<i>idx1</i>	First channel index. Used as reference
<i>idx2</i>	Second channel index
<i>filter_rate_</i>	Filter rate of low pass filter
<i>mismatch_time_</i> ↵ <i>constant_</i>	Time constant of low pass filter

4.158.2 Member Function Documentation

4.158.2.1 update_mismatch() [1/2] `mha_real_t level_matching::channel_pair::update←`
`_mismatch (`
 `const mha_wave_t & signal_)`

Calculates the filtered rms level mismatch.

Parameters

<i>signal</i>	Input signal
---------------	--------------

Returns

Low-pass filtered mismatch ratio

```
4.158.2.2 update_mismatch() [2/2] mha_real_t level_matching::channel_pair::update←
_mismatch (
    const mha_spec_t & signal_,
    unsigned fftlen_ )
```

Calculates the filtered rms level mismatch.

Parameters

<i>signal</i>	Input signal
---------------	--------------

Returns

Low-pass filtered mismatch ratio

```
4.158.2.3 get_mismatch() mha_real_t level_matching::channel_pair::get_mismatch (
) const
```

Get last filter result.

Returns

Last filter result

```
4.158.2.4 get_idx() const std::pair<int,int>& level_matching::channel_pair::get_idx
( ) const [inline]
```

4.158.3 Member Data Documentation

4.158.3.1 **idx** std::pair<int,int> level_matching::channel_pair::idx [private]

Indices of channels.

4.158.3.2 **mismatch** MHAFilter::o1flt_lowpass_t level_matching::channel_pair::mismatch [private]

Low-pass filtered level mismatch.

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

- **level_matching.hh**
- **level_matching.cpp**

4.159 level_matching::level_matching_config_t Class Reference

Public Member Functions

- **level_matching_config_t** (const **mhaconfig_t** &cfg_, const std::vector< std::vector< int >> &pairs_, **mha_real_t** signal_lp_fpass_, **mha_real_t** signal_fstop_, unsigned signal_lp_order_, **mha_real_t** level_tau_, **mha_real_t** range_)
RT-config c'tor.
- **~level_matching_config_t** ()=default
- **mha_wave_t * process** (**mha_wave_t** *)
- **mha_spec_t * process** (**mha_spec_t** *)

Private Attributes

- **MHASignal::waveform_t tmp**
Temporary storage for input signal to use waveform_t helper functions.
- **std::vector< channel_pair > pairings**
Channel pairings.
- **MHAFilter::iir_filter_state_t lp**
Nth-order low pass filter used to exclude high frequencies from level matching.
- **mha_real_t range**
Maximum matching range. Maximum adjustment done.
- **unsigned fftlen**
fftlen in spectral domain

4.159.1 Constructor & Destructor Documentation

```
4.159.1.1 level_matching_config_t() level_matching::level_matching_config_t::level←
←_matching_config_t (
    const mhaconfig_t & cfg_,
    const std::vector< std::vector< int >> & pairs_,
    mha_real_t signal_lp_fpass_,
    mha_real_t signal_lp_fstop_,
    unsigned signal_lp_order_,
    mha_real_t level_tau_,
    mha_real_t range_ )
```

RT-config c'tor.

Parameters

<i>cfg</i>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
------------	---

Precondition

Size of param pairs must be two

Parameters

<i>pairs</i> _	Matrix containing the channel indices of the microphone pairs. Two entries per row.
<i>signal_tau</i>	Time constant of the signal low pass filter in seconds.
<i>level_tau</i>	Time constant of the level mismatch averaging filter in seconds.

```
4.159.1.2 ~level_matching_config_t() level_matching::level_matching_config_t::~level←
←_matching_config_t ( ) [default]
```

4.159.2 Member Function Documentation

4.159.2.1 process() [1/2] `mha_wave_t * level_matching::level_matching_config_t::process (mha_wave_t * s)`

4.159.2.2 process() [2/2] `mha_spec_t * level_matching::level_matching_config_t::process (mha_spec_t * s)`

4.159.3 Member Data Documentation

4.159.3.1 tmp `MHASignal::waveform_t level_matching::level_matching_config_t::tmp` [private]

Temporary storage for input signal to use waveform_t helper functions.

4.159.3.2 pairings `std::vector< channel_pair > level_matching::level_matching_config_t::pairings` [private]

Channel pairings.

4.159.3.3 lp `MHAFilter::iir_filter_state_t level_matching::level_matching_config_t::lp` [private]

Nth-order low pass filter used to exclude high frequencies from level matching.

4.159.3.4 range `mha_real_t level_matching::level_matching_config_t::range` [private]

Maximum matching range. Maximum adjustment done.

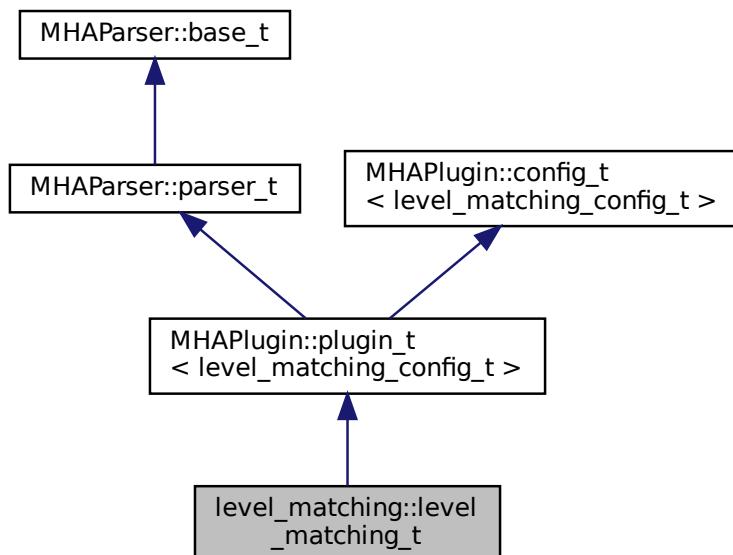
4.159.3.5 fftlen `unsigned level_matching::level_matching_config_t::ffflen` [private]
 fftlen in spectral domain

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

- `level_matching.hh`
- `level_matching.cpp`

4.160 level_matching::level_matching_t Class Reference

Inheritance diagram for `level_matching::level_matching_t`:



Public Member Functions

- `level_matching_t (algo_comm_t & ac, const std::string &chain_name, const std::string &algo_name)`
Plugin interface ctor.
- `~level_matching_t ()`
- `mha_wave_t * process (mha_wave_t *)`
Processing callback.
- `mha_spec_t * process (mha_spec_t *)`
Processing callback.
- `void prepare (mhaconfig_t &)`
Plugin preparation callback.
- `void release (void)`

Private Member Functions

- void **update_cfg ()**

Private Attributes

- **MHAParser::mint_t channels** ={"channels","[[0 1]]"}
- **MHAParser::float_t range** ={"Maximum matching range in dB","4","[0,["]}
- **MHAParser::float_t lp_signal_fpass** ={"Upper edge of the lp pass band for the signal in Hz","4000","[0,["]}
- **MHAParser::float_t lp_signal_fstop** ={"Stop band lower edge frequency for the signal in Hz","8000","[0,["]}
- **MHAParser::float_t lp_signal_order** ={"Signal lp order","24","[1,["]}
- **MHAParser::float_t lp_level_tau** ={"Low pass time constant for the mismatch in s","1","[0,["]}
- **MHAEvents::patchbay_t< level_matching_t > patchbay**

Additional Inherited Members

4.160.1 Constructor & Destructor Documentation

4.160.1.1 `level_matching_t()` `level_matching::level_matching_t::level_matching_t (algo_comm_t & ac,`
`const std::string & chain_name,`
`const std::string & algo_name)`

Plugin interface ctor.

Parameters

<i>ac</i>	
-----------	--

4.160.1.2 `~level_matching_t()` `level_matching::level_matching_t::~level_matching_t ()`

4.160.2 Member Function Documentation

4.160.2.1 process() [1/2] `mha_wave_t * level_matching::level_matching_t::process (mha_wave_t * signal_)`

Processing callback.

Parameters

<code>signal</code>	Input signal
---------------------	--------------

4.160.2.2 process() [2/2] `mha_spec_t * level_matching::level_matching_t::process (mha_spec_t * signal_)`

Processing callback.

Parameters

<code>signal</code>	Input signal
---------------------	--------------

4.160.2.3 prepare() `void level_matching::level_matching_t::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation callback.

Parameters

<code>signal_info</code>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------------	---

Implements **MHAPlugin::plugin_t< level_matching_config_t >** (p. 1148).

4.160.2.4 release() `void level_matching::level_matching_t::release (void) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< level_matching_config_t >** (p. 1149).

4.160.2.5 update_cfg() void level_matching::level_matching_t::update_cfg () [private]

4.160.3 Member Data Documentation

4.160.3.1 channels **MHAParser::mint_t** level_matching::level_matching_t::channels
= {"channels", "[[0 1]]"} [private]

4.160.3.2 range **MHAParser::float_t** level_matching::level_matching_t::range = {"Maximum matching range in dB", "4", "[0,["} [private]

4.160.3.3 lp_signal_fpass **MHAParser::float_t** level_matching::level_matching_t::lp_signal_fpass = {"Upper edge of the lp pass band for the signal in Hz", "4000", "[0,["} [private]

4.160.3.4 lp_signal_fstop **MHAParser::float_t** level_matching::level_matching_t::lp_signal_fstop = {"Stop band lower edge frequency for the signal in Hz", "8000", "[0,["} [private]

4.160.3.5 lp_signal_order **MHAParser::float_t** level_matching::level_matching_t::lp_signal_order = {"Signal lp order", "24", "[1,["} [private]

4.160.3.6 lp_level_tau **MHAParser::float_t** level_matching::level_matching_t::lp_level_tau = {"Low pass time constant for the mismatch in s", "1", "[0,["} [private]

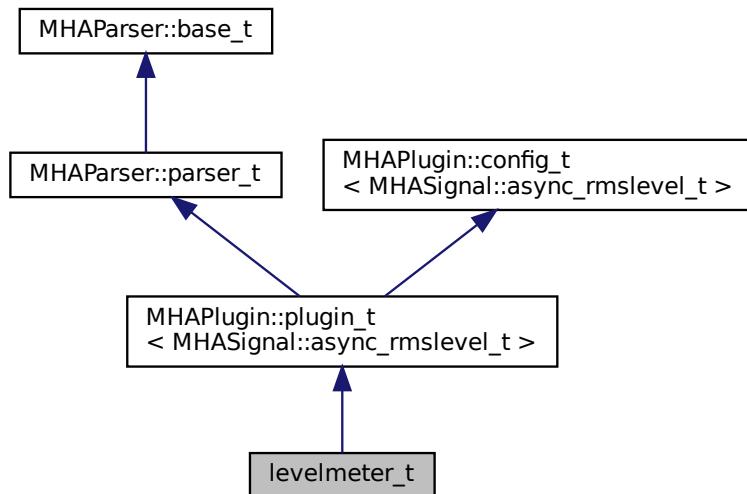
4.160.3.7 patchbay `MHAEEvents::patchbay_t< level_matching_t> level_matching<= ::level_matching_t::patchbay [private]`

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

- `level_matching.hh`
- `level_matching.cpp`

4.161 levelmeter_t Class Reference

Inheritance diagram for levelmeter_t:



Public Member Functions

- `levelmeter_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_tau ()`
- `void query_rms ()`
- `void query_peak ()`

Private Attributes

- **MHAParser::float_t tau**
- **MHAParser::kw_t mode**
- **MHAParser::vfloat_mon_t rms**
- **MHAParser::vfloat_mon_t peak**
- **MHAEvents::patchbay_t< levelmeter_t > patchbay**

Additional Inherited Members**4.161.1 Constructor & Destructor Documentation**

4.161.1.1 levelmeter_t() `levelmeter_t::levelmeter_t (`
`const algo_comm_t & iac,`
`const std::string & ,`
`const std::string)`

4.161.2 Member Function Documentation

4.161.2.1 process() `mha_wave_t * levelmeter_t::process (`
`mha_wave_t * s)`

4.161.2.2 prepare() `void levelmeter_t::prepare (`
`mhaconfig_t & cf) [virtual]`

Implements **MHAPlugin::plugin_t< MHASignal::async_rmslevel_t >** (p. [1148](#)).

4.161.2.3 update_tau() `void levelmeter_t::update_tau () [private]`

4.161.2.4 query_rms() void levelmeter_t::query_rms () [private]

4.161.2.5 query_peak() void levelmeter_t::query_peak () [private]

4.161.3 Member Data Documentation

4.161.3.1 tau **MHAParser::float_t** levelmeter_t::tau [private]

4.161.3.2 mode **MHAParser::kw_t** levelmeter_t::mode [private]

4.161.3.3 rms **MHAParser::vfloat_mon_t** levelmeter_t::rms [private]

4.161.3.4 peak **MHAParser::vfloat_mon_t** levelmeter_t::peak [private]

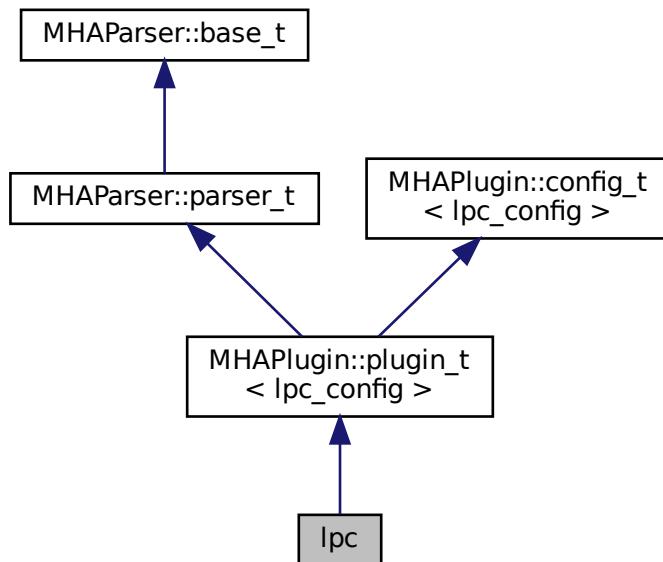
4.161.3.5 patchbay **MHAEvents::patchbay_t< levelmeter_t>** levelmeter_t::patchbay [private]

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

- **levelmeter.cpp**

4.162 Ipc Class Reference

Inheritance diagram for Ipc:



Public Member Functions

- **Ipc (algo_comm_t & ac, const std::string &chain_name, const std::string & algo_name)**
Constructs our plugin.
- **~Ipc ()**
- **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

4.162.1 Constructor & Destructor Documentation

```
4.162.1.1 lpc() lpc::lpc (
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs our plugin.

```
4.162.1.2 ~lpc() lpc::~lpc ( )
```

4.162.2 Member Function Documentation

```
4.162.2.1 process() mha_wave_t * lpc::process (
    mha_wave_t * signal )
```

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

```
4.162.2.2 prepare() 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. 1148).

4.162.2.3 release() void lpc::release (
void) [inline], [virtual]

Reimplemented from **MHAPlugIn::plugin_t< lpc_config >** (p. 1149).

4.162.2.4 update_cfg() void lpc::update_cfg () [private]

4.162.3 Member Data Documentation

4.162.3.1 algo_name std::string lpc::algo_name [private]

4.162.3.2 lpc_order MHAParser::int_t lpc::lpc_order [private]

4.162.3.3 lpc_buffer_size MHAParser::int_t lpc::lpc_buffer_size [private]

4.162.3.4 shift MHAParser::bool_t lpc::shift [private]

4.162.3.5 comp_each_iter `MHAParser::int_t lpc::comp_each_iter [private]`

4.162.3.6 norm `MHAParser::bool_t lpc::norm [private]`

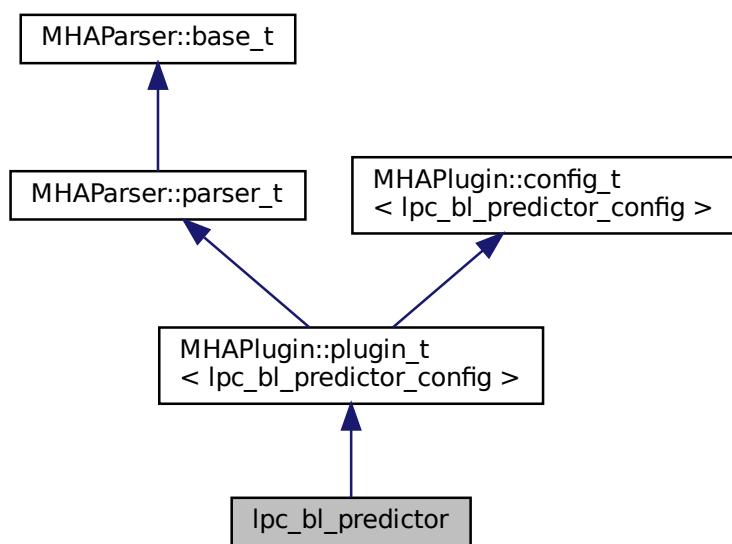
4.162.3.7 patchbay `MHAEEvents::patchbay_t< lpc> lpc::patchbay [private]`

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

- [lpc.h](#)
- [lpc.cpp](#)

4.163 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

4.163.1 Constructor & Destructor Documentation

4.163.1.1 lpc_bl_predictor() `lpc_bl_predictor::lpc_bl_predictor (`
`algo_comm_t & ac,`
`const std::string & chain_name,`
`const std::string & algo_name)`

Constructs our plugin.

4.163.1.2 ~lpc_bl_predictor() `lpc_bl_predictor::~lpc_bl_predictor ()`**4.163.2 Member Function Documentation****4.163.2.1 process()** `mha_wave_t * lpc_bl_predictor::process (mha_wave_t * signal)`

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

4.163.2.2 prepare() `void lpc_bl_predictor::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	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. [1148](#)).

4.163.2.3 release() `void lpc_bl_predictor::release (void) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< lpc_bl_predictor_config >** (p. [1149](#)).

4.163.2.4 update_cfg() `void lpc_bl_predictor::update_cfg () [private]`

4.163.3 Member Data Documentation

4.163.3.1 lpc_order `MHAParser::int_t lpc_bl_predictor::lpc_order`

4.163.3.2 name_kappa `MHAParser::string_t lpc_bl_predictor::name_kappa`

4.163.3.3 name_lpc_f `MHAParser::string_t lpc_bl_predictor::name_lpc_f`

4.163.3.4 name_lpc_b `MHAParser::string_t lpc_bl_predictor::name_lpc_b`

4.163.3.5 name_f `MHAParser::string_t lpc_bl_predictor::name_f`

4.163.3.6 name_b `MHAParser::string_t lpc_bl_predictor::name_b`

4.163.3.7 patchbay `MHAEEvents::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`

4.164 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`

4.164.1 Constructor & Destructor Documentation

4.164.1.1 lpc_bl_predictor_config() `lpc_bl_predictor_config::lpc_bl_predictor_config (algo_comm_t & iac,`
`const mhaconfig_t in_cfg,`
`lpc_bl_predictor * _lpc)`

4.164.1.2 ~lpc_bl_predictor_config() `lpc_bl_predictor_config::~lpc_bl_predictor_config ()`

4.164.2 Member Function Documentation

4.164.2.1 process() `mha_wave_t * lpc_bl_predictor_config::process (`
`mha_wave_t * wave)`

4.164.3 Member Data Documentation

4.164.3.1 ac `algo_comm_t lpc_bl_predictor_config::ac` [private]

4.164.3.2 f_est `MHA_AC::waveform_t lpc_bl_predictor_config::f_est` [private]

4.164.3.3 b_est `MHA_AC::waveform_t lpc_bl_predictor_config::b_est` [private]

4.164.3.4 forward `MHASignal::waveform_t lpc_bl_predictor_config::forward` [private]

4.164.3.5 backward `MHASignal::waveform_t lpc_bl_predictor_config::backward` [private]

4.164.3.6 lpc_order `int lpc_bl_predictor_config::lpc_order` [private]

4.164.3.7 name_km `std::string lpc_bl_predictor_config::name_km` [private]

4.164.3.8 name_f std::string lpc_bl_predictor_config::name_f [private]

4.164.3.9 name_b std::string lpc_bl_predictor_config::name_b [private]

4.164.3.10 km mha_wave_t lpc_bl_predictor_config::km [private]

4.164.3.11 s_f mha_wave_t lpc_bl_predictor_config::s_f [private]

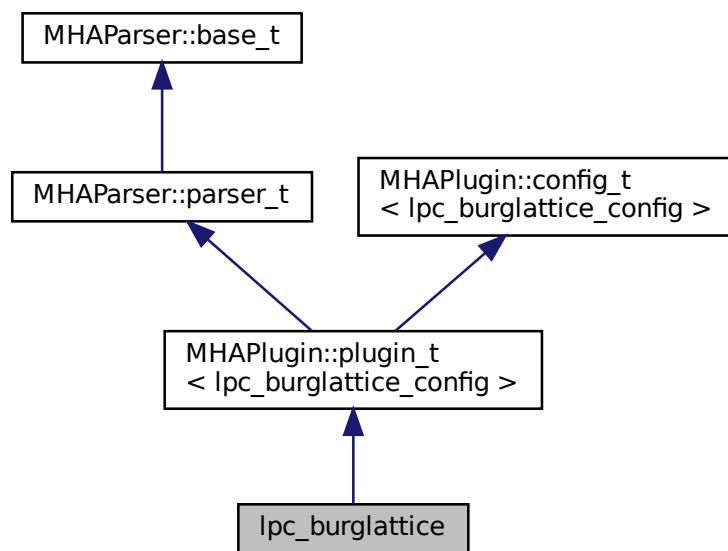
4.164.3.12 s_b mha_wave_t lpc_bl_predictor_config::s_b [private]

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

- [lpc_bl_predictor.h](#)
- [lpc_bl_predictor.cpp](#)

4.165 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

4.165.1 Constructor & Destructor Documentation

4.165.1.1 lpc_burglattice() `lpc_burglattice::lpc_burglattice (algo_comm_t & ac, const std::string & chain_name, const std::string & algo_name)`

Constructs our plugin.

4.165.1.2 ~lpc_burglattice() `lpc_burglattice::~lpc_burglattice ()`**4.165.2 Member Function Documentation****4.165.2.1 process()** `mha_wave_t * lpc_burglattice::process (mha_wave_t * signal)`

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

4.165.2.2 prepare() `void lpc_burglattice::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	Structure containing a description of the form of the signal (domain, number of channels, frames per block, sampling rate).
--------------------------	---

Implements **MHAPeripheral::MHAPlugin::plugin_t< lpc_burglattice_config >** (p. [1148](#)).

4.165.2.3 release() `void lpc_burglattice::release (void) [inline], [virtual]`

Reimplemented from **MHAPeripheral::MHAPlugin::plugin_t< lpc_burglattice_config >** (p. [1149](#)).

4.165.2.4 update_cfg() `void lpc_burglattice::update_cfg () [private]`

4.165.3 Member Data Documentation

4.165.3.1 lpc_order `MHAParser::int_t lpc_burglattice::lpc_order`

4.165.3.2 name_kappa `MHAParser::string_t lpc_burglattice::name_kappa`

4.165.3.3 name_f `MHAParser::string_t lpc_burglattice::name_f`

4.165.3.4 name_b `MHAParser::string_t lpc_burglattice::name_b`

4.165.3.5 lambda `MHAParser::float_t lpc_burglattice::lambda`

4.165.3.6 patchbay `MHAEVENTS::patchbay_t< lpc_burglattice> lpc_burglattice<::patchbay [private]`

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

- `lpc_burg-lattice.h`
- `lpc_burg-lattice.cpp`

4.166 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`

4.166.1 Constructor & Destructor Documentation

```
4.166.1.1 lpc_burglattice_config() lpc_burglattice_config::lpc_burglattice_config (
    algo_comm_t & iac,
    const mhaconfig_t in_cfg,
    lpc_burglattice * _lpc )
```

```
4.166.1.2 ~lpc_burglattice_config() lpc_burglattice_config::~lpc_burglattice_config
( )
```

4.166.2 Member Function Documentation

```
4.166.2.1 process() mha_wave_t * lpc_burglattice_config::process (
    mha_wave_t * wave )
```

4.166.3 Member Data Documentation

4.166.3.1 ac algo_comm_t lpc_burglattice_config::ac [private]

4.166.3.2 forward MHASignal::waveform_t lpc_burglattice_config::forward [private]

4.166.3.3 backward MHASignal::waveform_t lpc_burglattice_config::backward [private]

4.166.3.4 kappa MHASignal::waveform_t lpc_burglattice_config::kappa [private]

4.166.3.5 kappa_block MHA_AC::waveform_t lpc_burglattice_config::kappa_block [private]

4.166.3.6 dm MHASignal::waveform_t lpc_burglattice_config::dm [private]

4.166.3.7 nm MHASignal::waveform_t lpc_burglattice_config::nm [private]

4.166.3.8 lambda mha_real_t lpc_burglattice_config::lambda [private]

4.166.3.9 lpc_order int lpc_burglattice_config::lpc_order [private]

4.166.3.10 name_f std::string lpc_burglattice_config::name_f [private]

4.166.3.11 name_b std::string lpc_burglattice_config::name_b [private]

4.166.3.12 s_f mha_wave_t lpc_burglattice_config::s_f [private]

4.166.3.13 s_b mha_wave_t lpc_burglattice_config::s_b [private]

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

- **lpc_burg-lattice.h**
- **lpc_burg-lattice.cpp**

4.167 **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**

4.167.1 Constructor & Destructor Documentation

4.167.1.1 `lpc_config()` `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 )
```

4.167.1.2 `~lpc_config()` `lpc_config::~lpc_config ()`

4.167.2 Member Function Documentation

4.167.2.1 `process()` `mha_wave_t * lpc_config::process (`

```
    mha_wave_t * wave )
```

4.167.2.2 `insert()` `void lpc_config::insert ()`

4.167.3 Member Data Documentation

4.167.3.1 `norm` `bool lpc_config::norm [private]`

4.167.3.2 shift bool lpc_config::shift [private]

4.167.3.3 comp_each_iter unsigned int lpc_config::comp_each_iter [private]

4.167.3.4 order unsigned int lpc_config::order [private]

4.167.3.5 lpc_buffer_size unsigned int lpc_config::lpc_buffer_size [private]

4.167.3.6 N unsigned int lpc_config::N [private]

4.167.3.7 comp_iter unsigned int lpc_config::comp_iter [private]

4.167.3.8 sample mha_wave_t lpc_config::sample [private]

4.167.3.9 R std::vector< mha_real_t > lpc_config::R [private]

4.167.3.10 A std::vector< mha_real_t > lpc_config::A [private]

4.167.3.11 inwave `MHASignal::ringbuffer_t lpc_config::inwave` [private]

4.167.3.12 lpc_out `MHA_AC::waveform_t lpc_config::lpc_out` [private]

4.167.3.13 corr_out `MHA_AC::waveform_t lpc_config::corr_out` [private]

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

- `lpc.h`
- `lpc.cpp`

4.168 `Isl2ac::cfg_t` Class Reference

Runtime configuration class of the **Isl2ac** (p. 96) plugin.

Public Member Functions

- `cfg_t (const algo_comm_t &ac_, overrun_behavior overrun_, int bufsize_ ← , int chunksize_, const std::vector< std::string > &streamnames_, int nchannels_, int nsamples_)`
*C'tor of **Isl2ac** (p. 96) run time configuration.*
- `void process ()`

Private Attributes

- `std::map< std::string, std::unique_ptr< save_var_t > > varlist`
Maps variable name to unique ptr's of Isl to ac bridges.

4.168.1 Detailed Description

Runtime configuration class of the **Isl2ac** (p. 96) plugin.

4.168.2 Constructor & Destructor Documentation

```
4.168.2.1 cfg_t() cfg_t::cfg_t (
    const algo_comm_t & ac_,
    overrun_behavior overrun_,
    int bufsize_,
    int chunksize_,
    const std::vector< std::string > & streamnames_,
    int nchannels_,
    int nsamples_ )
```

C'tor of **Isl2ac** (p. 96) run time configuration.

Parameters

<i>ac_</i>	AC space, data from LSL will be inserted as AC variables
<i>overrun_</i>	Overrun behavior
<i>bufsize_</i>	Buffer size of the LSL stream inlet
<i>chunksize_</i>	Chunk size of the LSL stream
<i>streamnames_</i>	Names of LSL streams to be subscribed to
<i>nchannels_</i>	Number of channels to expect in the the LSL streams. Zero means accept any.
<i>nsamples_</i>	Number of samples per channel in the AC variable. Zero means resize as needed.

4.168.3 Member Function Documentation

4.168.3.1 process() void cfg_t::process ()

4.168.4 Member Data Documentation

4.168.4.1 varlist `std::map<std::string, std::unique_ptr< save_var_t > > lsl2ac->;cfg_t::varlist [private]`

Maps variable name to unique ptr's of lsl to ac bridges.

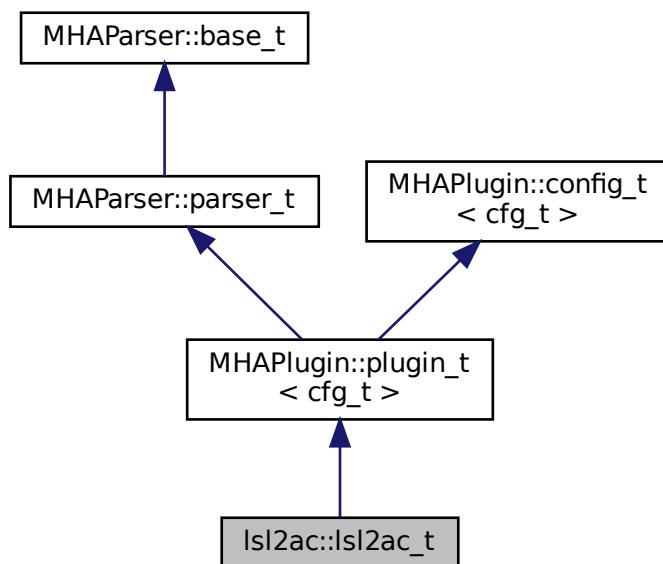
The documentation for this class was generated from the following files:

- `lsl2ac.hh`
- `lsl2ac.cpp`

4.169 `lsl2ac::lsl2ac_t` Class Reference

Plugin class of `lsl2ac` (p. 96).

Inheritance diagram for `lsl2ac::lsl2ac_t`:



Public Member Functions

- `lsl2ac_t (algo_comm_t iac, const char *chain, const char *algo)`
- `void prepare (mhaconfig_t &)`
 - Prepare constructs the vector of bridge variables and locks the configuration, then calls `update()` (p. 683).*
- `mha_wave_t * process (mha_wave_t *s)`

- **mha_spec_t * process (mha_spec_t *s)**
Processing fct for waveforms.
- **void process ()**
Processing fct for spectra.
- **void release ()**
Release fct.

Private Member Functions

- **void get_all_stream_names ()**
Retrieves all stream names from LSL and fills them into available_streams.
- **void update ()**
Construct new runtime configuration.
- **void setlock (bool lock_)**
Convencience function lock/unlock all config variables.

Private Attributes

- **MHAParser::vstring_t streams** ={"List of LSL streams to be saved, empty for all.", "[]"}
Config variable for list of streams to be saved.
- **MHAParser::bool_t activate** ={"Receive from network?", "yes"}
Config variable for activation/deactivation of plugin.
- **MHAParser::kw_t overrun_behavior** ={"How to handle overrun", "discard", "[discard ignore]"}
Config variable for overrun behavior.
- **MHAParser::int_t buffersize**
Config variable for maximum buffer size of LSL.
- **MHAParser::int_t chunksize**
Config variable for maximum chunk size of LSL.
- **MHAParser::int_t nchannels**
Config variable for number of channels to expect from LSL stream.
- **MHAParser::int_t nsamples**
Config variable for maximum chunk size of LSL.
- **MHAEvents::patchbay_t < Isl2ac_t > patchbay**
Patchbay for configuration callbacks.
- **MHAParser::vstring_mon_t available_streams** ={"List of all available LSL streams"}
Monitor variable containing all available streams.

Additional Inherited Members

4.169.1 Detailed Description

Plugin class of **Isl2ac** (p. 96).

4.169.2 Constructor & Destructor Documentation

4.169.2.1 `lsl2ac_t()` `lsl2ac::lsl2ac_t::lsl2ac_t (algo_comm_t iac, const char * chain, const char * algo)`

4.169.3 Member Function Documentation

4.169.3.1 `prepare()` `void lsl2ac::lsl2ac_t::prepare (mhaconfig_t & cf) [virtual]`

Prepare constructs the vector of bridge variables and locks the configuration, then calls **update()** (p. 683).

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1148).

4.169.3.2 `process() [1/3]` `mha_wave_t* lsl2ac::lsl2ac_t::process (mha_wave_t * s) [inline]`

Processing fct for waveforms.

Calls **process(void)** (p. 682).

4.169.3.3 `process() [2/3]` `mha_spec_t* lsl2ac::lsl2ac_t::process (mha_spec_t * s) [inline]`

Processing fct for spectra.

Calls **process(void)** (p. 682).

4.169.3.4 `process() [3/3]` `void lsl2ac::lsl2ac_t::process ()`

Process function.

4.169.3.5 release() void lsl2ac::lsl2ac_t::release () [virtual]

Release fct.

Unlocks variable name list

Reimplemented from **MHAPlugIn::plugin_t< cfg_t >** (p. 1149).

4.169.3.6 get_all_stream_names() void lsl2ac::lsl2ac_t::get_all_stream_names () [private]

Retrieves all stream names from LSL and fills them into available_streams.

4.169.3.7 update() void lsl2ac::lsl2ac_t::update () [private]

Construct new runtime configuration.

4.169.3.8 setlock() void lsl2ac::lsl2ac_t::setlock (bool lock_) [private]

Convencience function lock/unlock all config variables.

Parameters

<i>lock</i> ↪	True to lock, False for unlock
_	

4.169.4 Member Data Documentation**4.169.4.1 streams** **MHAParser::vstring_t** lsl2ac::lsl2ac_t::streams ={"List of LSL streams to be saved, empty for all.", "[]"} [private]

Config variable for list of streams to be saved.

4.169.4.2 activate `MHAParser::bool_t lsl2ac::lsl2ac_t::activate = {"Receive from network?", "yes"} [private]`

Config variable for activation/deactivation of plugin.

4.169.4.3 overrun_behavior `MHAParser::kw_t lsl2ac::lsl2ac_t::overrun_behavior = {"How to handle overrun", "discard", "[discard ignore]"} [private]`

Config variable for overrun behavior.

4.169.4.4 bufsize `MHAParser::int_t lsl2ac::lsl2ac_t::bufsize [private]`

Config variable for maximum buffer size of LSL.

4.169.4.5 chunksize `MHAParser::int_t lsl2ac::lsl2ac_t::chunksize [private]`

Config variable for maximum chunk size of LSL.

4.169.4.6 nchannels `MHAParser::int_t lsl2ac::lsl2ac_t::nchannels [private]`

Config variable for number of channels to expect from LSL stream.

4.169.4.7 nsamples `MHAParser::int_t lsl2ac::lsl2ac_t::nsamples [private]`

Config variable for maximum chunk size of LSL.

4.169.4.8 patchbay `MHAEVENTS::patchbay_t< lsl2ac_t> lsl2ac::lsl2ac_t::patchbay`
[private]

Patchbay for configuration callbacks.

4.169.4.9 available_streams `MHAPARSER::vstring_mon_t lsl2ac::lsl2ac_t::available<-streams ={"List of all available LSL streams"} [private]`

Monitor variable containing all available streams.

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

- `lsl2ac.hh`
- `lsl2ac.cpp`

4.170 **lsl2ac::save_var_t** Class Reference

LSL to AC bridge variable.

Public Member Functions

- `save_var_t (const save_var_t &)=delete`
- `save_var_t (save_var_t &&)=delete`
- `save_var_t & operator= (const save_var_t &)=delete`
- `save_var_t & operator= (save_var_t &&)=delete`
- `save_var_t (const Isl::stream_info &info_, const algo_comm_t &ac_, overrun_behavior ob_, int buffersize_, int chunksize_, int nchannels_, int nsamples_)`
C'tor of Isl to ac bridge.
- `~save_var_t ()=default`
- `Isl::stream_info info ()`
Get stream info object from stream inlet.
- `void receive_frame ()`
Receive a samples from Isl and copy to AC space.

Private Member Functions

- `void pull_samples_ignore ()`
Pull new samples, ignore overrun.
- `void pull_samples_discard ()`
Pull new samples as long as there are samples ready for pickup in the LSL buffers.
- `void get_time_correction ()`
Refresh time correction value every 5s.
- `void insert_vars ()`
Insert stream value, time stamp and time offset into ac space.

Private Attributes

- `Isl::stream_inlet stream`
LSL stream outlet.
- `std::vector< mha_real_t > buf`
Data buffer of the ac variable.
- `std::vector< double > ts_buf`
Timestamp buffer.
- `const algo_comm_t & ac`
Handle to AC space.
- `comm_var_t cv`
Timeseries AC variable.
- `comm_var_t ts`
Timestamp AC variable.
- `double tc =0.0`
Current time correction.
- `std::string ts_name`
Timestamp AC variable name.
- `std::string tc_name`
Time correction AC variable name.
- `std::string new_name`
Number of new samples AC variable name.
- `std::chrono::time_point< std::chrono::steady_clock > tic`
time point of last time correction pull
- `bool skip =false`
Should the variable be skipped in future process calls? Only true when error occurred.
- `overrun_behavior ob`
Behavior on stream overrun.
- `const std::string name`
Name of stream.
- `int32_t nchannels`
Number of channels.
- `std::size_t nsamples`
Number of samples per channel in the AC variable.
- `std::size_t chunksize`
Maximal chunk size of Isl stream.
- `std::size_t bufsize`
Total buffer size of the ac variable buffer.
- `int n_new_samples`
Number of most recently pulled samples per channel.

4.170.1 Detailed Description

LSL to AC bridge variable.

4.170.2 Constructor & Destructor Documentation

4.170.2.1 save_var_t() [1/3] `lsl2ac::save_var_t::save_var_t (`
`const save_var_t &) [delete]`

4.170.2.2 save_var_t() [2/3] `lsl2ac::save_var_t::save_var_t (`
`save_var_t &&) [delete]`

4.170.2.3 save_var_t() [3/3] `lsl2ac::save_var_t::save_var_t (`
`const lsl::stream_info & info_,`
`const algo_comm_t & ac_,`
`overrun_behavior ob_,`
`int buffersize_,`
`int chunksize_,`
`int nchannels_,`
`int nsamples_)`

C'tor of Isl to ac bridge.

Parameters

<code>name_</code>	Name of LSL stream to be received
<code>info_</code>	LSL stream info object containing metadata
<code>ac_</code>	Handle to ac space
<code>ub_</code>	Underrun behavior. 0=Zero out, 1=Copy, 2=Abort
<code>ob_</code>	Overrun behavior. 0=Discard oldest, 1=Ignore
<code>buffersize_</code>	LSL buffer size
<code>chunksize_</code>	LSL chunk size
<code>nchannels_</code>	Number of channels in the AC variable must be zero or equal to number of channels in LSL stream
<code>nsamples_</code>	Number of samples per channel in the AC variable. Zero means resize as needed

4.170.2.4 ~save_var_t() `lsl2ac::save_var_t::~save_var_t () [default]`

4.170.3 Member Function Documentation

4.170.3.1 operator=() [1/2] `save_var_t& lsl2ac::save_var_t::operator= (const save_var_t &) [delete]`

4.170.3.2 operator=() [2/2] `save_var_t& lsl2ac::save_var_t::operator= (save_var_t &&) [delete]`

4.170.3.3 info() `lsl::stream_info lsl2ac::save_var_t::info ()`

Get stream info object from stream inlet.

4.170.3.4 receive_frame() `void lsl2ac::save_var_t::receive_frame ()`

Receive a samples from Isl and copy to AC space.

Handling of underrun is configuration-dependent

4.170.3.5 pull_samples_ignore() `void lsl2ac::save_var_t::pull_samples_ignore () [private]`

Pull new samples, ignore overrun.

If nsamples=0, leaves the buffers in a state where the newest samples are at the beginning of the buffers, the state of the older samples is undefined and n_new_samples contains the number of new samples per channel. If nsamples is non-zero, the buffers are rotated so the oldest samples are the first in the buffer and n_new_samples contains the number of new samples per channel.

4.170.3.6 pull_samples_discard() void lsl2ac::save_var_t::pull_samples_discard ()
[private]

Pull new samples as long as there are samples ready for pickup in the LSL buffers.

If nsamples=0, leaves the buffers in a state where the newest samples are at the beginning of the buffers, the state of the older samples is undefined and n_new_samples contains total number of new samples per channel. If nsamples is non-zero, the buffers are rotated so the oldest samples come first. n_samples_new then contains the total number of new samples per channel. If n_new_samples is larger than the number of samples in the AC variable that means samples had be discarded.

4.170.3.7 get_time_correction() void lsl2ac::save_var_t::get_time_correction ()
[private]

Refresh time correction value every 5s.

4.170.3.8 insert_vars() void lsl2ac::save_var_t::insert_vars () [private]

Insert stream value, time stamp and time offset into ac space.

4.170.4 Member Data Documentation

4.170.4.1 stream lsl::stream_inlet lsl2ac::save_var_t::stream [private]

LSL stream outlet.

Interface to lsl

4.170.4.2 buf std::vector< **mha_real_t**> lsl2ac::save_var_t::buf [private]

Data buffer of the ac variable.

4.170.4.3 ts_buf std::vector<double> lsl2ac::save_var_t::ts_buf [private]

Timestamp buffer.

4.170.4.4 ac const algo_comm_t& lsl2ac::save_var_t::ac [private]

Handle to AC space.

4.170.4.5 cv comm_var_t lsl2ac::save_var_t::cv [private]

Timeseries AC variable.

4.170.4.6 ts comm_var_t lsl2ac::save_var_t::ts [private]

Timestamp AC variable.

4.170.4.7 tc double lsl2ac::save_var_t::tc =0.0 [private]

Current time correction.

4.170.4.8 ts_name std::string lsl2ac::save_var_t::ts_name [private]

Timestamp AC variable name.

4.170.4.9 tc_name std::string lsl2ac::save_var_t::tc_name [private]

Time correction AC variable name.

4.170.4.10 new_name std::string lsl2ac::save_var_t::new_name [private]

Number of new samples AC variable name.

4.170.4.11 tic std::chrono::time_point<std::chrono::steady_clock> lsl2ac::save_var_t::tic [private]

time point of last time correction pull

4.170.4.12 skip bool lsl2ac::save_var_t::skip =false [private]

Should the variable be skipped in future process calls? Only true when error occurred.

4.170.4.13 ob overrun_behavior lsl2ac::save_var_t::ob [private]

Behavior on stream overrun.

4.170.4.14 name const std::string lsl2ac::save_var_t::name [private]

Name of stream.

Must be saved separately because the stream info might be unrecoverable in error cases

4.170.4.15 nchannels int32_t lsl2ac::save_var_t::nchannels [private]

Number of channels.

4.170.4.16 nsamples std::size_t lsl2ac::save_var_t::nsamples [private]

Number of samples per channel in the AC variable.

Zero means resize as needed.

4.170.4.17 chunksize std::size_t lsl2ac::save_var_t::chunksize [private]

Maximal chunk size of lsl stream.

4.170.4.18 bufsize std::size_t lsl2ac::save_var_t::bufsize [private]

Total buffer size of the ac variable buffer.

nsamples*nchannels if nsamples is set, chunksize*nchannels otherwise if chunksize is set, nchannels if neither nsamples and chunksize are set.

4.170.4.19 n_new_samples int lsl2ac::save_var_t::n_new_samples [private]

Number of most recently pulled samples per channel.

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

- **lsl2ac.hh**
- **lsl2ac.cpp**

4.171 matlab_wrapper::callback Class Reference

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

Public Member Functions

- **callback** (matlab_wrapper_t *parent_, user_config_t *user_config_, MHParse \leftarrow
::mfloat_t *var_)
Ctor.
- **void on_writeaccess ()**
Write access callback.

Private Attributes

- **user_config_t * user_config**
Ptr to user_config_t.
- **MHParse::mfloat_t * var**
Ptr to parser.
- **matlab_wrapper_t * parent**
Ptr to parent plugin.

4.171.1 Detailed Description

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

Provides the callback. Every time the a user defined configuration parser is changed, this class makes sure that the corresponding 'matlab-side' struct is also changed and a new real time config is created and pushed.

4.171.2 Constructor & Destructor Documentation

4.171.2.1 `callback()` matlab_wrapper::callback::callback (

```
    matlab_wrapper_t * parent_,  
    user_config_t * user_config_,  
    MHAParser::mffloat_t * var_ )
```

Ctor.

Parameters

<code>parent_</code>	Ptr to the parent parser. Used to signal finished change.
<code>user_config_</code>	Ptr to user_config_t struct holding the 'matlab side' of the configuration variable
<code>var_</code>	Ptr to the configuration parser corresponding to user_config_

4.171.3 Member Function Documentation

4.171.3.1 `on_writeaccess()` void matlab_wrapper::callback::on_writeaccess ()

Write access callback.

To be called on every writeaccess of *var. Synchronises *var and *user_config.

4.171.4 Member Data Documentation

4.171.4.1 user_config `user_config_t* matlab_wrapper::callback::user_config [private]`

Ptr to `user_config_t`.

4.171.4.2 var `MHAParser::mfloat_t* matlab_wrapper::callback::var [private]`

Ptr to parser.

4.171.4.3 parent `matlab_wrapper_t* matlab_wrapper::callback::parent [private]`

Ptr to parent plugin.

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

- `matlab_wrapper.hh`
- `matlab_wrapper.cpp`

4.172 matlab_wrapper::matlab_wrapper_rt_cfg_t Class Reference

Thin wrapper around the emxArray containing the user defined configuration variables.

Public Member Functions

- **matlab_wrapper_rt_cfg_t** (`emxArray_user_config_t *user_config_`)
Ctor of the real time configuration class.
- **~matlab_wrapper_rt_cfg_t ()**
Dtor.

Public Attributes

- `emxArray_user_config_t * user_config`
User configuration to be used in the process callback.

4.172.1 Detailed Description

Thin wrapper around the emxArray containing the user defined configuration variables.

This wrapper holds a copy of the user configuration which is used by the process callback. On write access to a variable, the user configuration is changed and a new copy is created and exchanged in a real time safe manner.

4.172.2 Constructor & Destructor Documentation

4.172.2.1 matlab_wrapper_rt_cfg_t() matlab_wrapper::matlab_wrapper_rt_cfg_t::matlab_wrapper_rt_cfg_t (emxArray_user_config_t * user_config_) [explicit]

Ctor of the real time configuration class.

Creates a local copy of the user config

Parameters

<i>user_config_</i>	Pointer to the array containing the user config
---------------------	---

4.172.2.2 ~matlab_wrapper_rt_cfg_t() matlab_wrapper::matlab_wrapper_rt_cfg_t::~matlab_wrapper_rt_cfg_t ()

Dtor.

Calls the appropriate c-style destructors of the emx_ types

4.172.3 Member Data Documentation

4.172.3.1 `user_config`

`emxArray_user_config_t* matlab_wrapper::matlab_wrapper_rt_<cfg_t>::user_config`

User configuration to be used in the process callback.

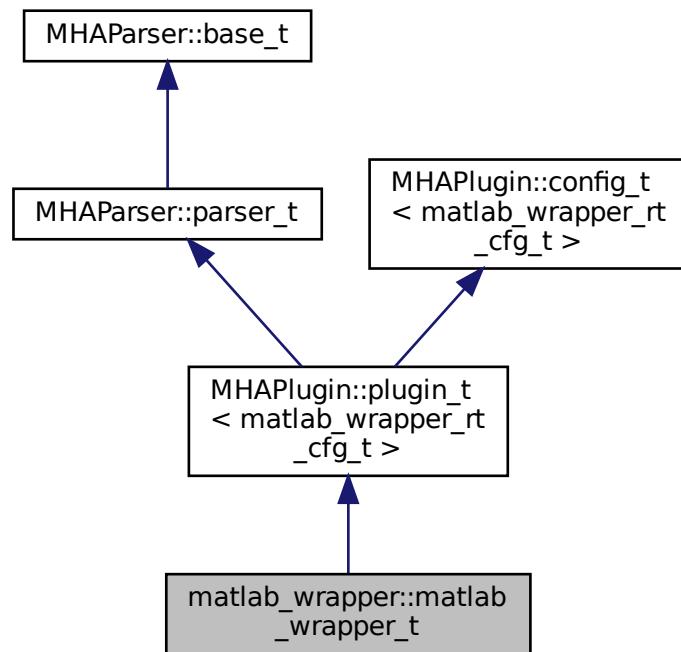
The documentation for this class was generated from the following files:

- `matlab_wrapper.hh`
- `matlab_wrapper.cpp`

4.173 `matlab_wrapper::matlab_wrapper_t` Class Reference

Matlab wraper plugin interface class.

Inheritance diagram for `matlab_wrapper::matlab_wrapper_t`:



Classes

- class `wrapped_plugin_t`

Wrapper class around the matlab-generated library.

Public Member Functions

- **matlab_wrapper_t** (const **algo_comm_t** &iac, const std::string &, const std::string &)
Ctor of the plugin interface class.
- **void process** (**mha_wave_t** *sin, **mha_wave_t** **sout)
Pure waveform processing.
- **void process** (**mha_spec_t** *sin, **mha_spec_t** **sout)
Pure spectrum processing.
- **void process** (**mha_wave_t** *sin, **mha_spec_t** **sout)
Signal processing with domain transformation from waveform to spectrum.
- **void process** (**mha_spec_t** *sin, **mha_wave_t** **sout)
Signal processing with domain transformation from spectrum to waveform.
- **void prepare** (**mhaconfig_t** &signal_info)
Prepare callback.
- **void release** ()
Release callback.

Private Member Functions

- **void insert_monitors** ()
- **void update_monitors** ()
- **void insert_config_vars** ()
- **void load_lib** ()
Create new library wrapper and load library.
- **void update** ()
Create new real time safe user config from user config.

Private Attributes

- friend **callback**
- **MHAParser::string_t library_name** {"Name of matlab generated library",""}
Configuration variable holding the file name of the matlab generated library.
- **MHAEvents::patchbay_t< matlab_wrapper_t > patchbay**
Patchbay for the interface plugins.
- **MHAEvents::patchbay_t< callback > cb_patchbay**
Patchbay for the custom callbacks.
- **std::unique_ptr< wrapped_plugin_t > plug**
Unique ptr holding the instance of the plugin wrapper class.
- **std::vector< MHAParser::mfloat_t > vars**
Vector holding the user defined configuration variables.
- **std::vector< callback > callbacks**
Vector holding the callbacks for the user defined variables' write access.
- **std::vector< MHAParser::mfloat_mon_t > monitors**
Vector holding the monitors corresponding to user state.

Additional Inherited Members

4.173.1 Detailed Description

Matlab wraper plugin interface class.

4.173.2 Constructor & Destructor Documentation

```
4.173.2.1 matlab_wrapper_t() matlab_wrapper::matlab_wrapper_t::matlab_wrapper_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

Ctor of the plugin interface class.

Parameters

<i>iac</i>	Handle to ac space
------------	--------------------

4.173.3 Member Function Documentation

```
4.173.3.1 process() [1/4] void matlab_wrapper::matlab_wrapper_t::process (
    mha_wave_t * sin,
    mha_wave_t ** sout )
```

Pure waveform processing.

Parameters

<i>sin</i>	input waveform signal
<i>sout</i>	output waveform signal

```
4.173.3.2 process() [2/4] void matlab_wrapper::matlab_wrapper_t::process (
    mha_spec_t * sin,
    mha_spec_t ** sout )
```

Pure spectrum processing.

Parameters

<i>sin</i>	input spectrum signal
<i>sout</i>	output spectrum signal

```
4.173.3.3 process() [3/4] void matlab_wrapper::matlab_wrapper_t::process (
    mha_wave_t * sin,
    mha_spec_t ** sout )
```

Signal processing with domain transformation from waveform to spectrum.

Parameters

<i>sin</i>	input waveform signal
<i>sout</i>	output spectrum signal

```
4.173.3.4 process() [4/4] void matlab_wrapper::matlab_wrapper_t::process (
    mha_spec_t * sin,
    mha_wave_t ** sout )
```

Signal processing with domain transformation from spectrum to waveform.

Parameters

<i>sin</i>	input spectrum signal
<i>sout</i>	output waveform signal

```
4.173.3.5 prepare() void matlab_wrapper::matlab_wrapper_t::prepare (
    mhaconfig_t & signal_info ) [virtual]
```

Prepare callback.

Parameters

<code>signal_info</code>	struct holding the input/output signal dimensions
--------------------------	---

Implements **MHAPlugIn::plugin_t< matlab_wrapper_rt_cfg_t >** (p. [1148](#)).

4.173.3.6 `release()` `void matlab_wrapper::matlab_wrapper_t::release () [virtual]`

Release callback.

Reimplemented from **MHAPlugIn::plugin_t< matlab_wrapper_rt_cfg_t >** (p. [1149](#)).

4.173.3.7 `insert_monitors()` `void matlab_wrapper::matlab_wrapper_t::insert_monitors () [private]`

4.173.3.8 `update_monitors()` `void matlab_wrapper::matlab_wrapper_t::update_monitors () [private]`

4.173.3.9 `insert_config_vars()` `void matlab_wrapper::matlab_wrapper_t::insert_config_vars () [private]`

4.173.3.10 `load_lib()` `void matlab_wrapper::matlab_wrapper_t::load_lib () [private]`

Create new library wrapper and load library.

To be called write access to `library_name`.

4.173.3.11 `update()` `void matlab_wrapper::matlab_wrapper_t::update () [private]`

Create new real time safe user config from user config.

Called by the custom callbacks.

4.173.4 Member Data Documentation

4.173.4.1 callback friend matlab_wrapper::matlab_wrapper_t::callback [private]

4.173.4.2 library_name `MHAParser::string_t matlab_wrapper::matlab_wrapper_t::library_name` {"Name of matlab generated library", ""} [private]

Configuration variable holding the file name of the matlab generated library.

4.173.4.3 patchbay `MHAEvents::patchbay_t< matlab_wrapper_t> matlab_wrapper::matlab_wrapper_t::patchbay` [private]

Patchbay for the interface plugins.

4.173.4.4 cb_patchbay `MHAEvents::patchbay_t< callback> matlab_wrapper::matlab_wrapper_t::cb_patchbay` [private]

Patchbay for the custom callbacks.

Can use normal patchbay bc/ of the interface of patchbay_t

4.173.4.5 plug `std::unique_ptr< wrapped_plugin_t> matlab_wrapper::matlab_wrapper_t::plug` [private]

Unique ptr holding the instance of the plugin wrapper class.

4.173.4.6 vars `std::vector< MHAParser::mfloat_t> matlab_wrapper::matlab_wrapper_t::vars` [private]

Vector holding the user defined configuration variables.

4.173.4.7 callbacks std::vector< **callback**> matlab_wrapper::matlab_wrapper_t::callbacks
[private]

Vector holding the callbacks for the user defined variables' write access.

4.173.4.8 monitors std::vector< **MHAParser::mfloating_mon_t**> matlab_wrapper::matlab_wrapper_t::monitors [private]

Vector holding the monitors corresponding to user state.

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

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

4.174 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t Class Reference

Wrapper class around the matlab-generated library.

Public Member Functions

- **wrapped_plugin_t ()=delete**
- **wrapped_plugin_t (const char *name_)**

Ctor.
- **virtual ~wrapped_plugin_t ()**

Dtor.
- **mha_wave_t * process_ww (mha_wave_t *s, emxArray_user_config_t *user_config_)**

Process callback.
- **mha_spec_t * process_ss (mha_spec_t *s, emxArray_user_config_t *user_config_)**

Process callback.
- **mha_spec_t * process_ws (mha_wave_t *s, emxArray_user_config_t *user_config_)**

Process callback.
- **mha_wave_t * process_sw (mha_spec_t *s, emxArray_user_config_t *user_config_)**

Process callback.
- **void prepare (mhaconfig_t &config)**

Prepare callback.
- **void release ()**

Release callback.

Public Attributes

- emxArray_user_config_t * **user_config** =nullptr
Ptr to user config array.
- emxArray_user_config_t * **state** =nullptr
Ptr to state array.

Private Attributes

- **dynamiclib_t library_handle**
Handle to matlab generated shared library.
- void(*) **fcn_terminate** ()() =nullptr
Handle to matlab generated cleanup function.
- void(*) **fcn_init**)(emxArray_user_config_t *, emxArray_user_config_t *) =nullptr
Handle to matlab generated init function.
- void(*) **fcn_process_ww**)(const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr
Handle to matlab generated wave to wave process function.
- void(*) **fcn_process_ss**)(const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr
Handle to matlab generated spectrum to spectrum process function.
- void(*) **fcn_process_ws**)(const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr
Handle to matlab generated wave to spectrum process function.
- void(*) **fcn_process_sw**)(const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr
Handle to matlab generated spectrum to wave process function.
- void(*) **fcn_prepare**)(signal_dimensions_t *, emxArray_user_config_t *, emxArray_user_config_t *) =nullptr
Handle to matlab generated prepare function.
- void(*) **fcn_release** ()() =nullptr
- signal_dimensions_t **signal_dimensions** {0,'U',0,0,0,0}
Signal dimensions.
- emxArray_real_T * **wave_in** =nullptr
Ptr to emxArray holding the input signal.
- emxArray_real_T * **wave_out** =nullptr
Ptr to emxArray holding the output signal.
- emxArray_creal_T * **spec_in** =nullptr
Ptr to emxArray holding the input signal.
- emxArray_creal_T * **spec_out** =nullptr
Ptr to emxArray holding the output signal.
- std::unique_ptr< **MHASignal::waveform_t** > **mha_wave_out**
MHA waveform holding the output signal.
- std::unique_ptr< **MHASignal::spectrum_t** > **mha_spec_out**
MHA waveform holding the output signal.

4.174.1 Detailed Description

Wrapper class around the matlab-generated library.

4.174.2 Constructor & Destructor Documentation

4.174.2.1 wrapped_plugin_t() [1/2] `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wrapped_plugin_t() [delete]`

4.174.2.2 wrapped_plugin_t() [2/2] `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wrapped_plugin_t(const char * name_) [explicit]`

Ctor.

Opens matlab-generated library and resolves functions

Parameters

<code>name</code>	file name of shared library
_	

4.174.2.3 ~wrapped_plugin_t() `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::~wrapped_plugin_t() [virtual]`

Dtor.

4.174.3 Member Function Documentation

```
4.174.3.1 process_ww() mha_wave_t * matlab_wrapper::matlab_wrapper_t::wrapped_↔
plugin_t::process_ww (
    mha_wave_t * s,
    emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type wave to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

<i>s</i>	input/output signal
<i>user_↔ config_</i>	Ptr to user configuration array

```
4.174.3.2 process_ss() mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_↔
plugin_t::process_ss (
    mha_spec_t * s,
    emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type spectrum to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

<i>s</i>	input/output signal
<i>user_↔ config_</i>	Ptr to user configuration array

```
4.174.3.3 process_ws() mha_spec_t * matlab_wrapper::matlab_wrapper_t::wrapped_↔
plugin_t::process_ws (
    mha_wave_t * s,
    emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type wave to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

<i>s</i>	input/output signal
<i>user_</i> \leftrightarrow <i>config_</i>	Ptr to user configuration array

```
4.174.3.4 process_sw() mha_wave_t * matlab_wrapper::matlab_wrapper_t::wrapped_<-
plugin_t::process_sw (
    mha_spec_t * s,
    emxArray_user_config_t * user_config_ )
```

Process callback.

Converts mha-type spectrum to matlab array and calls wrapped process callback, then convert output to mha type

Parameters

<i>s</i>	input/output signal
<i>user_</i> \leftrightarrow <i>config_</i>	Ptr to user configuration array

```
4.174.3.5 prepare() void matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::prepare
(
    mhaconfig_t & config )
```

Prepare callback.

Calls wrapped prepare function if necessary and determines output signal dimensions

```
4.174.3.6 release() void matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::release
( )
```

Release callback.

Cleans up io arrays and calls wrapped release if necessary.

4.174.4 Member Data Documentation

4.174.4.1 user_config emxArray_user_config_t* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::user_config =nullptr

Ptr to user config array.

4.174.4.2 state emxArray_user_config_t* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::state =nullptr

Ptr to state array.

4.174.4.3 library_handle dynamiclib_t matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::library_handle [private]

Handle to matlab generated shared library.

4.174.4.4 fcn_terminate void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_terminate) () =nullptr [private]

Handle to matlab generated cleanup function.

4.174.4.5 fcn_init void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_init) (emxArray_user_config_t *, emxArray_user_config_t *) =nullptr [private]

Handle to matlab generated init function.

4.174.4.6 fcn_process_ww void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ww) (const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr [private]

Handle to matlab generated wave to wave process function.

4.174.4.7 fcn_process_ss void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ss) (const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr [private]

Handle to matlab generated spectrum to spectrum process function.

4.174.4.8 fcn_process_ws void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_ws) (const emxArray_real_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_creal_T *) =nullptr [private]

Handle to matlab generated wave to spectrum process function.

4.174.4.9 fcn_process_sw void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_process_sw) (const emxArray_creal_T *, const signal_dimensions_t *, const emxArray_user_config_t *, emxArray_user_config_t *, emxArray_real_T *) =nullptr [private]

Handle to matlab generated spectrum to wave process function.

4.174.4.10 fcn_prepare void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_prepare) (signal_dimensions_t *, emxArray_user_config_t *, emxArray_user_config_t *) =nullptr [private]

Handle to matlab generated prepare function.

4.174.4.11 fcn_release void(* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::fcn_release) () =nullptr [private]

4.174.4.12 signal_dimensions signal_dimensions_t matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::signal_dimensions {0,'U',0,0,0,0} [private]

Signal dimensions.

Matlab-equivalent to mha_config_t

4.174.4.13 wave_in emxArray_real_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wave_in =nullptr [private]

Ptr to emxArray holding the input signal.

4.174.4.14 wave_out emxArray_real_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::wave_out =nullptr [private]

Ptr to emxArray holding the output signal.

4.174.4.15 spec_in emxArray_creal_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::spec_in =nullptr [private]

Ptr to emxArray holding the input signal.

4.174.4.16 spec_out emxArray_creal_T* matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t::spec_out =nullptr [private]

Ptr to emxArray holding the output signal.

4.174.4.17 mha_wave_out std::unique_ptr< **MHASignal::waveform_t::matlab_wrapper_t::wrapped_plugin_t::mha_wave_out [private]**

MHA waveform holding the output signal.

4.174.4.18 mha_spec_out std::unique_ptr< **MHASignal::spectrum_t**> matlab_wrapper<-
::matlab_wrapper_t::wrapped_plugin_t::mha_spec_out [private]

MHA waveform holding the output signal.

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

- **matlab_wrapper.hh**
- **matlab_wrapper.cpp**

4.175 matlab_wrapper::types< T > Struct Template Reference

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

- **matlab_wrapper.hh**

4.176 matlab_wrapper::types< MHA_SPECTRUM > Struct Reference

Public Types

- **typedef emxArray_creal_T array_type**
- **typedef mha_spec_t signal_type**
- **typedef MHASignal::spectrum_t class_signal_type**

4.176.1 Member Typedef Documentation

4.176.1.1 array_type **typedef emxArray_creal_T matlab_wrapper::types< MHA_SPECTRUM >::array_type**

4.176.1.2 signal_type `typedef mha_spec_t matlab_wrapper::types< MHA_SPECTRUM >:: signal_type`

4.176.1.3 class_signal_type `typedef MHASignal::spectrum_t matlab_wrapper::types< MHA_SPECTRUM >:: class_signal_type`

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

- `matlab_wrapper.hh`

4.177 matlab_wrapper::types< MHA_WAVEFORM > Struct Reference

Public Types

- `typedef emxArray_real_T array`
- `typedef mha_wave_t signal_type`
- `typedef MHASignal::waveform_t class_signal_type`

4.177.1 Member Typedef Documentation

4.177.1.1 array `typedef emxArray_real_T matlab_wrapper::types< MHA_WAVEFORM >:: array`

4.177.1.2 signal_type `typedef mha_wave_t matlab_wrapper::types< MHA_WAVEFORM >:: signal_type`

4.177.1.3 class_signal_type `typedef MHASignal::waveform_t matlab_wrapper::types< MHA_WAVEFORM >:: class_signal_type`

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

- `matlab_wrapper.hh`

4.178 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**

4.178.1 Constructor & Destructor Documentation

```
4.178.1.1 cfg_t() cfg_t::cfg_t (
    std::vector< std::vector< float > > imixer,
    unsigned int ci,
    unsigned int co,
    unsigned int fragsize,
    unsigned int nfft )
```

4.178.2 Member Function Documentation

```
4.178.2.1 process() [1/2] mha_wave_t * cfg_t::process (
    mha_wave_t * s )
```

```
4.178.2.2 process() [2/2] mha_spec_t * cfg_t::process (
    mha_spec_t * s )
```

4.178.3 Member Data Documentation

4.178.3.1 m `MHASignal::waveform_t` `matrixmixer::cfg_t::m` [private]

4.178.3.2 wout `MHASignal::waveform_t` `matrixmixer::cfg_t::wout` [private]

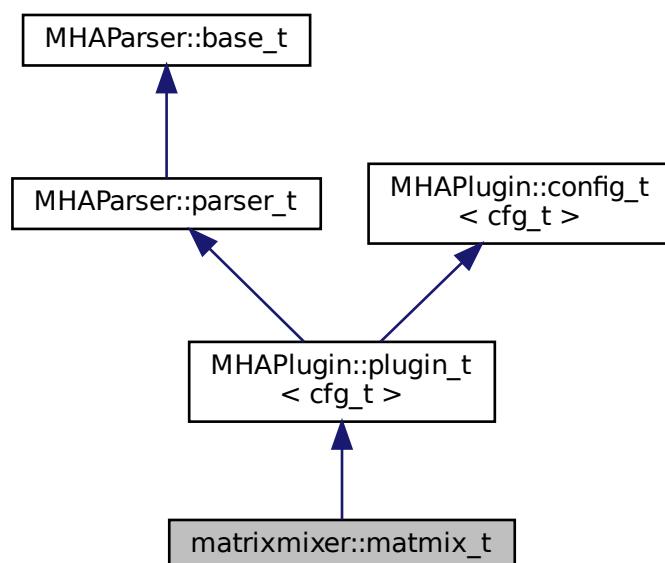
4.178.3.3 sout `MHASignal::spectrum_t` `matrixmixer::cfg_t::sout` [private]

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

- `matrixmixer.cpp`

4.179 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

4.179.1 Constructor & Destructor Documentation

4.179.1.1 `matmix_t()` `matrixmixer::matmix_t::matmix_t (`
`const algo_comm_t & iac,`
`const std::string & ,`
`const std::string &)`

4.179.2 Member Function Documentation

4.179.2.1 `prepare()` `void matrixmixer::matmix_t::prepare (`
`mhaconfig_t & tf) [virtual]`

Implements `MHAPlugin::plugin_t< cfg_t >` (p. 1148).

4.179.2.2 process() [1/2] `mha_wave_t * matrixmixer::matmix_t::process (mha_wave_t * s)`

4.179.2.3 process() [2/2] `mha_spec_t * matrixmixer::matmix_t::process (mha_spec_t * s)`

4.179.2.4 update_m() `void matrixmixer::matmix_t::update_m () [private]`

4.179.3 Member Data Documentation

4.179.3.1 patchbay `MHAEvents::patchbay_t< matmix_t> matrixmixer::matmix_t::patchbay [private]`

4.179.3.2 mixer `MHAParser::mfloat_t matrixmixer::matmix_t::mixer [private]`

4.179.3.3 ci `unsigned int matrixmixer::matmix_t::ci [private]`

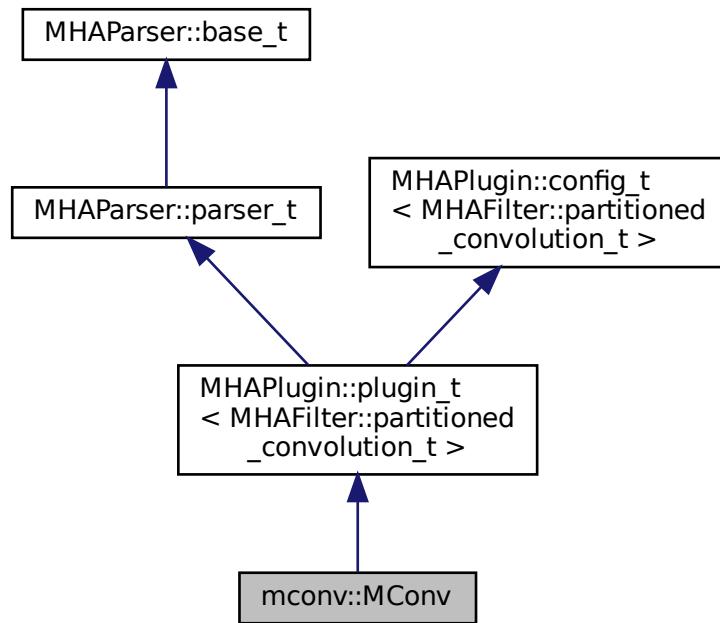
4.179.3.4 co `unsigned int matrixmixer::matmix_t::co [private]`

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

- `matrixmixer.cpp`

4.180 mconv::MConv Class Reference

Inheritance diagram for mconv::MConv:



Public Member Functions

- **MConv** (const **algo_comm_t** &iac, const std::string &chainname, const std::string &algoname)
Plugin constructor.
- void **prepare** (**mhaconfig_t** &mhaconfig)
Prepare this plugin for processing.
- void **release** ()
- **mha_wave_t** * **process** (**mha_wave_t** *)

Private Member Functions

- void **update** ()
*Update is needed only once, since this plugin allows only change of irs after **prepare()** (p. 718).*
- void **update_irs** ()
*This function updates the irs without allowing a change of its size after **prepare()** (p. 718).*

Private Attributes

- **MHAParser::int_t nchannels_out**
Number of output channels to produce.
- **MHAParser::vint_t inch**
Vector of input channel indices.
- **MHAParser::vint_t outch**
Vector of output channel indices.
- **MHAParser::mfloat_t irs**
Impulse responses, one per row.
- **unsigned int nchannels_in**
Number of input channels, set during prepare.
- **unsigned int fragsize**
Fragsize, set during prepare, is used as the partition length in the partitioned convolution.
- **MHAEvents::patchbay_t< MConv > patchbay**

Additional Inherited Members

4.180.1 Detailed Description

class implements plugin for partitioned convolution. A matrix of impulse responses, filtering n input channels to m output channels, is supported.

4.180.2 Constructor & Destructor Documentation

```
4.180.2.1 MConv() mconv::MConv::MConv (
    const algo_comm_t & iac,
    const std::string & chainname,
    const std::string & algoname )
```

Plugin constructor.

Parameters

<i>iac</i>	handle and function pointers for algorithm communication
<i>chainname</i>	Name of processing chain
<i>algoname</i>	The name by which the chain refers to this algorithm

4.180.3 Member Function Documentation

4.180.3.1 `prepare()` `void mconv::MConv::prepare (mhaconfig_t & mhaconfig) [virtual]`

Prepare this plugin for processing.

Parameters

<code>mhaconfig</code>	Configuration for this plugin (Input/Output parameter) Sample rate, fragment size, number of channels are detailed here.
------------------------	--

Implements `MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >` (p. [1148](#)).

4.180.3.2 `release()` `void mconv::MConv::release () [virtual]`

Reimplemented from `MHAPlugin::plugin_t< MHAFilter::partitioned_convolution_t >` (p. [1149](#)).

4.180.3.3 `process()` `mha_wave_t * mconv::MConv::process (mha_wave_t * s_in)`

4.180.3.4 `update()` `void mconv::MConv::update () [private]`

Update is needed only once, since this plugin allows only change of irs after `prepare()` (p. [718](#)).

4.180.3.5 `update_irs()` `void mconv::MConv::update_irs () [private]`

This function updates the irs without allowing a change of its size after `prepare()` (p. [718](#)).

4.180.4 Member Data Documentation

4.180.4.1 **nchannels_out** `MHAParser::int_t mconv::MConv::nchannels_out [private]`

Number of output channels to produce.

4.180.4.2 **inch** `MHAParser::vint_t mconv::MConv::inch [private]`

Vector of input channel indices.

Each element in this vector identifies the input channel to which to apply the corresponding impulse response in irs.

4.180.4.3 **outch** `MHAParser::vint_t mconv::MConv::outch [private]`

Vector of output channel indices.

Each element in this vector identifies the output channel to which the result of filtering with the corresponding impulse response in irs is mixed.

4.180.4.4 **irs** `MHAParser::mfloat_t mconv::MConv::irs [private]`

Impulse responses, one per row.

For each row, the corresponding element of inch identifies the source channel, and the corresponding element of outch identifies the target channel.

4.180.4.5 **nchannels_in** `unsigned int mconv::MConv::nchannels_in [private]`

Number of input channels, set during prepare.

4.180.4.6 **fragsize** `unsigned int mconv::MConv::fragsize [private]`

Fragsize, set during prepare, is used as the partition length in the partitioned convolution.

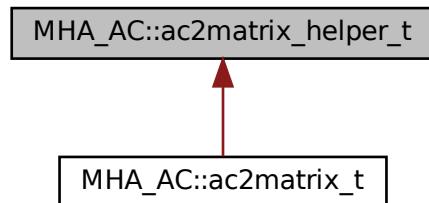
4.180.4.7 patchbay MHAEvents::patchbay_t< MConv> mconv::MConv::patchbay [private]

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

- **mconv.cpp**

4.181 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**

4.181.1 Constructor & Destructor Documentation

4.181.1.1 ac2matrix_helper_t() `MHA_AC::ac2matrix_helper_t::ac2matrix_helper_t (algo_comm_t iac,
const std::string & iname)`

4.181.2 Member Function Documentation

4.181.2.1 getvar() `void MHA_AC::ac2matrix_helper_t::getvar ()`

4.181.3 Member Data Documentation

4.181.3.1 ac `algo_comm_t MHA_AC::ac2matrix_helper_t::ac`

4.181.3.2 name `std::string MHA_AC::ac2matrix_helper_t::name`

4.181.3.3 username `std::string MHA_AC::ac2matrix_helper_t::username`

4.181.3.4 size `MHASignal::uint_vector_t MHA_AC::ac2matrix_helper_t::size`

4.181.3.5 **is_complex** `bool MHA_AC::ac2matrix_helper_t::is_complex`

4.181.3.6 **acvar** `comm_var_t MHA_AC::ac2matrix_helper_t::acvar [protected]`

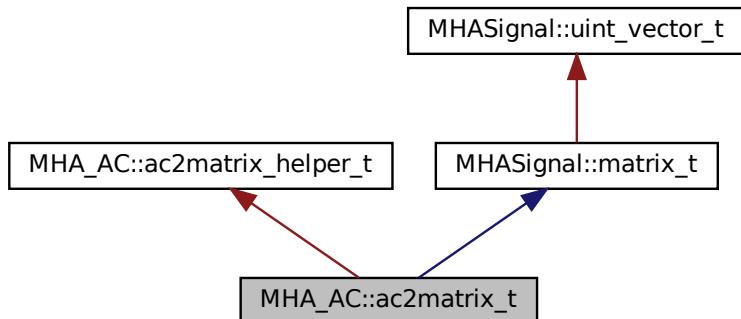
The documentation for this class was generated from the following files:

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

4.182 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

4.182.1 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. 723) function is real-time safe.

4.182.2 Constructor & Destructor Documentation

```
4.182.2.1 ac2matrix_t() 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

4.182.3 Member Function Documentation

```
4.182.3.1 update() void MHA_AC::ac2matrix_t::update ( )
```

Update contents of the matrix from the AC space.

This function is real-time safe. The copy operation performance is of the order of the number of elements in the matrix.

```
4.182.3.2 getname() const std::string& MHA_AC::ac2matrix_t::getname ( ) const [inline]
```

Return name of AC variable/matrix.

4.182.3.3 getusername() const std::string& MHA_AC::ac2matrix_t::getusername ()
const [inline]

Return user specified name of AC variable/matrix.

4.182.3.4 insert() void MHA_AC::ac2matrix_t::insert (algo_comm_t ac)

Insert matrix into an AC space (other than source AC space)

Parameters

ac	AC space handle to insert data
----	--------------------------------

Note

The AC variable data buffer points to the data of the matrix. Modifications of the AC variable directly modify the data of the matrix; after deletion of the matrix, the data buffer is invalid.

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

- [mha_algo_comm.h](#)
- [mha_algo_comm.cpp](#)

4.183 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**

4.183.1 Detailed Description

Copy all or a subset of all numeric AC variables into an array of matrixes.

4.183.2 Constructor & Destructor Documentation

4.183.2.1 acspace2matrix_t() [1/2] MHA_AC::acspace2matrix_t::acspace2matrix_t (

algo_comm_t	<i>ac,</i>
const std::vector<	<i>std::string > & names</i>)

Constructor.

Scan all given AC variables and allocate corresponding matrixes.

Parameters

ac	AC handle.
names	Names of AC variables, or empty for all.

4.183.2.2 acspace2matrix_t() [2/2] `MHA_AC::acspace2matrix_t::acspace2matrix_t (const MHA_AC::acspace2matrix_t & src)`

Constructor with initialization from an instance.

Parameters

<code>src</code>	Instance to be copied.
------------------	------------------------

4.183.2.3 ~acspace2matrix_t() `MHA_AC::acspace2matrix_t::~acspace2matrix_t ()`

4.183.3 Member Function Documentation

4.183.3.1 operator=() `MHA_AC::acspace2matrix_t & MHA_AC::acspace2matrix_t::operator= (const MHA_AC::acspace2matrix_t & src)`

Copy all contents (deep copy).

Parameters

<code>src</code>	Array of matrixes to be copied.
------------------	---------------------------------

4.183.3.2 operator[]() [1/2] `MHA_AC::ac2matrix_t& MHA_AC::acspace2matrix_t::operator[] (unsigned int k) [inline]`

Access operator.

Parameters

<code>k</code>	index into array; should not exceed size() (p. 727)-1.
----------------	--

Return values

<i>Reference</i>	to matrix.
------------------	------------

4.183.3.3 operator[]() [2/2] 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. 727)-1.
----------	---

Return values

<i>Constant</i>	reference to matrix.
-----------------	----------------------

4.183.3.4 update() void MHA_AC::acspace2matrix_t::update () [inline]

Update function.

This function updates all matrixes from their corresponding AC variables. It can be called from the MHA Framework prepare function or in the processing callback.

4.183.3.5 size() unsigned int MHA_AC::acspace2matrix_t::size () const [inline]

Number of matrixes in AC space.

4.183.3.6 frame() unsigned int MHA_AC::acspace2matrix_t::frame () const [inline]

Actual frame number.

4.183.3.7 insert() void MHA_AC::acspace2matrix_t::insert (algo_comm_t *ac*)

Insert AC space copy into an AC space (other than source AC space)

Parameters

<code>ac</code>	AC space handle to insert data
-----------------	--------------------------------

4.183.4 Member Data Documentation

4.183.4.1 len `unsigned int MHA_AC::acspace2matrix_t::len [private]`

4.183.4.2 data `MHA_AC::ac2matrix_t** MHA_AC::acspace2matrix_t::data [private]`

4.183.4.3 frameno `unsigned int MHA_AC::acspace2matrix_t::frameno [private]`

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

- `mha_algo_comm.h`
- `mha_algo_comm.cpp`

4.184 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 ()`
- `void insert ()`

Public Attributes

- **double data**
Floating point (double precision) value variable.

Private Attributes

- `algo_comm_t ac`
- `std::string name`

4.184.1 Detailed Description

Insert a double precision floating point variable into the AC space.

The variable is automatically removed on destruction.

4.184.2 Constructor & Destructor Documentation

4.184.2.1 `double_t()` `MHA_AC::double_t::double_t (`
 `algo_comm_t ac,`
 `std::string name,`
 `double val = 0)`

4.184.2.2 `~double_t()` `MHA_AC::double_t::~double_t ()`

4.184.3 Member Function Documentation

4.184.3.1 `insert()` `void MHA_AC::double_t::insert ()`

4.184.4 Member Data Documentation

4.184.4.1 data double MHA_AC::double_t::data

Floating point (double precision) value variable.

4.184.4.2 ac algo_comm_t MHA_AC::double_t::ac [private]**4.184.4.3 name** std::string MHA_AC::double_t::name [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

4.185 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 ()**
- **void insert ()**

Public Attributes

- float **data**
Floating point value variable.

Private Attributes

- **algo_comm_t ac**
- **std::string name**

4.185.1 Detailed Description

Insert a float point variable into the AC space.

The variable is automatically removed on destruction.

4.185.2 Constructor & Destructor Documentation

```
4.185.2.1 float_t() MHA_AC::float_t::float_t (
    algo_comm_t ac,
    std::string name,
    float val = 0 )
```

Constructor.

```
4.185.2.2 ~float_t() MHA_AC::float_t::~float_t ( )
```

4.185.3 Member Function Documentation

```
4.185.3.1 insert() void MHA_AC::float_t::insert ( )
```

4.185.4 Member Data Documentation

```
4.185.4.1 data float MHA_AC::float_t::data
```

Floating point value variable.

4.185.4.2 ac algo_comm_t MHA_AC::float_t::ac [private]

4.185.4.3 name std::string MHA_AC::float_t::name [private]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

4.186 MHA_AC::int_t Class Reference

Insert a integer variable into the AC space.

Public Member Functions

- **int_t (algo_comm_t ac, std::string name, int val=0)**
- **~int_t ()**
- **void insert ()**

Public Attributes

- **int data**
Integer value variable.

Private Attributes

- **algo_comm_t ac**
- **std::string name**

4.186.1 Detailed Description

Insert a integer variable into the AC space.

The variable is automatically removed on destruction.

4.186.2 Constructor & Destructor Documentation

4.186.2.1 int_t() `MHA_AC::int_t::int_t (algo_comm_t ac, std::string name, int val = 0)`

4.186.2.2 ~int_t() `MHA_AC::int_t::~int_t ()`

4.186.3 Member Function Documentation

4.186.3.1 insert() `void MHA_AC::int_t::insert ()`

4.186.4 Member Data Documentation

4.186.4.1 data `int MHA_AC::int_t::data`

Integer value variable.

4.186.4.2 ac `algo_comm_t MHA_AC::int_t::ac [private]`

4.186.4.3 **name** std::string MHA_AC::int_t::name [private]

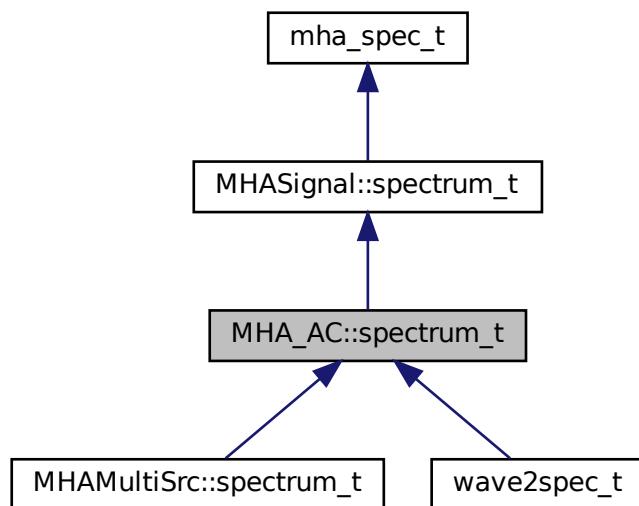
The documentation for this class was generated from the following files:

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

4.187 MHA_AC::spectrum_t Class Reference

Insert a **MHASignal::spectrum_t** (p. 1244) 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

4.187.1 Detailed Description

Insert a `MHASignal::spectrum_t` (p. 1244) class into the AC space.

The variable is automatically removed on destruction.

4.187.2 Constructor & Destructor Documentation

```
4.187.2.1 spectrum_t() 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

<code>ac</code>	AC handle
<code>name</code>	Name of variable in AC space
<code>bins</code>	Number of FFT bins in the <code>waveform_t</code> (p. 738) class
<code>channels</code>	Number of audio channels in the <code>waveform_t</code> (p. 738) class
<code>insert_now</code>	Insert implicitly in the constructor (true) or explicitly in the <code>insert()</code> (p. 736) function (false)

```
4.187.2.2 ~spectrum_t() MHA_AC::spectrum_t::~spectrum_t ( ) [virtual]
```

Reimplemented from `MHASignal::spectrum_t` (p. 1246).

4.187.3 Member Function Documentation

4.187.3.1 **insert()** void MHA_AC::spectrum_t::insert ()

Insert AC variable into AC space.

4.187.4 Member Data Documentation

4.187.4.1 **ac algo_comm_t** MHA_AC::spectrum_t::ac [protected]

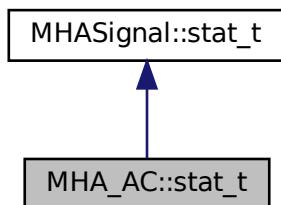
4.187.4.2 **name std::string** MHA_AC::spectrum_t::name [protected]

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

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

4.188 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**

4.188.1 Constructor & Destructor Documentation

```
4.188.1.1 stat_t() MHA_AC::stat_t::stat_t (
    algo_comm_t ac,
    const std::string & name,
    const unsigned int & frames,
    const unsigned int & channels,
    bool insert_now )
```

4.188.2 Member Function Documentation

```
4.188.2.1 update() void MHA_AC::stat_t::update ( )
```

```
4.188.2.2 insert() void MHA_AC::stat_t::insert ( )
```

4.188.3 Member Data Documentation

4.188.3.1 **mean** MHA_AC::waveform_t MHA_AC::stat_t::mean [private]

4.188.3.2 **std** MHA_AC::waveform_t MHA_AC::stat_t::std [private]

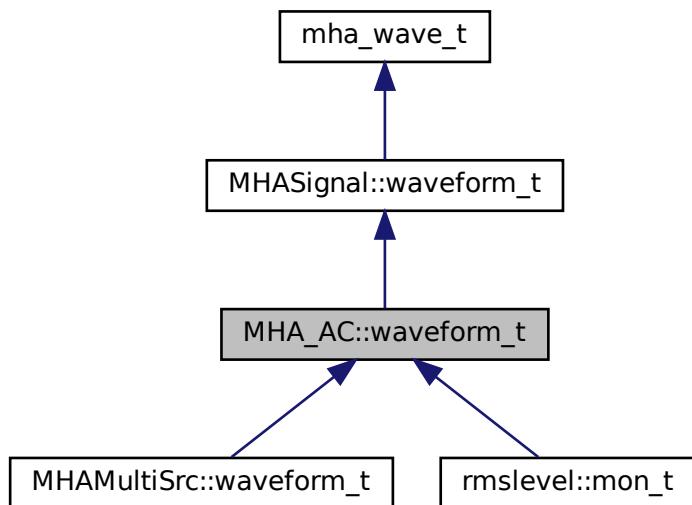
The documentation for this class was generated from the following files:

- **mha_algo_comm.h**
- **mha_algo_comm.cpp**

4.189 MHA_AC::waveform_t Class Reference

Insert a **MHASignal::waveform_t** (p. 1259) 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

4.189.1 Detailed Description

Insert a `MHASignal::waveform_t` (p. 1259) class into the AC space.

The variable is automatically removed on destruction.

4.189.2 Constructor & Destructor Documentation

```
4.189.2.1 waveform_t() 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

<code>ac</code>	AC handle
<code>name</code>	Name of variable in AC space
<code>frames</code>	Number of frames in the <code>waveform_t</code> (p. 738) class
<code>channels</code>	Number of audio channels in the <code>waveform_t</code> (p. 738) class
<code>insert_now</code>	Insert implicitly in the constructor (true) or explicitly in the <code>insert()</code> (p. 740) function (false)

```
4.189.2.2 ~waveform_t() MHA_AC::waveform_t::~waveform_t ( ) [virtual]
```

Reimplemented from `MHASignal::waveform_t` (p. 1262).

4.189.3 Member Function Documentation

4.189.3.1 `insert()` `void MHA_AC::waveform_t::insert()`

Insert AC variable into AC space.

4.189.4 Member Data Documentation

4.189.4.1 `ac` `algo_comm_t MHA_AC::waveform_t::ac` [protected]

4.189.4.2 `name` `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`

4.190 `mha_audio_descriptor_t` Struct Reference

Description of an audio fragment (planned as a replacement of `mhaconfig_t` (p. 850)).

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.

4.190.1 Detailed Description

Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 850)).

4.190.2 Member Data Documentation

4.190.2.1 n_samples `unsigned int mha_audio_descriptor_t::n_samples`

Number of samples.

4.190.2.2 n_channels `unsigned int mha_audio_descriptor_t::n_channels`

Number of audio channels.

4.190.2.3 n_freqs `unsigned int mha_audio_descriptor_t::n_freqs`

Number of frequency bands.

4.190.2.4 is_complex `unsigned int mha_audio_descriptor_t::is_complex`

Flag about sample type.

4.190.2.5 dt `mha_real_t mha_audio_descriptor_t::dt`

Time distance between samples (only equidistant samples allowed)

4.190.2.6 cf mha_real_t* mha_audio_descriptor_t::cf

Center frequencies of frequency bands.

4.190.2.7 chdir mha_real_t* mha_audio_descriptor_t::chdir

Hint on source direction of channel, values below zero is left, values above zero is right, zero means unknown.

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

- **mha.hh**

4.191 mha_audio_t Struct Reference

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793)).

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. 741) is unset.*
- **mha_complex_t * cdata**
*Data pointer if flag **mha_audio_descriptor_t::is_complex** (p. 741) is set.*

4.191.1 Detailed Description

An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793)).

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. 839) and **mha_spec_t** (p. 793) data pointers into corresponding **mha_audio_t** (p. 742) objects.

4.191.2 Member Data Documentation

4.191.2.1 descriptor `mha_audio_descriptor_t` `mha_audio_t::descriptor`

Dimension and description of the data.

4.191.2.2 rdata `mha_real_t*` `mha_audio_t::rdata`

Data pointer if flag `mha_audio_descriptor_t::is_complex` (p. 741) is unset.

4.191.2.3 cdata `mha_complex_t*` `mha_audio_t::cdata`

Data pointer if flag `mha_audio_descriptor_t::is_complex` (p. 741) is set.

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

- `mha.hh`

4.192 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.

4.192.1 Detailed Description

Channel information structure.

4.192.2 Member Data Documentation

4.192.2.1 id int mha_channel_info_t::id

channel id

4.192.2.2 idstr char mha_channel_info_t::idstr[32]

channel id

4.192.2.3 side unsigned int mha_channel_info_t::side

side (left/right)

4.192.2.4 dir mha_direction_t mha_channel_info_t::dir

source direction

4.192.2.5 peaklevel mha_real_t mha_channel_info_t::peaklevel

Peak level corresponds to this SPL (dB) level.

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

- **mha.hh**

4.193 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.

4.193.1 Detailed Description

Type for complex floating point values.

4.193.2 Member Data Documentation

4.193.2.1 re mha_real_t mha_complex_t::re

Real part.

4.193.2.2 im mha_real_t mha_complex_t::im

Imaginary part.

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

- **mha.hh**

4.194 mha_complex_test_array_t Struct Reference

Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 744) `c[]` to an array of `mha_real_t r[]` with `r[0] == c[0].re` `r[1] == c[0].im` `r[2] == c[1].re` ...

Public Attributes

- **mha_complex_t c [2]**

4.194.1 Detailed Description

Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 744) `c[]` to an array of **mha_real_t** `r[]` with `r[0] == c[0].re` `r[1] == c[0].im` `r[2] == c[1].re` ...

Check these expectations in static asserts.

4.194.2 Member Data Documentation

4.194.2.1 C **mha_complex_t** `mha_complex_test_array_t::c[2]`

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

- **mha.hh**

4.195 **mha dblbuf_t< FIFO >** Class Template Reference

The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.

Public Types

- `typedef FIFO::value_type value_type`

The datatype exchanged by the FIFO and this doublebuffer.

Public Member Functions

- virtual unsigned **get_inner_size** () const
- virtual unsigned **get_outer_size** () const
- virtual unsigned **get_delay** () const
- virtual unsigned **get_fifo_size** () const
- virtual unsigned **get_input_channels** () const
- virtual unsigned **get_output_channels** () const
- virtual unsigned **get_input_fifo_fill_count** () const
- virtual unsigned **get_output_fifo_fill_count** () const
- virtual unsigned **get_input_fifo_space** () const
- virtual unsigned **get_output_fifo_space** () const
- virtual **MHA_Error** * **get_inner_error** () const
- virtual void **provoke_inner_error** (const **MHA_Error** &)
- virtual void **provoke_outer_error** (const **MHA_Error** &)
- **mha_dblbuf_t** (unsigned **outer_size**, unsigned **inner_size**, unsigned **delay**, unsigned **input_channels**, unsigned **output_channels**, const **value_type** &**delay_data**)

Constructor creates FIFOs with specified delay.

- virtual ~**mha_dblbuf_t** ()
- virtual void **process** (const **value_type** ***input_signal**, **value_type** ***output_signal**, unsigned count)

The outer process has to call this method to propagate the input signal to the inner process, and receives back the output signal.

- virtual void **input** (**value_type** ***input_signal**)

The inner process has to call this method to receive its input signal.

- virtual void **output** (const **value_type** ***output_signal**)

The outer process has to call this method to deliver its output signal.

Private Attributes

- unsigned **outer_size**
The block size used by the outer process.
- unsigned **inner_size**
The block size used by the inner process.
- unsigned **delay**
The delay introduced by bidirectional buffer size adaptation.
- unsigned **fifo_size**
The size of each of the FIFOs.
- unsigned **input_channels**
The number of input channels.
- unsigned **output_channels**
The number of output channels.
- FIFO **input_fifo**
The FIFO for transporting the input signal from the outer process to the inner process.
- FIFO **output_fifo**

The FIFO for transporting the output signal from the inner process to the outer process.

- **MHA_Error * inner_error**

Owned copy of exception to be thrown in inner thread.

- **MHA_Error * outer_error**

Owned copy of exception to be thrown in outer thread.

4.195.1 Detailed Description

```
template<class FIFO>
class mha_dblbuf_t< FIFO >
```

The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.

This class introduces the channels concept. Input and output may have different channel counts.

4.195.2 Member Typedef Documentation

```
4.195.2.1 value_type template<class FIFO >
typedef FIFO::value_type mha_dblbuf_t< FIFO >:: value_type
```

The datatype exchanged by the FIFO and this doublebuffer.

4.195.3 Constructor & Destructor Documentation

```
4.195.3.1 mha_dblbuf_t() template<class FIFO >
mha_dblbuf_t< FIFO >:: mha_dblbuf_t (
    unsigned outer_size,
    unsigned inner_size,
    unsigned delay,
    unsigned input_channels,
    unsigned output_channels,
    const value_type & delay_data )
```

Constructor creates FIFOs with specified delay.

Warning

The doublebuffer may block or raise an exception if the delay is too small. To avoid this, the delay should be

$$\text{delay} \geq (\text{inner_size} - \text{gcd}(\text{inner_size}, \text{outer_size}))$$

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.

4.195.3.2 ~mha_dblbuf_t() template<class FIFO >
mha_dblbuf_t< FIFO >::~ mha_dblbuf_t [virtual]

4.195.4 Member Function Documentation

4.195.4.1 get_inner_size() template<class FIFO >
virtual unsigned mha_dblbuf_t< FIFO >::get_inner_size () const [inline], [virtual]

4.195.4.2 get_outer_size() template<class FIFO >
virtual unsigned mha_dblbuf_t< FIFO >::get_outer_size () const [inline], [virtual]

4.195.4.3 get_delay() template<class FIFO >
virtual unsigned mha_dblbuf_t< FIFO >::get_delay () const [inline], [virtual]

4.195.4.4 get_fifo_size() template<class FIFO >
virtual unsigned mha_dblbuf_t< FIFO >::get_fifo_size () const [inline], [virtual]

4.195.4.5 get_input_channels() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_input_channels () const [inline],
[virtual]

4.195.4.6 get_output_channels() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_output_channels () const [inline],
[virtual]

4.195.4.7 get_input_fifo_fill_count() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_input_fifo_fill_count () const [inline],
[virtual]

4.195.4.8 get_output_fifo_fill_count() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_output_fifo_fill_count () const [inline],
[virtual]

4.195.4.9 get_input_fifo_space() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_input_fifo_space () const [inline],
[virtual]

4.195.4.10 get_output_fifo_space() template<class FIFO >
virtual unsigned **mha_dbdbuf_t**< FIFO >::get_output_fifo_space () const [inline],
[virtual]

4.195.4.11 get_inner_error() template<class FIFO >
virtual **MHA_Error*** **mha_dbdbuf_t**< FIFO >::get_inner_error () const [inline],
[virtual]

```
4.195.4.12 provoke_inner_error() template<class FIFO >
void mha_dblbuf_t< FIFO >::provoke_inner_error (
    const MHA_Error & error ) [virtual]
```

```
4.195.4.13 provoke_outer_error() template<class FIFO >
void mha_dblbuf_t< FIFO >::provoke_outer_error (
    const MHA_Error & error ) [virtual]
```

```
4.195.4.14 process() template<class FIFO >
void mha_dblbuf_t< FIFO >::process (
    const value_type * input_signal,
    value_type * output_signal,
    unsigned count ) [virtual]
```

The outer process has to call this method to propagate the input signal to the inner process, and receives back the output signal.

Parameters

<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. 763)	When count is > outer_size as given to constructor or the underlying fifo implementation detects an error.
---------------------------	--

```
4.195.4.15 input() template<class FIFO >
void mha_dblbuf_t< FIFO >::input (
    value_type * input_signal ) [virtual]
```

The inner process has to call this method to receive its input signal.

Parameters

<i>input_signal</i>	Array where the doublebuffer can store the signal.
---------------------	--

Exceptions

MHA_Error (p. 763)	When the underlying fifo implementation detects an error.
---------------------------	---

```
4.195.4.16 output() template<class FIFO >
void mha_dblbuf_t< FIFO >::output (
    const value_type * output_signal ) [virtual]
```

The outer process has to call this method to deliver its output signal.

Parameters

<i>output_signal</i>	Array from which doublebuffer reads outputsignal.
----------------------	---

Exceptions

MHA_Error (p. 763)	When the underlying fifo implementation detects an error.
---------------------------	---

4.195.5 Member Data Documentation

```
4.195.5.1 outer_size template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::outer_size [private]
```

The block size used by the outer process.

```
4.195.5.2 inner_size template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::inner_size [private]
```

The block size used by the inner process.

4.195.5.3 delay template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::delay [private]

The delay introduced by bidirectional buffer size adaptation.

4.195.5.4 fifo_size template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::fifo_size [private]

The size of each of the FIFOs.

4.195.5.5 input_channels template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::input_channels [private]

The number of input channels.

4.195.5.6 output_channels template<class FIFO >
unsigned mha_dblbuf_t< FIFO >::output_channels [private]

The number of output channels.

4.195.5.7 input_fifo template<class FIFO >
FIFO mha_dblbuf_t< FIFO >::input_fifo [private]

The FIFO for transporting the input signal from the outer process to the inner process.

4.195.5.8 output_fifo template<class FIFO >
FIFO mha_dblbuf_t< FIFO >::output_fifo [private]

The FIFO for transporting the output signal from the inner process to the outer process.

4.195.5.9 inner_error template<class FIFO >
`MHA_Error* mha_dbdbuf_t< FIFO >::inner_error [private]`

Owned copy of exception to be thrown in inner thread.

4.195.5.10 outer_error template<class FIFO >
`MHA_Error* mha_dbdbuf_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](#)

4.196 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

4.196.1 Detailed Description

Channel source direction structure.

4.196.2 Member Data Documentation

4.196.2.1 azimuth mha_real_t mha_direction_t::azimuth

azimuth in radians

4.196.2.2 elevation mha_real_t mha_direction_t::elevation

elevation in radians

4.196.2.3 distance mha_real_t mha_direction_t::distance

distance in meters

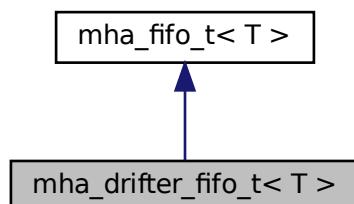
The documentation for this struct was generated from the following file:

- mha.hh

4.197 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. 763) 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. 763) 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. 758) or **mha_drifter_fifo_t<T>::write** (p. 758) when their xrun in succession counter exceeds its limit.*
- virtual void **starting** ()
*Called by **mha_drifter_fifo_t<T>::read** (p. 758) or **mha_drifter_fifo_t<T>::write** (p. 758) when the respective flag (**mha_drifter_fifo_t<T>::reader_started** (p. 761) or **mha_drifter_fifo_t<T>::writer_started** (p. 761)) 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. 761) and **mha_drifter_fifo_t< T >::writer_started** (p. 761) are true, then first **mha_drifter_fifo_t< T >::desired_fill_count** (p. 761) instances of **mha_drifter_fifo_t< T >::null_data** (p. 762) are delivered to the reader.*

Additional Inherited Members

4.197.1 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.

4.197.2 Constructor & Destructor Documentation

4.197.2.1 mha_drifter_fifo_t() [1/2] template<class T >

```
mha_drifter_fifo_t< T >:: mha_drifter_fifo_t (
    unsigned min_fill_count,
    unsigned desired_fill_count,
    unsigned max_fill_count )
```

Create drifter FIFO.

```
4.197.2.2 mha_drifter_fifo_t() [2/2] template<class T >
mha_drifter_fifo_t< T >::: mha_drifter_fifo_t (
    unsigned min_fill_count,
    unsigned desired_fill_count,
    unsigned max_fill_count,
    const T & t )
```

Create drifter FIFO where all (initially unused) copies of T are initialized as copies of t.

4.197.3 Member Function Documentation

```
4.197.3.1 write() template<class T >
void mha_drifter_fifo_t< T >::write (
    const T * data,
    unsigned count ) [virtual]
```

write data to fifo

Sets **writer_started** (p. 761) to true.

When processing has started, i.e. both **reader_started** (p. 761) and **writer_started** (p. 761) are true, write specified amount 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. 762) exceeds **maximum_writer_xruns_in_succession_before_stop** (p. 762).

Parameters

<i>data</i>	Pointer to source data.
<i>count</i>	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 778).

```
4.197.3.2 read() template<class T >
void mha_drifter_fifo_t< T >::read (
    T * buf,
    unsigned count ) [virtual]
```

Read data from fifo.

Sets **reader_started** (p. 761) to true.

When processing has started, i.e. both **reader_started** (p. 761) and **writer_started** (p. 761) are true, then read specified amount of data from the fifo. As long as **startup_zeros** (p. 763) is > 0 , **null_data** (p. 762) is delivered to the reader and **startup_zeros** (p. 763) is diminished. Only when **startup_zeros** (p. 763) 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. 760), then only so much data are read that **minimum_fill_count** (p. 760) entries remain in the fifo, the missing data is replaced with **null_data** (p. 762), and the reader xrun counters are increased.

Processing is stopped when **reader_xruns_in_succession** (p. 762) exceeds **maximum_reader_xruns_in_succession_before_stop** (p. 762).

Parameters

<i>buf</i>	Pointer to the target buffer
<i>count</i>	Number of instances to copy

Reimplemented from **mha_fifo_t< T >** (p. 779).

```
4.197.3.3 get_fill_count() template<class T >
unsigned mha_drifter_fifo_t< T >::get_fill_count [virtual]
```

Return fill_count, adding **mha_drifter_fifo_t<T>::startup_zeros** (p. 763) to the number of samples actually in the fifo's buffer.

Reimplemented from **mha_fifo_t< T >** (p. 779).

```
4.197.3.4 get_available_space() template<class T >
unsigned mha_drifter_fifo_t< T >::get_available_space [virtual]
```

Return available space, subtracting number of **mha_drifter_fifo_t<T>::startup_zeros** (p. 763) 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. 779).

4.197.3.5 `get_des_fill_count()` template<class T >
 virtual unsigned `mha_drifter_fifo_t< T >::get_des_fill_count () const [inline],`
`[virtual]`

The desired fill count of this fifo.

4.197.3.6 `get_min_fill_count()` template<class T >
 virtual unsigned `mha_drifter_fifo_t< T >::get_min_fill_count () const [inline],`
`[virtual]`

The minimum fill count of this fifo.

4.197.3.7 `stop()` template<class T >
 void `mha_drifter_fifo_t< T >::stop [virtual]`

Called by `mha_drifter_fifo_t<T>::read` (p. 758) or `mha_drifter_fifo_t<T>::write` (p. 758) when their xrun in succession counter exceeds its limit.

Called by `read` (p. 758) or `write` (p. 758) when their xrun in succession counter exceeds its limit.

May also be called explicitly.

4.197.3.8 `starting()` template<class T >
 void `mha_drifter_fifo_t< T >::starting [virtual]`

Called by `mha_drifter_fifo_t<T>::read` (p. 758) or `mha_drifter_fifo_t<T>::write` (p. 758) when the respective flag (`mha_drifter_fifo_t<T>::reader_started` (p. 761) or `mha_drifter_fifo_t<T>::writer_started` (p. 761)) is about to be toggled from false to true.

The fifo's buffer is emptied, this method resets `startup_zeros` (p. 763) to `desired_fill_count` (p. 761), and it also resets `reader_xruns_since_start` (p. 762) and `writer_xruns_since_start` (p. 761) to 0.

4.197.4 Member Data Documentation

4.197.4.1 minimum_fill_count template<class T >
const unsigned **mha_drifter_fifo_t< T >::minimum_fill_count** [private]

The minimum fill count of this fifo.

4.197.4.2 desired_fill_count template<class T >
const unsigned **mha_drifter_fifo_t< T >::desired_fill_count** [private]

The desired fill count of the fifo.

The fifo is initialized with this amount of data when data transmission starts.

4.197.4.3 writer_started template<class T >
bool **mha_drifter_fifo_t< T >::writer_started** [private]

Flag set to true when write is called the first time.

4.197.4.4 reader_started template<class T >
bool **mha_drifter_fifo_t< T >::reader_started** [private]

Flag set to true when read is called for the first time.

4.197.4.5 writer_xruns_total template<class T >
unsigned **mha_drifter_fifo_t< T >::writer_xruns_total** [private]

The number of xruns seen by the writer since object instantiation.

4.197.4.6 reader_xruns_total template<class T >
unsigned **mha_drifter_fifo_t< T >::reader_xruns_total** [private]

The number of xruns seen by the reader since object instantiation.

4.197.4.7 writer_xruns_since_start template<class T >
 unsigned **mha_drifter_fifo_t**< T >::writer_xruns_since_start [private]

The number of xruns seen by the writer since the last start of processing.

4.197.4.8 reader_xruns_since_start template<class T >
 unsigned **mha_drifter_fifo_t**< T >::reader_xruns_since_start [private]

The number of xruns seen by the reader since the last start of processing.

4.197.4.9 writer_xruns_in_succession template<class T >
 unsigned **mha_drifter_fifo_t**< T >::writer_xruns_in_succession [private]

The number of xruns seen by the writer in succession.

Reset to 0 every time a write succeeds without xrun.

4.197.4.10 reader_xruns_in_succession template<class T >
 unsigned **mha_drifter_fifo_t**< T >::reader_xruns_in_succession [private]

The number of xruns seen by the reader in succession.

Reset to 0 every time a read succeeds without xrun.

4.197.4.11 maximum_writer_xruns_in_succession_before_stop template<class T >
 unsigned **mha_drifter_fifo_t**< T >::maximum_writer_xruns_in_succession_before_stop
 [private]

A limit to the number of xruns seen in succession during write before the data transmission through the FIFO is stopped.

4.197.4.12 maximum_reader_xruns_in_succession_before_stop template<class T >
 unsigned **mha_drifter_fifo_t**< T >::maximum_reader_xruns_in_succession_before_stop
 [private]

A limit to the number of xruns seen in succession during read before the data transmission through the FIFO is stopped.

```
4.197.4.13 null_data template<class T >
mha_fifo_t<T>:: value_type mha_drifter_fifo_t< T >::null_data [private]
```

The value used in place of missing data.

```
4.197.4.14 startup_zeros template<class T >
unsigned mha_drifter_fifo_t< T >::startup_zeros [private]
```

When processing starts, that is when both **mha_drifter_fifo_t<T>::reader_started** (p. 761) and **mha_drifter_fifo_t<T>::writer_started** (p. 761) are true, then first **mha_drifter_fifo_t< T >::desired_fill_count** (p. 761) instances of **mha_drifter_fifo_t<T>::null_data** (p. 762) are delivered to the reader.

These **null_data** (p. 762) instances are not transmitted through the fifo because filling the fifo with enough **null_data** (p. 762) might not be realtime safe and this filling has to be initiated by **starting** (p. 760) or **stop** (p. 760) (this implementation: **starting** (p. 760)) which are be called with realtime constraints.

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

- **mha_fifo.h**

4.198 MHA_Error Class Reference

Error reporting exception class.

Inherits exception.

Public Member Functions

- **MHA_Error** (const char *file, int line, const char *fmt,...) **__attribute__((__format__(printf
Create an instance of a **MHA_Error** (p. 763).**
- **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**

4.198.1 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. 763) class holds source file name, line number and an error message.

4.198.2 Constructor & Destructor Documentation

```
4.198.2.1 MHA_Error() [1/2] MHA_Error::MHA_Error (
    const char * s_file,
    int l,
    const char * fmt,
    ... )
```

Create an instance of a **MHA_Error** (p. 763).

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)

```
4.198.2.2 MHA_Error() [2/2] MHA_Error::MHA_Error (
    const MHA_Error & p )
```

4.198.2.3 ~MHA_Error() `MHA_Error::~MHA_Error () throw ()`

4.198.3 Member Function Documentation

4.198.3.1 operator=() `MHA_Error & MHA_Error::operator= (const MHA_Error & p)`

4.198.3.2 get_msg() `const char* MHA_Error::get_msg () const [inline]`

Return the error message without source position.

4.198.3.3 get_longmsg() `const char* MHA_Error::get_longmsg () const [inline]`

Return the error message with source position.

4.198.3.4 what() `const char* MHA_Error::what () const throw () [inline]`

overwrite std::exception::what()

4.198.4 Member Data Documentation

4.198.4.1 msg `char* MHA_Error::msg [private]`

4.198.4.2 **longmsg** `char* MHA_Error::longmsg [private]`

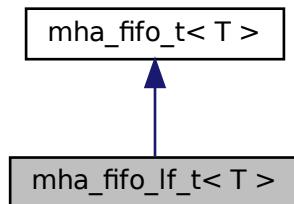
The documentation for this class was generated from the following files:

- `mha_error.hh`
- `mha_error.cpp`

4.199 **mha_fifo_if_t< T >** Class Template Reference

A lock-free FIFO class for transferring data from a producer thread to a consumer thread.

Inheritance diagram for `mha_fifo_if_t< T >`:



Public Member Functions

- virtual void **write** (const T *data, unsigned count) override
Write specified amount of data to the fifo.
- virtual void **read** (T *outbuf, unsigned count) override
Read data from fifo.
- virtual unsigned **get_fill_count** () const override
get_fill_count() (p. 768) must only be called by the reader thread
- virtual unsigned **get_available_space** () const override
get_available_space() (p. 768) must only be called by the writer thread
- **mha_fifo_if_t** (unsigned max_fill_count, const T &t=T())
Create FIFO with fixed buffer size.

Private Attributes

- std::atomic< const T * > **atomic_write_ptr**
atomic copy of the write_ptr, only modified by the producer thread
- std::atomic< const T * > **atomic_read_ptr**
atomic copy of the read ptr, only modified by the consumer thread

Additional Inherited Members

4.199.1 Detailed Description

```
template<class T>
class mha_fifo_lf_t< T >
```

A lock-free FIFO class for transferring data from a producer thread to a consumer thread.

Inherits basic functionality from **mha_fifo_t** (p. 775), adds release-acquire semantics to ensure consumer that the fill count or free space deduced from read and write pointers is consistent with the actual data. Copying, moving, and assignment of FIFO are forbidden by base class.

4.199.2 Constructor & Destructor Documentation

```
4.199.2.1 mha_fifo_lf_t() template<class T >
mha_fifo_lf_t< T >:: mha_fifo_lf_t (
    unsigned max_fill_count,
    const T & t = T() ) [inline], [explicit]
```

Create FIFO with fixed buffer size.

All (initially unused) instances of T are initialized as copies of t

4.199.3 Member Function Documentation

```
4.199.3.1 write() template<class T >
virtual void mha_fifo_lf_t< T >::write (
    const T * data,
    unsigned count ) [inline], [override], [virtual]
```

Write specified amount of data to the fifo.

Must only be called by the writer thread.

Parameters

<i>data</i>	Pointer to source data.
<i>count</i>	Number of instances to copy.

Exceptions

MHA_Error (p. 763)	when there is not enough space available.
---------------------------	---

Reimplemented from **mha_fifo_t< T >** (p. 778).

```
4.199.3.2 read() template<class T >
virtual void mha_fifo_lf_t< T >::read (
    T * outbuf,
    unsigned count) [inline], [override], [virtual]
```

Read data from fifo.

Must only be called by the reader thread.

Parameters

<i>outbuf</i>	Pointer to the target buffer.
<i>count</i>	Number of instances to copy.

Exceptions

MHA_Error (p. 763)	when there is not enough data available.
---------------------------	--

Reimplemented from **mha_fifo_t< T >** (p. 779).

```
4.199.3.3 get_fill_count() template<class T >
virtual unsigned mha_fifo_lf_t< T >::get_fill_count () const [inline], [override],
[virtual]
```

get_fill_count() (p. 768) must only be called by the reader thread

Reimplemented from **mha_fifo_t< T >** (p. 779).

```
4.199.3.4 get_available_space() template<class T >
virtual unsigned mha_fifo_lf_t< T >::get_available_space ( ) const [inline], [override],
[virtual]
```

get_available_space() (p. 768) must only be called by the writer thread

Reimplemented from **mha_fifo_t< T >** (p. 779).

4.199.4 Member Data Documentation

```
4.199.4.1 atomic_write_ptr template<class T >
std::atomic<const T *> mha_fifo_lf_t< T >::atomic_write_ptr [private]
```

atomic copy of the write_ptr, only modified by the producer thread

```
4.199.4.2 atomic_read_ptr template<class T >
std::atomic<const T *> mha_fifo_lf_t< T >::atomic_read_ptr [private]
```

atomic copy of the read_ptr, only modified by the consumer thread

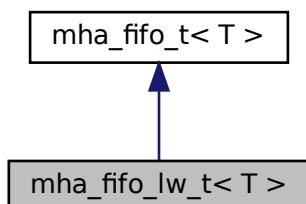
The documentation for this class was generated from the following file:

- **mha_fifo.h**

4.200 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 amount 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

4.200.1 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.

4.200.2 Constructor & Destructor Documentation

```
4.200.2.1 mha_fifo_lw_t() template<class T >
mha_fifo_lw_t< T >:: mha_fifo_lw_t (
    unsigned max_fill_count ) [explicit]
```

Create FIFO with fixed buffer size.

```
4.200.2.2 ~mha_fifo_lw_t() template<class T >
mha_fifo_lw_t< T >::~ mha_fifo_lw_t [virtual]
```

release synchronization object

4.200.3 Member Function Documentation

4.200.3.1 write() template<class T >

```
void mha_fifo_lw_t< T >::write (
    const T * data,
    unsigned count ) [virtual]
```

write specified amount 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

MHA_Error (p. 763)	when detecting a deadlock situation.
--	--------------------------------------

Reimplemented from **mha_fifo_t< T >** (p. [778](#)).

4.200.3.2 read() template<class T >

```
void mha_fifo_lw_t< T >::read (
    T * buf,
    unsigned count ) [virtual]
```

read data from fifo.

If there is not enough data, then wait for more data.

Parameters

<i>buf</i>	Pointer to the target buffer.
<i>count</i>	Number of instances to copy.

Exceptions

MHA_Error (p. 763)	when detecting a deadlock situation.
---------------------------	--------------------------------------

Reimplemented from **mha_fifo_t< T >** (p. 779).

```
4.200.3.3 set_error() template<class T >
void mha_fifo_lw_t< T >::set_error (
    unsigned index,
    MHA_Error * error ) [virtual]
```

Process waiting for more data or space should bail out, throwing this error.

Parameters

<i>index</i>	Use 0 for terminating reader, 1 for terminating writer.
<i>error</i>	MHA_Error (p. 763) to be thrown

4.200.4 Member Data Documentation

```
4.200.4.1 sync template<class T >
mha_fifo_thread_platform_t* mha_fifo_lw_t< T >::sync [private]
platform specific thread synchronization
```

```
4.200.4.2 error template<class T >
MHA_Error* mha_fifo_lw_t< T >::error[2] [private]
```

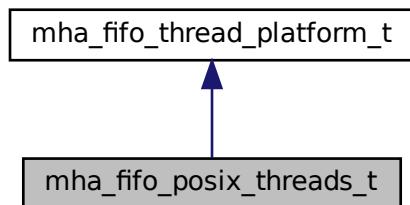
If waiting for synchronization should be aborted then exception to be thrown by reader process (index 0) or writer process (index 1) has to be placed here.

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

- **mha_fifo.h**
- **mha_fifo.cpp**

4.201 mha_fifo_posix_threads_t Class Reference

Inheritance diagram for mha_fifo_posix_threads_t:



Public Member Functions

- **mha_fifo_posix_threads_t ()**
- virtual void **aquire_mutex ()**
- virtual void **release_mutex ()**
- virtual void **wait_for_decrease ()**
- virtual void **wait_for_increase ()**
- virtual void **increment ()**
- virtual void **decrement ()**
- virtual ~**mha_fifo_posix_threads_t ()**

Private Attributes

- pthread_mutex_t **mutex**
- pthread_cond_t **decrease_condition**
- pthread_cond_t **increase_condition**

4.201.1 Constructor & Destructor Documentation

4.201.1.1 mha_fifo_posix_threads_t() mha_fifo_posix_threads_t::mha_fifo_posix_posix_threads_t () [inline]

4.201.1.2 ~mha_fifo_posix_threads_t() virtual mha_fifo_posix_threads_t::~mha_fifo_posix_threads_t () [inline], [virtual]

4.201.2 Member Function Documentation

4.201.2.1 acquire_mutex() virtual void mha_fifo_posix_threads_t::acquire_mutex () [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. [785](#)).

4.201.2.2 release_mutex() virtual void mha_fifo_posix_threads_t::release_mutex () [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. [785](#)).

4.201.2.3 wait_for_decrease() virtual void mha_fifo_posix_threads_t::wait_for_decrease () [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. [785](#)).

4.201.2.4 wait_for_increase() virtual void mha_fifo_posix_threads_t::wait_for_increase () [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. [786](#)).

4.201.2.5 increment() virtual void mha_fifo_posix_threads_t::increment () [inline], [virtual]

Implements **mha_fifo_thread_platform_t** (p. [786](#)).

4.201.2.6 decrement() virtual void mha_fifo_posix_threads_t::decrement () [inline], [virtual]

Implements [mha_fifo_thread_platform_t](#) (p. 786).

4.201.3 Member Data Documentation

4.201.3.1 mutex pthread_mutex_t mha_fifo_posix_threads_t::mutex [private]

4.201.3.2 decrease_condition pthread_cond_t mha_fifo_posix_threads_t::decrease_< condition [private]

4.201.3.3 increase_condition pthread_cond_t mha_fifo_posix_threads_t::increase_< condition [private]

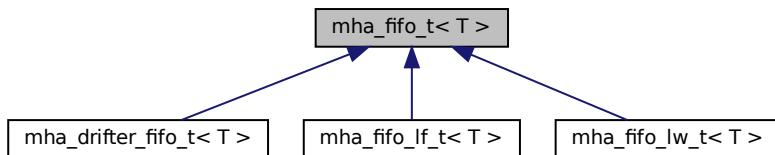
The documentation for this class was generated from the following file:

- [mha_fifo.h](#)

4.202 mha_fifo_t< T > Class Template Reference

A FIFO class.

Inheritance diagram for mha_fifo_t< T >:



Public Types

- `typedef std::vector< T >:: value_type value_type`
The data type exchanged by this fifo.

Public Member Functions

- `virtual void write (const T *data, unsigned count)`
write specified amount of data to the fifo.
- `virtual void read (T *outbuf, unsigned count)`
read data from fifo
- `virtual unsigned get_fill_count () const`
Read-only access to fill_count.
- `virtual unsigned get_available_space () const`
Read-only access to available_space.
- `virtual unsigned get_max_fill_count () const`
The capacity of this fifo.
- `mha_fifo_t (unsigned max_fill_count, const T &t=T())`
Create FIFO with fixed buffer size, where all (initially unused) instances of T are initialized as copies of t.
- `virtual ~mha_fifo_t ()=default`
Make destructor virtual.
- `mha_fifo_t (const mha_fifo_t &)=delete`
Copy constructor.
- `mha_fifo_t (mha_fifo_t &&)=delete`
Move constructor.
- `mha_fifo_t< T > & operator= (const mha_fifo_t< T > &)=delete`
Assignment operator.
- `mha_fifo_t< T > & operator= (mha_fifo_t< T > &&)=delete`
Move assignment operator.

Protected Member Functions

- `void clear ()`
Empty the fifo at once.
- `const T * get_write_ptr () const`
read-only access to the write pointer for derived classes
- `const T * get_read_ptr () const`
read-only access to read pointer for derived classes
- `unsigned get_fill_count (const T *wp, const T *rp) const`
Compute fill count from given write pointer and read pointer.

Private Attributes

- std::vector< T > **buf**
The memory allocated to store the data in the fifo.
- T * **write_ptr**
points to location where to write next
- const T * **read_ptr**
points to location where to read next

4.202.1 Detailed Description

```
template<class T>
class mha_fifo_t< T >
```

A FIFO class.

Synchronization: None. Use external synchronisation or synchronization in inheriting class.
Assignment, copy and move constructors are disabled.

4.202.2 Member Typedef Documentation

```
4.202.2.1 value_type template<class T >
typedef std::vector<T>:: value_type mha_fifo_t< T >:: value_type
```

The data type exchanged by this fifo.

4.202.3 Constructor & Destructor Documentation

```
4.202.3.1 mha_fifo_t() [1/3] template<class T >
mha_fifo_t< T >:: mha_fifo_t (
    unsigned max_fill_count,
    const T & t = T() ) [explicit]
```

Create FIFO with fixed buffer size, where all (initially unused) instances of T are initialized as copies of t.

Parameters

<i>max_fill_count</i>	The maximum number of instances of T that can be held at the same time inside the fifo.
<i>The</i>	fifo allocates a vector of <i>max_fill_count+1</i> instances of T for storage, one of which is always unused.

4.202.3.2 ~mha_fifo_t() template<class T >
 virtual **mha_fifo_t**< T >::~ **mha_fifo_t** () [virtual], [default]

Make destructor virtual.

4.202.3.3 mha_fifo_t() [2/3] template<class T >
mha_fifo_t< T >:: **mha_fifo_t** (
 const **mha_fifo_t**< T > &) [delete]

Copy constructor.

4.202.3.4 mha_fifo_t() [3/3] template<class T >
mha_fifo_t< T >:: **mha_fifo_t** (
mha_fifo_t< T > &&) [delete]

Move constructor.

4.202.4 Member Function Documentation

4.202.4.1 write() template<class T >
 void **mha_fifo_t**< T >::write (
 const T * *data*,
 unsigned *count*) [virtual]

write specified amount of data to the fifo.

Parameters

<i>data</i>	Pointer to source data.
<i>count</i>	Number of instances to copy

Exceptions

MHA_Error (p. 763)	when there is not enough space available.
---------------------------	---

Reimplemented in **mha_fifo_lf_t< T >** (p. 767), **mha_fifo_lw_t< T >** (p. 771), **mha_drifter_fifo_t< T >** (p. 758), **mha_fifo_lf_t< mha_real_t >** (p. 767), **mha_fifo_lw_t< mha_real_t >** (p. 771), and **mha_fifo_lf_t< MHA_AC::acspace2matrix_t >** (p. 767).

4.202.4.2 `read()` template<class T >
void mha_fifo_t< T >::read (
 *T * outbuf,*
 unsigned count) [virtual]

read data from fifo

Parameters

<i>outbuf</i>	Pointer to the target buffer.
<i>count</i>	Number of instances to copy.

Exceptions

MHA_Error (p. 763)	when there is not enough data available.
---------------------------	--

Reimplemented in **mha_fifo_lf_t< T >** (p. 768), **mha_fifo_lw_t< T >** (p. 771), **mha_drifter_fifo_t< T >** (p. 758), **mha_fifo_lf_t< mha_real_t >** (p. 768), **mha_fifo_lw_t< mha_real_t >** (p. 771), and **mha_fifo_lf_t< MHA_AC::acspace2matrix_t >** (p. 768).

4.202.4.3 `get_fill_count()` [1/2] template<class T >
virtual unsigned mha_fifo_t< T >::get_fill_count () const [inline], [virtual]

Read-only access to fill_count.

Reimplemented in **mha_fifo_lf_t< T >** (p. 768), **mha_fifo_lf_t< mha_real_t >** (p. 768), **mha_fifo_lf_t< MHA_AC::acspace2matrix_t >** (p. 768), and **mha_drifter_fifo_t< T >** (p. 759).

4.202.4.4 `get_available_space()` template<class T >
`unsigned mha_fifo_t< T >::get_available_space [virtual]`

Read-only access to available_space.

Reimplemented in `mha_fifo_if_t< T >` (p. 768), `mha_fifo_if_t< mha_real_t >` (p. 768), `mha_fifo_if_t< MHA_AC::acspace2matrix_t >` (p. 768), and `mha_drifter_fifo_t< T >` (p. 759).

4.202.4.5 `get_max_fill_count()` template<class T >
`virtual unsigned mha_fifo_t< T >::get_max_fill_count () const [inline], [virtual]`

The capacity of this fifo.

4.202.4.6 `operator=()` [1/2] template<class T >
`mha_fifo_t<T>& mha_fifo_t< T >::operator= (`
`const mha_fifo_t< T > &) [delete]`

Assignment operator.

4.202.4.7 `operator=()` [2/2] template<class T >
`mha_fifo_t<T>& mha_fifo_t< T >::operator= (`
`mha_fifo_t< T > &&) [delete]`

Move assignment operator.

4.202.4.8 `clear()` template<class T >
`void mha_fifo_t< T >::clear () [inline], [protected]`

Empty the fifo at once.

Should be called by the reader, or when the reader is inactive.

4.202.4.9 get_write_ptr() template<class T >

```
const T* mha_fifo_t< T >::get_write_ptr ( ) const [inline], [protected]
```

read-only access to the write pointer for derived classes

4.202.4.10 get_read_ptr() template<class T >

```
const T* mha_fifo_t< T >::get_read_ptr ( ) const [inline], [protected]
```

read-only access to read pointer for derived classes

4.202.4.11 get_fill_count() [2/2] template<class T >

```
unsigned mha_fifo_t< T >::get_fill_count (
    const T * wp,
    const T * rp ) const [inline], [protected]
```

Compute fill count from given write pointer and read pointer.

Parameters

<i>wp</i>	Write pointer.
<i>rp</i>	Read pointer.

Precondition

wp and *rp* must point to an instance of *T* inside *buf*.

Returns

Number of elements that can be read from the fifo when *wp* and *rp* have the given values.

4.202.5 Member Data Documentation

4.202.5.1 buf template<class T >

```
std::vector<T> mha_fifo_t< T >::buf [private]
```

The memory allocated to store the data in the fifo.

At least one location in *buf* is always unused, because we have *max_fill_count* + 1 possible fillcounts [0:*max_fill_count*] that we need to distinguish.

4.202.5.2 write_ptr template<class T >
T* **mha_fifo_t**< T >::write_ptr [private]

points to location where to write next

4.202.5.3 read_ptr template<class T >
const T* **mha_fifo_t**< T >::read_ptr [private]

points to location where to read next

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

- **mha_fifo.h**

4.203 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**

4.203.1 Detailed Description

Simple Mutex Guard Class.

4.203.2 Constructor & Destructor Documentation

```
4.203.2.1 mha_fifo_thread_guard_t() mha_fifo_thread_guard_t::mha_fifo_thread_<-
guard_t (
    mha_fifo_thread_platform_t * sync ) [inline]
```

```
4.203.2.2 ~mha_fifo_thread_guard_t() mha_fifo_thread_guard_t::~mha_fifo_thread_<-
guard_t () [inline]
```

4.203.3 Member Data Documentation

```
4.203.3.1 sync mha_fifo_thread_platform_t* mha_fifo_thread_guard_t::sync [private]
```

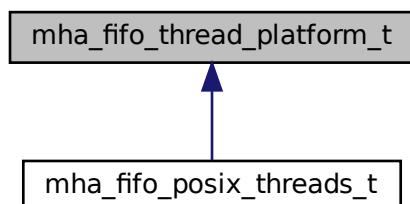
The documentation for this class was generated from the following file:

- [mha_fifo.h](#)

4.204 mha_fifo_thread_platform_t Class Reference

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Inheritance diagram for mha_fifo_thread_platform_t:



Public Member Functions

- virtual void **aquire_mutex** ()=0
Calling thread waits until it aquires the lock.
- virtual void **release_mutex** ()=0
Calling thread releases the lock.
- virtual void **wait_for_decrease** ()=0
Calling producer thread must own the lock.
- virtual void **wait_for_increase** ()=0
Calling consumer thread must own the lock.
- virtual void **increment** ()=0
To be called by producer thread after producing.
- virtual void **decrement** ()=0
To be called by consumer thread after consuming.
- virtual ~**mha_fifo_thread_platform_t** ()
Make destructor virtual.
- **mha_fifo_thread_platform_t** ()
Make default constructor accessible.

Private Member Functions

- **mha_fifo_thread_platform_t** (const **mha_fifo_thread_platform_t** &)
- **mha_fifo_thread_platform_t** & **operator=** (const **mha_fifo_thread_platform_t** &)

4.204.1 Detailed Description

Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.

Works only with single producer and single consumer.

4.204.2 Constructor & Destructor Documentation

4.204.2.1 ~mha_fifo_thread_platform_t() virtual **mha_fifo_thread_platform_t**::~**mha_fifo_thread_platform_t** () [inline], [virtual]

Make destructor virtual.

```
4.204.2.2 mha_fifo_thread_platform_t() [1/2] mha_fifo_thread_platform_t::mha_fifo->
_thread_platform_t (
    const mha_fifo_thread_platform_t & ) [private]
```

```
4.204.2.3 mha_fifo_thread_platform_t() [2/2] mha_fifo_thread_platform_t::mha_fifo->
_thread_platform_t ( ) [inline]
```

Make default constructor accessible.

4.204.3 Member Function Documentation

```
4.204.3.1 acquire_mutex() virtual void mha_fifo_thread_platform_t::acquire_mutex ( )
[pure virtual]
```

Calling thread waits until it aquires the lock.

Must not be called when the lock is already aquired.

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

```
4.204.3.2 release_mutex() virtual void mha_fifo_thread_platform_t::release_mutex ( )
[pure virtual]
```

Calling thread releases the lock.

May only be called when lock is owned.

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

4.204.3.3 wait_for_decrease() virtual void mha_fifo_thread_platform_t::wait_for_decrease () [pure virtual]

Calling producer thread must own the lock.

Method releases lock, and waits for consumer thread to call decrease(). Then reacquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

4.204.3.4 wait_for_increase() virtual void mha_fifo_thread_platform_t::wait_for_increase () [pure virtual]

Calling consumer thread must own the lock.

Method releases lock, and waits for producer thread to call increase(). Then reacquires lock and returns

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

4.204.3.5 increment() virtual void mha_fifo_thread_platform_t::increment () [pure virtual]

To be called by producer thread after producing.

Producer thread needs to own the lock to call this method.

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

4.204.3.6 decrement() virtual void mha_fifo_thread_platform_t::decrement () [pure virtual]

To be called by consumer thread after consuming.

Consumer thread needs to own the lock to call this method.

Implemented in **mha_fifo_posix_threads_t** (p. [774](#)).

```
4.204.3.7 operator=() mha_fifo_thread_platform_t& mha_fifo_thread_platform_t<
::operator= (
    const mha_fifo_thread_platform_t & ) [private]
```

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

- [mha_fifo.h](#)

4.205 mha_real_test_array_t Struct Reference

Public Attributes

- [mha_real_t r \[4\]](#)

4.205.1 Member Data Documentation

4.205.1.1 r mha_real_t mha_real_test_array_t::r[4]

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

- [mha.hh](#)

4.206 mha_rt_fifo_element_t< T > Class Template Reference

Object wrapper for [mha_rt_fifo_t](#) (p. 789).

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.

4.206.1 Detailed Description

```
template<class T>
class mha_rt_fifo_element_t< T >
```

Object wrapper for **mha_rt_fifo_t** (p. 789).

4.206.2 Constructor & Destructor Documentation

```
4.206.2.1 mha_rt_fifo_element_t() template<class T >
mha_rt_fifo_element_t< T >:: mha_rt_fifo_element_t (
    T * data ) [inline]
```

Constructor.

This element assumes ownership of user data.

Parameters

data	User data. Has to be allocated on the heap with standard operator new, because it will be deleted in this element's destructor.
-------------	---

```
4.206.2.2 ~mha_rt_fifo_element_t() template<class T >
mha_rt_fifo_element_t< T >::~ mha_rt_fifo_element_t ( ) [inline]
```

4.206.3 Member Data Documentation

```
4.206.3.1 next template<class T >
mha_rt_fifo_element_t<T>* mha_rt_fifo_element_t< T >::next
```

Pointer to next fifo element. NULL for the last (newest) fifo element.

```
4.206.3.2 abandonned template<class T >
bool mha_rt_fifo_element_t< T >::abandonned
```

Indicates that this element will no longer be used and may be deleted.

```
4.206.3.3 data template<class T >
T* mha_rt_fifo_element_t< T >::data
```

Pointer to user data.

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

- **mha_fifo.h**

4.207 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_abandonned ()**
Deletes abandonned elements.
- void **remove_all ()**
Deletes all elements.

Private Attributes

- **mha_rt_fifo_element_t< T > * root**
The first element in the fifo. Deleting elements starts here.
- **mha_rt_fifo_element_t< T > * current**
*The element most recently returned by **poll** (p. 791) or **poll_1** (p. 791).*

4.207.1 Detailed Description

```
template<class T>
class mha_rt_fifo_t< T >
```

Template class for thread safe, half real time safe fifo without explixit locks.

Reading from this fifo is realtime safe, writing to it is not. This fifo is designed for objects that were constructed on the heap. It assumes ownership of these objects and calls delete on them when they are no longer used. Objects remain inside the Fifo while being used by the reader.

A new fifo element is inserted by using **push** (p. 791). The push operation is not real time safe, it allocates and deallocates memory. The latest element is retrieved by calling **poll** (p. 791). This operation will skip fifo elements if more than one **push** (p. 791) has been occured since the last poll. To avoid skipping, call the **poll_1** (p. 791) operation instead.

4.207.2 Constructor & Destructor Documentation

4.207.2.1 mha_rt_fifo_t() template<class T >
`mha_rt_fifo_t< T >:: mha_rt_fifo_t () [inline]`

Construct empty fifo.

4.207.2.2 ~mha_rt_fifo_t() template<class T >
T* mha_rt_fifo_t< T >::~ mha_rt_fifo_t () [inline]

Destructor will delete all data currently in the fifo.

4.207.3 Member Function Documentation

4.207.3.1 poll() template<class T >
T* mha_rt_fifo_t< T >::poll () [inline]

Retrieve the latest element in the Fifo.

Will skip fifo elements if more than one element has been added since last poll invocation. Will return the same element as on last call if no elements have been added in the mean time. Marks former elements as abandonned.

Returns

The latest element in this Fifo. Returns NULL if the Fifo is empty.

4.207.3.2 poll_1() template<class T >
T* mha_rt_fifo_t< T >::poll_1 () [inline]

Retrieve the next element in the Fifo, if there is one, and mark the previous element as abandoned.

Else, if there is no newer element, returns the same element as on last **poll()** (p. 791) or **poll_1()** (p. 791) invocation.

Returns

The next element in this Fifo, if there is one, or the same as before. Returns NULL if the Fifo is empty.

4.207.3.3 push() template<class T >
void mha_rt_fifo_t< T >::push (
 T * data) [inline]

Add element to the Fifo.

Deletes abandonned elements in the fifo.

Parameters

<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. <i>data</i> must have been allocated on the heap with standard operator new.
-------------	---

4.207.3.4 **remove_abandonned()** template<class T >

```
void mha_rt_fifo_t< T >::remove_abandonned ( ) [inline], [private]
```

Deletes abandonned elements.

4.207.3.5 **remove_all()** template<class T >

```
void mha_rt_fifo_t< T >::remove_all ( ) [inline], [private]
```

Deletes all elements.

4.207.4 Member Data Documentation

4.207.4.1 **root** template<class T >

```
mha_rt_fifo_element_t<T>* mha_rt_fifo_t< T >::root [private]
```

The first element in the fifo. Deleting elements starts here.

4.207.4.2 **current** template<class T >

```
mha_rt_fifo_element_t<T>* mha_rt_fifo_t< T >::current [private]
```

The element most recently returned by **poll** (p. 791) or **poll_1** (p. 791).

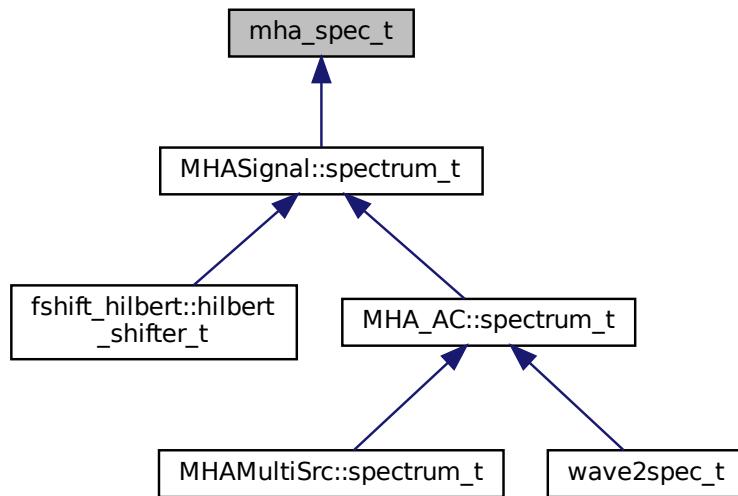
Searching for new elements starts here.

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

- **mha_fifo.h**

4.208 mha_spec_t Struct Reference

Inheritance diagram for mha_spec_t:



Public Attributes

- **mha_complex_t * buf**
signal buffer
- **unsigned int num_channels**
number of channels
- **unsigned int num_frames**
number of frames in each channel
- **mha_channel_info_t * channel_info**
detailed channel description

4.208.1 Detailed Description

```
\ingroup mhasignal
\brief Spectrum signal structure
```

This structure contains the short time fourier transform output of the windowed input signal. The member `num_frames` describes the number of frequency bins in each channel. For an even FFT length N , this is $N/2 + 1$. With odd FFT lengths, it is $(N + 1)/2$. The imaginary part

of the first bin is zero. For even FFT lengths, also the imaginary part at the Nyquist frequency is zero.

buf[k].re	$Re(0)$	$Re(1)$	$Re(2)$	$Re(3)$	$Re(4)$	\cdots	$Re(n/2-1)$	$Re(n/2)$
buf[k].im		$Im(1)$	$Im(2)$	$Im(3)$	$Im(4)$	\cdots	$Im(n/2-1)$	
k	0	1	2	3	4		$n/2-1$	$n/2$

Figure 4 Data order of FFT spectrum.

Only the FFT bins for the positive frequencies, 0, and the Nyquist frequency are stored in this structure. The negative frequencies are not stored, because for a real-valued time signal they are the complex conjugates of the positive frequencies.

The negative frequencies still contribute to the signal's level. Refer to **Central Calibration** (p. 3) for a description of the scaling and how the level would be computed from the spectrum. It is recommended to use the library function **MHASignal::rmslevel** (p. 53) to compute the unweighted level correctly in Pascal, or **MHASignal::colored_intensity** (p. 54) to compute a possibly weighted intensity.

4.208.2 Member Data Documentation

4.208.2.1 **buf** `mha_complex_t* mha_spec_t::buf`

signal buffer

4.208.2.2 **num_channels** `unsigned int mha_spec_t::num_channels`

number of channels

4.208.2.3 **num_frames** `unsigned int mha_spec_t::num_frames`

number of frames in each channel

4.208.2.4 channel_info mha_channel_info_t* mha_spec_t::channel_info

detailed channel description

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

- **mha.hh**

4.209 mha_stash_environment_variable_t Class Reference

This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.

Public Member Functions

- **mha_stash_environment_variable_t** (const std::string & **variable_name**, const std::string & **new_content**)
- **~mha_stash_environment_variable_t** ()

Private Attributes

- const bool **existed_before**
Flag indicates if the environment variable existed before constructor.
- const std::string **variable_name**
Name of environment variable.
- const std::string **original_content**
Content of environment variable before constructor executed.

4.209.1 Detailed Description

This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.

Can be used for testing functionality related to environment variables.

4.209.2 Constructor & Destructor Documentation

4.209.2.1 mha_stash_environment_variable_t() `mha_stash_environment_variable_t::mha_stash_environment_variable_t (const std::string & variable_name, const std::string & new_content) [inline]`

4.209.2.2 ~mha_stash_environment_variable_t() `mha_stash_environment_variable_t::~mha_stash_environment_variable_t () [inline]`

4.209.3 Member Data Documentation

4.209.3.1 existed_before `const bool mha_stash_environment_variable_t::existed_before [private]`

Flag indicates if the environment variable existed before constructor.

4.209.3.2 variable_name `const std::string mha_stash_environment_variable_t::variable_name [private]`

Name of environment variable.

4.209.3.3 original_content `const std::string mha_stash_environment_variable_t::original_content [private]`

Content of environment variable before constructor executed.

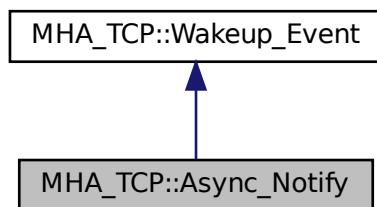
The documentation for this class was generated from the following file:

- **mha_os.h**

4.210 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

4.210.1 Detailed Description

Portable Multiplexable cross-thread notification.

4.210.2 Constructor & Destructor Documentation

4.210.2.1 `Async_Notify()` `Async_Notify::Async_Notify ()`

4.210.2.2 `~Async_Notify()` `Async_Notify::~Async_Notify () [virtual]`

4.210.3 Member Function Documentation

4.210.3.1 `reset()` `void Async_Notify::reset () [virtual]`

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 837).

4.210.3.2 `set()` `void Async_Notify::set () [virtual]`

4.210.4 Member Data Documentation

4.210.4.1 `pipe` `int MHA_TCP::Async_Notify::pipe[2] [private]`

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.211 **mha_tcp::buffered_socket_t Class Reference**

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

Inherits socket, and enable_shared_from_this< buffered_socket_t >.

Public Member Functions

- `asio::streambuf & get_buffer ()`
Access to associated streambuf.
- `void queue_write (const std::string &message)`
Send the given message through this connection to the client asynchronously.

Private Attributes

- `asio::streambuf streambuf`
associated streambuf object to collect received pieces into lines
- `std::string current_message`
The message that is currently sent back to the client.
- `std::string next_message`
A buffer for the next message(s) that must be sent back to the client after the sending of current_message has completed.

4.211.1 Detailed Description

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

Used for communicating with MHA TCP clients. The life time of the connection objects is managed with shared pointers registered together with callbacks in the asio event loop. This is a common idiom in asio. To support this, we inherit from enable_shared_from_this which is also common in code that uses asio.

4.211.2 Member Function Documentation

4.211.2.1 `get_buffer()` `asio::streambuf& mha_tcp::buffered_socket_t::get_buffer ()` [inline]

Access to associated streambuf.

Needed to invoke `async_read`.

Returns

associated streambuf object by reference

4.211.2.2 `queue_write()` `void mha_tcp::buffered_socket_t::queue_write (` `const std::string & message)`

Send the given message through this connection to the client asynchronously.

Parameters

<i>message</i>	The text to send. Method copies the message before returning.
----------------	---

4.211.3 Member Data Documentation

4.211.3.1 `streambuf` `asio::streambuf mha_tcp::buffered_socket_t::streambuf` [private]
 associated streambuf object to collect received pieces into lines

4.211.3.2 `current_message` `std::string mha_tcp::buffered_socket_t::current_message` [private]

The message that is currently sent back to the client.

4.211.3.3 `next_message` `std::string mha_tcp::buffered_socket_t::next_message` [private]

A buffer for the next message(s) that must be sent back to the client after the sending of `current_message` has completed.

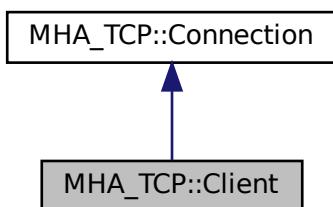
The documentation for this class was generated from the following files:

- `mha_tcp_server.hh`
- `mha_tcp_server.cpp`

4.212 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

4.212.1 Detailed Description

A portable class for a tcp client connections.

4.212.2 Constructor & Destructor Documentation

4.212.2.1 Client() [1/2] Client::Client (

host	The hostname of the TCP Server (p. 814).
port	The port or the TCP Server (p. 814).

Constructor connects to host, port via TCP.

Parameters

host	The hostname of the TCP Server (p. 814).
port	The port or the TCP Server (p. 814).

4.212.2.2 Client() [2/2] Client::Client (

host	
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. 814).
<i>port</i>	The port or the TCP Server (p. 814).
<i>timeout_watcher</i>	an Event watcher that implements a timeout.

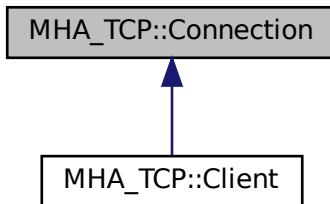
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.213 MHA_TCP::Connection Class Reference

Connection (p. 802) handles Communication between client and server, is used on both sides.

Inheritance diagram for MHA_TCP::Connection:



Public Member Functions

- **Sockread_Event * get_read_event ()**
Get peer's IP Address.
- **Sockwrite_Event * get_write_event ()**
- **std::string get_peer_address ()**
Get peer's IP Address.
- **unsigned short get_peer_port ()**
Get peer's TCP port.
- **SOCKET get_fd () const**
Return the (protected) file descriptor of the connection.
- **virtual ~Connection ()**
Destructor closes the underlying file descriptor.
- **bool eof ()**

- Checks if the peer has closed the connection.
- bool **can_read_line** (char delim='\n')
 - Checks if a full line of text has arrived by now.*
- bool **can_read_bytes** (unsigned howmany)
 - Checks if the specified amount 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 amount of data 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 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**

4.213.1 Detailed Description

Connection (p. 802) handles Communication between client and server, is used on both sides.

4.213.2 Constructor & Destructor Documentation

4.213.2.1 Connection() `MHA_TCP::Connection::Connection (`
`SOCKET _fd) [protected]`

Create a connection instance from a socket filedescriptor.

Parameters

 <code>fd</code>	The file descriptor of the TCP Socket. This file descriptor is closed again in the destructor.
--	--

Exceptions

MHA_Error (p. 763)	If the file descriptor is < 0.
---------------------------	--------------------------------

4.213.2.2 ~Connection() `Connection::~Connection () [virtual]`

Destructor closes the underlying file descriptor.

4.213.3 Member Function Documentation

4.213.3.1 init_peer_data() void MHA_TCP::Connection::init_peer_data () [private]

determine peer address and port

4.213.3.2 can_sysread() bool Connection::can_sysread () [private]

Determine whether at least 1 byte can be read without blocking.

4.213.3.3 can_syswrite() bool Connection::can_syswrite () [private]

Determine whether at least 1 byte can be written without blocking.

4.213.3.4 sysread() std::string Connection::sysread (unsigned bytes) [private]

Call the system's read function and try to read bytes.

This will block in a situation where can_sysread returns false.

Parameters

<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.

4.213.3.5 syswrite() std::string Connection::syswrite (const std::string & data) [private]

Call the system's write function and try to write all characters in the string data.

May write fewer characters, but will at least write one character.

Parameters

data	A string of characters to write to the socket.
-------------	--

Returns

The rest of the characters that have not yet been written.

4.213.3.6 `get_read_event()` `Sockread_Event * Connection::get_read_event ()`**4.213.3.7 `get_write_event()`** `Sockwrite_Event * Connection::get_write_event ()`**4.213.3.8 `get_peer_address()`** `std::string Connection::get_peer_address ()`

Get peer's IP Address.

4.213.3.9 `get_peer_port()` `unsigned short Connection::get_peer_port ()`

Get peer's TCP port.

4.213.3.10 `get_fd()` `SOCKET MHA_TCP::Connection::get_fd () const [inline]`

Return the (protected) file descriptor of the connection.

Will be required for SSL.

4.213.3.11 `eof()` `bool Connection::eof ()`

Checks if the peer has closed the connection.

As a side effect, this method fills the internal "incoming" buffer if it was empty and the socket is readable and not eof.

4.213.3.12 `can_read_line()` `bool Connection::can_read_line (char delim = '\n')`

Checks if a full line of text has arrived by now.

This method reads data from the socket into the internal "incoming" buffer if it can be done without blocking.

Parameters

<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.

4.213.3.13 can_read_bytes() `bool Connection::can_read_bytes (unsigned howmany)`

Checks if the specified amount 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 amount of data is present in the internal buffer after this method call, false otherwise

4.213.3.14 read_line() `std::string Connection::read_line (char delim = '\n')`

Reads a single line of data from the socket.

Blocks if necessary.

Parameters

<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.

4.213.3.15 `read_bytes()` `std::string Connection::read_bytes (`
`unsigned howmany)`

Reads the specified amount of data from the socket.

Blocks if necessary.

Parameters

<code>howmany</code>	The number of bytes to read.
----------------------	------------------------------

Returns

The string of characters read. The string may be shorter if EOF is encountered.

4.213.3.16 `try_write()` `void Connection::try_write (`
`const std::string & data = "")`

Adds data to the internal "outgoing" buffer, and then tries to write as much data from that buffer to the socket as possible without blocking.

Parameters

<code>data</code>	data to send over the socket.
-------------------	-------------------------------

4.213.3.17 `write()` `void Connection::write (`
`const std::string & data = "")`

Adds data to the internal "outgoing" buffer, and then writes that buffer to the socket, regardless of blocking.

Parameters

<i>data</i>	data to send over the socket.
-------------	-------------------------------

4.213.3.18 needs_write() bool Connection::needs_write ()

Checks if the internal "outgoing" buffer contains data.

4.213.3.19 buffered_incoming_bytes() unsigned Connection::buffered_incoming_bytes () const

Returns the number of bytes in the internal "incoming" buffer.

4.213.3.20 buffered_outgoing_bytes() unsigned Connection::buffered_outgoing_bytes () const

Returns the number of bytes in the internal "outgoing" buffer.

4.213.4 Member Data Documentation**4.213.4.1 outbuf** std::string MHA_TCP::Connection::outbuf [private]**4.213.4.2 inbuf** std::string MHA_TCP::Connection::inbuf [private]**4.213.4.3 read_event** Sockread_Event* MHA_TCP::Connection::read_event [private]

4.213.4.4 write_event `Sockwrite_Event*` `MHA_TCP::Connection::write_event` [private]

4.213.4.5 closed `bool` `MHA_TCP::Connection::closed` [private]

4.213.4.6 peer_addr `struct sockaddr_in` `MHA_TCP::Connection::peer_addr` [private]

4.213.4.7 fd `SOCKET` `MHA_TCP::Connection::fd` [protected]

The file descriptor of the TCP Socket.

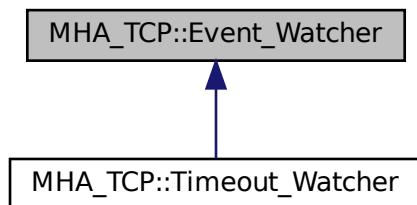
The documentation for this class was generated from the following files:

- `mha_tcp.hh`
- `mha_tcp.cpp`

4.214 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.

4.214.1 Detailed Description

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

4.214.2 Member Typedef Documentation

4.214.2.1 Events `typedef std::set< Wakeup_Event*> MHA_TCP::Event_Watcher::Events`

4.214.2.2 iterator `typedef std::set< Wakeup_Event*>:: iterator MHA_TCP::Event_Watcher::iterator`

4.214.3 Constructor & Destructor Documentation

4.214.3.1 ~Event_Watcher() Event_Watcher::~Event_Watcher () [virtual]

4.214.4 Member Function Documentation

4.214.4.1 observe() void Event_Watcher::observe (
 Wakeup_Event * event)

Add an event to this observer.

4.214.4.2 ignore() void Event_Watcher::ignore (
 Wakeup_Event * event)

Remove an event from this observer.

4.214.4.3 wait() std::set< Wakeup_Event * > Event_Watcher::wait ()

\ Wait for some event to occur.

Return all events that are ready

4.214.5 Member Data Documentation

4.214.5.1 events std::set< **Wakeup_Event***> MHA_TCP::Event_Watcher::events [private]

The list of events to watch.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.215 MHA_TCP::OS_EVENT_TYPE Struct Reference

Public Types

- enum { **R** =0, **W** =1, **X** =2, **T** }

Public Attributes

- enum MHA_TCP::OS_EVENT_TYPE:: { ... } **mode**
- union {
 - int **fd**
 - double **timeout**};

4.215.1 Member Enumeration Documentation

4.215.1.1 anonymous enum anonymous enum

Enumerator

R	
W	
X	
T	

4.215.2 Member Data Documentation

4.215.2.1 mode enum { ... } MHA_TCP::OS_EVENT_TYPE::mode

4.215.2.2 fd int MHA_TCP::OS_EVENT_TYPE::fd

4.215.2.3 timeout double MHA_TCP::OS_EVENT_TYPE::timeout

4.215.2.4 "@5 union { ... }

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

- **mha_tcp.hh**

4.216 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. 810).*
- **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**

4.216.1 Constructor & Destructor Documentation

4.216.1.1 Server() [1/2] Server::Server (

```
    unsigned short port = 0,
    const std::string & iface = "0.0.0.0" )
```

Create a TCP server socket.

Parameters

<i>port</i>	The TCP port to listen to.
<i>iface</i>	The network interface to bind to.

4.216.1.2 Server() [2/2] 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.

4.216.1.3 ~Server() `Server::~Server ()`

Close the TCP server socket.

4.216.2 Member Function Documentation**4.216.2.1 initialize()** `void Server::initialize (const std::string & iface, unsigned short port) [private]`**4.216.2.2 get_interface()** `std::string Server::get_interface () const`

Get the name given in the constructor for the network interface.

4.216.2.3 get_port() `unsigned short Server::get_port () const`

Get the port that the TCP server socket currently listens to.

4.216.2.4 get_accept_event() `Sockaccept_Event * Server::get_accept_event ()`

Produces an event that can be observed by an **Event_Watcher** (p. 810).

This event signals incoming connections that can be accepted.

4.216.2.5 accept() `Connection * Server::accept ()`

Accept an incoming connection.

blocks if necessary.

Returns

The new TCP connection. The connection has to be deleted by the caller.

4.216.2.6 try_accept() `Connection * Server::try_accept ()`

Accept an incoming connection if it can be done without blocking.

Returns

The new TCP connection or 0 if there is no immediate connection. The connection has to be deleted by the caller.

4.216.3 Member Data Documentation**4.216.3.1 sock_addr** `sockaddr_in MHA_TCP::Server::sock_addr [private]`**4.216.3.2 serversocket** `SOCKET MHA_TCP::Server::serversocket [private]`**4.216.3.3 iface** `std::string MHA_TCP::Server::iface [private]`**4.216.3.4 port** `unsigned short MHA_TCP::Server::port [private]`**4.216.3.5 accept_event** `Sockaccept_Event* MHA_TCP::Server::accept_event [private]`

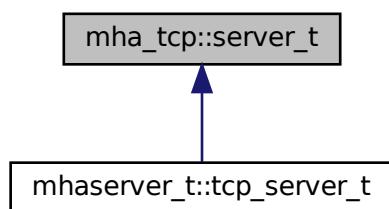
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.217 mha_tcp::server_t Class Reference

Class for accepting TCP connections from clients.

Inheritance diagram for mha_tcp::server_t:



Public Member Functions

- **server_t** (const std::string &interface, uint16_t port)

Allocates a TCP server.
- uint16_t **get_port** () const
- asio::ip::tcp::endpoint **get_endpoint** () const
- asio::ip::address **get_address** () const
- size_t **get_num_accepted_connections** () const
- void **run** ()

Accepts connections on the TCP port and serves them.
- virtual bool **on_received_line** (std::shared_ptr< buffered_socket_t > c, const std::string &l)

This method is invoked when a line of text is received on one of the accepted connections.
- virtual void **shutdown** ()

Shuts down the server: Close the acceptor (no new connections), shuts down the receiving direction of all accepted connections (no new commands, but responses can still be finished), registers a timer event that will cause event loop termination and return from the run() (p. 821) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.
- virtual ~**server_t** ()=default

Make destructor virtual.
- asio::io_context & **get_context** ()

Private Member Functions

- void **trigger_accept ()**
Triggers the acceptance of the next connection.
- void **post_trigger_read_line (std::shared_ptr< buffered_socket_t > c)**
Call trigger_read_line with a single detour through asio's event loop to avoid starving other connections when one connection floods us.
- void **trigger_read_line (std::shared_ptr< buffered_socket_t > c)**
Triggers the reading of the next line from a connection.
- void **add_connection (std::shared_ptr< buffered_socket_t > connection)**
Add new connection to the list of connections, retire stale pointers.

Private Attributes

- asio::io_context **io_context**
The io context used to run event loops.
- std::shared_ptr<asio::ip::tcp::acceptor> **acceptor**
The underlying asio object used to accept incoming TCP connections.
- bool **async_accept_has_been_triggered** = false
Set to true when async_acceptance is triggered in trigger_accept (Only one accept can be in process at any time).
- size_t **num_accepted_connections** = 0U
Number of accepted connections (not necessarily still existing)
- std::vector<std::weak_ptr< buffered_socket_t >> **connections**
Weak pointers to the existing connections.

4.217.1 Detailed Description

Class for accepting TCP connections from clients.

4.217.2 Constructor & Destructor Documentation

4.217.2.1 **server_t()** mha_tcp::server_t::server_t (

```
const std::string & interface,
uint16_t port )
```

Allocates a TCP server.

Parameters

<i>interface</i>	Host name of the network interface to listen on. Can be "localhost" or "127.0.0.1" for localhost, "0.0.0.0" for any ipv4 interface, ...
<i>port</i>	TCP port to open for incoming connections. If <i>port</i> ==0, then the operating system will select a free port.

Exceptions

<i>system_error</i>	if the name given in <i>interface</i> cannot be resolved
<i>system_error</i>	if we cannot bind to the requested interface

4.217.2.2 ~server_t() virtual mha_tcp::server_t::~server_t () [virtual], [default]

Make destructor virtual.

4.217.3 Member Function Documentation

4.217.3.1 get_port() uint16_t mha_tcp::server_t::get_port () const

Returns

The port number of the TCP port that has been opened. If the port specified in the constructor was 0, this will return the port that the operating system has selected.

4.217.3.2 get_endpoint() asio::ip::tcp::endpoint mha_tcp::server_t::get_endpoint () const

Returns

The local endpoint of the acceptor.

4.217.3.3 get_address() asio::ip::address mha_tcp::server_t::get_address () const**Returns**

The ip address that the server is bound to.

4.217.3.4 get_num_accepted_connections() size_t mha_tcp::server_t::get_num_accepted_connections () const**Returns**

the number of TCP connections that have been accepted

4.217.3.5 run() void mha_tcp::server_t::run ()

Accepts connections on the TCP port and serves them.

Triggers the acceptance of the next connection to start things off.

4.217.3.6 on_received_line() bool mha_tcp::server_t::on_received_line (std::shared_ptr< buffered_socket_t > c, const std::string & l) [virtual]

This method is invoked when a line of text is received on one of the accepted connections.

Override this method to process the communication with the client.

Parameters

c	the connection that has received this line
l	the line that has been received, without the line ending

Returns

client should return true when client wants to read another line of text, else false.

Reimplemented in **mhaserver_t::tcp_server_t** (p. 1194).

4.217.3.7 shutdown() `void mha_tcp::server_t::shutdown () [virtual]`

Shuts down the server: Close the acceptor (no new connections), shuts down the receiving direction of all accepted connections (no new commands, but responses can still be finished), registers a timer event that will cause event loop termination and return from the **run()** (p. 821) method 1 second in the future, giving us reasonably enough time for the pending responses to be sent out.

4.217.3.8 get_context() `asio::io_context & mha_tcp::server_t::get_context ()`

Returns

the asio io context used to run the event loop

4.217.3.9 trigger_accept() `void mha_tcp::server_t::trigger_accept () [private]`

Triggers the acceptance of the next connection.

Called from run to accept the first connection and from the accept handler to accept each next connection. Once a connection from a client is accepted, the accept handler will register it with the event loop for receiving a line of text.

4.217.3.10 post_trigger_read_line() `void mha_tcp::server_t::post_trigger_read_line (std::shared_ptr< buffered_socket_t > c) [private]`

Call trigger_read_line with a single detour through asio's event loop to avoid starving other connections when one connection floods us.

4.217.3.11 trigger_read_line() `void mha_tcp::server_t::trigger_read_line (std::shared_ptr< buffered_socket_t > c) [private]`

Triggers the reading of the next line from a connection.

Parameters

<code>c</code>	The connection where the incoming data is expected from.
----------------	--

4.217.3.12 add_connection() `void mha_tcp::server_t::add_connection (std::shared_ptr< buffered_socket_t > connection) [inline], [private]`

Add new connection to the list of connections, retire stale pointers.

4.217.4 Member Data Documentation

4.217.4.1 io_context `asio::io_context mha_tcp::server_t::io_context [private]`

The io context used to run event loops.

4.217.4.2 acceptor `std::shared_ptr<asio::ip::tcp::acceptor> mha_tcp::server_t::acceptor [private]`

The underlying asio object used to accept incoming TCP connections.

4.217.4.3 async_accept_has_been_triggered `bool mha_tcp::server_t::async_accept_has_been_triggered = false [private]`

Set to true when async_acceptance is triggered in trigger_accept (Only one accept can be in process at any time).

4.217.4.4 num_accepted_connections `size_t mha_tcp::server_t::num_accepted_connections = 0U [private]`

Number of accepted connections (not necessarily still existing)

4.217.4.5 connections `std::vector<std::weak_ptr< buffered_socket_t > > mha_tcp->::server_t::connections [private]`

Weak pointers to the existing connections.

Needed to shutdown the active connections for incoming data when server shuts down.

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

- `mha_tcp_server.hh`
- `mha_tcp_server.cpp`

4.218 MHA_TCP::sock_init_t Class Reference

Public Member Functions

- `sock_init_t ()`

4.218.1 Constructor & Destructor Documentation

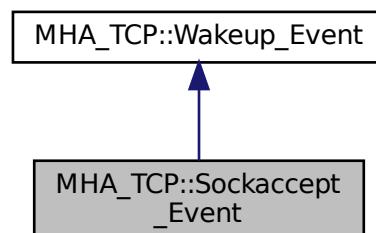
4.218.1.1 `sock_init_t()` `MHA_TCP::sock_init_t::sock_init_t () [inline]`

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

- `mha_tcp.cpp`

4.219 MHA_TCP::Sockaccept_Event Class Reference

Inheritance diagram for MHA_TCP::Sockaccept_Event:



Public Member Functions

- **Sockaccept_Event (SOCKET)**

Additional Inherited Members

4.219.1 Constructor & Destructor Documentation

4.219.1.1 Sockaccept_Event() `MHA_TCP::Sockaccept_Event::Sockaccept_Event (SOCKET s)`

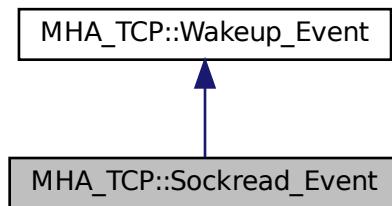
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.220 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

4.220.1 Detailed Description

Watch socket for incoming data.

4.220.2 Constructor & Destructor Documentation

4.220.2.1 **Sockread_Event()** `MHA_TCP::Sockread_Event::Sockread_Event (SOCKET s)`

Set socket to watch for.

Parameters

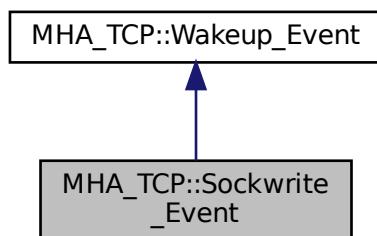
<code>s</code>	The socket to observe incoming data on.
----------------	---

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

- `mha_tcp.hh`
- `mha_tcp.cpp`

4.221 MHA_TCP::Sockwrite_Event Class Reference

Inheritance diagram for MHA_TCP::Sockwrite_Event:



Public Member Functions

- **Sockwrite_Event (SOCKET s)**

Additional Inherited Members

4.221.1 Constructor & Destructor Documentation

4.221.1.1 Sockwrite_Event() `MHA_TCP::Sockwrite_Event::Sockwrite_Event (SOCKET s)`

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.222 MHA_TCP::Thread Class Reference

A very simple class for portable threads.

Public Types

- enum { **PREPARED**, **RUNNING**, **FINISHED** }
The current state of the thread.
- typedef void `*(* thr_f) (void *)`
The thread function signature to use with this class.

Public Member Functions

- **Thread (thr_f func, void * arg=0)**
Constructor starts a new thread.
- virtual **~Thread ()**
*The destructor should only be called when the **Thread** (p. 827) is finished.*
- virtual void **run ()**
*The internal method that delegated the new thread to the registered **Thread** (p. 827) 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. 763) 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.

4.222.1 Detailed Description

A very simple class for portable threads.

4.222.2 Member Typedef Documentation

4.222.2.1 thr_f `typedef void*(* MHA_TCP::Thread::thr_f) (void *)`

The thread function signature to use with this class.

Derive from this class and call protected standard constructor to start threads differently.

4.222.3 Member Enumeration Documentation**4.222.3.1 anonymous enum** `anonymous enum`

The current state of the thread.

Enumerator

PREPARED	
RUNNING	
FINISHED	

4.222.4 Constructor & Destructor Documentation**4.222.4.1 Thread() [1/2]** `MHA_TCP::Thread::Thread () [protected]`

Default constructor may only be used by derived classes that want to start the thread themselves.

4.222.4.2 Thread() [2/2] `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.

4.222.4.3 ~Thread() Thread::~Thread () [virtual]

The destructor should only be called when the **Thread** (p. 827) is finished.

There is preliminary support for forceful thread cancellation in the destructor, but probably not very robust or portable..

4.222.5 Member Function Documentation

4.222.5.1 run() void Thread::run () [virtual]

The internal method that delegated the new thread to the registered **Thread** (p. 827) function.

4.222.6 Member Data Documentation

4.222.6.1 thread_handle pthread_t MHA_TCP::Thread::thread_handle [private]

The posix thread handle.

4.222.6.2 thread_attr pthread_attr_t MHA_TCP::Thread::thread_attr [private]

The posix thread attribute structure.

Required for starting a thread in detached state. Detachment is required to eliminate the need for joining this thread.

4.222.6.3 arg void* MHA_TCP::Thread::arg [protected]

The argument for the client's thread function.

4.222.6.4 return_value void* MHA_TCP::Thread::return_value [protected]

The return value from the client's thread function is stored here When that function returns.

4.222.6.5 thread_finish_event **Async_Notify** MHA_TCP::Thread::thread_finish_event

Event will be triggered when the thread exits.

4.222.6.6 state enum { ... } MHA_TCP::Thread::state

The current state of the thread.

4.222.6.7 thread_func **thr_f** MHA_TCP::Thread::thread_func

The thread function that the client has registered.

4.222.6.8 thread_arg void* MHA_TCP::Thread::thread_arg

The argument that the client wants to be handed through to the thread function.

4.222.6.9 error **MHA_Error*** MHA_TCP::Thread::error

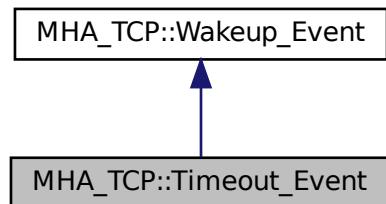
The **MHA_Error** (p. 763) that caused the thread to abort, if any.

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

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.223 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

4.223.1 Constructor & Destructor Documentation

4.223.1.1 Timeout_Event() `Timeout_Event::Timeout_Event (`
`double interval)`

4.223.2 Member Function Documentation

4.223.2.1 get_os_event() `os_EVENT_TYPE Timeout_Event::get_os_event () [virtual]`

Reimplemented from **MHA_TCP::Wakeup_Event** (p. 837).

4.223.3 Member Data Documentation**4.223.3.1 end_time** `double MHA_TCP::Timeout_Event::end_time [private]`

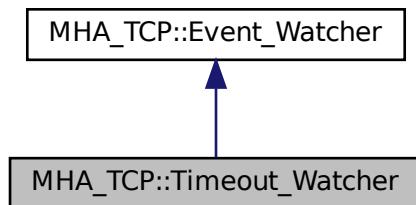
The documentation for this class was generated from the following files:

- **mha_tcp.hh**
- **mha_tcp.cpp**

4.224 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

4.224.1 Detailed Description

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

4.224.2 Constructor & Destructor Documentation

4.224.2.1 `Timeout_Watcher()` `Timeout_Watcher::Timeout_Watcher (double interval) [explicit]`

4.224.2.2 `~Timeout_Watcher()` `Timeout_Watcher::~Timeout_Watcher () [virtual]`

4.224.3 Member Data Documentation

4.224.3.1 `timeout` `Timeout_Event MHA_TCP::Timeout_Watcher::timeout [private]`

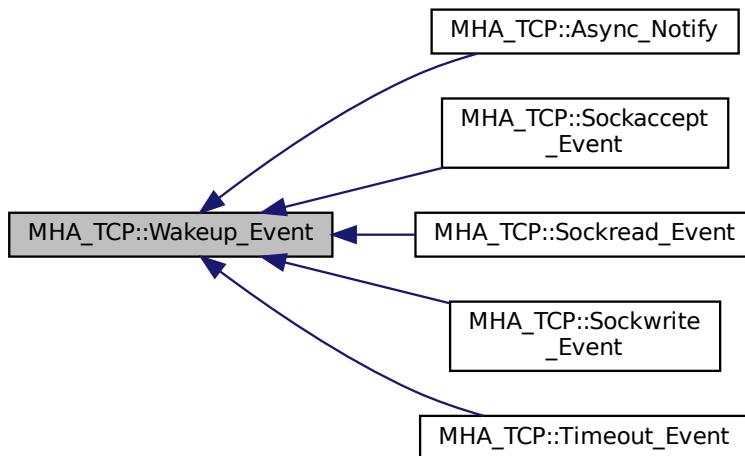
The documentation for this class was generated from the following files:

- `mha_tcp.hh`
- `mha_tcp.cpp`

4.225 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. 810) when this event is added to its list of observed events.*
- virtual void **ignored_by (Event_Watcher *observer)**
*Called by the **Event_Watcher** (p. 810) 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 whether 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. 810) instances that this **Wakeup_Event** (p. 835) is observed by (stored here for proper deregistering).*

4.225.1 Detailed Description

A base class for asynchronous wakeup events.

4.225.2 Constructor & Destructor Documentation

4.225.2.1 **Wakeup_Event()** `Wakeup_Event::Wakeup_Event ()`

Event Constructor.

The new event has invalid state.

4.225.2.2 **~Wakeup_Event()** `Wakeup_Event::~Wakeup_Event () [virtual]`

Destructor deregisters from observers.

4.225.3 Member Function Documentation

4.225.3.1 **observed_by()** `void Wakeup_Event::observed_by (` `Event_Watcher * observer) [virtual]`

Called by the **Event_Watcher** (p. 810) when this event is added to its list of observed events.

4.225.3.2 ignored_by() void Wakeup_Event::ignored_by (Event_Watcher * observer) [virtual]

Called by the **Event_Watcher** (p. 810) when this event is removed from its list of observed events.

4.225.3.3 get_os_event() os_EVENT_TYPE Wakeup_Event::get_os_event () [virtual]

Get necessary information for the Event Watcher.

Reimplemented in **MHA_TCP::Timeout_Event** (p. 832).

4.225.3.4 reset() void Wakeup_Event::reset () [virtual]

For pure notification events, reset the "signalled" status.

Reimplemented in **MHA_TCP::Async_Notify** (p. 798).

4.225.3.5 status() bool Wakeup_Event::status () [virtual]

Query whether the event is in signalled state now.

4.225.4 Member Data Documentation

4.225.4.1 observers std::set<class Event_Watcher *> MHA_TCP::Wakeup_Event::observers [private]

A list of all **Event_Watcher** (p. 810) instances that this **Wakeup_Event** (p. 835) is observed by (stored here for proper deregistering).

4.225.4.2 os_event `os_EVENT_TYPE MHA_TCP::Wakeup_Event::os_event` [protected]

4.225.4.3 os_event_valid `bool MHA_TCP::Wakeup_Event::os_event_valid` [protected]

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

- `mha_tcp.hh`
- `mha_tcp.cpp`

4.226 mha_tictoc_t Struct Reference

Public Attributes

- `struct timeval tv1`
- `struct timeval tv2`
- `struct timezone tz`
- `float t`

4.226.1 Member Data Documentation

4.226.1.1 tv1 `struct timeval mha_tictoc_t::tv1`

4.226.1.2 tv2 `struct timeval mha_tictoc_t::tv2`

4.226.1.3 tz `struct timezone mha_tictoc_t::tz`

4.226.1.4 t float mha_tictoc_t::t

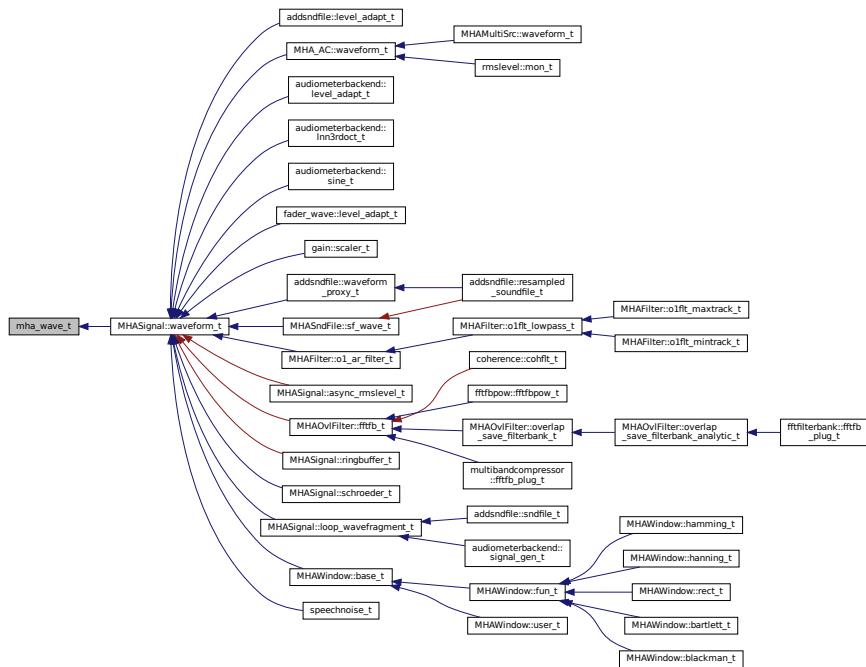
The documentation for this struct was generated from the following file:

- **mha_profiling.h**

4.227 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

4.227.1 Detailed Description

Waveform signal structure.

This structure contains one fragment of a waveform signal. The member num_frames describes the number of audio samples in each audio channel.

In a calibrated openMHA, audio samples are stored in unit Pascal, see **Central Calibration** (p. 3).

The field channel_info must be an array of num_channels entries or NULL.

4.227.2 Member Data Documentation

4.227.2.1 buf `mha_real_t*` `mha_wave_t::buf`

signal buffer

4.227.2.2 num_channels `unsigned int mha_wave_t::num_channels`

number of channels

4.227.2.3 num_frames `unsigned int mha_wave_t::num_frames`

number of frames in each channel

4.227.2.4 channel_info `mha_channel_info_t*` `mha_wave_t::channel_info`

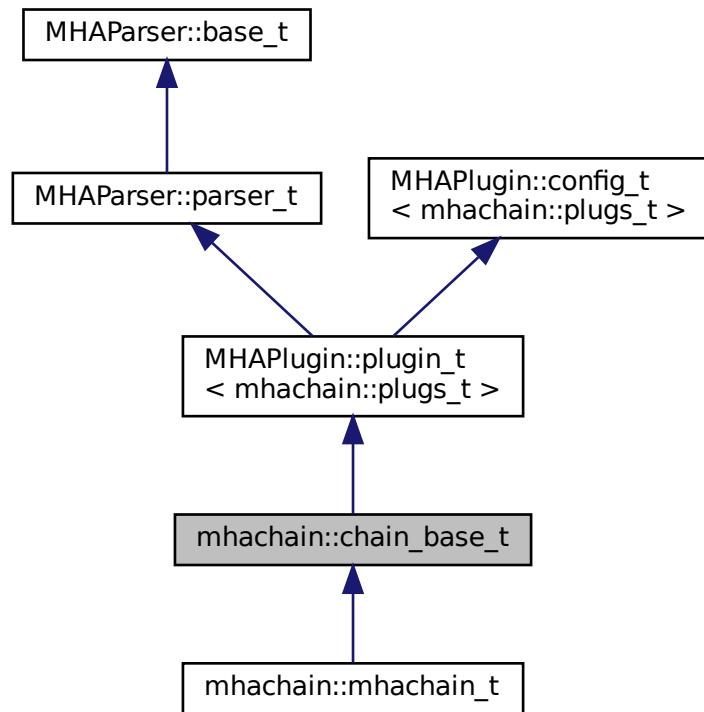
detailed channel description

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

- `mha.hh`

4.228 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

4.228.1 Constructor & Destructor Documentation

4.228.1.1 chain_base_t() mhachain::chain_base_t::chain_base_t (algo_comm_t *iac*, const std::string & *ichain*, const std::string & *ialgo*)

4.228.2 Member Function Documentation

4.228.2.1 process() [1/4] void mhachain::chain_base_t::process (mha_wave_t * *sin*, mha_wave_t ** *sout*)

4.228.2.2 process() [2/4] void mhachain::chain_base_t::process (mha_spec_t * *sin*, mha_wave_t ** *sout*)

4.228.2.3 process() [3/4] void mhachain::chain_base_t::process (

```
mha_wave_t * sin,
mha_spec_t ** sout )
```

4.228.2.4 process() [4/4] void mhachain::chain_base_t::process (

```
mha_spec_t * sin,
mha_spec_t ** sout )
```

4.228.2.5 prepare() void mhachain::chain_base_t::prepare (

```
mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugIn::plugin_t< mhachain::plugs_t >** (p. 1148).

4.228.2.6 release() void mhachain::chain_base_t::release () [virtual]

Reimplemented from **MHAPlugIn::plugin_t< mhachain::plugs_t >** (p. 1149).

4.228.2.7 update() void mhachain::chain_base_t::update () [private]

4.228.3 Member Data Documentation

4.228.3.1 bprofiling MHAParser::bool_t mhachain::chain_base_t::bprofiling [protected]

4.228.3.2 algos MHAParser::vstring_t mhachain::chain_base_t::algos [protected]

4.228.3.3 old_algos std::vector<std::string> mhachain::chain_base_t::old_algos [private]

4.228.3.4 patchbay MHAEvents::patchbay_t< mhachain::chain_base_t > mhachain< ::chain_base_t::patchbay [private]

4.228.3.5 cfin mhaconfig_t mhachain::chain_base_t::cfin [private]

4.228.3.6 cfout mhaconfig_t mhachain::chain_base_t::cfout [private]

4.228.3.7 b_prepared bool mhachain::chain_base_t::b_prepared [private]

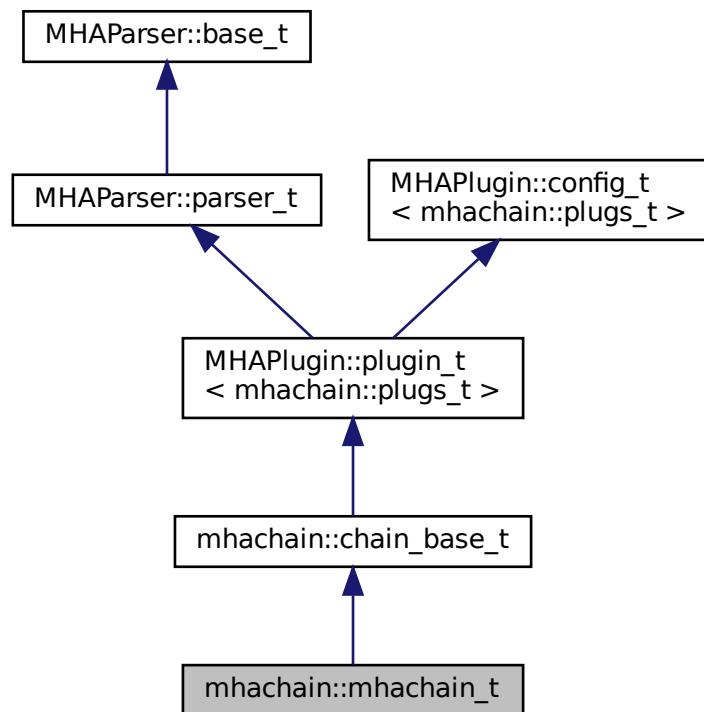
4.228.3.8 chain std::string mhachain::chain_base_t::chain [private]

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

- [mha_generic_chain.h](#)
- [mha_generic_chain.cpp](#)

4.229 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

4.229.1 Constructor & Destructor Documentation

4.229.1.1 `mhachain_t()` `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`

4.230 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

4.230.1 Constructor & Destructor Documentation

```
4.230.1.1 plugs_t() mhachain::plugs_t::plugs_t (
    std::vector< std::string > algos,
    mhaconfig_t cfin,
    mhaconfig_t cfout,
    bool do_prepare,
    MHAParser::parser_t & p,
    algo_comm_t iac,
    std::string ichain,
    bool use_profiling )
```

4.230.1.2 ~plugs_t() mhachain::plugs_t::~plugs_t ()

4.230.2 Member Function Documentation

```
4.230.2.1 prepare() void mhachain::plugs_t::prepare (
    mhaconfig_t & tf )
```

4.230.2.2 release() void mhachain::plugs_t::release ()

```
4.230.2.3 process() void mhachain::plugs_t::process (
    mha_wave_t * win,
    mha_spec_t * sin,
    mha_wave_t ** wout,
    mha_spec_t ** sout )
```

4.230.2.4 prepared() bool mhachain::plugs_t::prepared () const [inline]

4.230.2.5 alloc_plugs() void mhachain::plugs_t::alloc_plugs (std::vector< std::string > algos) [private]

4.230.2.6 cleanup_plugs() void mhachain::plugs_t::cleanup_plugs () [private]

4.230.2.7 update_proc_load() void mhachain::plugs_t::update_proc_load () [private]

4.230.3 Member Data Documentation

4.230.3.1 b_prepared bool mhachain::plugs_t::b_prepared [private]

4.230.3.2 algos std::vector< **PluginLoader::mhapluginloader_t*** > mhachain::plugs_t::algos [private]

4.230.3.3 parser **MHAParser::parser_t&** mhachain::plugs_t::parser [private]

4.230.3.4 ac **algo_comm_t** mhachain::plugs_t::ac [private]

4.230.3.5 chain std::string mhachain::plugs_t::chain [private]

4.230.3.6 profiling `MHAParser::parser_t` `mhachain::plugs_t::profiling` [private]

4.230.3.7 prof_algos `MHAParser::vstring_mon_t` `mhachain::plugs_t::prof_algos` [private]

4.230.3.8 prof_init `MHAParser::vfloat_mon_t` `mhachain::plugs_t::prof_init` [private]

4.230.3.9 prof_prepare `MHAParser::vfloat_mon_t` `mhachain::plugs_t::prof_prepare` [private]

4.230.3.10 prof_release `MHAParser::vfloat_mon_t` `mhachain::plugs_t::prof_release` [private]

4.230.3.11 prof_process `MHAParser::vfloat_mon_t` `mhachain::plugs_t::prof_process` [private]

4.230.3.12 prof_process_tt `MHAParser::float_mon_t` `mhachain::plugs_t::prof_process←_tt` [private]

4.230.3.13 prof_process_load `MHAParser::vfloat_mon_t` `mhachain::plugs_t::prof←_process_load` [private]

4.230.3.14 proc_cnt `unsigned int mhachain::plugs_t::proc_cnt [private]`

4.230.3.15 prof_cfg `mhaconfig_t mhachain::plugs_t::prof_cfg [private]`

4.230.3.16 prof_load_con `MHAEvents::connector_t< mhachain::plugs_t> mhachain<::plugs_t::prof_load_con [private]`

4.230.3.17 prof_tt_con `MHAEvents::connector_t< mhachain::plugs_t> mhachain::plugs_t::prof_tt_con [private]`

4.230.3.18 b_use_profiling `bool mhachain::plugs_t::b_use_profiling [private]`

4.230.3.19 tictoc `mha_platform_tictoc_t mhachain::plugs_t::tictoc [private]`

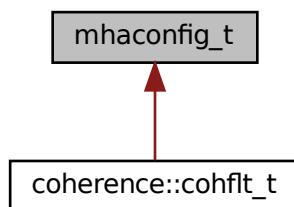
The documentation for this class was generated from the following files:

- `mha_generic_chain.h`
- `mha_generic_chain.cpp`

4.231 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 **ffflen**
FFT length of spectral data.
- **mha_real_t srate**
Sampling rate in Hz.

4.231.1 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).

4.231.2 Member Data Documentation

4.231.2.1 channels unsigned int mhaconfig_t::channels

Number of audio channels.

4.231.2.2 domain unsigned int mhaconfig_t::domain

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

4.231.2.3 fragsize `unsigned int mhaconfig_t::fragsize`

Fragment size of waveform data.

4.231.2.4 wndlen `unsigned int mhaconfig_t::wndlen`

Window length of spectral data.

4.231.2.5 fftlen `unsigned int mhaconfig_t::fftlens`

FFT length of spectral data.

4.231.2.6 srate `mha_real_t mhaconfig_t::srate`

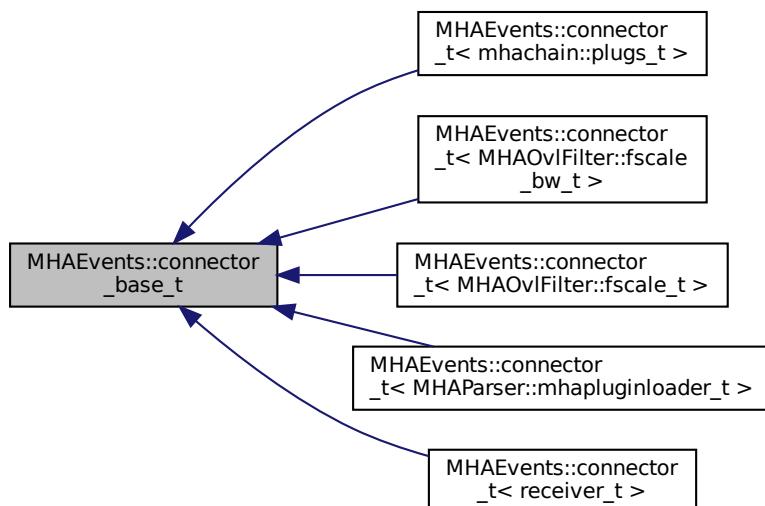
Sampling rate in Hz.

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

- **mha.hh**

4.232 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`

4.232.1 Constructor & Destructor Documentation

4.232.1.1 `connector_base_t()` MHAEvents::connector_base_t::connector_base_t ()

4.232.1.2 `~connector_base_t()` MHAEvents::connector_base_t::~connector_base_t () [virtual]

4.232.2 Member Function Documentation

4.232.2.1 `emit_event()` [1/3] `void MHAEvents::connector_base_t::emit_event () [virtual]`

Reimplemented in `MHAEvents::connector_t< receiver_t >` (p. 856), `MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >` (p. 856), `MHAEvents::connector_t< MHAParser::mhapluginloader_t >` (p. 856), `MHAEvents::connector_t< mhachain::plugs_t >` (p. 856), and `MHAEvents::connector_t< MHAOvlFilter::fscale_t >` (p. 856).

4.232.2.2 emit_event() [2/3] void MHAEvents::connector_base_t::emit_event (const std::string &) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 857), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 857), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 857), **MHAEvents::connector_t< mhachain::plugs_t >** (p. 857), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 857).

4.232.2.3 emit_event() [3/3] void MHAEvents::connector_base_t::emit_event (const std::string & , unsigned int , unsigned int) [virtual]

Reimplemented in **MHAEvents::connector_t< receiver_t >** (p. 857), **MHAEvents::connector_t< MHAOvlFilter::fscale_bw_t >** (p. 857), **MHAEvents::connector_t< MHAParser::mhapluginloader_t >** (p. 857), **MHAEvents::connector_t< mhachain::plugs_t >** (p. 857), and **MHAEvents::connector_t< MHAOvlFilter::fscale_t >** (p. 857).

4.232.2.4 emitter_die() void MHAEvents::connector_base_t::emitter_die ()

4.232.3 Member Data Documentation

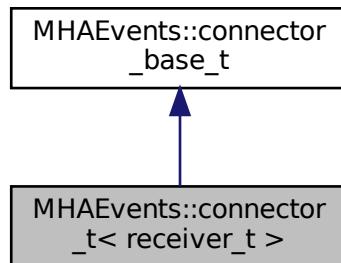
4.232.3.1 emitter_is_alive bool MHAEvents::connector_base_t::emitter_is_alive [protected]

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

- **mha_event_emitter.h**
- **mha_events.cpp**

4.233 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

4.233.1 Constructor & Destructor Documentation

4.233.1.1 **connector_t()** [1/3] template<class receiver_t >

```
MHAEEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(rfunc )
```

4.233.1.2 **connector_t()** [2/3] template<class receiver_t >

```
MHAEEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(const std::string &) rfunc )
```

4.233.1.3 **connector_t()** [3/3] template<class receiver_t >

```
MHAEEvents::connector_t< receiver_t >:: connector_t (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*)(const std::string &, unsigned int, unsigned int)
rfun )
```

4.233.1.4 ~**connector_t()** template<class receiver_t >

```
MHAEEvents::connector_t< receiver_t >::~ connector_t
```

4.233.2 Member Function Documentation

```
4.233.2.1 emit_event() [1/3] template<class receiver_t >
void MHAEvents::connector_t< receiver_t >::emit_event [private], [virtual]
```

Reimplemented from **MHAEvents::connector_base_t** (p. 853).

```
4.233.2.2 emit_event() [2/3] template<class receiver_t >
void MHAEvents::connector_t< receiver_t >::emit_event (
    const std::string & arg ) [private], [virtual]
```

Reimplemented from **MHAEvents::connector_base_t** (p. 853).

```
4.233.2.3 emit_event() [3/3] template<class receiver_t >
void MHAEvents::connector_t< receiver_t >::emit_event (
    const std::string & arg,
    unsigned int arg2,
    unsigned int arg3 ) [private], [virtual]
```

Reimplemented from **MHAEvents::connector_base_t** (p. 854).

4.233.3 Member Data Documentation

```
4.233.3.1 emitter template<class receiver_t >
emitter_t* MHAEvents::connector_t< receiver_t >::emitter [private]
```

```
4.233.3.2 receiver template<class receiver_t >
receiver_t* MHAEvents::connector_t< receiver_t >::receiver [private]
```

```
4.233.3.3 eventhandler template<class receiver_t >
void(receiver_t::* MHAEvents::connector_t< receiver_t >::eventhandler) () [private]
```

4.233.3.4 evenhandler_s template<class receiver_t >
void(receiver_t::* **MHAEVENTS::CONNECTOR_T**< receiver_t >::evenhandler_s) (const
std::string &) [private]

4.233.3.5 evenhandler_suu template<class receiver_t >
void(receiver_t::* **MHAEVENTS::CONNECTOR_T**< receiver_t >::evenhandler_suu) (const
std::string &, unsigned int, unsigned int) [private]

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

- **mha_events.h**

4.234 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**

4.234.1 Detailed Description

Class for emitting openMHA events.

Use the template class **MHAEVENTS::patchbay_t** (p. 860) for connecting to an emitter.

4.234.2 Constructor & Destructor Documentation

4.234.2.1 ~emitter_t() MHAEvents::emitter_t::~emitter_t ()

4.234.3 Member Function Documentation

4.234.3.1 operator()() [1/3] void MHAEvents::emitter_t::operator() ()

Emit an event without parameter.

4.234.3.2 operator()() [2/3] void MHAEvents::emitter_t::operator() (const std::string & arg)

Emit an event with string parameter.

4.234.3.3 operator()() [3/3] void MHAEvents::emitter_t::operator() (const std::string & arg, unsigned int arg2, unsigned int arg3)

Emit an event with string parameter and two unsigned int parameters.

4.234.3.4 connect() void MHAEvents::emitter_t::connect (connector_base_t * c)

4.234.3.5 disconnect() void MHAEvents::emitter_t::disconnect (connector_base_t * c)

4.234.4 Member Data Documentation

4.234.4.1 connections `std::list< connector_base_t*> MHAEvents::emitter_t::connections`
 [private]

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

- `mha_event_emitter.h`
- `mha_events.cpp`

4.235 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`

4.235.1 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. 861) function are held 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.

4.235.2 Constructor & Destructor Documentation

4.235.2.1 ~patchbay_t() template<class receiver_t >
MHAEvents::patchbay_t< receiver_t >::~ patchbay_t

4.235.3 Member Function Documentation

4.235.3.1 connect() [1/3] template<class receiver_t >
void **MHAEvents::patchbay_t< receiver_t >::connect** (
 emitter_t * *e*,
 receiver_t * *r*,
 void(receiver_t::*)() *rfun*)

Connect a receiver member function **void (receiver_t::*)()** with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

<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

4.235.3.2 connect() [2/3] template<class receiver_t >
void **MHAEvents::patchbay_t< receiver_t >::connect** (
 emitter_t * *e*,
 receiver_t * *r*,
 void(receiver_t::*)(const std::string &) *rfun*)

Connect a receiver member function **void (receiver_t::*)(const std::string&)** with an event emitter.

Create a connection.

The connection is removed when the patchbay is destructed.

Parameters

<code>e</code>	Pointer to an event emitter
<code>r</code>	Pointer to the receiver
<code>rfunc</code>	Pointer to a member function of the receiver class

4.235.3.3 `connect()` [3/3] template<class receiver_t >

```
void MHAEvents::patchbay_t< receiver_t >::connect (
    emitter_t * e,
    receiver_t * r,
    void(receiver_t::*) (const std::string &, unsigned int, unsigned int)
rfunc )
```

4.235.4 Member Data Documentation

4.235.4.1 `cons` template<class receiver_t >

```
std::list< connector_t<receiver_t>*> MHAEvents::patchbay_t< receiver_t >::cons
[private]
```

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

- `mha_events.h`

4.236 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`

4.236.1 Constructor & Destructor Documentation

4.236.1.1 adapt_filter_param_t() MHAFilter::adapt_filter_param_t::adapt_filter_param_t (

```
mha_real_t imu,
bool ierr_in )
```

4.236.2 Member Data Documentation

4.236.2.1 mu mha_real_t MHAFilter::adapt_filter_param_t::mu

4.236.2.2 err_in bool MHAFilter::adapt_filter_param_t::err_in

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.237 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**

4.237.1 Constructor & Destructor Documentation

4.237.1.1 adapt_filter_state_t() MHAFilter::adapt_filter_state_t::adapt_filter_state_t (int ntaps, int nchannels)

4.237.2 Member Function Documentation

4.237.2.1 filter() void MHAFilter::adapt_filter_state_t::filter (mha_wave_t y, mha_wave_t e, mha_wave_t x, mha_wave_t d, mha_real_t mu, bool err_in)

4.237.3 Member Data Documentation

4.237.3.1 ntaps int MHAFilter::adapt_filter_state_t::ntaps [private]

4.237.3.2 nchannels int MHAFilter::adapt_filter_state_t::nchannels [private]

4.237.3.3 W MHASignal::waveform_t MHAFilter::adapt_filter_state_t::W [private]

4.237.3.4 X `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::X` [private]

4.237.3.5 od `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::od` [private]

4.237.3.6 oy `MHASignal::waveform_t` `MHAFilter::adapt_filter_state_t::oy` [private]

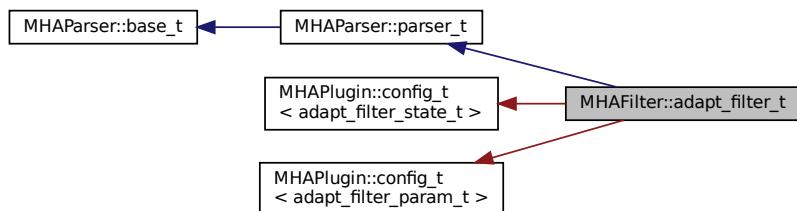
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

4.238 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

4.238.1 Detailed Description

Adaptive filter.

4.238.2 Constructor & Destructor Documentation

4.238.2.1 adapt_filter_t() `MHAFilter::adapt_filter_t::adapt_filter_t (std::string help)`

4.238.3 Member Function Documentation

4.238.3.1 filter() `void MHAFilter::adapt_filter_t::filter (mha_wave_t y, mha_wave_t e, mha_wave_t x, mha_wave_t d)`

4.238.3.2 set_channelcnt() `void MHAFilter::adapt_filter_t::set_channelcnt (unsigned int nch)`

4.238.3.3 update_mu() void MHAFilter::adapt_filter_t::update_mu () [private]

4.238.3.4 update_ntaps() void MHAFilter::adapt_filter_t::update_ntaps () [private]

4.238.4 Member Data Documentation

4.238.4.1 mu MHAParser::float_t MHAFilter::adapt_filter_t::mu [private]

4.238.4.2 ntaps MHAParser::int_t MHAFilter::adapt_filter_t::ntaps [private]

4.238.4.3 err_in MHAParser::bool_t MHAFilter::adapt_filter_t::err_in [private]

4.238.4.4 connector MHAEVENTS::patchbay_t< adapt_filter_t> MHAFilter::adapt_filter_t::connector [private]

4.238.4.5 nchannels unsigned int MHAFilter::adapt_filter_t::nchannels [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.239 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**

4.239.1 Detailed Description

A class that does polyphase resampling and takes into account block processing.

4.239.2 Constructor & Destructor Documentation

```
4.239.2.1 blockprocessing_polyphase_resampling_t() MHAFilter::blockprocessing_<br>
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.

4.239.2.2 ~blockprocessing_polyphase_resampling_t() virtual MHAFilter::blockprocessing<~polyphase_resampling_t>::~blockprocessing_polyphase_resampling_t() [inline], [virtual]

4.239.3 Member Function Documentation

4.239.3.1 write() 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. 763)	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
---------------------------	--

4.239.3.2 `read()` `void MHAFilter::blockprocessing_polyphase_resampling_t::read (mha_wave_t & signal)`

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

<code>signal</code>	buffer to write the resampled signal to.
---------------------	--

Exceptions

<code>MHA_Error</code> (p. 763)	Raises exception if there is not enough input signal, if the number of channels of frames does not match.
---	---

4.239.3.3 `can_read()` `bool MHAFilter::blockprocessing_polyphase_resampling_t::can_read () const [inline]`

Checks if the resampling ring buffer can produce another output signal block.

4.239.4 Member Data Documentation

4.239.4.1 `resampling` `polyphase_resampling_t* MHAFilter::blockprocessing_polyphase_resampling_t::resampling [private]`

4.239.4.2 `fragsize_in` `unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_in [private]`

4.239.4.3 fragsize_out unsigned MHAFilter::blockprocessing_polyphase_resampling_t::fragsize_out [private]

4.239.4.4 num_channels unsigned MHAFilter::blockprocessing_polyphase_resampling_t::num_channels [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.240 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**

4.240.1 Detailed Description

Complex bandpass filter.

4.240.2 Constructor & Destructor Documentation

```
4.240.2.1 complex_bandpass_t() MHAFilter::complex_bandpass_t::complex_bandpass_t
(
    std::vector< mha_complex_t > A,
    std::vector< mha_complex_t > B )
```

Constructor with filter coefficients (one per channel)

Parameters

A	complex filter coefficients, one per band
B	complex weights

4.240.3 Member Function Documentation

```
4.240.3.1 creator_A() std::vector< mha_complex_t > MHAFilter::complex_bandpass_<→  
t::creator_A ( std::vector< mha_real_t > cf,  
std::vector< mha_real_t > bw,  
mha_real_t srate,  
unsigned int order ) [static]
```

```
4.240.3.2 creator_B() std::vector< mha_complex_t > MHAFilter::complex_bandpass_<→  
t::creator_B ( std::vector< mha_complex_t > A,  
unsigned int order ) [static]
```

```
4.240.3.3 set_state() [1/3] void MHAFilter::complex_bandpass_t::set_state ( mha_real_t val )
```

```
4.240.3.4 set_state() [2/3] void MHAFilter::complex_bandpass_t::set_state ( std::vector< mha_real_t > val )
```

```
4.240.3.5 set_state() [3/3] void MHAFilter::complex_bandpass_t::set_state ( mha_complex_t val )
```

```
4.240.3.6 set_weights() void MHAFilter::complex_bandpass_t::set_weights ( std::vector< mha_complex_t > new_B )
```

Allow to modify the input weights at a later stage.

```
4.240.3.7 get_weights() std::vector< mha_complex_t > MHAFilter::complex_bandpass_<→  
t::get_weights ( ) const [inline]
```

4.240.3.8 filter() [1/4] void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & X, mha_spec_t & Y) [inline]

Filter method for real value input.

4.240.3.9 filter() [2/4] void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & X, mha_wave_t & Yre, mha_wave_t & Yim) [inline]

Filter method for real value input.

4.240.3.10 filter() [3/4] void MHAFilter::complex_bandpass_t::filter (const mha_spec_t & X, mha_spec_t & Y) [inline]

Filter method for complex value input.

4.240.3.11 filter() [4/4] void MHAFilter::complex_bandpass_t::filter (const mha_wave_t & Xre, const mha_wave_t & Xim, mha_wave_t & Yre, mha_wave_t & Yim) [inline]

Filter method for complex value input.

4.240.3.12 inspect() std::string MHAFilter::complex_bandpass_t::inspect () const [inline]

4.240.4 Member Data Documentation

4.240.4.1 A_ std::vector< mha_complex_t> MHAFilter::complex_bandpass_t::A_ [private]

4.240.4.2 B_ std::vector< mha_complex_t> MHAFilter::complex_bandpass_t::B_ [private]

4.240.4.3 Yn std::vector< mha_complex_t> MHAFilter::complex_bandpass_t::Yn [private]

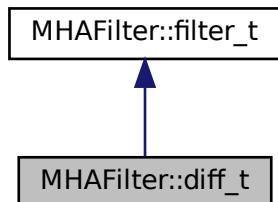
The documentation for this class was generated from the following files:

- **complex_filter.h**
- **complex_filter.cpp**

4.241 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

4.241.1 Detailed Description

Differentiator class (non-normalized)

4.241.2 Constructor & Destructor Documentation

4.241.2.1 diff_t() MHAFilter::diff_t::diff_t (unsigned int ch)

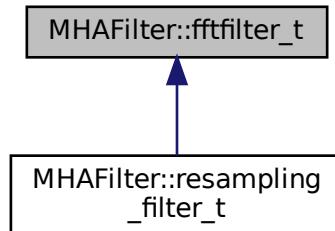
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

4.242 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 **ffflen**)
Constructor.
- **~fftfilter_t ()**
- void **update_coeffs** (const **mha_wave_t** *pwIRS)
Update the set of coefficients.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut, const **mha_wave_t** *pwIRS)
Apply filter with changing coefficients to a waveform fragment.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut)
Apply filter to waveform fragment, without changing the coefficients.
- void **filter** (const **mha_wave_t** *pwIn, **mha_wave_t** **ppwOut, const **mha_spec_t** *psWeights)
Apply filter with changing coefficients to a waveform fragment.

Private Attributes

- unsigned int **fragsize**
- unsigned int **channels**
- unsigned int **ffflen**
- **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**

4.242.1 Detailed Description

FFT based FIR filter implementation.

The maximal number of coefficients can be FFT length - fragsize + 1.

4.242.2 Constructor & Destructor Documentation

4.242.2.1 fftfilter_t() MHAFilter::fftfilter_t::fftfilter_t (

```
    unsigned int fragsize,
    unsigned int channels,
    unsigned int fftlen )
```

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>ffflen</i>	FFT length of filter.

4.242.2.2 ~fftfilter_t() MHAFilter::fftfilter_t::~fftfilter_t ()

4.242.3 Member Function Documentation

4.242.3.1 update_coeffs() 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).

4.242.3.2 filter() [1/3] void MHAFilter::fftfilter_t::filter (

```
const mha_wave_t * pwIn,
mha_wave_t ** ppwOut,
const mha_wave_t * pwIRS )
```

Apply filter with changing coefficients to a waveform fragment.

Parameters

<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.
--------------	--

```
4.242.3.3 filter() [2/3] void MHAFilter::fftfilter_t::filter (
    const mha_wave_t * pwIn,
    mha_wave_t ** ppwOut )
```

Apply filter to waveform fragment, without changing the coefficients.

Parameters

<i>pw</i> ← <i>In</i>	Input signal pointer.
--------------------------	-----------------------

Return values

<i>ppwOut</i>	Pointer to output signal pointer, will be set to a valid signal
---------------	---

```
4.242.3.4 filter() [3/3] void MHAFilter::fftfilter_t::filter (
    const mha_wave_t * pwIn,
    mha_wave_t ** ppwOut,
    const mha_spec_t * psWeights )
```

Apply filter with changing coefficients to a waveform fragment.

Parameters

<i>pw</i> ← <i>In</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.
------------------	--------------------------------------

4.242.4 Member Data Documentation

4.242.4.1 fragsize `unsigned int MHAFilter::fftfilter_t::fragsize` [private]

4.242.4.2 channels `unsigned int MHAFilter::fftfilter_t::channels` [private]

4.242.4.3 fftlen `unsigned int MHAFilter::fftfilter_t::ffflen` [private]

4.242.4.4 wInput_fft `MHASignal::waveform_t MHAFilter::fftfilter_t::wInput_fft` [private]

4.242.4.5 wInput `mha_wave_t MHAFilter::fftfilter_t::wInput` [private]

4.242.4.6 wOutput_fft `MHASignal::waveform_t MHAFilter::fftfilter_t::wOutput_fft` [private]

4.242.4.7 wOutput `mha_wave_t MHAFilter::fftfilter_t::wOutput` [private]

4.242.4.8 sInput `MHASignal::spectrum_t MHAFilter::fftfilter_t::sInput` [private]

4.242.4.9 sWeights `MHASignal::spectrum_t MHAFilter::fftfilter_t::sWeights` [private]

4.242.4.10 wIRS_fft `MHASignal::waveform_t MHAFilter::fftfilter_t::wIRS_fft [private]`**4.242.4.11 fft** `mha_fft_t MHAFilter::fftfilter_t::fft [private]`

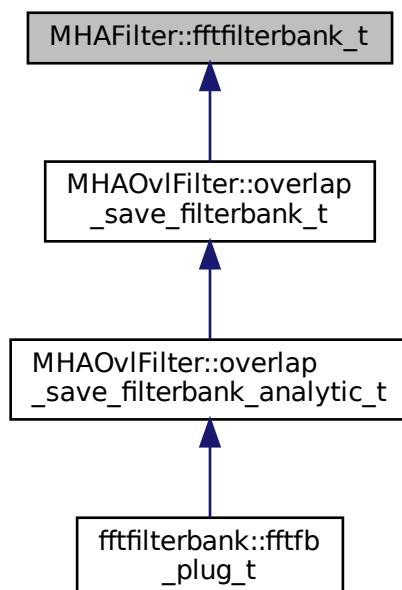
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

4.243 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 **ffflen**)

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_ir** () const

Return the current IRS.

Private Attributes

- unsigned int **fragsize**
- unsigned int **inputchannels**
- unsigned int **firchannels**
- unsigned int **outputchannels**
- unsigned int **ffflen**
- **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*

4.243.1 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.

4.243.2 Constructor & Destructor Documentation

```
4.243.2.1 fftfilterbank_t() MHAFilter::fftfilterbank_t::fftfilterbank_t (
    unsigned int fragsize,
    unsigned int inputchannels,
    unsigned int firchannels,
    unsigned int fftlen )
```

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*.

```
4.243.2.2 ~fftfilterbank_t() MHAFilter::fftfilterbank_t::~fftfilterbank_t ( )
```

4.243.3 Member Function Documentation

```
4.243.3.1 update_coeffs() 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.

4.243.3.2 filter() [1/2] void MHAFilter::fftfilterbank_t::filter (

```
const mha_wave_t * s_in,
mha_wave_t ** s_out,
const mha_wave_t * h )
```

Apply filter with changing coefficients to a waveform fragment.

Parameters

<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
----------	------------------

4.243.3.3 filter() [2/2] void MHAFilter::fftfilterbank_t::filter (

```
const mha_wave_t * s_in,
mha_wave_t ** s_out )
```

Apply filter to waveform fragment, without changing the coefficients.

Parameters

<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
------------------------	---

4.243.3.4 get_irs() const mha_wave_t* MHAFilter::fftfilterbank_t::get_irs () const
[inline]

Return the current IRS.

4.243.4 Member Data Documentation

4.243.4.1 fragsize unsigned int MHAFilter::fftfilterbank_t::fragsize [private]

4.243.4.2 inputchannels unsigned int MHAFilter::fftfilterbank_t::inputchannels [private]

4.243.4.3 firchannels unsigned int MHAFilter::fftfilterbank_t::firchannels [private]

4.243.4.4 outputchannels unsigned int MHAFilter::fftfilterbank_t::outputchannels [private]

4.243.4.5 fftlen unsigned int MHAFilter::fftfilterbank_t::ffflen [private]

4.243.4.6 hw MHASignal::waveform_t MHAFilter::fftfilterbank_t::hw [private]

4.243.4.7 Hs MHASignal::spectrum_t MHAFilter::fftfilterbank_t::Hs [private]

4.243.4.8 XW MHASignal::waveform_t MHAFilter::fftfilterbank_t::xw [private]

4.243.4.9 Xs `MHASignal::spectrum_t` `MHAFilter::fftfilterbank_t::Xs` [private]

4.243.4.10 yw `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::yw` [private]

4.243.4.11 Ys `MHASignal::spectrum_t` `MHAFilter::fftfilterbank_t::Ys` [private]

4.243.4.12 yw_temp `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::yw_temp` [private]

4.243.4.13 tail `MHASignal::waveform_t` `MHAFilter::fftfilterbank_t::tail` [private]

4.243.4.14 fft `mha_fft_t` `MHAFilter::fftfilterbank_t::fft` [private]

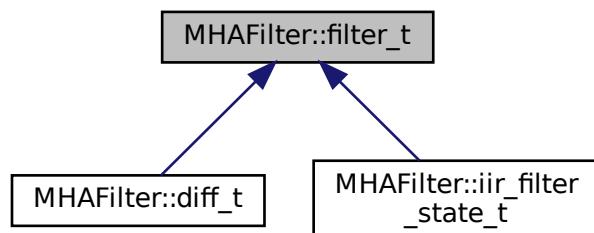
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

4.244 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**

4.244.1 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. 839) structs. The filter coefficients can be directly accessed.

4.244.2 Constructor & Destructor Documentation

4.244.2.1 filter_t() [1/3] 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

4.244.2.2 filter_t() [2/3] MHAFilter::filter_t::filter_t (

```
    unsigned int ch,
    const std::vector< mha_real_t > & vA,
    const std::vector< mha_real_t > & vB )
```

Constructor with initialization of coefficients.

Parameters

<i>ch</i>	Number of channels.
<i>vA</i>	Recursive coefficients.
<i>vB</i>	Non-recursive coefficients.

4.244.2.3 filter_t() [3/3] MHAFilter::filter_t::filter_t (

```
    const MHAFilter::filter_t & src )
```

4.244.2.4 ~filter_t() MHAFilter::filter_t::~filter_t ()

4.244.3 Member Function Documentation

4.244.3.1 filter() [1/3] void MHAFilter::filter_t::filter (

<code>mha_wave_t * out,</code>
<code>const mha_wave_t * in)</code>

Filter all channels in a waveform structure.

Parameters

<i>out</i>	Output signal
<i>in</i>	Input signal

4.244.3.2 filter() [2/3] void MHAFilter::filter_t::filter (

<code>mha_real_t * dest,</code>
<code>const mha_real_t * src,</code>
<code>unsigned int dframes,</code>
<code>unsigned int frame_dist,</code>
<code>unsigned int channel_dist,</code>
<code>unsigned int channel_begin,</code>
<code>unsigned int channel_end)</code>

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

4.244.3.3 filter() [3/3] mha_real_t MHAFilter::filter_t::filter (

<code>mha_real_t x,</code>
<code>unsigned int ch)</code>

Filter one sample.

Parameters

<i>x</i>	Input value
<i>ch</i>	Channel number to use in filter state

4.244.3.4 get_len_A() `unsigned int MHAFilter::filter_t::get_len_A () const [inline]`

Return length of recursive coefficients.

4.244.3.5 get_len_B() `unsigned int MHAFilter::filter_t::get_len_B () const [inline]`

Return length of non-recursive coefficients.

4.244.4 Member Data Documentation

4.244.4.1 A `double* MHAFilter::filter_t::A`

Pointer to recursive coefficients.

4.244.4.2 B `double* MHAFilter::filter_t::B`

Pointer to non-recursive coefficients.

4.244.4.3 len_A `unsigned int MHAFilter::filter_t::len_A [private]`

4.244.4.4 len_B unsigned int MHAFilter::filter_t::len_B [private]

4.244.4.5 len unsigned int MHAFilter::filter_t::len [private]

4.244.4.6 channels unsigned int MHAFilter::filter_t::channels [private]

4.244.4.7 state double* MHAFilter::filter_t::state [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.245 MHAFilter::gamma_flt_t Class Reference

Class for gammatone filter.

Public Member Functions

- **gamma_flt_t** (std::vector< **mha_real_t** > cf, std::vector< **mha_real_t** > bw, **mha_real_t** srate, unsigned int order)
Constructor.
- **~gamma_flt_t** ()
- void **operator()** (**mha_wave_t** &X, **mha_spec_t** &Y)
Filter method.
- void **operator()** (**mha_wave_t** &X, **mha_wave_t** &Yre, **mha_wave_t** &Yim)
Filter method.
- void **operator()** (**mha_wave_t** &Yre, **mha_wave_t** &Yim, unsigned int stage)
Filter method for specific stage.
- void **phase_correction** (unsigned int desired_delay, unsigned int inchannels)
- void **set_weights** (std::vector< **mha_complex_t** > new_B)
- void **set_weights** (unsigned int stage, std::vector< **mha_complex_t** > new_B)
- std::vector< **mha_complex_t** > **get_weights** () const
- std::vector< **mha_complex_t** > **get_weights** (unsigned int stage) const
- std::vector< **mha_real_t** > **get_resynthesis_gain** () const
- void **reset_state** ()
- const std::vector< **mha_complex_t** > & **get_A** ()
- std::string **inspect** () const

Private Attributes

- std::vector< **mha_complex_t** > **A**
- std::vector< **complex_bandpass_t** > **GF**
- **MHASignal::delay_t** * **delay**
- std::vector< int > **envelope_delay**
- std::vector< **mha_real_t** > **resynthesis_gain**
- std::vector< **mha_real_t** > **cf_**
- std::vector< **mha_real_t** > **bw_**
- **mha_real_t** **srate_**

4.245.1 Detailed Description

Class for gammatone filter.

4.245.2 Constructor & Destructor Documentation

```
4.245.2.1 gamma_flt_t() MHAFilter::gamma_flt_t::gamma_flt_t (
    std::vector< mha_real_t > cf,
    std::vector< mha_real_t > bw,
    mha_real_t srate,
    unsigned int order )
```

Constructor.

Parameters

<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.

```
4.245.2.2 ~gamma_flt_t() MHAFilter::gamma_flt_t::~gamma_flt_t ( )
```

4.245.3 Member Function Documentation

4.245.3.1 operator()() [1/3] void MHAFilter::gamma_flt_t::operator() (

```
mha_wave_t & X,  
mha_spec_t & Y ) [inline]
```

Filter method.

4.245.3.2 operator()() [2/3] void MHAFilter::gamma_flt_t::operator() (

```
mha_wave_t & X,  
mha_wave_t & Yre,  
mha_wave_t & Yim ) [inline]
```

Filter method.

4.245.3.3 operator()() [3/3] void MHAFilter::gamma_flt_t::operator() (

```
mha_wave_t & Yre,  
mha_wave_t & Yim,  
unsigned int stage ) [inline]
```

Filter method for specific stage.

4.245.3.4 phase_correction() void MHAFilter::gamma_flt_t::phase_correction (

```
unsigned int desired_delay,  
unsigned int inchannels )
```

4.245.3.5 set_weights() [1/2] void MHAFilter::gamma_flt_t::set_weights (

```
std::vector< mha_complex_t > new_B )
```

4.245.3.6 set_weights() [2/2] void MHAFilter::gamma_flt_t::set_weights (

```
unsigned int stage,  
std::vector< mha_complex_t > new_B )
```

4.245.3.7 get_weights() [1/2] std::vector< **mha_complex_t**> MHAFilter::gamma_flt_t::get_weights () const [inline]

4.245.3.8 get_weights() [2/2] std::vector< **mha_complex_t**> MHAFilter::gamma_flt_t::get_weights (unsigned int stage) const [inline]

4.245.3.9 get_resynthesis_gain() std::vector< **mha_real_t**> MHAFilter::gamma_flt_t::get_resynthesis_gain () const [inline]

4.245.3.10 reset_state() void MHAFilter::gamma_flt_t::reset_state ()

4.245.3.11 get_A() const std::vector< **mha_complex_t**>& MHAFilter::gamma_flt_t::get_A () [inline]

4.245.3.12 inspect() std::string MHAFilter::gamma_flt_t::inspect () const [inline]

4.245.4 Member Data Documentation

4.245.4.1 A std::vector< **mha_complex_t**> MHAFilter::gamma_flt_t::A [private]

4.245.4.2 GF std::vector< **complex_bandpass_t**> MHAFilter::gamma_flt_t::GF [private]

4.245.4.3 delay `MHASignal::delay_t*` `MHAFilter::gamma_flt_t::delay` [private]

4.245.4.4 envelope_delay `std::vector<int>` `MHAFilter::gamma_flt_t::envelope_delay` [private]

4.245.4.5 resynthesis_gain `std::vector< mha_real_t>` `MHAFilter::gamma_flt_t::resynthesis->gain` [private]

4.245.4.6 cf_ `std::vector< mha_real_t>` `MHAFilter::gamma_flt_t::cf_` [private]

4.245.4.7 bw_ `std::vector< mha_real_t>` `MHAFilter::gamma_flt_t::bw_` [private]

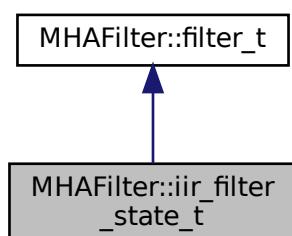
4.245.4.8 srate_ `mha_real_t` `MHAFilter::gamma_flt_t::srate_` [private]

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

- **complex_filter.h**
- **complex_filter.cpp**

4.246 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

4.246.1 Constructor & Destructor Documentation

4.246.1.1 iir_filter_state_t() MHAFilter::iir_filter_state_t::iir_filter_state_t (unsigned int *channels*, std::vector< float > *cf_A*, std::vector< float > *cf_B*)

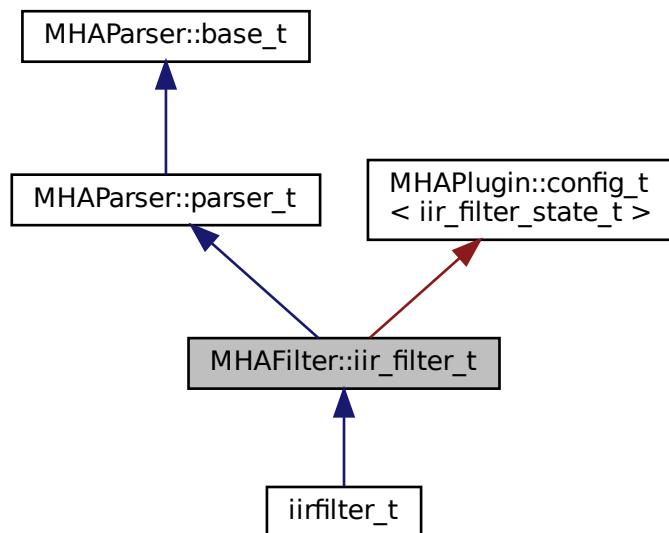
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

4.247 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

4.247.1 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. 1097), 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.

4.247.2 Constructor & Destructor Documentation

```
4.247.2.1 iir_filter_t() MHAFilter::iir_filter_t::iir_filter_t (
    std::string help = "IIR filter structure",
    std::string def_A = "[1]",
    std::string def_B = "[1]",
    unsigned int channels = 1 )
```

Constructor of the IIR filter.

Initialises the sub-parser structure and the memory for holding the filter's state.

Parameters

<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.

4.247.3 Member Function Documentation

```
4.247.3.1 filter() [1/2] void MHAFilter::iir_filter_t::filter (
    mha_wave_t * y,
    const mha_wave_t * x )
```

The filter processes the audio signal.

All channels in the audio signal are processed using the same filter coefficients. 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 <i>x</i> . In-place processing (<i>y</i> and <i>x</i> 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. 899) method.

```
4.247.3.2 filter() [2/2] mha_real_t MHAFilter::iir_filter_t::filter (
    mha_real_t x,
    unsigned int ch )
```

Filter a single audio sample.

Parameters

<i>x</i>	The single audio sample
<i>ch</i>	Zero-based channel index. Use and change the state of channel <i>ch</i> . <i>ch</i> has to be less than the number of channels given to the constructor or the resize (p. 899) method.

Returns

the filtered result sample.

```
4.247.3.3 resize() 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.
-----------------	---

```
4.247.3.4 update_filter() void MHAFilter::iir_filter_t::update_filter ( ) [private]
```

4.247.4 Member Data Documentation

```
4.247.4.1 A MHParse::vfloat_t MHAFilter::iir_filter_t::A [private]
```

4.247.4.2 B `MHAParser::vfloat_t MHAFilter::iir_filter_t::B` [private]

4.247.4.3 connector `MHAEvents::patchbay_t< iir_filter_t> MHAFilter::iir_filter< -t::connector` [private]

4.247.4.4 nchannels `unsigned int MHAFilter::iir_filter_t::nchannels` [private]

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

- `mha_filter.hh`
- `mha_filter.cpp`

4.248 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**

4.248.1 Detailed Description

First order recursive filter.

4.248.2 Constructor & Destructor Documentation

4.248.2.1 iir_ord1_real_t() [1/2] MHAFilter::iir_ord1_real_t::iir_ord1_real_t (std::vector< **mha_real_t** > *A*, std::vector< **mha_real_t** > *B*)

Constructor with filter coefficients (one per channel)

4.248.2.2 iir_ord1_real_t() [2/2] MHAFilter::iir_ord1_real_t::iir_ord1_real_t (std::vector< **mha_real_t** > *tau*, **mha_real_t** *srate*)

Constructor for low pass filter (one time constant per channel)

4.248.3 Member Function Documentation

4.248.3.1 set_state() [1/3] void MHAFilter::iir_ord1_real_t::set_state (**mha_real_t** *val*)

4.248.3.2 set_state() [2/3] void MHAFilter::iir_ord1_real_t::set_state (std::vector< mha_real_t > val)

4.248.3.3 set_state() [3/3] void MHAFilter::iir_ord1_real_t::set_state (mha_complex_t val)

4.248.3.4 operator()() [1/5] mha_real_t MHAFilter::iir_ord1_real_t::operator() (unsigned int ch, mha_real_t x) [inline]

Filter method for real value input, one element.

4.248.3.5 operator()() [2/5] mha_complex_t MHAFilter::iir_ord1_real_t::operator() (unsigned int ch, mha_complex_t x) [inline]

Filter method for complex input, one element.

4.248.3.6 operator()() [3/5] void MHAFilter::iir_ord1_real_t::operator() (const mha_wave_t & X, mha_wave_t & Y) [inline]

Filter method for real value input.

4.248.3.7 operator()() [4/5] void MHAFilter::iir_ord1_real_t::operator() (const mha_spec_t & X, mha_spec_t & Y) [inline]

Filter method for complex value input.

```
4.248.3.8 operator()() [5/5] void MHAFilter::iir_ord1_real_t::operator() (  
    const mha_wave_t & Xre,  
    const mha_wave_t & Xim,  
    mha_wave_t & Yre,  
    mha_wave_t & Yim ) [inline]
```

Filter method for complex value input.

4.248.4 Member Data Documentation

4.248.4.1 A_ std::vector< mha_real_t> MHAFilter::iir_ord1_real_t::A_ [private]

4.248.4.2 B_ std::vector< mha_real_t> MHAFilter::iir_ord1_real_t::B_ [private]

4.248.4.3 Yn std::vector< mha_complex_t> MHAFilter::iir_ord1_real_t::Yn [private]

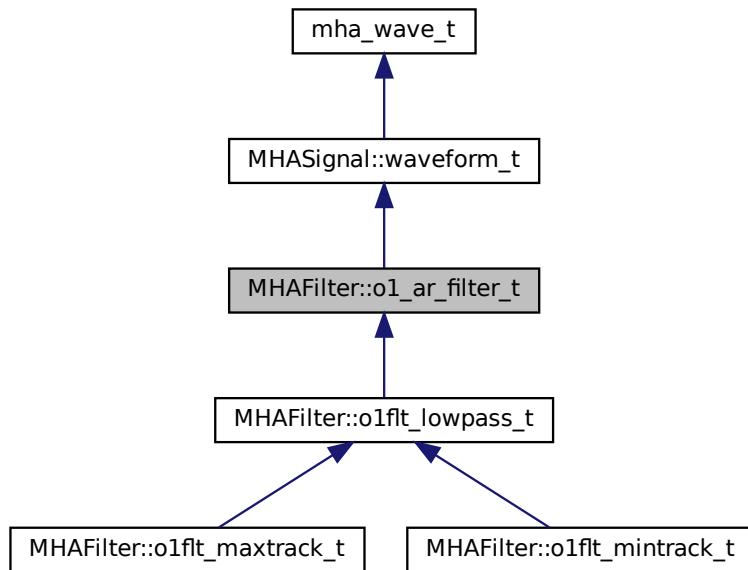
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

4.249 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. 839) 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

4.249.1 Detailed Description

First order attack-release lowpass filter.

This filter is the base of first order lowpass filter, maximum tracker and minimum tracker.

4.249.2 Constructor & Destructor Documentation

```
4.249.2.1 o1_ar_filter_t() MHAFilter::o1_ar_filter_t::o1_ar_filter_t (
    unsigned int channels,
    mha_real_t fs = 1.0f,
    std::vector< mha_real_t > tau_a = std::vector<float>(1, 0.0f),
    std::vector< mha_real_t > tau_r = std::vector<float>(1, 0.0f) )
```

Constructor, setting all taus to zero.

The filter state can be accessed through the member functions of **MHASignal::waveform_t** (p. [1259](#)).

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)

4.249.3 Member Function Documentation

```
4.249.3.1 set_tau_attack() void MHAFilter::o1_ar_filter_t::set_tau_attack (
    unsigned int ch,
    mha_real_t tau )
```

Set the attack time constant.

Parameters

<i>ch</i>	Channel number
<i>tau</i>	Time constant

4.249.3.2 set_tau_release() void MHAFilter::ol_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

4.249.3.3 operator()() [1/2] **mha_real_t** MHAFilter::ol_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

4.249.3.4 operator()() [2/2] void MHAFilter::ol_ar_filter_t::operator() (const **mha_wave_t** & *in*, **mha_wave_t** & *out*) [inline]

Apply filter to a **mha_wave_t** (p. 839) data.

Parameters

<i>in</i>	Input signal
<i>out</i>	Output signal

The number of channels must match the number of filter bands.

4.249.4 Member Data Documentation

4.249.4.1 c1_a `MHASignal::waveform_t` `MHAFilter::o1_ar_filter_t::c1_a` [protected]

4.249.4.2 c2_a `MHASignal::waveform_t` `MHAFilter::o1_ar_filter_t::c2_a` [protected]

4.249.4.3 c1_r `MHASignal::waveform_t` `MHAFilter::o1_ar_filter_t::c1_r` [protected]

4.249.4.4 c2_r `MHASignal::waveform_t` `MHAFilter::o1_ar_filter_t::c2_r` [protected]

4.249.4.5 fs `mha_real_t` `MHAFilter::o1_ar_filter_t::fs` [protected]

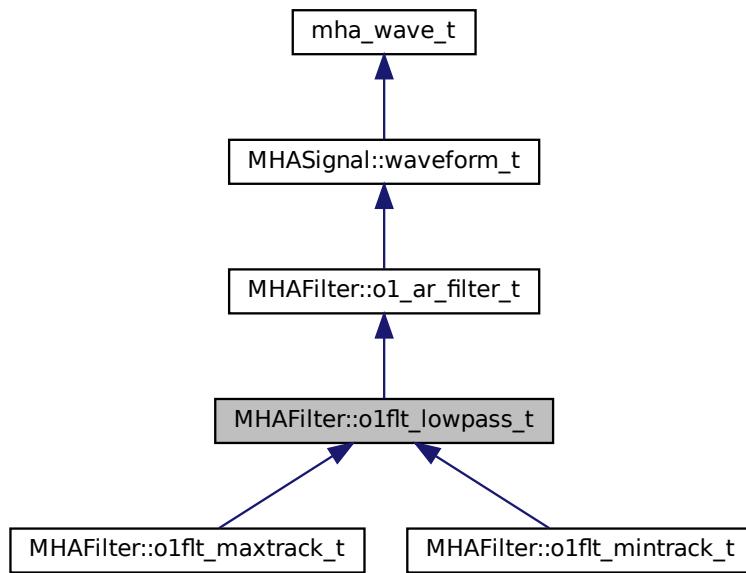
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

4.250 MHAFilter::o1flt_lowpass_t Class Reference

First order low pass filter.

Inheritance diagram for MHAFilter::o1flt_lowpass_t:



Public Member Functions

- **o1flt_lowpass_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
Constructor of low pass filter, sets sampling rate and time constants.
- **o1flt_lowpass_t** (const std::vector< **mha_real_t** > &tau, **mha_real_t** fs, const std::vector< **mha_real_t** > &startval)
Constructor of low pass filter, sets sampling rate and time constants.
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
set time constant in all channels to tau
- **mha_real_t get_c1** (unsigned int ch) const
- **mha_real_t get_last_output** (unsigned int ch) const

Additional Inherited Members

4.250.1 Detailed Description

First order low pass filter.

4.250.2 Constructor & Destructor Documentation

4.250.2.1 o1flt_lowpass_t() [1/2] MHAFilter::o1flt_lowpass_t::o1flt_lowpass_t (

```
const std::vector< mha_real_t > & tau,
mha_real_t fs,
mha_real_t startval = 0 )
```

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

<i>tau</i>	Vector of time constants
<i>fs</i>	Sampling rate
<i>startval</i>	Initial internal state value

4.250.2.2 o1flt_lowpass_t() [2/2] MHAFilter::o1flt_lowpass_t::o1flt_lowpass_t (

```
const std::vector< mha_real_t > & tau,
mha_real_t fs,
const std::vector< mha_real_t > & startval )
```

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

<i>tau</i>	Vector of time constants
<i>fs</i>	Sampling rate
<i>startval</i>	Initial internal state value

4.250.3 Member Function Documentation

4.250.3.1 set_tau() [1/2] void MHAFilter::o1flt_lowpass_t::set_tau (

```
unsigned int ch,
mha_real_t tau )
```

change the time constant in one channel

4.250.3.2 set_tau() [2/2] void MHAFilter::o1flt_lowpass_t::set_tau (mha_real_t tau)

set time constant in all channels to tau

4.250.3.3 get_c1() mha_real_t MHAFilter::o1flt_lowpass_t::get_c1 (unsigned int ch) const [inline]

4.250.3.4 get_last_output() mha_real_t MHAFilter::o1flt_lowpass_t::get_last_output (unsigned int ch) const [inline]

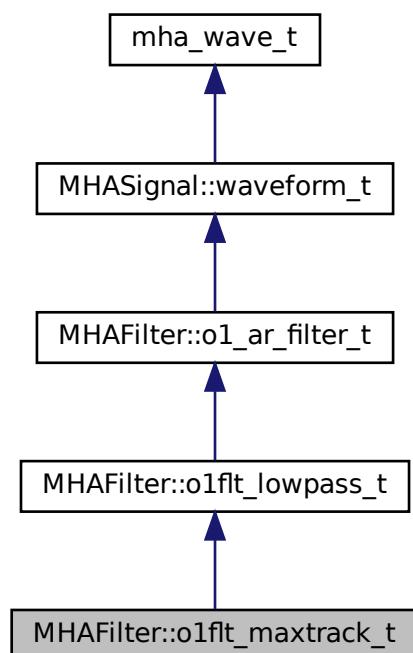
The documentation for this class was generated from the following files:

- **mha_filter.hh**
- **mha_filter.cpp**

4.251 MHAFilter::o1flt_maxtrack_t Class Reference

First order maximum tracker.

Inheritance diagram for MHAFilter::o1flt_maxtrack_t:



Public Member Functions

- **o1flt_maxtrack_t** (const std::vector< **mha_real_t** > &, **mha_real_t**, **mha_real_t**=0)
Constructor of low pass filter, sets sampling rate and time constants.
- **o1flt_maxtrack_t** (const std::vector< **mha_real_t** > &tau, **mha_real_t** fs, const std::vector< **mha_real_t** > &startval)
- void **set_tau** (unsigned int ch, **mha_real_t** tau)
change the time constant in one channel
- void **set_tau** (**mha_real_t** tau)
set time constant in all channels to tau

Additional Inherited Members

4.251.1 Detailed Description

First order maximum tracker.

4.251.2 Constructor & Destructor Documentation

```
4.251.2.1 o1flt_maxtrack_t() [1/2] MHAFilter::o1flt_maxtrack_t::o1flt_maxtrack_t (
    const std::vector< mha_real_t > & tau,
    mha_real_t fs,
    mha_real_t startval = 0 )
```

Constructor of low pass filter, sets sampling rate and time constants.

Parameters

<i>tau</i>	Vector of time constants
<i>fs</i>	Sampling rate
<i>startval</i>	Initial internal state value

```
4.251.2.2 o1flt_maxtrack_t() [2/2] MHAFilter::o1flt_maxtrack_t::o1flt_maxtrack_t (
    const std::vector< mha_real_t > & tau,
    mha_real_t fs,
    const std::vector< mha_real_t > & startval )
```

4.251.3 Member Function Documentation

4.251.3.1 set_tau() [1/2] void MHAFilter::o1flt_maxtrack_t::set_tau (unsigned int ch, mha_real_t tau)

change the time constant in one channel

4.251.3.2 set_tau() [2/2] void MHAFilter::o1flt_maxtrack_t::set_tau (mha_real_t tau)

set time constant in all channels to tau

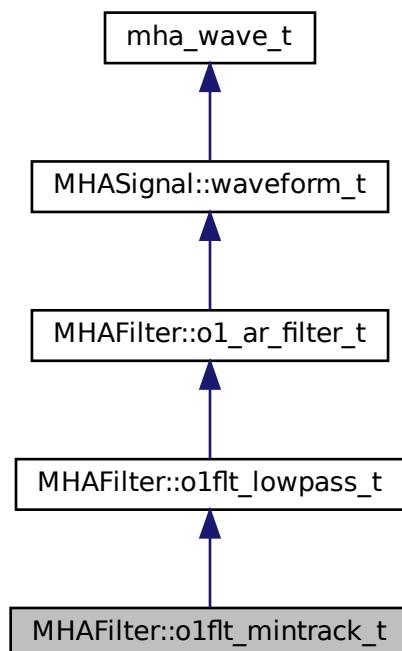
The documentation for this class was generated from the following files:

- [mha_filter.hh](#)
- [mha_filter.cpp](#)

4.252 MHAFilter::o1flt_mintrack_t Class Reference

First order minimum tracker.

Inheritance diagram for MHAFilter::o1flt_mintrack_t:



Public Member Functions

- `o1flt_mintrack_t (const std::vector< mha_real_t > &, mha_real_t, mha_real_t=0)`
- `o1flt_mintrack_t (const std::vector< mha_real_t > &, mha_real_t, const std::vector< mha_real_t > &)`
- `void set_tau (unsigned int ch, mha_real_t tau)`
change the time constant in one channel
- `void set_tau (mha_real_t tau)`
set time constant in all channels to tau

Additional Inherited Members

4.252.1 Detailed Description

First order minimum tracker.

4.252.2 Constructor & Destructor Documentation

4.252.2.1 o1flt_mintrack_t() [1/2] MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (

```
const std::vector< mha_real_t > & tau,
mha_real_t fs,
mha_real_t startval = 0 )
```

4.252.2.2 o1flt_mintrack_t() [2/2] MHAFilter::o1flt_mintrack_t::o1flt_mintrack_t (

```
const std::vector< mha_real_t > & tau,
mha_real_t fs,
const std::vector< mha_real_t > & startval )
```

4.252.3 Member Function Documentation

4.252.3.1 set_tau() [1/2] void MHAFilter::olflt_mintrack_t::set_tau (unsigned int ch, mha_real_t tau)

change the time constant in one channel

4.252.3.2 set_tau() [2/2] void MHAFilter::olflt_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**

4.253 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.

4.253.1 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.

4.253.2 Constructor & Destructor Documentation

```
4.253.2.1 partitioned_convolution_t() MHAFilter::partitioned_convolution_t::partitioned_convolution_t (
    unsigned int fragsize,
    unsigned int nchannels_in,
    unsigned int nchannels_out,
    const transfer_matrix_t & transfer )
```

Create a new partitioned convolver.

Parameters

<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.

```
4.253.2.2 ~partitioned_convolution_t() MHAFilter::partitioned_convolution_t::~partitioned_convolution_t ( )
```

Free fftw resource allocated in constructor.

4.253.3 Member Function Documentation

```
4.253.3.1 process() mha_wave_t * MHAFilter::partitioned_convolution_t::process (
    const mha_wave_t * s_in )
```

processing

4.253.4 Member Data Documentation

```
4.253.4.1 fragsize unsigned int MHAFilter::partitioned_convolution_t::fragsize
```

Audio fragment size, always equal to partition size.

4.253.4.2 nchannels_in `unsigned int MHAFilter::partitioned_convolution_t::nchannels_in`

Number of audio input channels.

4.253.4.3 nchannels_out `unsigned int MHAFilter::partitioned_convolution_t::nchannels_out`

Number of audio output channels.

4.253.4.4 output_partitions `unsigned int MHAFilter::partitioned_convolution_t::output_partitions`

The maximum number of partitions in any of the impulse responses.

Determines the size if the delay line.

4.253.4.5 filter_partitions `unsigned int MHAFilter::partitioned_convolution_t::filter_partitions`

The total number of non-zero impulse response partitions.

4.253.4.6 input_signal_wave `MHASignal::waveform_t MHAFilter::partitioned_convolution_t::input_signal_wave`

Buffer for input signal.

Has `nchannels_in` channels and `fragsize*2` frames

4.253.4.7 current_input_signal_buffer_half_index `unsigned int MHAFilter::partitioned_convolution_t::current_input_signal_buffer_half_index`

A counter modulo 2.

Indicates the buffer half in input signal wave into which to copy the current input signal.

4.253.4.8 input_signal_spec `MHASignal::spectrum_t` `MHAFilter::partitioned_convolution_t::input_signal_spec`

Buffer for FFT transformed input signal.

Has nchannels_in channels and fragsize+1 frames (fft bins).

4.253.4.9 frequency_response `MHASignal::spectrum_t` `MHAFilter::partitioned_convolution_t::frequency_response`

Buffers for frequency response spectra of impulse response partitions.

Each "channel" contains another partition of some impulse response. The bookkeeping array is used to keep track what to do with these frequency responses. This container has filter_partitions channels and fragsize+1 frames (fft bins).

4.253.4.10 bookkeeping `std::vector< index_t >` `MHAFilter::partitioned_convolution_t::bookkeeping`

Keeps track of input channels, output channels, impulse response partition, and delay.

The index into this array is the same as the "channel" index into the frequency_response array. Array has filter_partitions entries.

4.253.4.11 output_signal_spec `std::vector< MHASignal::spectrum_t >` `MHAFilter::partitioned_convolution_t::output_signal_spec`

Buffers for FFT transformed output signal.

For each array member, Number of channels is equal to nchannels_out, number of frames (fft bins) is equal to fragsize+1. Array size is equal to output_partitions.

4.253.4.12 current_output_partition_index `unsigned int` `MHAFilter::partitioned_convolution_t::current_output_partition_index`

A counter modulo output_partitions, indexing the "current" output partition.

4.253.4.13 output_signal_wave `MHASignal::waveform_t` `MHAFilter::partitioned_convolution_t::output_signal_wave`

Buffer for the wave output signal.

Number of channels is equal to nchannels_out, number of frames is equal to fragsize

4.253.4.14 fft mha_fft_t MHAFilter::partitioned_convolution_t::fft

The FFT transformer.

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.254 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.

4.254.1 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.

4.254.2 Constructor & Destructor Documentation**4.254.2.1 index_t() [1/2] MHAFilter::partitioned_convolution_t::index_t::index_t (**

```
    unsigned int src,
    unsigned int tgt,
    unsigned int dly ) [inline]
```

Data constructor.

Parameters

<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

4.254.2.2 `index_t()` [2/2] `MHAFilter::partitioned_convolution_t::index_t::index_t ()`
 [inline]

Default constructor for STL compatibility.

4.254.3 Member Data Documentation

4.254.3.1 `source_channel_index` `unsigned int MHAFilter::partitioned_convolution_t::index_t::source_channel_index`

The input channel index to apply the current partition to.

4.254.3.2 `target_channel_index` `unsigned int MHAFilter::partitioned_convolution_t::index_t::target_channel_index`

The index of the output channel to which the filter result should go.

4.254.3.3 `delay` `unsigned int MHAFilter::partitioned_convolution_t::index_t::delay`

The delay (in blocks) of this partition.

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

- **mha_filter.hh**

4.255 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.

4.255.1 Detailed Description

A class that performs polyphase resampling.

Background information: When resampling from one sampling rate to another, it helps when one sampling rate is a multiple of the other sampling rate: In the case of upsampling, the samples at the original rate are copied to the upsampled signal spread out with a constant number of zero samples between the originally adjacent samples. The signal is then low-pass filtered to avoid frequency aliasing and to fill the zero-samples with interpolated values. In the case of down-sampling, the signal is first low-pass filtered for anti-aliasing, and only every n^{th}

sample of the filtered output is used for the signal at the new sample rate. Of course, for finite-impulse-response (FIR) filters this means that only every n^{th} sample needs to be computed.

When resampling from one sampling rate to another where neither is a multiple of the other, the signal first needs to be upsampled to a sampling rate that is a multiple of both (source and target) sampling rates, and then downsampled again to the target sampling rate. Instead of applying two separate lowpass filters directly after each other (one filter for upsampling and another for downsampling), it is sufficient to apply only one low-pass filter, when producing the output at the final target rate, with a cut-off frequency equal to the lower cut-off-frequency of the replaced two low-pass filters. Not filtering to produce a filtered signal already at the common multiple sampling rate has the side effect that this intermediate signal at the common multiple sampling rate keeps its filler zero samples unaltered. These zero samples can be taken advantage of when filtering to produce the output at the target rate: The zeros do not need to be multiplied with their corresponding filter coefficients, because the result is known to be zero again, and this zero product has no effect on the summation operation to compute a target sample at the target rate. To summarize, the following optimization techniques are available:

- The signal does not need to be stored in memory at the interpolation rate. It is sufficient to have the signal available at the source rate and to know where the zeros would be.
- The signal needs to be low-pass-filtered only once.
- The FIR low-pass filtering can take advantage of
 - computing only filter outputs for the required samples at the target rate,
 - skipping over zero-samples at the interpolation rate.

The procedure that takes advantage of these optimization possibilites is known as polyphase resampling.

This class implements polyphase resampling in this way for a source sampling rate and a target sampling rate that have common multiple, the interpolation sampling rate. Non-rational and drifting sample rates are outside the scope of this resampler.

4.255.2 Constructor & Destructor Documentation

```
4.255.2.1 polyphase_resampling_t() MHAFilter::polyphase_resampling_t::polyphase_←
resampling_t (
    unsigned n_up,
    unsigned n_down,
    mha_real_t nyquist_ratio,
    unsigned n_irs,
    unsigned n_ringbuffer,
    unsigned n_channels,
    unsigned n_prefill )
```

Construct a polyphase resampler instance.

Allocates a ringbuffer with the given capacity *n_ringbuffer*. Client that triggers the constructor must ensure that the capacity *n_ringbuffer* and the delay *n_prefill* are sufficient, i.e. enough old and new samples are always available to compute sufficient samples in using an impulse response of length *n_irs*. Audio block sizes at both sides of the resampler have to be taken into account. Class `MHASignal::blockprocessing_polyphase_resampling_t` takes care of this, and it is recommended to use this class for block-based processing.

Based on *n_up*, *n_down*, *n_irs* and *nyquist_ratio*, a suitable sinc impulse response is computed and windowed with a hanning window to limit its extent.

The actual source sampling rate, target sampling rate, and interpolation sampling rate are not parameters to this constructors, because only their ratios matter.

Parameters

<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

4.255.3 Member Function Documentation

```
4.255.3.1 write() void MHAFilter::polyphase_resampling_t::write (
    mha_wave_t & signal )
```

Write signal to the ringbuffer.

Signal contained in signal is appended to the audio frames already present in the ringbuffer.

Parameters

<i>signal</i>	input signal in original sampling rate
---------------	--

Exceptions

MHA_Error (p. 763)	Raises exception if there is not enough room or if the number of channels does not match.
---------------------------	---

4.255.3.2 `read()` `void MHAFilter::polyphase_resampling_t::read (mha_wave_t & signal)`

Read resampled signal.

Will perform the resampling and remove no longer needed samples from the input buffer.

Parameters

<i>signal</i>	buffer to write the resampled signal to.
---------------	--

Exceptions

MHA_Error (p. 763)	Raises exception if there is not enough input signal or if the number of channels is too high.
---------------------------	--

4.255.3.3 `readable_frames()` `unsigned MHAFilter::polyphase_resampling_t::readable_frames () const [inline]`

Number of frames at target sampling rate that can be produced.

This method only checks for enough future samples present, therefore, this number can be positive and a read operation can still fail if there are not enough past samples present to perform the filtering for the first output sample. This could only happen if the constructor parameters *n_ringbuffer* or *n_prefill* have been chosen too small, because otherwise the method **read** (p. 924) ensures that enough past samples are present to compute the next target sample.

4.255.4 Member Data Documentation

4.255.4.1 upsampling_factor `unsigned MHAFilter::polyphase_resampling_t::upsampling_factor [private]`

Integer upsampling factor.

Interpolation rate divided by source rate.

4.255.4.2 downsampling_factor `unsigned MHAFilter::polyphase_resampling_t::downsampling_factor [private]`

Integer downsampling factor.

Interpolation rate divided by target rate.

4.255.4.3 now_index `unsigned MHAFilter::polyphase_resampling_t::now_index [private]`

Index of "now" in the interpolated sampling rate.

4.255.4.4 underflow `bool MHAFilter::polyphase_resampling_t::underflow [private]`

Set to true when an underflow has occurred.

When this is true, then the object can no longer be used. Underflows have to be avoided by clients, e.g. by checking that enough **readable_frames** (p. 924) are present before calling **read** (p. 924)

4.255.4.5 impulse_response `MHAWindow::hanning_t MHAFilter::polyphase_resampling_t::impulse_response [private]`

Contains the impulse response of the lowpass filter needed for anti-aliasing.

The impulse response is stored at the interpolation sampling rate. We use an instance of **MHAWindow::hanning_t** (p. 1293) 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. 1293) can be altered with *=, which our constructor does.

4.255.4.6 ringbuffer `MHASignal::ringbuffer_t` `MHAFilter::polyphase_resampling_t` ←
`::ringbuffer` [private]

Storage of input signal.

Part of the polyphase resampling optimization is that apart from the FIR impulse response, nothing is stored at the interpolation rate, saving memory and computation cycles.

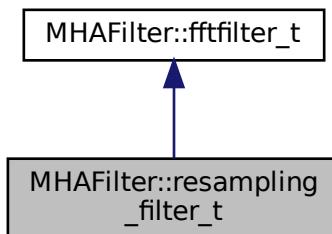
The documentation for this class was generated from the following files:

- `mha_filter.hh`
- `mha_filter.cpp`

4.256 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 `ffflen`, unsigned int `irlen`, unsigned int `channels`, unsigned int `Nup`, unsigned int `Ndown`, double `fCutOff`)
Constructor.

Static Public Member Functions

- static unsigned int **fragsize_validator** (unsigned int `ffflen`, unsigned int `irlen`)

Private Attributes

- unsigned int **fragsize**

4.256.1 Detailed Description

Hann shaped low pass filter for resampling.

This class uses FFT filter at upsampled rate.

4.256.2 Constructor & Destructor Documentation

```
4.256.2.1 resampling_filter_t() MHAFilter::resampling_filter_t
(
    unsigned int fftlen,
    unsigned int irslen,
    unsigned int channels,
    unsigned int Nup,
    unsigned int Ndown,
    double fCutOff )
```

Constructor.

Parameters

<i>fftlen</i>	FFT length.
<i>irslen</i>	Length of filter.
<i>channels</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)

4.256.3 Member Function Documentation

```
4.256.3.1 fragsize_validator() unsigned int MHAFilter::resampling_filter_t::fragsize←
_validator (
    unsigned int fftlen,
    unsigned int irslen ) [static]
```

4.256.4 Member Data Documentation

4.256.4.1 fragsize unsigned int MHAFilter::resampling_filter_t::fragsize [private]

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

- **mha_filter.hh**
- **mha_filter.cpp**

4.257 MHAFilter::smoothspec_t Class Reference

Smooth spectral gains, create a windowed impulse response.

Public Member Functions

- **smoothspec_t** (unsigned int **ffflen**, 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 **ffflen**
- 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**

4.257.1 Detailed Description

Smooth spectral gains, create a windowed impulse response.

Spectral gains are smoothed by multiplying the impulse response with a window function.

If a minimal phase is used, then the original phase is discarded and replaced by the minimal phase function. In this case, the window is applied to the beginning of the inverse Fourier transform of the input spectrum, and the remaining signal set to zero. If the original phase is kept, the window is applied symmetrically around zero, i.e. to the first and last samples of the inverse Fourier transform of the input spectrum. The **spec2fir()** (p. 931) function creates a causal impulse response by circularly shifting the impulse response by half of the window length.

The signal dimensions of the arguments of **smoothspec()** (p. 930) must correspond to the FFT length and number of channels provided in the constructor. The function **spec2fir()** (p. 931) can fill signal structures with more than window length frames.

4.257.2 Constructor & Destructor Documentation

```
4.257.2.1 smoothspec_t() MHAFilter::smoothspec_t::smoothspec_t (
    unsigned int fftlen,
    unsigned int nchannels,
    const MHAWindow::base_t & window,
    bool minphase,
    bool linphase_asym = false )
```

Constructor.

Parameters

<i>ffflen</i>	FFT length of input spectrum (ffflen/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

4.257.2.2 ~smoothspec_t() `MHAFilter::smoothspec_t::~smoothspec_t ()`

4.257.3 Member Function Documentation

4.257.3.1 smoothspec() [1/2] `void MHAFilter::smoothspec_t::smoothspec (`

```
const mha_spec_t & s_in,
mha_spec_t & s_out )
```

Create a smoothed spectrum.

Parameters

<i>s_in</i>	Input spectrum
-------------	----------------

Return values

<i>s_out</i>	Output spectrum
--------------	-----------------

4.257.3.2 smoothspec() [2/2] `void MHAFilter::smoothspec_t::smoothspec (`
`mha_spec_t & spec) [inline]`

Create a smoothed spectrum (in place)

Parameters

<i>spec</i>	Spectrum to be smoothed.
-------------	--------------------------

4.257.3.3 spec2fir() 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
------------	---

4.257.3.4 internal_fir() void MHAFilter::smoothspec_t::internal_fir (const mha_spec_t & *s_in*) [private]

4.257.4 Member Data Documentation

4.257.4.1 fftlen unsigned int MHAFilter::smoothspec_t::fftlens [private]

4.257.4.2 nchannels unsigned int MHAFilter::smoothspec_t::nchannels [private]

4.257.4.3 window `MHAWindow::base_t MHAFilter::smoothspec_t::window` [private]

4.257.4.4 tmp_wave `MHASignal::waveform_t MHAFilter::smoothspec_t::tmp_wave` [private]

4.257.4.5 tmp_spec `MHASignal::spectrum_t MHAFilter::smoothspec_t::tmp_spec` [private]

4.257.4.6 minphase `MHASignal::minphase_t* MHAFilter::smoothspec_t::minphase` [private]

4.257.4.7 _linphase_asym `bool MHAFilter::smoothspec_t::_linphase_asym` [private]

4.257.4.8 fft `mha_fft_t MHAFilter::smoothspec_t::fft` [private]

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

- `mha_filter.hh`
- `mha_filter.cpp`

4.258 MHAFilter::thirddoctave_analyzer_t Class Reference

Public Member Functions

- `thirddoctave_analyzer_t (mhaconfig_t cfg)`
- `mha_wave_t * process (mha_wave_t *)`
- `unsigned int nbands ()`
- `unsigned int nchannels ()`
- `std::vector< mha_real_t > get_cf_hz ()`

Static Public Member Functions

- static std::vector< **mha_real_t** > **cf_generator** (**mhaconfig_t** cfg)
- static std::vector< **mha_real_t** > **bw_generator** (**mhaconfig_t** cfg)
- static std::vector< **mha_real_t** > **dup** (std::vector< **mha_real_t** >, **mhaconfig_t** cfg)

Private Attributes

- **mhaconfig_t** **cfg_**
- std::vector< **mha_real_t** > **cf**
- **MHAFilter::gamma_flt_t** **fb**
- **MHASignal::waveform_t** **out_chunk**
- **MHASignal::waveform_t** **out_chunk_im**

4.258.1 Constructor & Destructor Documentation

4.258.1.1 thirddoctave_analyzer_t() MHAFilter::thirddoctave_analyzer_t::thirddoctave_analyzer_t (**mhaconfig_t** cfg)

4.258.2 Member Function Documentation

4.258.2.1 process() **mha_wave_t** * MHAFilter::thirddoctave_analyzer_t::process (**mha_wave_t** * sIn)

4.258.2.2 nbands() unsigned int MHAFilter::thirddoctave_analyzer_t::nbands ()

4.258.2.3 nchannels() unsigned int MHAFilter::thirddoctave_analyzer_t::nchannels ()

4.258.2.4 get_cf_hz() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_t::get_cf_hz ()

4.258.2.5 cf_generator() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_t::cf_generator (**mhaconfig_t** cfg) [static]

4.258.2.6 bw_generator() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_t::bw_generator (**mhaconfig_t** cfg) [static]

4.258.2.7 dup() std::vector< **mha_real_t** > MHAFilter::thirdoctave_analyzer_t::dup (std::vector< **mha_real_t** > vec, **mhaconfig_t** cfg) [static]

4.258.3 Member Data Documentation

4.258.3.1 cfg_ **mhaconfig_t** MHAFilter::thirdoctave_analyzer_t::cfg_ [private]

4.258.3.2 cf std::vector< **mha_real_t**> MHAFilter::thirdoctave_analyzer_t::cf [private]

4.258.3.3 fb **MHAFilter::gamma_flt_t** MHAFilter::thirdoctave_analyzer_t::fb [private]

4.258.3.4 out_chunk `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk [private]`

4.258.3.5 out_chunk_im `MHASignal::waveform_t MHAFilter::thirdoctave_analyzer_t::out_chunk_im [private]`

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

- `complex_filter.h`
- `complex_filter.cpp`

4.259 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.

4.259.1 Detailed Description

a structure containing a source channel number, a target channel number, and an impulse response.

4.259.2 Constructor & Destructor Documentation

4.259.2.1 transfer_function_t() [1/2] MHAFilter::transfer_function_t::transfer_↔
function_t () const [inline]

Default constructor for STL conformity.

Not used.

4.259.2.2 transfer_function_t() [2/2] MHAFilter::transfer_function_t::transfer_↔
function_t (unsigned int *source_channel_index*,
unsigned int *target_channel_index*,
const std::vector< float > & *impulse_response*)

Data constructor.

Parameters

<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

4.259.3 Member Function Documentation

4.259.3.1 partitions() unsigned int MHAFilter::transfer_function_t::partitions (unsigned int *fragsize*) const [inline]

for the given partition size, return the number of partitions of the impulse response.

Parameters

<i>fragsize</i>	partition size
-----------------	----------------

Returns

number of partitions occupied by the impulse response

```
4.259.3.2 non_empty_partitions() unsigned int MHAFilter::transfer_function_t::non_empty_partitions (
    unsigned int fragsize ) const [inline]
```

for the given partition size, return the number of non-empty partitions of the impulse response.

Parameters

<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.

```
4.259.3.3 isempty() bool MHAFilter::transfer_function_t::isempty (
    unsigned int fragsize,
    unsigned int index ) const [inline]
```

checks if the partition contains only zeros

Parameters

<i>fragsize</i>	partition size
<i>index</i>	partition index

Returns

true when this partition of the impulse response contains only zeros.

4.259.4 Member Data Documentation

4.259.4.1 source_channel_index `unsigned int MHAFilter::transfer_function_t::source_channel_index`

Source audio channel index for this transfer function.

4.259.4.2 target_channel_index `unsigned int MHAFilter::transfer_function_t::target_channel_index`

Target audio channel index for this transfer function.

4.259.4.3 impulse_response `std::vector<float> MHAFilter::transfer_function_t::impulse_response`

Impulse response of transfer from source to target channel.

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

- `mha_filter.hh`
- `mha_filter.cpp`

4.260 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. 939) 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. 939) method on every matrix member.

4.260.1 Detailed Description

A sparse matrix of transfer function partitions.

Each matrix element knows its position in the matrix, so they can be stored as a vector.

4.260.2 Member Function Documentation

```
4.260.2.1 partitions() std::valarray<unsigned int> MHAFilter::transfer_matrix_t<→
::partitions (
    unsigned fragsize ) const [inline]
```

Returns an array of the results of calling the **partitions()** (p. 939) method on every matrix member.

```
4.260.2.2 non_empty_partitions() std::valarray<unsigned int> MHAFilter::transfer←
_matrix_t::non_empty_partitions (
    unsigned int fragsize ) const [inline]
```

Returns an array of the results of calling the **non_empty_partitions()** (p. 939) method on every matrix member.

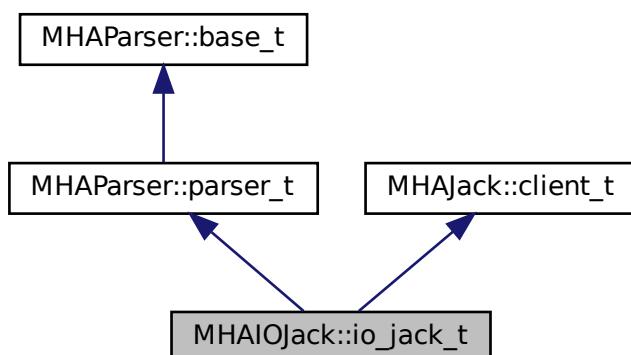
The documentation for this struct was generated from the following file:

- **mha_filter.hh**

4.261 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

4.261.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

4.261.2 Constructor & Destructor Documentation

```
4.261.2.1 io_jack_t() io_jack_t::io_jack_t (
    unsigned int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle )
```

4.261.3 Member Function Documentation

```
4.261.3.1 prepare() void io_jack_t::prepare (
    int nch_in,
    int nch_out )
```

Allocate buffers, activate JACK client and install internal ports.

```
4.261.3.2 release() void io_jack_t::release ( )
```

4.261.3.3 reconnect_inports() void io_jack_t::reconnect_inports () [private]

Connect the input ports when connection variable is accessed.

4.261.3.4 reconnect_outports() void io_jack_t::reconnect_outports () [private]

Connect the output ports when connection variable is accessed.

4.261.3.5 get_physical_input_ports() void io_jack_t::get_physical_input_ports () [private]**4.261.3.6 get_physical_output_ports()** void io_jack_t::get_physical_output_ports () [private]**4.261.3.7 get_all_input_ports()** void io_jack_t::get_all_input_ports () [private]**4.261.3.8 get_all_output_ports()** void io_jack_t::get_all_output_ports () [private]**4.261.3.9 get_delays_in()** void io_jack_t::get_delays_in () [private]**4.261.3.10 get_delays_out()** void io_jack_t::get_delays_out () [private]

4.261.3.11 `read_get_cpu_load()` void io_jack_t::read_get_cpu_load () [private]

4.261.3.12 `read_get_xruns()` void io_jack_t::read_get_xruns () [private]

4.261.3.13 `read_get_scheduler()` void io_jack_t::read_get_scheduler () [private]

4.261.4 Member Data Documentation

4.261.4.1 `fw_fragsize` unsigned int MHAIOJack::io_jack_t::fw_fragsize [private]

4.261.4.2 `fw_samplerate` float MHAIOJack::io_jack_t::fw_samplerate [private]

4.261.4.3 `servername` `MHAParser::string_t` MHAIOJack::io_jack_t::servername [private]

4.261.4.4 `clientname` `MHAParser::string_t` MHAIOJack::io_jack_t::clientname [private]

4.261.4.5 `connections_in` `MHAParser::vstring_t` MHAIOJack::io_jack_t::connections_in [private]

4.261.4.6 delays_in `MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_in` [private]

4.261.4.7 connections_out `MHAParser::vstring_t MHAIOJack::io_jack_t::connections_out` [private]

4.261.4.8 delays_out `MHAParser::vint_mon_t MHAIOJack::io_jack_t::delays_out` [private]

4.261.4.9 portnames_in `MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_in` [private]

4.261.4.10 portnames_out `MHAParser::vstring_t MHAIOJack::io_jack_t::portnames_out` [private]

4.261.4.11 ports_in_physical `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_physical` [private]

4.261.4.12 ports_out_physical `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_out_physical` [private]

4.261.4.13 ports_in_all `MHAParser::vstring_mon_t MHAIOJack::io_jack_t::ports_in_all` [private]

4.261.4.14 ports_out_all `MHAParser::vstring_mon_t` `MHAIOJack::io_jack_t::ports_out_all` [private]

4.261.4.15 ports_parser `MHAParser::parser_t` `MHAIOJack::io_jack_t::ports_parser` [private]

4.261.4.16 state_cupload `MHAParser::float_mon_t` `MHAIOJack::io_jack_t::state_cupload` [private]

4.261.4.17 state_xruns `MHAParser::int_mon_t` `MHAIOJack::io_jack_t::state_xruns` [private]

4.261.4.18 state_priority `MHAParser::int_mon_t` `MHAIOJack::io_jack_t::state_priority` [private]

4.261.4.19 state_scheduler `MHAParser::string_mon_t` `MHAIOJack::io_jack_t::state_scheduler` [private]

4.261.4.20 state_parser `MHAParser::parser_t` `MHAIOJack::io_jack_t::state_parser` [private]

4.261.4.21 patchbay `MHAEVENTS::patchbay_t< io_jack_t>` `MHAIOJack::io_jack_t::patchbay` [private]

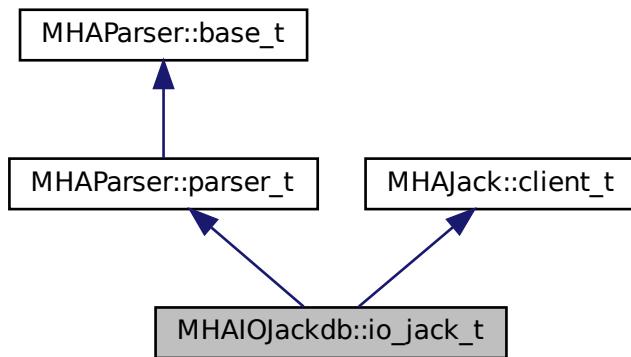
The documentation for this class was generated from the following file:

- `MHAIOJack.cpp`

4.262 MHAIOJackdb::io_jack_t Class Reference

Main class for JACK IO.

Inheritance diagram for MHAIOJackdb::io_jack_t:



Public Member Functions

- **io_jack_t** (unsigned int **fragsize**, float **samplerate**, **IOProcessEvent_t** **proc_event**, void * **proc_handle**, **IOStartedEvent_t** **start_event**, void * **start_handle**, **IOStoppedEvent_t** **stop_event**, void * **stop_handle**)
- void **prepare** (int, int)

Allocate buffers, activate JACK client and install internal ports.
- void **release** ()
- bool **fail_on_async_jackerror** () const

Private Member Functions

- int **IOProcessEvent_inner** (**mha_wave_t** ***sIn**, **mha_wave_t** ****sOut**)
- void **reconnect_inports** ()

Connect the input ports when connection variable is accessed.
- void **reconnect_outports** ()

Connect the output ports when connection variable is accessed.
- void **get_physical_input_ports** ()
- void **get_physical_output_ports** ()
- void **get_all_input_ports** ()
- void **get_all_output_ports** ()
- void **read_get_cpu_load** ()
- void **read_get_xruns** ()
- void **read_get_scheduler** ()
- void **set_use_jack_transport** ()
- void **set_locate** ()

Static Private Member Functions

- static int **IOProcessEvent_inner** (void *handle, **mha_wave_t** *sIn, **mha_wave_t** **sOut)

Private Attributes

- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- unsigned int **mha_fragsize**
- float **mha_samplerate**
- unsigned int **fragsize_ratio**
- **MHAParser::string_t servername**
- **MHAParser::string_t clientname**
- **MHAParser::vstring_t connections_in**
- **MHAParser::vstring_t connections_out**
- **MHAParser::vstring_t portnames_in**
- **MHAParser::vstring_t portnames_out**
- **MHAParser::bool_t fail_on_async_jackerr**
- **MHAParser::bool_t use_jack_transport**
- **MHAParser::float_t locate**
- **MHAParser::float_mon_t server_srate**
- **MHAParser::int_mon_t server_fragsize**
- **MHAParser::vstring_mon_t ports_in_physical**
- **MHAParser::vstring_mon_t ports_out_physical**
- **MHAParser::vstring_mon_t ports_in_all**
- **MHAParser::vstring_mon_t ports_out_all**
- **MHAParser::parser_t ports_parser**
- **MHAParser::float_mon_t state_cpupload**
- **MHAParser::int_mon_t state_xruns**
- **MHAParser::int_mon_t state_priority**
- **MHAParser::string_mon_t state_scheduler**
- **MHAParser::parser_t state_parser**
- **MHASignal::waveform_t * pwinner_out**
- **MHAEvents::patchbay_t< io_jack_t > patchbay**

Additional Inherited Members

4.262.1 Detailed Description

Main class for JACK IO.

This class registers a JACK client. JACK and framework states are managed by this class.

4.262.2 Constructor & Destructor Documentation

4.262.2.1 `io_jack_t()` `io_jack_t::io_jack_t (`
 `unsigned int fragsize,`
 `float samplerate,`
 `IOProcessEvent_t proc_event,`
 `void * proc_handle,`
 `IOStartedEvent_t start_event,`
 `void * start_handle,`
 `IOStoppedEvent_t stop_event,`
 `void * stop_handle)`

4.262.3 Member Function Documentation

4.262.3.1 `prepare()` `void io_jack_t::prepare (`
 `int nch_in,`
 `int nch_out)`

Allocate buffers, activate JACK client and install internal ports.

4.262.3.2 `release()` `void io_jack_t::release ()`

4.262.3.3 `fail_on_async_jackerror()` `bool MHAIOJackdb::io_jack_t::fail_on_async_`←
`jackerror () const [inline]`

4.262.3.4 `IOProcessEvent_inner()` [1/2] `int io_jack_t::IOProcessEvent_inner (`

```
    void * handle,  
    mha_wave_t * sIn,  
    mha_wave_t ** sOut ) [static], [private]
```

4.262.3.5 IOProcessEvent_inner() [2/2]

```
int io_jack_t::IOProcessEvent_inner (
```

`mha_wave_t * sIn,`
`mha_wave_t ** sOut) [private]`

4.262.3.6 reconnect_inports()

```
void io_jack_t::reconnect_inports ( ) [private]
```

Connect the input ports when connection variable is accessed.

4.262.3.7 reconnect_outports()

```
void io_jack_t::reconnect_outports ( ) [private]
```

Connect the output ports when connection variable is accessed.

4.262.3.8 get_physical_input_ports()

```
void io_jack_t::get_physical_input_ports ( ) [private]
```

4.262.3.9 get_physical_output_ports()

```
void io_jack_t::get_physical_output_ports ( ) [private]
```

4.262.3.10 get_all_input_ports()

```
void io_jack_t::get_all_input_ports ( ) [private]
```

4.262.3.11 get_all_output_ports()

```
void io_jack_t::get_all_output_ports ( ) [private]
```

4.262.3.12 read_get_cpu_load()

```
void io_jack_t::read_get_cpu_load ( ) [private]
```

4.262.3.13 `read_get_xruns()` void io_jack_t::read_get_xruns () [private]

4.262.3.14 `read_get_scheduler()` void io_jack_t::read_get_scheduler () [private]

4.262.3.15 `set_use_jack_transport()` void io_jack_t::set_use_jack_transport () [private]

4.262.3.16 `set_locate()` void io_jack_t::set_locate () [private]

4.262.4 Member Data Documentation

4.262.4.1 `proc_event` IOProcessEvent_t MHAIOJackdb::io_jack_t::proc_event [private]

4.262.4.2 `proc_handle` void* MHAIOJackdb::io_jack_t::proc_handle [private]

4.262.4.3 `mha_fragsize` unsigned int MHAIOJackdb::io_jack_t::mha_fragsize [private]

4.262.4.4 `mha_samplerate` float MHAIOJackdb::io_jack_t::mha_samplerate [private]

4.262.4.5 fragsize_ratio `unsigned int MHAIOJackdb::io_jack_t::fragsize_ratio [private]`

4.262.4.6 servername `MHAParser::string_t MHAIOJackdb::io_jack_t::servername [private]`

4.262.4.7 clientname `MHAParser::string_t MHAIOJackdb::io_jack_t::clientname [private]`

4.262.4.8 connections_in `MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_in [private]`

4.262.4.9 connections_out `MHAParser::vstring_t MHAIOJackdb::io_jack_t::connections_out [private]`

4.262.4.10 portnames_in `MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_in [private]`

4.262.4.11 portnames_out `MHAParser::vstring_t MHAIOJackdb::io_jack_t::portnames_out [private]`

4.262.4.12 fail_on_async_jackerr `MHAParser::bool_t MHAIOJackdb::io_jack_t::fail_on_async_jackerr [private]`

4.262.4.13 use_jack_transport `MHAParser::bool_t MHAIOJackdb::io_jack_t::use_←
jack_transport [private]`

4.262.4.14 locate `MHAParser::float_t MHAIOJackdb::io_jack_t::locate [private]`

4.262.4.15 server_srate `MHAParser::float_mon_t MHAIOJackdb::io_jack_t::server_←
srate [private]`

4.262.4.16 server_fragsize `MHAParser::int_mon_t MHAIOJackdb::io_jack_t::server_←
fragsize [private]`

4.262.4.17 ports_in_physical `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←
::ports_in_physical [private]`

4.262.4.18 ports_out_physical `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←
::ports_out_physical [private]`

4.262.4.19 ports_in_all `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←
in_all [private]`

4.262.4.20 ports_out_all `MHAParser::vstring_mon_t MHAIOJackdb::io_jack_t::ports_←
out_all [private]`

4.262.4.21 ports_parser `MHAParser::parser_t` MHAIOJackdb::io_jack_t::ports_parser
[private]

4.262.4.22 state_cupload `MHAParser::float_mon_t` MHAIOJackdb::io_jack_t::state_cupload
[private]

4.262.4.23 state_xruns `MHAParser::int_mon_t` MHAIOJackdb::io_jack_t::state_xruns
[private]

4.262.4.24 state_priority `MHAParser::int_mon_t` MHAIOJackdb::io_jack_t::state_priority
[private]

4.262.4.25 state_scheduler `MHAParser::string_mon_t` MHAIOJackdb::io_jack_t::state_scheduler
[private]

4.262.4.26 state_parser `MHAParser::parser_t` MHAIOJackdb::io_jack_t::state_parser
[private]

4.262.4.27 pwinner_out `MHASignal::waveform_t*` MHAIOJackdb::io_jack_t::pwinner_out
[private]

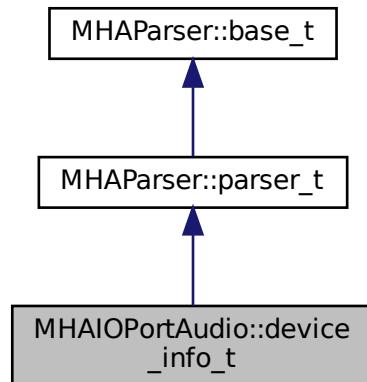
4.262.4.28 patchbay `MHAEVENTS::patchbay_t< io_jack_t>` MHAIOJackdb::io_jack_t::patchbay
[private]

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

- **MHAIOJackdb.cpp**

4.263 MHAIOPortAudio::device_info_t Class Reference

Inheritance diagram for MHAIOPortAudio::device_info_t:



Public Member Functions

- `device_info_t ()`
- `void fill_info ()`

Public Attributes

- `MHPParser::int_mon_t numDevices`
- `MHPParser::vint_mon_t structVersion`
- `MHPParser::vstring_mon_t name`
- `MHPParser::vint_mon_t hostApi`
- `MHPParser::vint_mon_t maxInputChannels`
- `MHPParser::vint_mon_t maxOutputChannels`
- `MHPParser::vfloat_mon_t defaultLowInputLatency`
- `MHPParser::vfloat_mon_t defaultLowOutputLatency`
- `MHPParser::vfloat_mon_t defaultHighInputLatency`
- `MHPParser::vfloat_mon_t defaultHighOutputLatency`
- `MHPParser::vfloat_mon_t defaultSampleRate`

Additional Inherited Members

4.263.1 Constructor & Destructor Documentation

4.263.1.1 device_info_t() MHAIOPortAudio::device_info_t::device_info_t () [inline]

4.263.2 Member Function Documentation

4.263.2.1 fill_info() void MHAIOPortAudio::device_info_t::fill_info () [inline]

4.263.3 Member Data Documentation

4.263.3.1 numDevices MHAParser::int_mon_t MHAIOPortAudio::device_info_t::numDevices

4.263.3.2 structVersion MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::structVersion

4.263.3.3 name MHAParser::vstring_mon_t MHAIOPortAudio::device_info_t::name

4.263.3.4 hostApi MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::hostApi

4.263.3.5 maxInputChannels MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxInputChannels

4.263.3.6 maxOutputChannels `MHAParser::vint_mon_t MHAIOPortAudio::device_info_t::maxOutputChannels`

4.263.3.7 defaultLowInputLatency `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowInputLatency`

4.263.3.8 defaultLowOutputLatency `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultLowOutputLatency`

4.263.3.9 defaultHighInputLatency `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighInputLatency`

4.263.3.10 defaultHighOutputLatency `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultHighOutputLatency`

4.263.3.11 defaultSampleRate `MHAParser::vfloat_mon_t MHAIOPortAudio::device_info_t::defaultSampleRate`

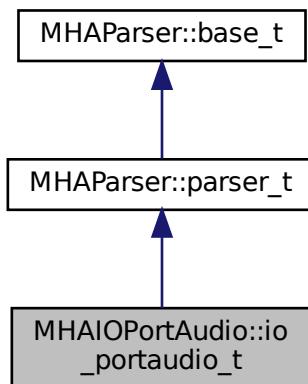
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

4.264 MHAIOPortAudio::io_portaudio_t Class Reference

Main class for Portaudio sound IO.

Inheritance diagram for MHAIOPortAudio::io_portaudio_t:



Public Member Functions

- `io_portaudio_t (unsigned int fragsize, float samplerate, IOProcessEvent_t proc_event, void * proc_handle, IOStartedEvent_t start_event, void * start_handle, IOStoppedEvent_t stop_event, void * stop_handle)`
- `void device_name_in_updated ()`
- `void device_name_out_updated ()`
- `void device_index_in_updated ()`
- `void device_index_out_updated ()`
- `~io_portaudio_t ()`
- `void cmd_prepare (int, int)`
- `void cmd_start ()`
- `void cmd_stop ()`
- `void cmd_release ()`
- `int portaudio_callback (const void *input, void *output, unsigned long frame_count, const PaStreamCallbackTimeInfo *time_info, PaStreamCallbackFlags status_flags)`

Private Attributes

- `device_info_t device_info`
- `MHASignal::waveform_t * s_in`
- `mha_wave_t * s_out`
- `float samplerate`
- `unsigned int nchannels_out`
- `unsigned int nchannels_in`
- `unsigned int fragsize`
- `IOProcessEvent_t proc_event`
- `void * proc_handle`
- `IOStartedEvent_t start_event`
- `void * start_handle`
- `IOStoppedEvent_t stop_event`
- `void * stop_handle`
- `PaStream * portaudio_stream`
- `MHAParser::string_t device_name_in`
- `MHAParser::int_t device_index_in`
- `MHAParser::string_t device_name_out`
- `MHAParser::int_t device_index_out`
- `MHAEvents::patchbay_t< io_portaudio_t > patchbay`

Additional Inherited Members

4.264.1 Detailed Description

Main class for Portaudio sound IO.

4.264.2 Constructor & Destructor Documentation

```
4.264.2.1 io_portaudio_t() MHAIOPortAudio::io_portaudio_t::io_portaudio_t (
    unsigned int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle ) [inline]
```

4.264.2.2 ~io_portaudio_t() MHAIOPortAudio::io_portaudio_t::~io_portaudio_t ()
[inline]

4.264.3 Member Function Documentation

4.264.3.1 device_name_in_updated() void MHAIOPortAudio::io_portaudio_t::device_name_in_updated () [inline]

4.264.3.2 device_name_out_updated() void MHAIOPortAudio::io_portaudio_t::device_name_out_updated () [inline]

4.264.3.3 device_index_in_updated() void MHAIOPortAudio::io_portaudio_t::device_index_in_updated () [inline]

4.264.3.4 device_index_out_updated() void MHAIOPortAudio::io_portaudio_t::device_index_out_updated () [inline]

4.264.3.5 cmd_prepare() void MHAIOPortAudio::io_portaudio_t::cmd_prepare (int *nchannels_in*, int *nchannels_out*)

4.264.3.6 cmd_start() void MHAIOPortAudio::io_portaudio_t::cmd_start ()

4.264.3.7 cmd_stop() void MHAIOPortAudio::io_portaudio_t::cmd_stop ()

4.264.3.8 cmd_release() void MHAIOPortAudio::io_portaudio_t::cmd_release ()

4.264.3.9 portaudio_callback() int MHAIOPortAudio::io_portaudio_t::portaudio_callback (

```
    const void * input,
    void * output,
    unsigned long frame_count,
    const PaStreamCallbackTimeInfo * time_info,
    PaStreamCallbackFlags status_flags )
```

4.264.4 Member Data Documentation

4.264.4.1 device_info device_info_t MHAIOPortAudio::io_portaudio_t::device_info [private]

4.264.4.2 s_in MHASignal::waveform_t* MHAIOPortAudio::io_portaudio_t::s_in [private]

4.264.4.3 s_out mha_wave_t* MHAIOPortAudio::io_portaudio_t::s_out [private]

4.264.4.4 samplerate float MHAIOPortAudio::io_portaudio_t::samplerate [private]

4.264.4.5 nchannels_out unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_←
out [private]

4.264.4.6 nchannels_in unsigned int MHAIOPortAudio::io_portaudio_t::nchannels_in
[private]

4.264.4.7 fragsize unsigned int MHAIOPortAudio::io_portaudio_t::fragsize [private]

4.264.4.8 proc_event IOProcessEvent_t MHAIOPortAudio::io_portaudio_t::proc_event
[private]

4.264.4.9 proc_handle void* MHAIOPortAudio::io_portaudio_t::proc_handle [private]

4.264.4.10 start_event IOStartedEvent_t MHAIOPortAudio::io_portaudio_t::start_←
event [private]

4.264.4.11 start_handle void* MHAIOPortAudio::io_portaudio_t::start_handle [private]

4.264.4.12 stop_event IOStoppedEvent_t MHAIOPortAudio::io_portaudio_t::stop_event
[private]

4.264.4.13 stop_handle void* MHAIOPortAudio::io_portaudio_t::stop_handle [private]

4.264.4.14 portaudio_stream PaStream* MHAIOPortAudio::io_portaudio_t::portaudio_stream [private]

4.264.4.15 device_name_in MHAParser::string_t MHAIOPortAudio::io_portaudio_t::device_name_in [private]

4.264.4.16 device_index_in MHAParser::int_t MHAIOPortAudio::io_portaudio_t::device_index_in [private]

4.264.4.17 device_name_out MHAParser::string_t MHAIOPortAudio::io_portaudio_t::device_name_out [private]

4.264.4.18 device_index_out MHAParser::int_t MHAIOPortAudio::io_portaudio_t::device_index_out [private]

4.264.4.19 patchbay MHAEvents::patchbay_t< io_portaudio_t> MHAIOPortAudio::io_portaudio_t::patchbay [private]

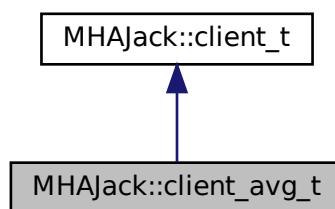
The documentation for this class was generated from the following file:

- **MHAIOPortAudio.cpp**

4.265 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 * **frag-size**=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

4.265.1 Detailed Description

Generic JACK client for averaging a system response across time.

4.265.2 Constructor & Destructor Documentation

4.265.2.1 `client_avg_t()` MHAJack::client_avg_t::client_avg_t (

```
const std::string & name_,  
const unsigned int & nrep_ )
```

Constructor for averaging client.

Parameters

<i>name</i> ↵	Name of JACK client
<i>nrep</i> ↵	Number of repetitions

4.265.3 Member Function Documentation

```
4.265.3.1 io() void MHAJack::client_avg_t::io (
    mha_wave_t * is_out,
    mha_wave_t * is_in,
    const std::vector< std::string > & p_out,
    const std::vector< std::string > & p_in,
    float * srate = NULL,
    unsigned int * fragsize = NULL )
```

Recording function.

long-description

Parameters

<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

```
4.265.3.2 proc() [1/2] int MHAJack::client_avg_t::proc (
    void * handle,
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [static], [private]
```

```
4.265.3.3 IOStoppedEvent() [1/2] void MHAJack::client_avg_t::IOStoppedEvent (
    void * handle,
    int proc_err,
    int io_err ) [static], [private]
```

```
4.265.3.4 proc() [2/2] void MHAJack::client_avg_t::proc (
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [private]
```

4.265.3.5 IOStoppedEvent() [2/2] void MHAJack::client_avg_t::IOStoppedEvent ()
[private]

4.265.4 Member Data Documentation

4.265.4.1 b_stopped bool MHAJack::client_avg_t::b_stopped [private]

4.265.4.2 pos unsigned int MHAJack::client_avg_t::pos [private]

4.265.4.3 sn_in mha_wave_t* MHAJack::client_avg_t::sn_in [private]

4.265.4.4 sn_out mha_wave_t* MHAJack::client_avg_t::sn_out [private]

4.265.4.5 name std::string MHAJack::client_avg_t::name [private]

4.265.4.6 frag_out MHASignal::waveform_t* MHAJack::client_avg_t::frag_out [private]

4.265.4.7 nrep const unsigned int MHAJack::client_avg_t::nrep [private]

4.265.4.8 n unsigned int MHAJack::client_avg_t::n [private]

4.265.4.9 b_ready bool MHAJack::client_avg_t::b_ready [private]

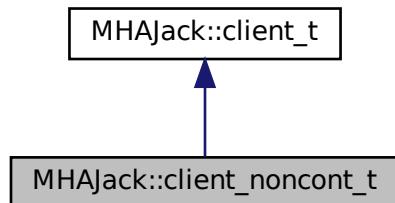
The documentation for this class was generated from the following files:

- **mhajack.h**
- **mhajack.cpp**

4.266 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 * **frag-size**=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

4.266.1 Detailed Description

Generic client for synchronous playback and recording of waveform fragments.

4.266.2 Constructor & Destructor Documentation

4.266.2.1 **client_noncont_t()** MHAJack::client_noncont_t::client_noncont_t (

```
const std::string & name,
bool use_jack_transport = false )
```

4.266.3 Member Function Documentation

4.266.3.1 **io()** void MHAJack::client_noncont_t::io (

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

4.266.3.2 proc() [1/2] int MHAJack::client_noncont_t::proc (void * *handle*,
 mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [static], [private]

4.266.3.3 IOStoppedEvent() [1/2] void MHAJack::client_noncont_t::IOStoppedEvent (void * *handle*,
 int *proc_err*,
 int *io_err*) [static], [private]

4.266.3.4 proc() [2/2] void MHAJack::client_noncont_t::proc (mha_wave_t * *sIn*,
 mha_wave_t ** *sOut*) [private]

4.266.3.5 IOStoppedEvent() [2/2] void MHAJack::client_noncont_t::IOStoppedEvent () [private]

4.266.4 Member Data Documentation

4.266.4.1 b_stopped bool MHAJack::client_noncont_t::b_stopped [private]

4.266.4.2 pos unsigned int MHAJack::client_noncont_t::pos [private]

4.266.4.3 sn_in mha_wave_t* MHAJack::client_noncont_t::sn_in [private]

4.266.4.4 sn_out `mha_wave_t*` `MHAJack::client_noncont_t::sn_out` [private]

4.266.4.5 name `std::string` `MHAJack::client_noncont_t::name` [private]

4.266.4.6 frag_out `MHASignal::waveform_t*` `MHAJack::client_noncont_t::frag_out` [private]

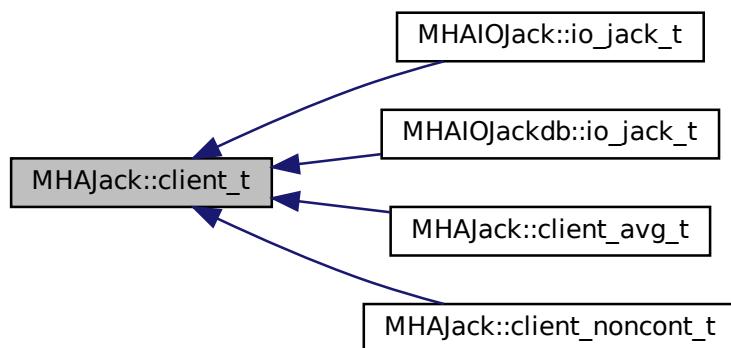
The documentation for this class was generated from the following files:

- `mhajack.h`
- `mhajack.cpp`

4.267 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)`
- `bool is_prepared () const`

Protected Attributes

- `jack_client_t * jc`

Private Member Functions

- `void prepare_impl (const char *server_name, const char *client_name, const unsigned int & nchannels_in, const unsigned int & nchannels_out)`
Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.
- `void internal_start ()`
- `void internal_stop ()`
- `void stopped (int, int)`
- `int jack_proc_cb (jack_nframes_t)`
This is the main processing callback.
- `int jack_xrun_cb ()`

Static Private Member Functions

- static int **jack_proc_cb** (jack_nframes_t, void *)
- static int **jack_xrun_cb** (void *)

Private Attributes

- unsigned long **num_xruns**
- unsigned int **fragsize**
- float **samplerate**
- unsigned int **nchannels_in**
- unsigned int **nchannels_out**
- **IOProcessEvent_t proc_event**
- void * **proc_handle**
- **IOStartedEvent_t start_event**
- void * **start_handle**
- **IOStoppedEvent_t stop_event**
- void * **stop_handle**
- **MHASignal::waveform_t * s_in**
- **mha_wave_t * s_out**
- **MHAJack::port_t ** inch**
- **MHAJack::port_t ** outch**
- unsigned int **flags**
- bool **b_prepared**
- bool **use_jack_transport**
- jack_transport_state_t **jstate_prev**
- std::vector< std::string > **input_portnames**
- std::vector< std::string > **output_portnames**
- bool **fail_on_async_jackerror**

4.267.1 Detailed Description

Generic asynchronous JACK client.

4.267.2 Constructor & Destructor Documentation

```
4.267.2.1 client_t() MHAJack::client_t::client_t (
    IOProcessEvent_t proc_event,
    void * proc_handle = NULL,
    IOStartedEvent_t start_event = NULL,
    void * start_handle = NULL,
    IOStoppedEvent_t stop_event = NULL,
    void * stop_handle = NULL,
    bool use_jack_transport = false )
```

4.267.3 Member Function Documentation

4.267.3.1 prepare() [1/2] void MHAJack::client_t::prepare (

const std::string & client_name,
const unsigned int & nch_in,
const unsigned int & nch_out)

Allocate buffers, activate JACK client and install internal ports.

Registers the jack client with the default jack server and activates it.

Parameters

<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

4.267.3.2 prepare() [2/2] void MHAJack::client_t::prepare (

const std::string & server_name,
const std::string & client_name,
const unsigned int & nch_in,
const unsigned int & nch_out)

Allocate buffers, ports, and activates JACK client.

Registers the jack client with specified jack server and activates it.

Parameters

<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

4.267.3.3 release() void MHAJack::client_t::release ()

Remove JACK client and deallocate internal ports and buffers.

4.267.3.4 start() void MHAJack::client_t::start (
 bool fail_on_async_jack_error = true)**4.267.3.5 stop()** void MHAJack::client_t::stop ()**4.267.3.6 connect_input()** void MHAJack::client_t::connect_input (
 const std::vector< std::string > & con)

Connect the input ports when connection variable is accessed.

4.267.3.7 connect_output() void MHAJack::client_t::connect_output (
 const std::vector< std::string > & con)

Connect the output ports when connection variable is accessed.

4.267.3.8 get_fragsize() unsigned int MHAJack::client_t::get_fragsize () const
[inline]**4.267.3.9 get_srate()** float MHAJack::client_t::get_srate () const [inline]

4.267.3.10 get_xruns() unsigned long MHAJack::client_t::get_xruns () [inline]

4.267.3.11 get_xruns_reset() unsigned long MHAJack::client_t::get_xruns_reset ()

4.267.3.12 str_error() std::string MHAJack::client_t::str_error (int err)

4.267.3.13 get_ports() void MHAJack::client_t::get_ports (std::vector< std::string > & res, unsigned long jack_flags)

Get a list of Jack ports.

Parameters

<i>res</i>	Result string vector
<i>jack_flags</i>	Jack port flags (JackPortInput etc.)

4.267.3.14 get_my_input_ports() std::vector< std::string > MHAJack::client_t::get_my_input_ports ()

4.267.3.15 get_my_output_ports() std::vector< std::string > MHAJack::client_t::get_my_output_ports ()

4.267.3.16 set_input_portnames() void MHAJack::client_t::set_input_portnames (const std::vector< std::string > & names)

4.267.3.17 set_output_portnames() void MHAJack::client_t::set_output_portnames (const std::vector< std::string > & names)

4.267.3.18 get_cpu_load() float MHAJack::client_t::get_cpu_load ()

4.267.3.19 set_use_jack_transport() void MHAJack::client_t::set_use_jack_transport (bool ut) [inline]

4.267.3.20 is_prepared() bool MHAJack::client_t::is_prepared () const [inline]

4.267.3.21 prepare_impl() void MHAJack::client_t::prepare_impl (const char * server_name, const char * client_name, const unsigned int & nch_in, const unsigned int & nch_out) [private]

Allocate buffers, activate JACK client and allocates jack ports Registers the jack client with the given server and activates it.

Parameters

<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

4.267.3.22 internal_start() void MHAJack::client_t::internal_start () [private]

4.267.3.23 internal_stop() void MHAJack::client_t::internal_stop () [private]

4.267.3.24 stopped() void MHAJack::client_t::stopped (int *proc_err*, int *io_err*) [private]

4.267.3.25 jack_proc_cb() [1/2] int MHAJack::client_t::jack_proc_cb (jack_nframes_t *n*, void * *h*) [static], [private]

4.267.3.26 jack_proc_cb() [2/2] int MHAJack::client_t::jack_proc_cb (jack_nframes_t *n*) [private]

This is the main processing callback.

Here happens double buffering and downsampling.

4.267.3.27 jack_xrun_cb() [1/2] int MHAJack::client_t::jack_xrun_cb (void * *h*) [static], [private]

4.267.3.28 jack_xrun_cb() [2/2] int MHAJack::client_t::jack_xrun_cb () [inline], [private]

4.267.4 Member Data Documentation

4.267.4.1 num_xruns unsigned long MHAJack::client_t::num_xruns [private]

4.267.4.2 fragsize unsigned int MHAJack::client_t::fragsize [private]

4.267.4.3 samplerate float MHAJack::client_t::samplerate [private]

4.267.4.4 nchannels_in unsigned int MHAJack::client_t::nchannels_in [private]

4.267.4.5 nchannels_out unsigned int MHAJack::client_t::nchannels_out [private]

4.267.4.6 proc_event IOProcessEvent_t MHAJack::client_t::proc_event [private]

4.267.4.7 proc_handle void* MHAJack::client_t::proc_handle [private]

4.267.4.8 start_event IOStartedEvent_t MHAJack::client_t::start_event [private]

4.267.4.9 start_handle void* MHAJack::client_t::start_handle [private]

4.267.4.10 stop_event IOStoppedEvent_t MHAJack::client_t::stop_event [private]

4.267.4.11 stop_handle void* MHAJack::client_t::stop_handle [private]

4.267.4.12 s_in MHASignal::waveform_t* MHAJack::client_t::s_in [private]

4.267.4.13 s_out mha_wave_t* MHAJack::client_t::s_out [private]

4.267.4.14 inch MHAJack::port_t** MHAJack::client_t::inch [private]

4.267.4.15 outch MHAJack::port_t** MHAJack::client_t::outch [private]

4.267.4.16 jc jack_client_t* MHAJack::client_t::jc [protected]

4.267.4.17 flags unsigned int MHAJack::client_t::flags [private]

4.267.4.18 b_prepared bool MHAJack::client_t::b_prepared [private]

4.267.4.19 use_jack_transport bool MHAJack::client_t::use_jack_transport [private]

4.267.4.20 jstate_prev jack_transport_state_t MHAJack::client_t::jstate_prev [private]

4.267.4.21 input_portnames std::vector<std::string> MHAJack::client_t::input_portnames [private]

4.267.4.22 output_portnames std::vector<std::string> MHAJack::client_t::output_portnames [private]

4.267.4.23 fail_on_async_jackerror bool MHAJack::client_t::fail_on_async_jackerror [private]

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

- **mhajack.h**
- **mhajack.cpp**

4.268 MHAJack::port_t Class Reference

Class for one channel/port.

Public Types

- enum **dir_t** { **input**, **output** }

Public Member Functions

- **port_t** (jack_client_t * **jc**, **dir_t** **dir**, int **id**)
- **port_t** (jack_client_t * **jc**, **dir_t** **dir**, const std::string &**id**)
Constructor to create port with specific name.
- **~port_t** ()
- void **read** (**mha_wave_t** ***s**, unsigned int **ch**)
- void **write** (**mha_wave_t** ***s**, unsigned int **ch**)
- void **mute** (unsigned int **n**)
- void **connect_to** (const char ***pn**)
- const char * **get_short_name** ()
Return the port name.

Private Attributes

- **dir_t dir_type**
- **jack_port_t * port**
- **jack_default_audio_sample_t * iob**
- **jack_client_t * jc**

4.268.1 Detailed Description

Class for one channel/port.

This class represents one JACK port. Double buffering for asynchronous process callbacks is managed by this class.

4.268.2 Member Enumeration Documentation

4.268.2.1 **dir_t** enum MHAJack::port_t::dir_t

Enumerator

input	
output	

4.268.3 Constructor & Destructor Documentation

4.268.3.1 **port_t()** [1/2] 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).

4.268.3.2 port_t() [2/2]

```
MHAJack::port_t::port_t (
    jack_client_t * jc,
    dir_t dir,
    const std::string & id )
```

Constructor to create port with specific name.

Parameters

<i>jc</i>	JACK client.
<i>dir</i>	Direction (input/output).
<i>id</i>	Port name.

4.268.3.3 ~port_t()

MHAJack::port_t::~port_t ()

4.268.4 Member Function Documentation

4.268.4.1 read()

```
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.

4.268.4.2 write()

```
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.

4.268.4.3 mute() 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).
----------	---

4.268.4.4 connect_to() void MHAJack::port_t::connect_to (const char * *pn*)

Parameters

<i>pn</i>	Port name to connect to
-----------	-------------------------

4.268.4.5 get_short_name() const char * MHAJack::port_t::get_short_name ()

Return the port name.

4.268.5 Member Data Documentation

4.268.5.1 dir_type *dir_t* MHAJack::port_t::dir_type [private]

4.268.5.2 port jack_port_t* MHAJack::port_t::port [private]

4.268.5.3 iob jack_default_audio_sample_t* MHAJack::port_t::iob [private]

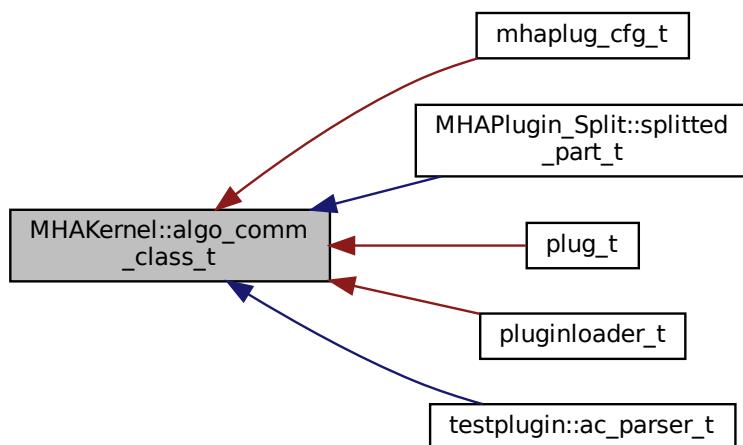
4.268.5.4 jc jack_client_t* MHAJack::port_t::jc [private]

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

- **mhajack.h**
- **mhajack.cpp**

4.269 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_vfloat(void *handle, const char *name, std::vector< float > &ivar)`
- static int `insert_var_float(void *, const char *, float *)`
- static int `insert_var_double(void *, const char *, double *)`
- 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_var_double(void *, const char *, double *)`
- 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`

4.269.1 Constructor & Destructor Documentation

4.269.1.1 algo_comm_class_t() MHAKernel::algo_comm_class_t::algo_comm_class_t ()

4.269.1.2 ~algo_comm_class_t() MHAKernel::algo_comm_class_t::~algo_comm_class_t () [virtual]

4.269.2 Member Function Documentation

4.269.2.1 get_c_handle() algo_comm_t MHAKernel::algo_comm_class_t::get_c_handle ()

4.269.2.2 insert_var() int MHAKernel::algo_comm_class_t::insert_var (void * handle,
const char * name,
comm_var_t var) [static]

4.269.2.3 insert_var_int() int MHAKernel::algo_comm_class_t::insert_var_int (void * handle,
const char * name,
int * ivar) [static]

4.269.2.4 insert_var_vfloat() int MHAKernel::algo_comm_class_t::insert_var_vfloat (void * handle,
const char * name,
std::vector< float > & ivar) [static]

4.269.2.5 insert_var_float() int MHAKernel::algo_comm_class_t::insert_var_float (void * *handle*, const char * *name*, float * *ivar*) [static]

4.269.2.6 insert_var_double() int MHAKernel::algo_comm_class_t::insert_var_double (void * *handle*, const char * *name*, double * *ivar*) [static]

4.269.2.7 remove_var() int MHAKernel::algo_comm_class_t::remove_var (void * *handle*, const char * *name*) [static]

4.269.2.8 remove_ref() int MHAKernel::algo_comm_class_t::remove_ref (void * *handle*, void * *ref*) [static]

4.269.2.9 is_var() int MHAKernel::algo_comm_class_t::is_var (void * *handle*, const char * *name*) [static]

4.269.2.10 get_var() int MHAKernel::algo_comm_class_t::get_var (void * *handle*, const char * *name*, comm_var_t * *var*) [static]

4.269.2.11 `get_var_int()` `int MHAKernel::algo_comm_class_t::get_var_int (void * handle,`
`const char * name,`
`int * ivar) [static]`

4.269.2.12 `get_var_float()` `int MHAKernel::algo_comm_class_t::get_var_float (void * handle,`
`const char * name,`
`float * ivar) [static]`

4.269.2.13 `get_var_double()` `int MHAKernel::algo_comm_class_t::get_var_double (void * handle,`
`const char * name,`
`double * ivar) [static]`

4.269.2.14 `get_entries()` `int MHAKernel::algo_comm_class_t::get_entries (void * handle,`
`char * ret,`
`unsigned int len) [static]`

4.269.2.15 `get_error()` `const char * MHAKernel::algo_comm_class_t::get_error (int e) [static]`

4.269.2.16 `local_insert_var()` `void MHAKernel::algo_comm_class_t::local_insert_var (const char * name,`
`comm_var_t var) [virtual]`

4.269.2.17 `local_remove_var()` `void MHAKernel::algo_comm_class_t::local_remove_var (const char * name) [virtual]`

4.269.2.18 local_remove_ref() void MHAKernel::algo_comm_class_t::local_remove_ref (void * *addr*) [virtual]

4.269.2.19 local_is_var() bool MHAKernel::algo_comm_class_t::local_is_var (const char * *name*) [virtual]

4.269.2.20 local_get_var() void MHAKernel::algo_comm_class_t::local_get_var (const char * *name*, comm_var_t * *var*) [virtual]

4.269.2.21 local_get_entries() std::string MHAKernel::algo_comm_class_t::local_get_entries () [virtual]

4.269.2.22 size() MHAKernel::comm_var_map_t::size_type MHAKernel::algo_comm_class_t::size () const [virtual]

4.269.3 Member Data Documentation

4.269.3.1 algo_comm_id_string char* MHAKernel::algo_comm_class_t::algo_comm_id_string

4.269.3.2 ac algo_comm_t MHAKernel::algo_comm_class_t::ac [private]

4.269.3.3 algo_comm_id_string_len int MHAKernel::algo_comm_class_t::algo_comm_id_string_len [private]

4.269.3.4 vars 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**

4.270 MHAKernel::comm_var_map_t Class Reference

Inherits map< std::string, comm_var_t >.

Public Member Functions

- bool **has_key** (const std::string &name)

4.270.1 Member Function Documentation

4.270.1.1 has_key() bool MHAKernel::comm_var_map_t::has_key (const std::string & name) [inline]

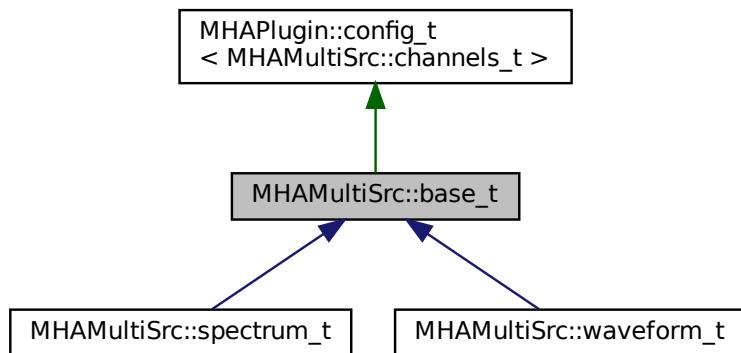
The documentation for this class was generated from the following file:

- **mha_algo_comm.hh**

4.271 MHAMultiSrc::base_t Class Reference

Base class for source selection.

Inheritance diagram for MHAMultiSrc::base_t:



Public Member Functions

- **base_t (algo_comm_t iac)**
- void **select_source** (const std::vector< std::string > &src, int in_channels)
Change the selection of input sources.

Protected Attributes

- **algo_comm_t ac**

Additional Inherited Members

4.271.1 Detailed Description

Base class for source selection.

See also

- MHAMultiSrc::channel_t (p. 993)**
- MHAMultiSrc::channels_t (p. 993)**

4.271.2 Constructor & Destructor Documentation

4.271.2.1 `base_t()` `MHAMultiSrc::base_t::base_t (algo_comm_t iac)`

4.271.3 Member Function Documentation

4.271.3.1 `select_source()` `void MHAMultiSrc::base_t::select_source (const std::vector< std::string > & src, int in_channels)`

Change the selection of input sources.

This function is real-time and thread safe.

Parameters

<code>src</code>	List of input sources
<code>in_channels</code>	Number of input channels in direct input (the processed signal)

4.271.4 Member Data Documentation

4.271.4.1 `ac algo_comm_t MHAMultiSrc::base_t::ac [protected]`

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

- `mha_multisrc.h`
- `mha_multisrc.cpp`

4.272 MHAMultiSrc::channel_t Class Reference

Public Attributes

- std::string **name**
- int **channel**

4.272.1 Member Data Documentation

4.272.1.1 **name** std::string MHAMultiSrc::channel_t::name

4.272.1.2 **channel** int MHAMultiSrc::channel_t::channel

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

- **mha_multisrc.h**

4.273 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.

4.273.1 Constructor & Destructor Documentation

4.273.1.1 **channels_t()** MHAMultiSrc::channels_t::channels_t (

```
const std::vector< std::string > & route,
int in_channels )
```

Separate a list of input sources into a parsable channel list.

The number of input channels if verified, a list of **MHAMultiSrc::channel_t** (p. 993) is filled.

Parameters

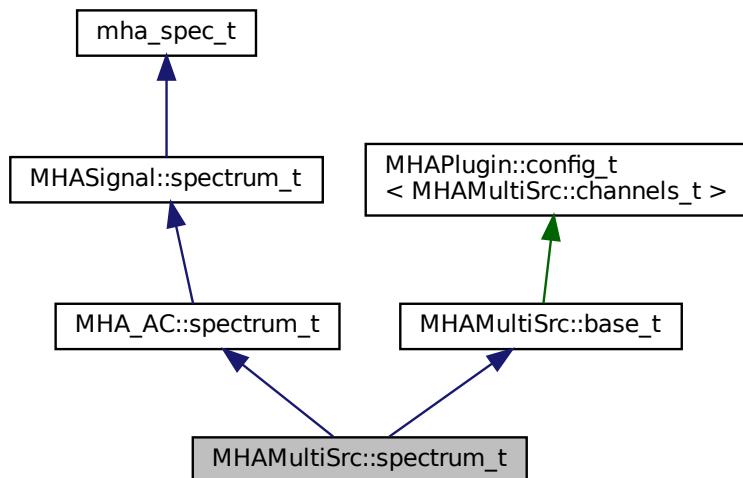
<i>route</i>	vector of source channel ids
<i>in_channels</i>	number of channels in the processed input signal

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

4.274 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

4.274.1 Constructor & Destructor Documentation

```
4.274.1.1 spectrum_t() MHAMultiSrc::spectrum_t::spectrum_t (
    algo_comm_t iac,
    std::string name,
    unsigned int frames,
    unsigned int channels )
```

4.274.2 Member Function Documentation

```
4.274.2.1 update() mha_spec_t * MHAMultiSrc::spectrum_t::update (
    mha_spec_t * s )
```

Update data of spectrum to hold actual input data.

Parameters

<code>s</code>	Input signal chunk
----------------	--------------------

Returns

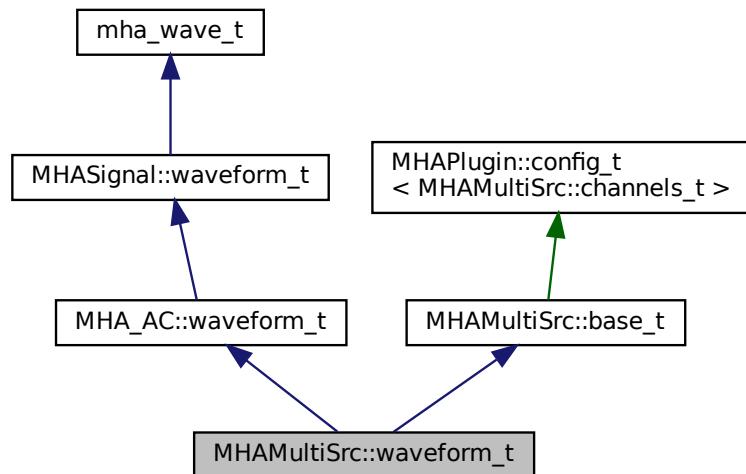
Return pointer to spectrum structure

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

- **mha_multisrc.h**
- **mha_multisrc.cpp**

4.275 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

4.275.1 Constructor & Destructor Documentation

4.275.1.1 waveform_t() MHAMultiSrc::waveform_t::waveform_t (

```

algo_comm_t iac,
std::string name,
unsigned int frames,
unsigned int channels )
  
```

4.275.2 Member Function Documentation

4.275.2.1 update() `mha_wave_t * MHAMultiSrc::waveform_t::update (mha_wave_t * s)`

Update data of waveform to hold actual input data.

Parameters

<code>s</code>	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`

4.276 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`

4.276.1 Member Data Documentation

4.276.1.1 cf_l `mha_real_t` `MHAOvlFilter::band_descriptor_t::cf_l`

4.276.1.2 ef_l `mha_real_t` `MHAOvlFilter::band_descriptor_t::ef_l`

4.276.1.3 cf `mha_real_t` `MHAOvlFilter::band_descriptor_t::cf`

4.276.1.4 ef_h `mha_real_t` `MHAOvlFilter::band_descriptor_t::ef_h`

4.276.1.5 cf_h `mha_real_t` `MHAOvlFilter::band_descriptor_t::cf_h`

4.276.1.6 low_side_flat `bool` `MHAOvlFilter::band_descriptor_t::low_side_flat`

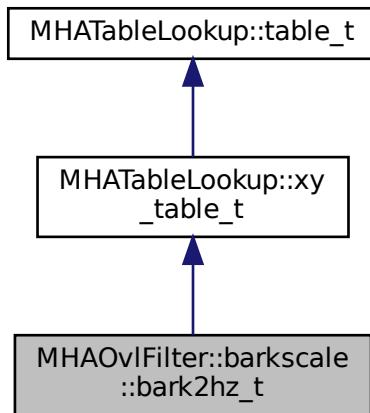
4.276.1.7 high_side_flat `bool` `MHAOvlFilter::band_descriptor_t::high_side_flat`

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

- `mha_fffb.hh`

4.277 MHAOvlFilter::barkscale::bark2hz_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::bark2hz_t:



Public Member Functions

- `bark2hz_t()`
- `~bark2hz_t()`

Additional Inherited Members

4.277.1 Constructor & Destructor Documentation

4.277.1.1 `bark2hz_t()` MHAOvlFilter::barkscale::bark2hz_t::bark2hz_t ()

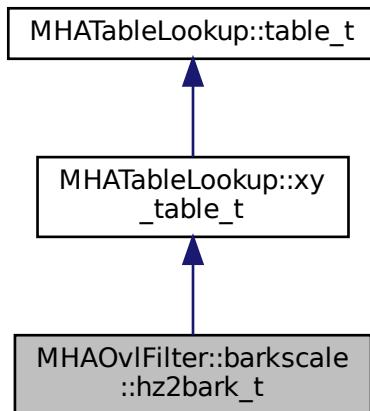
4.277.1.2 `~bark2hz_t()` MHAOvlFilter::barkscale::bark2hz_t::~bark2hz_t ()

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

- `mha_fffb.cpp`

4.278 MHAOvlFilter::barkscale::hz2bark_t Class Reference

Inheritance diagram for MHAOvlFilter::barkscale::hz2bark_t:



Public Member Functions

- **hz2bark_t ()**
- **~hz2bark_t ()**

Additional Inherited Members

4.278.1 Constructor & Destructor Documentation

4.278.1.1 **hz2bark_t()** MHAOvlFilter::barkscale::hz2bark_t::hz2bark_t ()

4.278.1.2 **~hz2bark_t()** MHAOvlFilter::barkscale::hz2bark_t::~hz2bark_t ()

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

- **mha_fffb.cpp**

4.279 MHAOvlFilter::fftfb_ac_info_t Class Reference

Public Member Functions

- `fftfb_ac_info_t (const MHAOvlFilter::fftfb_t &fb, algo_comm_t ac, const std::string &prefix)`
- `void insert ()`

Private Attributes

- **MHA_AC::waveform_t cfv**
vector of nominal center frequencies / Hz
- **MHA_AC::waveform_t efv**
vector of edge frequencies / Hz
- **MHA_AC::waveform_t bwv**
vector of band-weights (sum of squared fft-bin-weights)/num_frames
- **MHA_AC::waveform_t cLTASS**
vector of LTASS correction

4.279.1 Constructor & Destructor Documentation

```
4.279.1.1 ffftb_ac_info_t() MHAOvlFilter::fftfb_ac_info_t::fftfb_ac_info_t (
    const MHAOvlFilter::fftfb_t & fb,
    algo_comm_t ac,
    const std::string & prefix )
```

4.279.2 Member Function Documentation

```
4.279.2.1 insert() void MHAOvlFilter::fftfb_ac_info_t::insert ( )
```

4.279.3 Member Data Documentation

4.279.3.1 cfv MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cfv [private]

vector of nominal center frequencies / Hz

4.279.3.2 efv MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::efv [private]

vector of edge frequencies / Hz

4.279.3.3 bwv MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::bwv [private]

vector of band-weights (sum of squared fft-bin-weights)/num_frames

4.279.3.4 cLTASS MHA_AC::waveform_t MHAOvlFilter::fftfb_ac_info_t::cLTASS [private]

vector of LTASS correction

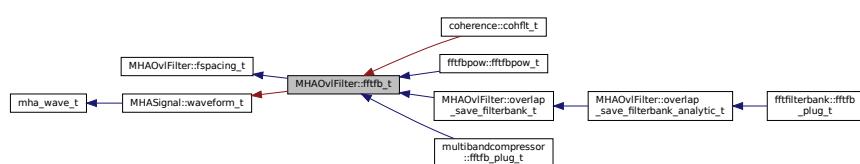
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

4.280 MHAOvlFilter::fftfb_t Class Reference

FFT based overlapping filter bank.

Inheritance diagram for MHAOvlFilter::fftfb_t:



Public Member Functions

- **fftfb_t** (**MHAOvlFilter::fftfb_vars_t** &par, unsigned int nfft, **mha_real_t** fs)
Constructor for a FFT-based overlapping filter bank.
- **~fftfb_t** ()
- void **apply_gains** (**mha_spec_t** *s_out, const **mha_spec_t** *s_in, const **mha_wave_t** *gains)
- void **get_fbpower** (**mha_wave_t** *fbpow, const **mha_spec_t** *s_in)
- void **get_fbpower_db** (**mha_wave_t** *fbpow, const **mha_spec_t** *s_in)
- std::vector< **mha_real_t** > **get_ltass_gain_db** () const
- unsigned int **bin1** (unsigned int band) const
Return index of first non-zero filter shape window.
- unsigned int **bin2** (unsigned int band) const
Return index of first zero filter shape window above center frequency.
- unsigned int **get_ffflen** () 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 **ffflen**
- **mha_real_t** **samplingrate**

Additional Inherited Members

4.280.1 Detailed Description

FFT based overlapping filter bank.

4.280.2 Constructor & Destructor Documentation

```
4.280.2.1 fftfb_t() MHAOvlFilter::fftfb_t::fftfb_t (
    MHAOvlFilter::fftfb_vars_t &par,
    unsigned int nfft,
    mha_real_t fs )
```

Constructor for a FFT-based overlapping filter bank.

Parameters

<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

4.280.2.2 ~fftfb_t() `MHAOvlFilter::fftfb_t::~fftfb_t ()`

4.280.3 Member Function Documentation

4.280.3.1 apply_gains() `void MHAOvlFilter::fftfb_t::apply_gains (`
 `mha_spec_t * s_out,`
 `const mha_spec_t * s_in,`
 `const mha_wave_t * gains)`

4.280.3.2 get_fbpower() `void MHAOvlFilter::fftfb_t::get_fbpower (`
 `mha_wave_t * fbpow,`
 `const mha_spec_t * s_in)`

4.280.3.3 get_fbpower_db() `void MHAOvlFilter::fftfb_t::get_fbpower_db (`
 `mha_wave_t * fbpow,`
 `const mha_spec_t * s_in)`

4.280.3.4 get_ltass_gain_db() `std::vector< float > MHAOvlFilter::fftfb_t::get_ltass_gain_db () const`

4.280.3.5 bin1() `unsigned int MHAOvlFilter::fftfb_t::bin1 (unsigned int band) const [inline]`

Return index of first non-zero filter shape window.

4.280.3.6 bin2() `unsigned int MHAOvlFilter::fftfb_t::bin2 (unsigned int band) const [inline]`

Return index of first zero filter shape window above center frequency.

4.280.3.7 get_ffflen() `unsigned int MHAOvlFilter::fftfb_t::get_ffflen () const [inline]`

Return fft length.

4.280.3.8 w() `mha_real_t MHAOvlFilter::fftfb_t::w (unsigned int k, unsigned int b) const [inline]`

Return filter shape window at index k in band b.

Parameters

<i>k</i>	Frequency index
<i>b</i>	Band index

4.280.4 Member Data Documentation

4.280.4.1 vbin1 `unsigned int* MHAOvlFilter::fftfb_t::vbin1 [private]`

4.280.4.2 vbin2 `unsigned int* MHAOvlFilter::fftfb_t::vbin2 [private]`

4.280.4.3 shape `mha_real_t (* MHAOvlFilter::fftfb_t::shape) (mha_real_t) [private]`

4.280.4.4 fftlen `unsigned int MHAOvlFilter::fftfb_t::ffflen [private]`

4.280.4.5 samplingrate `mha_real_t MHAOvlFilter::fftfb_t::samplingrate [private]`

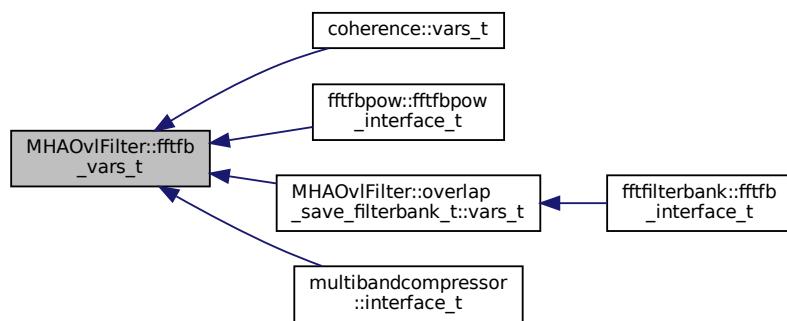
The documentation for this class was generated from the following files:

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

4.281 MHAOvlFilter::fftfb_vars_t Class Reference

Set of configuration variables for FFT-based overlapping filters.

Inheritance diagram for MHAOvlFilter::fftfb_vars_t:



Public Member Functions

- **fftfb_vars_t (MHParse::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 ovltpe**
Filter shape (rect/lin/hann).
- **MHParse::float_t plateau**
relative plateau width.
- **MHParse::kw_t ftype**
Flag to decide whether edge or center frequencies are used.
- **fscale_t f**
Frequency.
- **MHParse::bool_t normalize**
Normalize sum of channels.
- **MHParse::bool_t fail_on_nonmonotonic**
Fail if frequency entries are non-monotonic (otherwise sort)
- **MHParse::bool_t fail_on_unique_bins**
Fail if center frequencies share the same FFT bin.
- **MHParse::bool_t flag_allow_empty_bands**
Allow that frequency bands contain only zeros.
- **MHParse::vfloat_mon_t cf**
Final center frequencies in Hz.
- **MHParse::vfloat_mon_t ef**
Final edge frequencies in Hz.
- **MHParse::vfloat_mon_t cLTASS**
Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)
- **MHParse::mfloat_mon_t shapes**

4.281.1 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. 1006) 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.

4.281.2 Constructor & Destructor Documentation

4.281.2.1 `fftfb_vars_t()` `MHAOvlFilter::fftfb_vars_t::fftfb_vars_t (MHAParser::parser_t & p)`

construct a set of openMHA configuration language variables suitable for configuring the FFT-based overlapping filterbank.

Parameters

<code>p</code>	The node of the configuration tree where the variables created by this instance are inserted.
----------------	---

4.281.3 Member Data Documentation

4.281.3.1 `fscale scale_var_t` `MHAOvlFilter::fftfb_vars_t::fscale`

Frequency scale type (lin/bark/log/erb).

4.281.3.2 `ovltype scale_var_t` `MHAOvlFilter::fftfb_vars_t::ovltype`

Filter shape (rect/lin/hann).

4.281.3.3 `plateau MHAParser::float_t` `MHAOvlFilter::fftfb_vars_t::plateau`

relative plateau width.

4.281.3.4 ftype `MHAParser::kw_t MHAOvlFilter::fftfb_vars_t::ftype`

Flag to decide whether edge or center frequencies are used.

4.281.3.5 f `fscale_t MHAOvlFilter::fftfb_vars_t::f`

Frequency.

4.281.3.6 normalize `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::normalize`

Normalize sum of channels.

4.281.3.7 fail_on_nonmonotonic `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_nonmonotonic`

Fail if frequency entries are non-monotonic (otherwise sort)

4.281.3.8 fail_on_unique_bins `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::fail_on_unique_bins`

Fail if center frequencies share the same FFT bin.

4.281.3.9 flag_allow_empty_bands `MHAParser::bool_t MHAOvlFilter::fftfb_vars_t::flag_allow_empty_bands`

Allow that frequency bands contain only zeros.

4.281.3.10 cf `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cf`

Final center frequencies in Hz.

4.281.3.11 ef `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::ef`

Final edge frequencies in Hz.

4.281.3.12 cLTASS `MHAParser::vfloat_mon_t MHAOvlFilter::fftfb_vars_t::cLTASS`

Bandwidth correction for LTASS noise (level of 0 dB RMS LTASS noise)

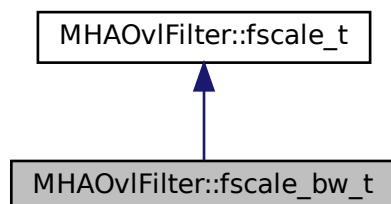
4.281.3.13 shapes `MHAParser::mfloat_mon_t MHAOvlFilter::fftfb_vars_t::shapes`

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

4.282 MHAOvlFilter::fscale_bw_t Class Reference

Inheritance diagram for MHAOvlFilter::fscale_bw_t:



Public Member Functions

- `fscale_bw_t (MHParse::parser_t &parent)`
- `std::vector< mha_real_t > get_bw_hz () const`

Protected Attributes

- `MHParse::vfloat_t bw`
- `MHParse::vfloat_mon_t bw_hz`

Private Member Functions

- `void update_hz ()`

Private Attributes

- `MHAEvents::connector_t< fscale_bw_t > updater`

Additional Inherited Members

4.282.1 Constructor & Destructor Documentation

4.282.1.1 `fscale_bw_t()` `MHAOvlFilter::fscale_bw_t::fscale_bw_t (MHParse::parser_t & parent)`

4.282.2 Member Function Documentation

4.282.2.1 `get_bw_hz()` `std::vector< mha_real_t > MHAOvlFilter::fscale_bw_t::get_bw_hz () const`

4.282.2.2 `update_hz()` `void MHAOvlFilter::fscale_bw_t::update_hz () [private]`

4.282.3 Member Data Documentation

4.282.3.1 bw `MHAParser::vfloat_t` `MHAOvlFilter::fscale_bw_t::bw` [protected]

4.282.3.2 bw_hz `MHAParser::vfloat_mon_t` `MHAOvlFilter::fscale_bw_t::bw_hz` [protected]

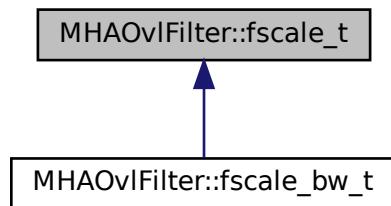
4.282.3.3 updater `MHAEvents::connector_t< fscale_bw_t>` `MHAOvlFilter::fscale_bw_t::updater` [private]

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

- `mha_fffb.hh`
- `mha_fffb.cpp`

4.283 MHAOvlFilter::fscale_t Class Reference

Inheritance diagram for MHAOvlFilter::fscale_t:



Public Member Functions

- `fscale_t (MHAParser::parser_t &parent)`
- `std::vector< mha_real_t > get_f_hz () const`

Public Attributes

- `scale_var_t unit`
- `MHAParser::vfloat_t f`
- `MHAParser::vfloat_mon_t f_hz`

Private Member Functions

- `void update_hz()`

Private Attributes

- `MHAEvents::connector_t< fscale_t > updater`

4.283.1 Constructor & Destructor Documentation

4.283.1.1 `fscale_t()` `MHAOvlFilter::fscale_t::fscale_t (`
`MHAParser::parser_t & parent)`

4.283.2 Member Function Documentation

4.283.2.1 `get_f_hz()` `std::vector< mha_real_t > MHAOvlFilter::fscale_t::get_f_hz (`
`) const`

4.283.2.2 `update_hz()` `void MHAOvlFilter::fscale_t::update_hz () [private]`

4.283.3 Member Data Documentation

4.283.3.1 unit scale_var_t MHAOvlFilter::fscale_t::unit

4.283.3.2 f MHAParser::vfloat_t MHAOvlFilter::fscale_t::f

4.283.3.3 f_hz MHAParser::vfloat_mon_t MHAOvlFilter::fscale_t::f_hz

4.283.3.4 updater MHAEvents::connector_t< fscale_t> MHAOvlFilter::fscale_t:::updater [private]

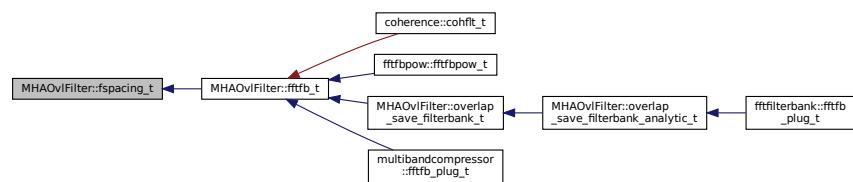
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

4.284 MHAOvlFilter::fspacing_t Class Reference

Class for frequency spacing, used by filterbank shape generator class.

Inheritance diagram for MHAOvlFilter::fspacing_t:



Public Member Functions

- **fspacing_t (const MHAOvlFilter::fftftb_vars_t &par, unsigned int nfft, mha_real_t fs)**
- **std::vector< unsigned int > get_cf_fftbin () const**
- **std::vector< mha_real_t > get_cf_hz () const**
- **std::vector< mha_real_t > get_ef_hz () const**
- **unsigned int nbands () const**

Return number of bands in filter bank.

Protected Member Functions

- void **fail_on_nonmonotonic_cf ()**
- void **fail_on_unique_fftbins ()**

Protected Attributes

- std::vector< **MHAOvlFilter::band_descriptor_t** > **bands**
- **mha_real_t**(* **symmetry_scale**)(**mha_real_t**)

Private Member Functions

- void **ef2bands** (std::vector< **mha_real_t** > vef)
- void **cf2bands** (std::vector< **mha_real_t** > vcf)
- void **equidist2bands** (std::vector< **mha_real_t** > vcf)

Private Attributes

- unsigned int **nfft_**
- **mha_real_t** **fs_**

4.284.1 Detailed Description

Class for frequency spacing, used by filterbank shape generator class.

4.284.2 Constructor & Destructor Documentation

```
4.284.2.1 fspacing_t() MHAOvlFilter::fspacing_t::fspacing_t (
    const MHAOvlFilter::fftfb_vars_t & par,
    unsigned int nfft,
    mha_real_t fs )
```

4.284.3 Member Function Documentation

4.284.3.1 get_cf_fftbin() std::vector< unsigned int > MHAOvlFilter::fspacing_t::get_cf_fftbin () const

4.284.3.2 get_cf_hz() std::vector< mha_real_t > MHAOvlFilter::fspacing_t::get_cf_hz () const

4.284.3.3 get_ef_hz() std::vector< mha_real_t > MHAOvlFilter::fspacing_t::get_ef_hz () const

4.284.3.4 nbands() unsigned int MHAOvlFilter::fspacing_t::nbands () const [inline]

Return number of bands in filter bank.

4.284.3.5 fail_on_nonmonotonic_cf() void MHAOvlFilter::fspacing_t::fail_on_nonmonotonic_cf () [protected]

4.284.3.6 fail_on_unique_fftbins() void MHAOvlFilter::fspacing_t::fail_on_unique_fftbins () [protected]

4.284.3.7 ef2bands() void MHAOvlFilter::fspacing_t::ef2bands (std::vector< mha_real_t > vef) [private]

4.284.3.8 cf2bands() void MHAOvlFilter::fspacing_t::cf2bands (std::vector< mha_real_t > vcf) [private]

4.284.3.9 equidist2bands() void MHAOvlFilter::fspacing_t::equidist2bands (std::vector< mha_real_t > vcf) [private]

4.284.4 Member Data Documentation

4.284.4.1 bands std::vector< MHAOvlFilter::band_descriptor_t > MHAOvlFilter::fspacing_t::bands [protected]

4.284.4.2 symmetry_scale mha_real_t (* MHAOvlFilter::fspacing_t::symmetry_scale) (mha_real_t) [protected]

4.284.4.3 nfft_ unsigned int MHAOvlFilter::fspacing_t::nfft_ [private]

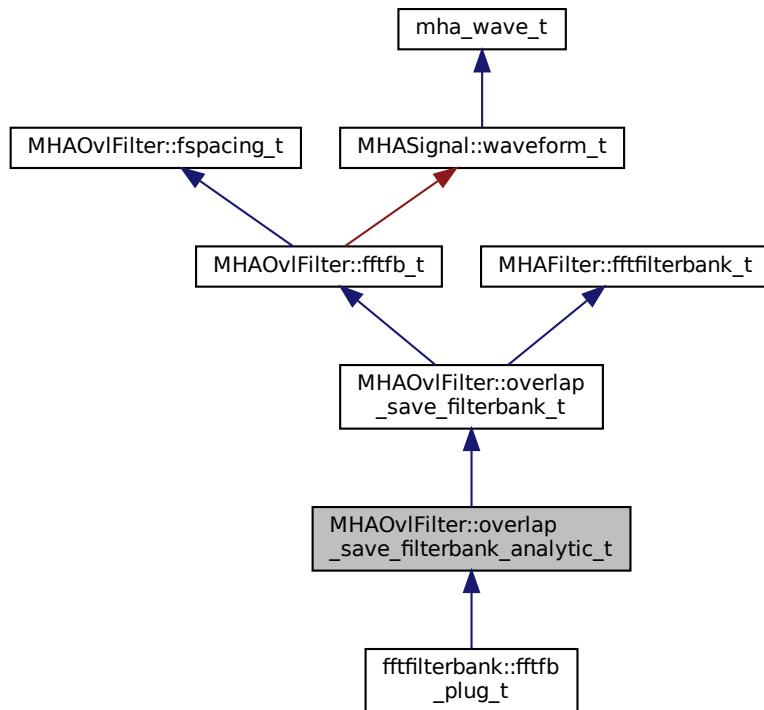
4.284.4.4 fs_ mha_real_t MHAOvlFilter::fspacing_t::fs_ [private]

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

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

4.285 MHAOvIFilter::overlap_save_filterbank_analytic_t Class Reference

Inheritance diagram for MHAOvIFilter::overlap_save_filterbank_analytic_t:



Public Member Functions

- **overlap_save_filterbank_analytic_t** (`MHAOvIFilter::overlap_save_filterbank_t::vars_t &fbpar, mhaconfig_t channelconfig_in)`
- void **filter_analytic** (const `mha_wave_t *sIn, mha_wave_t **fltRe, mha_wave_t **fltIm)`

Private Attributes

- **MHAFilter::fftfilterbank_t imagfb**

Additional Inherited Members

4.285.1 Constructor & Destructor Documentation

4.286 MHAOvlFilter::overlap_save_filterbank_t Class Reference

```
4.285.1.1 overlap_save_filterbank_analytic_t() MHAOvlFilter::overlap_save_filterbank_analytic_t::overlap_save_filterbank_analytic_t ( MHAOvlFilter::overlap_save_filterbank_t::vars_t & fbs, mhaconfig_t channelconfig_in )
```

4.285.2 Member Function Documentation

```
4.285.2.1 filter_analytic() void MHAOvlFilter::overlap_save_filterbank_analytic_t::filter_analytic ( const mha_wave_t * sIn, mha_wave_t ** fltRe, mha_wave_t ** fltIm )
```

4.285.3 Member Data Documentation

```
4.285.3.1 imagfb MHAFilter::fftfilterbank_t MHAOvlFilter::overlap_save_filterbank_analytic_t::imagfb [private]
```

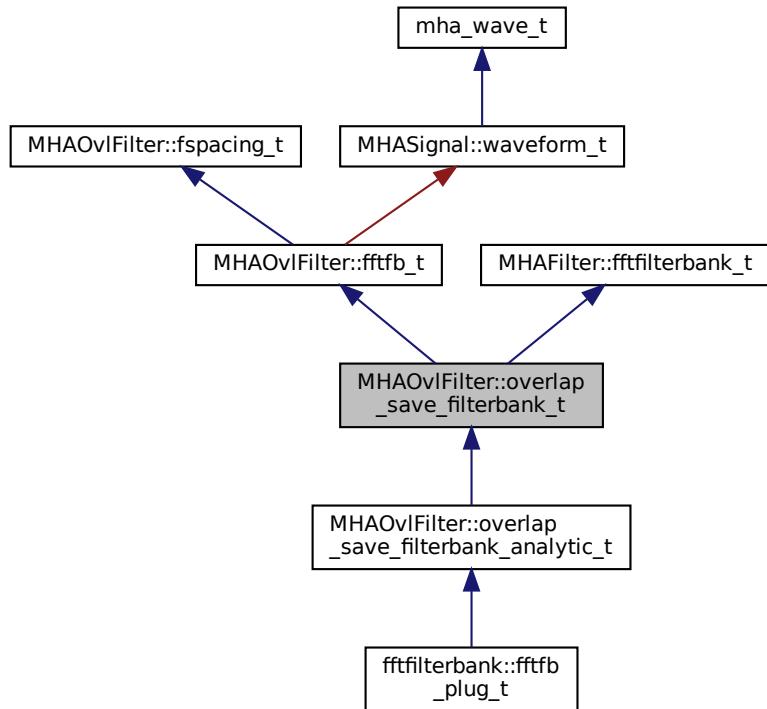
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

4.286 MHAOvlFilter::overlap_save_filterbank_t Class Reference

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1002).

Inheritance diagram for MHAOvlFilter::overlap_save_filterbank_t:



Classes

- class `vars_t`

Public Member Functions

- `overlap_save_filterbank_t (MHAOvlFilter::overlap_save_filterbank_t::vars_t &fb-
par, mhaconfig_t channelconfig_in)`
- `mhaconfig_t get_channelconfig () const`

Private Attributes

- `mhaconfig_t channelconfig_out_`

Additional Inherited Members

4.286.1 Detailed Description

A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1002).

4.286.2 Constructor & Destructor Documentation

4.286.2.1 overlap_save_filterbank_t() `MHAOvlFilter::overlap_save_filterbank_t` ↪
`::overlap_save_filterbank_t (`
 `MHAOvlFilter::overlap_save_filterbank_t::vars_t & fbspar,`
 `mhaconfig_t channelconfig_in)`

4.286.3 Member Function Documentation

4.286.3.1 get_channelconfig() `mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::get_channelconfig () const [inline]`

4.286.4 Member Data Documentation

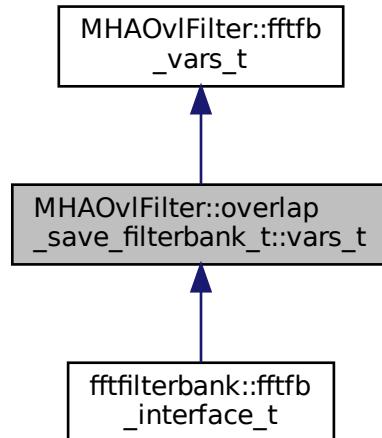
4.286.4.1 channelconfig_out_ `mhaconfig_t MHAOvlFilter::overlap_save_filterbank_t::channelconfig_out_ [private]`

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

- `mha_fftffb.hh`
- `mha_fftffb.cpp`

4.287 MHAOvlFilter::overlap_save_filterbank_t::vars_t Class Reference

Inheritance diagram for MHAOvlFilter::overlap_save_filterbank_t::vars_t:



Public Member Functions

- **vars_t (MHParse::parser_t &p)**

Public Attributes

- **MHParse::int_t fftlen**
- **MHParse::kw_t phasemode1**
- **MHParse::window_t irswnd**

4.287.1 Constructor & Destructor Documentation

4.287.1.1 vars_t() MHAOvlFilter::overlap_save_filterbank_t::vars_t::vars_t (MHParse::parser_t & p)

4.287.2 Member Data Documentation

4.287.2.1 fftlen `MHAParser::int_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::fftlen`

4.287.2.2 phasemode1 `MHAParser::kw_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::phasemode1`

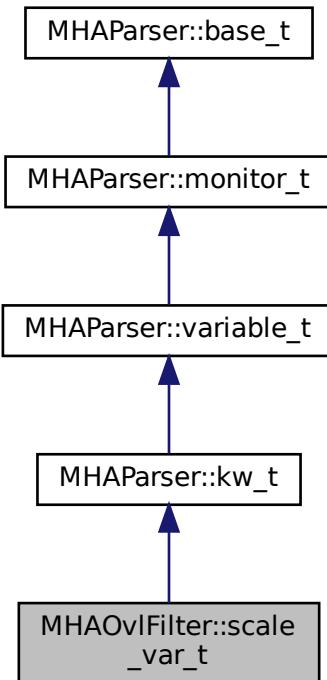
4.287.2.3 irswnd `MHAParser::window_t MHAOvlFilter::overlap_save_filterbank_t::vars_t::irswnd`

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

- `mha_fftfb.hh`
- `mha_fftfb.cpp`

4.288 MHAOvlFilter::scale_var_t Class Reference

Inheritance diagram for MHAOvlFilter::scale_var_t:



Public Member Functions

- **scale_var_t** (const std::string & **help**)
- void **add_fun** (const std::string &name, **scale_fun_t** *fun)
- std::string **get_name** () const
- **scale_fun_t** * **get_fun** () const
- **mha_real_t** **hz2unit** (**mha_real_t** x) const
- **mha_real_t** **unit2hz** (**mha_real_t** x) const

Private Attributes

- std::vector< std::string > **names**
- std::vector< **scale_fun_t** * > **fun**s

Additional Inherited Members

4.288.1 Constructor & Destructor Documentation

4.288.1.1 scale_var_t() MHAOvlFilter::scale_var_t::scale_var_t (const std::string & *help*)

4.288.2 Member Function Documentation

4.288.2.1 add_fun() void MHAOvlFilter::scale_var_t::add_fun (const std::string & *name*,
scale_fun_t * *fun*)

4.288.2.2 get_name() std::string MHAOvlFilter::scale_var_t::get_name () const [inline]

4.288.2.3 get_fun() **scale_fun_t*** MHAOvlFilter::scale_var_t::get_fun () const [inline]

4.288.2.4 hz2unit() **mha_real_t** MHAOvlFilter::scale_var_t::hz2unit (**mha_real_t** *x*) const

4.288.2.5 unit2hz() **mha_real_t** MHAOvlFilter::scale_var_t::unit2hz (**mha_real_t** *x*) const

4.288.3 Member Data Documentation

4.288.3.1 names std::vector<std::string> MHAOvlFilter::scale_var_t::names [private]

4.288.3.2 funs std::vector< scale_fun_t*> MHAOvlFilter::scale_var_t::funs [private]

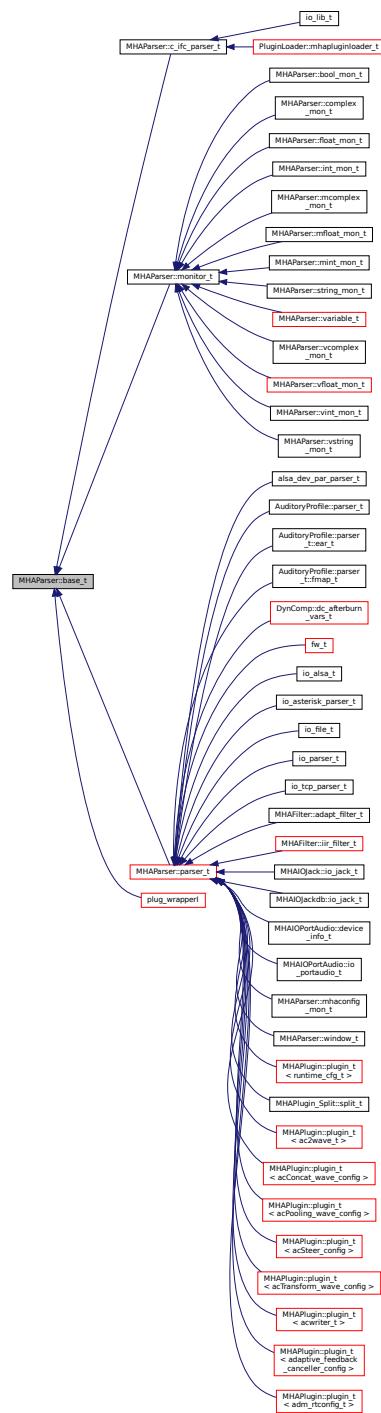
The documentation for this class was generated from the following files:

- **mha_fftfb.hh**
- **mha_fftfb.cpp**

4.289 MHAParser::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**

4.289.1 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. 1031) method. Parser proxy derivatives which overwrite any of the other **parse()** (p. 1031) methods to be the key method must make sure that the original **parse()** (p. 1031) method utilizes the new key method.

4.289.2 Member Typedef Documentation

4.289.2.1 repl_list_t `typedef std::vector< replace_t> MHPARSER::base_t::repl_< list_t> [private]`

4.289.3 Constructor & Destructor Documentation

4.289.3.1 base_t() [1/2] `MHPARSER::base_t::base_t (const std::string & h)`

Constructor for base class of all parser nodes.

Parameters

<code>h</code>	Help text describing this parser node. This help text is accessible to the configuration language through the "?help" query command.
----------------	--

4.289.3.2 base_t() [2/2] `MHPARSER::base_t::base_t (const base_t & src)`

4.289.3.3 ~base_t() `MHPARSER::base_t::~base_t () [virtual]`

4.289.4 Member Function Documentation

4.289.4.1 parse() [1/3] `std::string MHAParser::base_t::parse (const std::string & cs) [virtual]`

Causes this node to process a command in the openMHA configuration language.

Parameters

<code>cs</code>	The command to parse
-----------------	----------------------

Returns

The response to the command, if successful

Exceptions

MHA_Error (p. 763)	If the command cannot be executed successfully. The reason for failure is given in the message string of the exception.
--	---

Reimplemented in [plug_wrapper1](#) (p. [1350](#)), [PluginLoader::mhaplugloader_t](#) (p. [1368](#)), [plug_wrapper](#) (p. [1348](#)), [io_wrapper](#) (p. [642](#)), and [altplugs_t](#) (p. [302](#)).

4.289.4.2 parse() [2/3] `void MHAParser::base_t::parse (const char * cmd, char * retv, unsigned int len) [virtual]`

This function parses a command and writes the parsing result into a C character array.

This base class implementation delegates to [parse\(const std::string &\)](#) (p. [1031](#)).

Parameters

<code>cmd</code>	Command to be parsed
<code>retv</code>	Buffer for the result
<code>len</code>	Length of buffer

Reimplemented in **altplugs_t** (p. 302).

4.289.4.3 parse() [3/3] `void MHAParser::base_t::parse (const std::vector< std::string > & cs, std::vector< std::string > & retv) [virtual]`

4.289.4.4 op_subparse() `std::string MHAParser::base_t::op_subparse (expression_t &) [virtual]`

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1046), and **MHAParser::parser_t** (p. 1100).

4.289.4.5 op_setval() `std::string MHAParser::base_t::op_setval (expression_t &) [virtual]`

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1046), **MHAParser::mcomplex_t** (p. 1077), **MHAParser::mfloat_t** (p. 1082), **MHAParser::mint_t** (p. 1094), **MHAParser::vcomplex_t** (p. 1118), **MHAParser::vfloat_t** (p. 1124), **MHAParser::vint_t** (p. 1128), **MHAParser::complex_t** (p. 1053), **MHAParser::float_t** (p. 1060), **MHAParser::int_t** (p. 1065), **MHAParser::bool_t** (p. 1044), **MHAParser::vstring_t** (p. 1132), **MHAParser::string_t** (p. 1112), **MHAParser::kw_t** (p. 1072), **MHAParser::variable_t** (p. 1114), and **MHAParser::parser_t** (p. 1101).

4.289.4.6 op_query() `std::string MHAParser::base_t::op_query (expression_t &) [virtual]`

Reimplemented in **MHAParser::c_ifc_parser_t** (p. 1046), **MHAParser::monitor_t** (p. 1096), and **MHAParser::parser_t** (p. 1101).

4.289.4.7 query_dump() `std::string MHAParser::base_t::query_dump (const std::string & s) [virtual]`

Reimplemented in **MHAParser::monitor_t** (p. 1096), and **MHAParser::parser_t** (p. 1101).

4.289.4.8 query_entries() std::string MHAParser::base_t::query_entries (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1101).

4.289.4.9 query_perm() std::string MHAParser::base_t::query_perm (const std::string & s) [virtual]

Reimplemented in **MHAParser::variable_t** (p. 1114), and **MHAParser::monitor_t** (p. 1096).

4.289.4.10 query_range() std::string MHAParser::base_t::query_range (const std::string & s) [virtual]

Reimplemented in **MHAParser::kw_t** (p. 1073), and **MHAParser::range_var_t** (p. 1105).

4.289.4.11 query_type() std::string MHAParser::base_t::query_type (const std::string & s) [virtual]

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 1075), **MHAParser::vcomplex_mon_t** (p. 1116), **MHAParser::complex_mon_t** (p. 1051), **MHAParser::float_mon_t** (p. 1058), **MHAParser::mfloat_mon_t** (p. 1079), **MHAParser::vfloat_mon_t** (p. 1121), **MHAParser::mint_mon_t** (p. 1091), **MHAParser::vint_mon_t** (p. 1126), **MHAParser::vstring_mon_t** (p. 1131), **MHAParser::string_mon_t** (p. 1109), **MHAParser::bool_mon_t** (p. 1042), **MHAParser::int_mon_t** (p. 1063), **MHAParser::mcomplex_t** (p. 1077), **MHAParser::mfloat_t** (p. 1082), **MHAParser::mint_t** (p. 1094), **MHAParser::vcomplex_t** (p. 1119), **MHAParser::vfloat_t** (p. 1124), **MHAParser::vint_t** (p. 1128), **MHAParser::complex_t** (p. 1053), **MHAParser::float_t** (p. 1060), **MHAParser::int_t** (p. 1066), **MHAParser::bool_t** (p. 1044), **MHAParser::vstring_t** (p. 1132), **MHAParser::string_t** (p. 1112), **MHAParser::kw_t** (p. 1073), and **MHAParser::parser_t** (p. 1101).

4.289.4.12 query_val() std::string MHAParser::base_t::query_val (const std::string & s) [virtual]

Reimplemented in **MHAParser::mcomplex_mon_t** (p. 1075), **MHAParser::vcomplex_mon_t** (p. 1116), **MHAParser::complex_mon_t** (p. 1051), **MHAParser::float_mon_t** (p. 1057), **MHAParser::mfloat_mon_t** (p. 1079), **MHAParser::vfloat_mon_t** (p. 1121), **MHAParser::mint_mon_t** (p. 1091), **MHAParser::vint_mon_t** (p. 1126), **MHAParser::vstring_mon_t** (p. 1130), **MHAParser::string_mon_t** (p. 1109), **MHAParser::bool_mon_t** (p. 1041), **MHAParser::int_mon_t** (p. 1063), **MHAParser::mcomplex_t** (p. 1077), **MHAParser::mfloat_t** (p. 1082), **MHAParser::mint_t** (p. 1094), **MHAParser::vcomplex_t** (p. 1119), **MHAParser::vfloat_t** (p. 1124), **MHAParser::vint_t** (p. 1128), **MHAParser::complex_t** (p. 1053), **MHAParser::float_t** (p. 1061), **MHAParser::int_t** (p. 1066), **MHAParser::bool_t** (p. 1044), **MHAParser::vstring_t** (p. 1133), **MHAParser::string_t** (p. 1112), **MHAParser::kw_t** (p. 1073), and **MHAParser::parser_t** (p. 1102).

4.289.4.13 query_readfile() std::string MHAParser::base_t::query_readfile (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1101).

4.289.4.14 query_savefile() std::string MHAParser::base_t::query_savefile (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1102).

4.289.4.15 query_savefile_compact() std::string MHAParser::base_t::query_savefile_compact (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1102).

4.289.4.16 query_savemons() std::string MHAParser::base_t::query_savemons (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1102).

4.289.4.17 query_listids() std::string MHAParser::base_t::query_listids (const std::string & s) [virtual]

Reimplemented in **MHAParser::parser_t** (p. 1102).

4.289.4.18 query_version() std::string MHAParser::base_t::query_version (const std::string &)

4.289.4.19 query_id() std::string MHAParser::base_t::query_id (const std::string &)

4.289.4.20 query_subst() std::string MHAParser::base_t::query_subst (const std::string & s)

4.289.4.21 query_addsubst() std::string MHAParser::base_t::query_addsubst (const std::string & s)

4.289.4.22 query_help() std::string MHAParser::base_t::query_help (const std::string & s)

4.289.4.23 query_cmds() std::string MHAParser::base_t::query_cmds (const std::string & s)

4.289.4.24 set_node_id() void MHAParser::base_t::set_node_id (const std::string & s)

Set the identification string of this parser node.

The id can be queried from the configuration language using the ?id query command. Nodes can be found by id using the ?listid query command on a containing parser node.

Parameters

s	The new identification string.
----------	--------------------------------

4.289.4.25 `set_help()` void MHAParser::base_t::set_help (const std::string & s)

Set the help comment of a variable or parser.

Parameters

s	New help comment.
----------	-------------------

4.289.4.26 `add_parent_on_insert()` void MHAParser::base_t::add_parent_on_insert (**parser_t** * p, std::string n)

4.289.4.27 `rm_parent_on_remove()` void MHAParser::base_t::rm_parent_on_remove (**parser_t** *)

4.289.4.28 `fullname()` const std::string & MHAParser::base_t::fullname () const

Return the full dot-separated path name of this parser node in the openMHA configuration tree.

4.289.4.29 `activate_query()` void MHAParser::base_t::activate_query (const std::string & n, **query_t** a) [protected]

4.289.4.30 notify() void MHAParser::base_t::notify () [protected]

4.289.4.31 add_replace_pair() void MHAParser::base_t::add_replace_pair (const std::string & a, const std::string & b) [private]

4.289.4.32 oplist() std::string MHAParser::base_t::oplist () [private]

4.289.5 Member Data Documentation

4.289.5.1 writeaccess MHAEvents::emitter_t MHAParser::base_t::writeaccess

Event emitted on write access.

To connect a callback that is invoked on write access to this parser variable, use MHAEvents::patchbay_t<receiver_t> method connect(&writeaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

4.289.5.2 valuechanged MHAEvents::emitter_t MHAParser::base_t::valuechanged

Event emitted if the value has changed.

To connect a callback that is invoked when write access to this parser variable actually changes its value, use MHAEvents::patchbay_t<receiver_t> method connect(&valuechanged,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

4.289.5.3 readaccess MHAEvents::emitter_t MHAParser::base_t::readaccess

Event emitted on read access.

To connect a callback that is invoked after the value of this variable has been read through the configuration interface, use MHAEvents::patchbay_t<receiver_t> method connect(&readaccess,&receiver_t::callback) where callback is a method that expects no parameters and returns void.

4.289.5.4 prereadaccess `MHAEVENTS::EMITTER_T MHAPARSER::BASE_T::PREREADACCESS`

Event emitted on read access, before the data field is accessed.

To connect a callback that is invoked when the value of this variable is about to be read through the configuration interface, so that the callback can influence the value that is reported, use `MHAEVENTS::PATCHBAY_T<RECEIVER_T>` method `CONNECT(&PREREADACCESS,&RECEIVER_T::CALLBACK)` where callback is a method that expects no parameters and returns void.

4.289.5.5 queries `QUERY_MAP_T MHAPARSER::BASE_T::QUERIES [PROTECTED]`**4.289.5.6 data_is_initialized** `bool MHAPARSER::BASE_T::DATA_IS_INITIALIZED [PROTECTED]`**4.289.5.7 help** `std::string MHAPARSER::BASE_T::HELP [PRIVATE]`**4.289.5.8 id_str** `std::string MHAPARSER::BASE_T::ID_STR [PRIVATE]`**4.289.5.9 operators** `OPACT_MAP_T MHAPARSER::BASE_T::OPERATORS [PRIVATE]`**4.289.5.10 repl_list** `REPL_LIST_T MHAPARSER::BASE_T::REPL_LIST [PRIVATE]`**4.289.5.11 nested_lock** `bool MHAPARSER::BASE_T::NESTED_LOCK [PRIVATE]`

4.289.5.12 parent `parser_t*` MHAParser::base_t::parent [private]

4.289.5.13 thefullname `std::string` MHAParser::base_t::thefullname [private]

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

- `mha_parser.hh`
- `mha_parser.cpp`

4.290 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`

4.290.1 Constructor & Destructor Documentation

4.290.1.1 replace_t() `MHAParser::base_t::replace_t::replace_t (`
`const std::string & ia,`
`const std::string & ib)`

4.290.2 Member Function Documentation

4.290.2.1 `replace()` `void MHAParser::base_t::replace_t::replace (std::string & s)`

4.290.2.2 `get_a()` `const std::string& MHAParser::base_t::replace_t::get_a () const [inline]`

4.290.2.3 `get_b()` `const std::string& MHAParser::base_t::replace_t::get_b () const [inline]`

4.290.3 Member Data Documentation

4.290.3.1 `a` `std::string MHAParser::base_t::replace_t::a [private]`

4.290.3.2 `b` `std::string MHAParser::base_t::replace_t::b [private]`

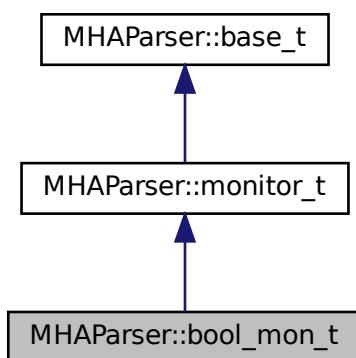
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.291 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

4.291.1 Detailed Description

Monitor with string value.

4.291.2 Constructor & Destructor Documentation

4.291.2.1 **bool_mon_t()** MHParse::bool_mon_t::bool_mon_t (const std::string & hlp)

Create a monitor variable for string values.

Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

4.291.3 Member Function Documentation

4.291.3.1 `query_val()` `std::string MHAParser::bool_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.291.3.2 `query_type()` `std::string MHAParser::bool_mon_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.291.4 Member Data Documentation

4.291.4.1 `data` `bool MHAParser::bool_mon_t::data`

Data field.

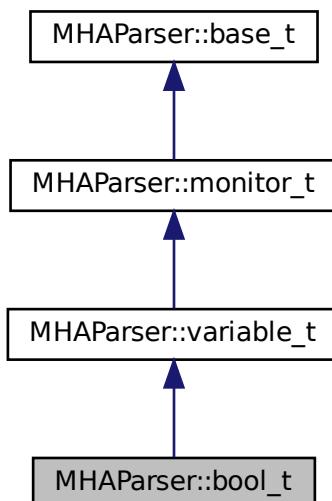
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.292 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

4.292.1 Detailed Description

Variable with a boolean value ("yes"/"no")

4.292.2 Constructor & Destructor Documentation

4.292.2.1 bool_t() MHParse::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".

4.292.3 Member Function Documentation

4.292.3.1 `op_setval()` std::string MHAParser::bool_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.292.3.2 `query_type()` std::string MHAParser::bool_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.292.3.3 `query_val()` std::string MHAParser::bool_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.292.4 Member Data Documentation

4.292.4.1 `data` bool MHAParser::bool_t::data

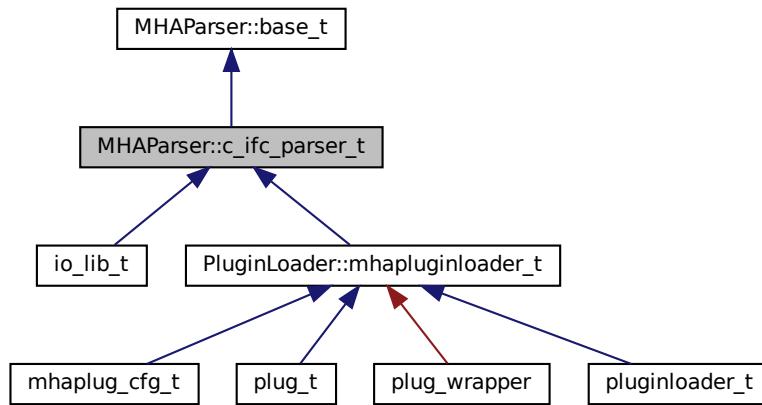
Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

4.293 MHAParser::c_ifc_parser_t Class Reference

Inheritance diagram for MHAParser::c_ifc_parser_t:



Public Member Functions

- `c_ifc_parser_t (const std::string &modulename_)`
- `~c_ifc_parser_t ()`
- `void set_parse_cb (c_parse_cmd_t, c_parse_err_t, void *)`

Protected Member Functions

- `std::string op_subparse (MHAParser::expression_t &)`
- `std::string op_setval (MHAParser::expression_t &)`
- `std::string op_query (MHAParser::expression_t &)`

Private Member Functions

- `void test_error ()`

Private Attributes

- `std::string modulename`
- `c_parse_cmd_t c_parse_cmd`
- `c_parse_err_t c_parse_err`
- `int liberr`
- `void * libdata`
- `unsigned int ret_size`
- `char * retv`

Additional Inherited Members

4.293.1 Constructor & Destructor Documentation

4.293.1.1 `c_ifc_parser_t()` `MHAParser::c_ifc_parser_t::c_ifc_parser_t (const std::string & modulename_)`

4.293.1.2 `~c_ifc_parser_t()` `MHAParser::c_ifc_parser_t::~c_ifc_parser_t ()`

4.293.2 Member Function Documentation

4.293.2.1 `set_parse_cb()` `void MHAParser::c_ifc_parser_t::set_parse_cb (c_parse_cmd_t , c_parse_err_t , void *)`

4.293.2.2 `op_subparse()` `std::string MHAParser::c_ifc_parser_t::op_subparse (MHAParser::expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1032).

4.293.2.3 `op_setval()` `std::string MHAParser::c_ifc_parser_t::op_setval (MHAParser::expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1032).

4.293.2.4 op_query() std::string MHAParser::c_ifc_parser_t::op_query (MHParse::expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.293.2.5 test_error() void MHAParser::c_ifc_parser_t::test_error () [private]

4.293.3 Member Data Documentation

4.293.3.1 modulename std::string MHAParser::c_ifc_parser_t::modulename [private]

4.293.3.2 c_parse_cmd c_parse_cmd_t MHAParser::c_ifc_parser_t::c_parse_cmd [private]

4.293.3.3 c_parse_err c_parse_err_t MHAParser::c_ifc_parser_t::c_parse_err [private]

4.293.3.4 liberr int MHAParser::c_ifc_parser_t::liberr [private]

4.293.3.5 libdata void* MHAParser::c_ifc_parser_t::libdata [private]

4.293.3.6 ret_size unsigned int MHAParser::c_ifc_parser_t::ret_size [private]

4.293.3.7 `retv` `char* MHAParser::c_ifc_parser_t::retv` [private]

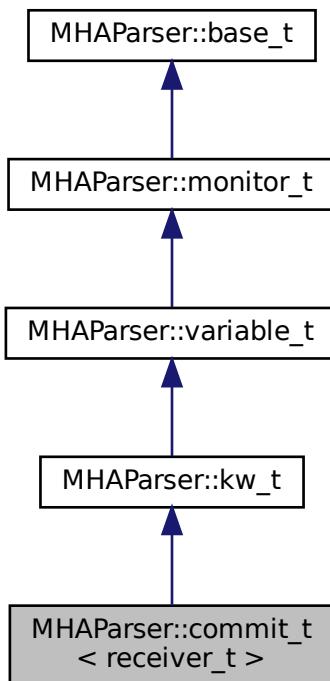
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.294 `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

4.294.1 Detailed Description

```
template<class receiver_t>
class MHParse::commit_t< receiver_t >
```

Parser variable with event-emission functionality.

The **commit_t** (p. 1048) variable can register an event receiver in its constructor, which is called whenever the variable is set to "commit".

4.294.2 Constructor & Destructor Documentation

```
4.294.2.1 commit_t() template<class receiver_t >
MHParse::commit_t< receiver_t >:: commit_t (
    receiver_t * r,
    void(receiver_t::* )() rfun,
    const std::string & help = "Variable changes action" )
```

4.294.3 Member Data Documentation

```
4.294.3.1 extern_connector template<class receiver_t >
MHAEvents::connector_t<receiver_t> MHParse::commit_t< receiver_t >::extern_<→
connector [private]
```

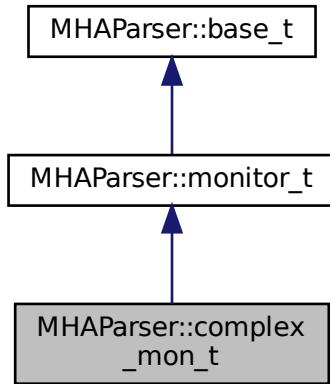
The documentation for this class was generated from the following file:

- mha_parser.hh

4.295 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

4.295.1 Detailed Description

Monitor with complex value.

4.295.2 Constructor & Destructor Documentation

4.295.2.1 complex_mon_t() `MHAParser::complex_mon_t::complex_mon_t (const std::string & hlp)`

Create a complex monitor variable.

Parameters

<code>hlp</code>	A help text describing this monitor variable.
------------------	---

4.295.3 Member Function Documentation

4.295.3.1 query_val() `std::string MHAParser::complex_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.295.3.2 query_type() `std::string MHAParser::complex_mon_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.295.4 Member Data Documentation

4.295.4.1 data mha_complex_t `MHAParser::complex_mon_t::data`

Data field.

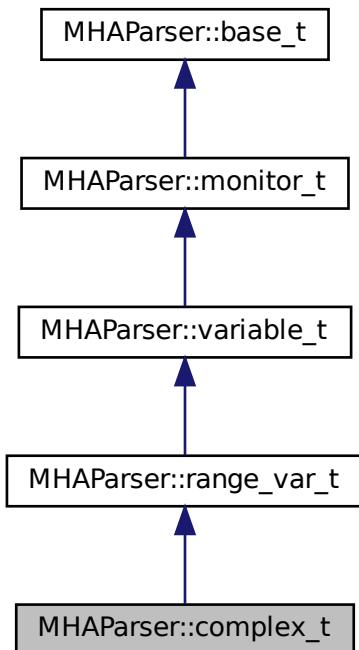
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.296 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

4.296.1 Detailed Description

Variable with complex value.

4.296.2 Constructor & Destructor Documentation

```
4.296.2.1 complex_t() MHAParser::complex_t::complex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "")
```

4.296.3 Member Function Documentation

```
4.296.3.1 op_setval() std::string MHAParser::complex_t::op_setval (
    expression_t & x) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1114).

```
4.296.3.2 query_type() std::string MHAParser::complex_t::query_type (
    const std::string & s) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1033).

```
4.296.3.3 query_val() std::string MHAParser::complex_t::query_val (
    const std::string & s) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1033).

4.296.4 Member Data Documentation

4.296.4.1 **data** `mha_complex_t` `MHAParser::complex_t::data`

Data field.

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

- `mha_parser.hh`
- `mha_parser.cpp`

4.297 `MHAParser::entry_t` Class Reference

Public Member Functions

- `entry_t (const std::string &, base_t *)`

Public Attributes

- `std::string name`
- `base_t * entry`

4.297.1 Constructor & Destructor Documentation

4.297.1.1 `entry_t()` `MHAParser::entry_t::entry_t (` `const std::string & n,` `base_t * e)`

4.297.2 Member Data Documentation

4.297.2.1 name std::string MHAParser::entry_t::name

4.297.2.2 entry base_t* MHAParser::entry_t::entry

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

- **mha_parser.hh**
- **mha_parser.cpp**

4.298 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**

4.298.1 Constructor & Destructor Documentation

4.298.1.1 expression_t() [1/2] expression_t::expression_t (

```
const std::string & s,
const std::string & o )
```

Constructor.

Parameters

s	String to be splitted
o	List of valid operators (single character only)

4.298.1.2 expression_t() [2/2] expression_t::expression_t ()**4.298.2 Member Data Documentation****4.298.2.1 lval** std::string MHAParser::expression_t::lval**4.298.2.2 rval** std::string MHAParser::expression_t::rval**4.298.2.3 op** std::string MHAParser::expression_t::op

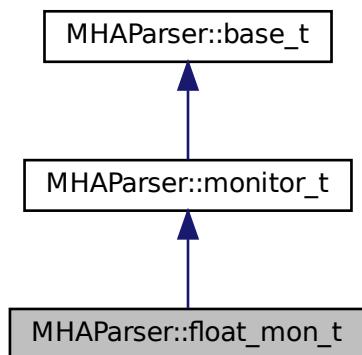
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.299 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

4.299.1 Detailed Description

Monitor with float value.

4.299.2 Constructor & Destructor Documentation

4.299.2.1 **float_mon_t()** MHParse::float_mon_t::float_mon_t (

```
const std::string & hlp )
```

Initialize a floating point (32 bits) monitor variable.

Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

4.299.3 Member Function Documentation

4.299.3.1 query_val() std::string MHAParser::float_mon_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.299.3.2 query_type() std::string MHAParser::float_mon_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.299.4 Member Data Documentation

4.299.4.1 data float MHAParser::float_mon_t::data

Data field.

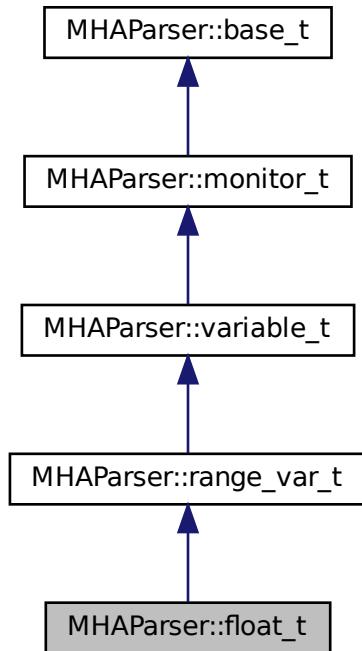
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.300 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

4.300.1 Detailed Description

Variable with float value.

4.300.2 Constructor & Destructor Documentation

```
4.300.2.1 float_t() MHAParser::float_t::float_t (
    const std::string & help_text,
    const std::string & initial_value,
    const std::string & range = "" )
```

Constructor for a configuration language variable for 32bit ieee floating-point values.

Parameters

<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<=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.

4.300.3 Member Function Documentation

```
4.300.3.1 op_setval() std::string MHAParser::float_t::op_setval (
    expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.300.3.2 query_type() std::string MHAParser::float_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.300.3.3 query_val() std::string MHAParser::float_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.300.4 Member Data Documentation

4.300.4.1 data float MHAParser::float_t::data

Data field.

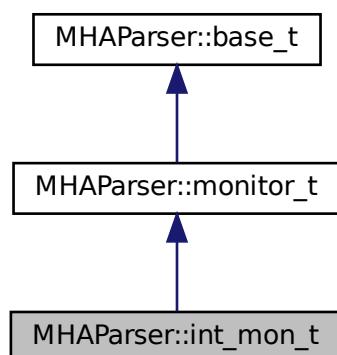
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.301 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

4.301.1 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.

4.301.2 Constructor & Destructor Documentation

4.301.2.1 **int_mon_t()** 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.
------------	---

4.301.3 Member Function Documentation

4.301.3.1 query_val() std::string MHAParser::int_mon_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.301.3.2 query_type() std::string MHAParser::int_mon_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.301.4 Member Data Documentation

4.301.4.1 data int MHAParser::int_mon_t::data

Data field.

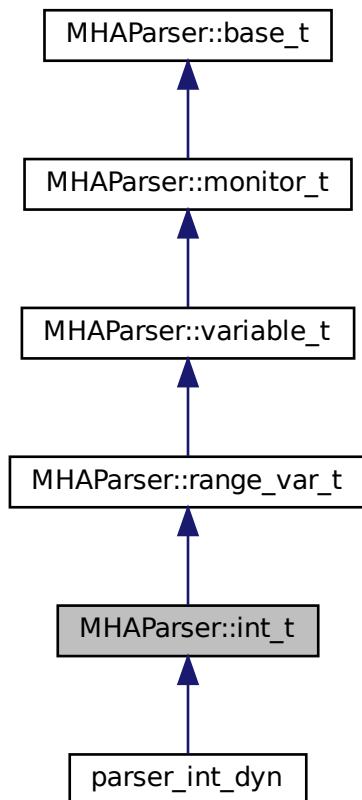
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.302 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

4.302.1 Detailed Description

Variable with integer value.

4.302.2 Constructor & Destructor Documentation

```
4.302.2.1 int_t() MHAParser::int_t::int_t (
    const std::string & help_text,
    const std::string & initial_value,
    const std::string & range = "" )
```

Constructor for a configuration language variable for integral values.

Parameters

<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<=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]).

4.302.3 Member Function Documentation

4.302.3.1 `op_setval()` std::string MHAParser::int_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.302.3.2 `query_type()` std::string MHAParser::int_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.302.3.3 `query_val()` std::string MHAParser::int_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.302.4 Member Data Documentation

4.302.4.1 `data` int MHAParser::int_t::data

Data field.

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

- **mha_parser.hh**
- **mha_parser.cpp**

4.303 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**

4.303.1 Detailed Description

Keyword list class.

The structure **keyword_list_t** (p. 1066) defines a keyword list (vector of strings) with an index into the list. Used as **MHAParser::kw_t** (p. 1070), it can be used to access a set of valid keywords through the parser (i.e. one of "pear apple banana").

4.303.2 Member Typedef Documentation

4.303.2.1 **size_t** `typedef std::vector<std::string>::size_type MHAParser::keyword_list_t::size_t`

4.303.3 Constructor & Destructor Documentation

4.303.3.1 keyword_list_t() `MHAParser::keyword_list_t::keyword_list_t ()`

Constructor.

4.303.4 Member Function Documentation

4.303.4.1 set_value() `void MHAParser::keyword_list_t::set_value (const std::string & s)`

Select a value from keyword list.

This function selects a value from the keyword list. The index is set to the last matching entry.

Parameters

<code>s</code>	Value to be selected.
----------------	-----------------------

4.303.4.2 set_entries() `void MHAParser::keyword_list_t::set_entries (const std::string & s)`

Set keyword list entries.

With this function, the keyword list can be set from a space separated string list.

Parameters

<code>s</code>	Space separated entry list.
----------------	-----------------------------

4.303.4.3 get_value() `const std::string & MHAParser::keyword_list_t::get_value ()`

const

Return selected value.

4.303.4.4 get_entries() const std::vector< std::string > & MHAParser::keyword_list_t::get_entries () const

Return keyword list.

4.303.4.5 get_index() const MHAParser::keyword_list_t::size_t & MHAParser::keyword_list_t::get_index () const

Return index of selected value.

4.303.4.6 set_index() void MHAParser::keyword_list_t::set_index (unsigned int idx)

4.303.4.7 validate() void MHAParser::keyword_list_t::validate () const

Check if index of selected value is valid.

4.303.4.8 add_entry() void MHAParser::keyword_list_t::add_entry (const std::string & en) [inline]

4.303.5 Member Data Documentation

4.303.5.1 index size_t MHAParser::keyword_list_t::index [private]

Index into list.

4.303.5.2 entries std::vector<std::string> MHAParser::keyword_list_t::entries [private]

List of valid entries.

4.303.5.3 empty_string std::string MHAParser::keyword_list_t::empty_string [private]

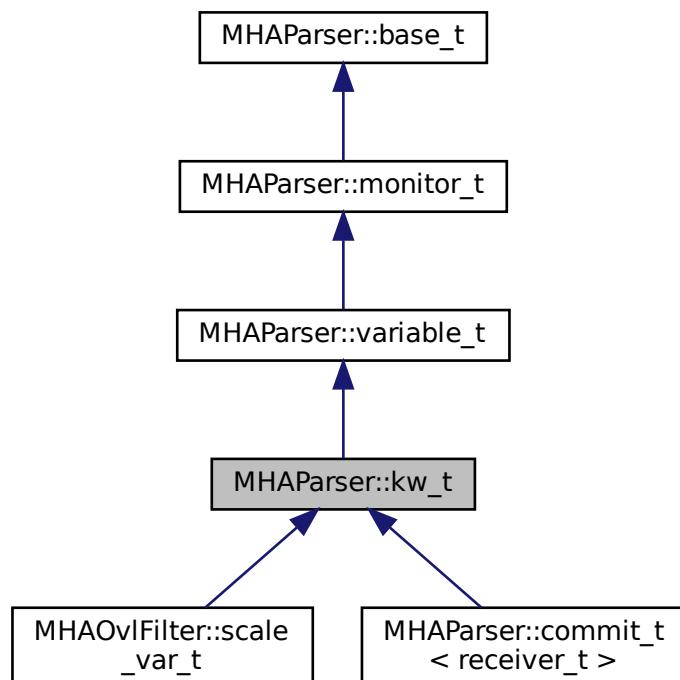
The documentation for this class was generated from the following files:

- [mha_parser.hh](#)
- [mha_parser.cpp](#)

4.304 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

4.304.1 Detailed Description

Variable with keyword list value.

4.304.2 Constructor & Destructor Documentation

4.304.2.1 kw_t() [1/2] MHParse::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 "[", "]".

4.304.2.2 kw_t() [2/2] `MHAParser::kw_t::kw_t (`
`const kw_t & src)`

Copy constructor.

4.304.3 Member Function Documentation

4.304.3.1 set_range() `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 "[", "]".
----------	--

4.304.3.2 isval() `bool MHAParser::kw_t::isval (`
`const std::string & testval) const`

Test if the given value is selected.

4.304.3.3 validate() `void MHAParser::kw_t::validate (`
`const keyword_list_t & s) [protected]`

4.304.3.4 op_setval() std::string MHAParser::kw_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.304.3.5 query_range() std::string MHAParser::kw_t::query_range (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.304.3.6 query_val() std::string MHAParser::kw_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.304.3.7 query_type() std::string MHAParser::kw_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.304.4 Member Data Documentation

4.304.4.1 data keyword_list_t MHAParser::kw_t::data

Variable data in its native type.

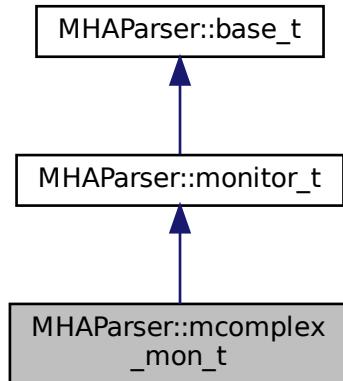
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.305 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

4.305.1 Detailed Description

Matrix of complex numbers monitor.

4.305.2 Constructor & Destructor Documentation

4.305.2.1 mcomplex_mon_t() `MHAParser::mcomplex_mon_t::mcomplex_mon_t (const std::string & hlp)`

Create a matrix of complex floating point monitor values.

Parameters

<code>hlp</code>	A help text describing this monitor variable.
------------------	---

4.305.3 Member Function Documentation

4.305.3.1 query_val() `std::string MHAParser::mcomplex_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.305.3.2 query_type() `std::string MHAParser::mcomplex_mon_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.305.4 Member Data Documentation

4.305.4.1 data `std::vector< std::vector< mha_complex_t > > MHAParser::mcomplex_mon_t::data`

Data field.

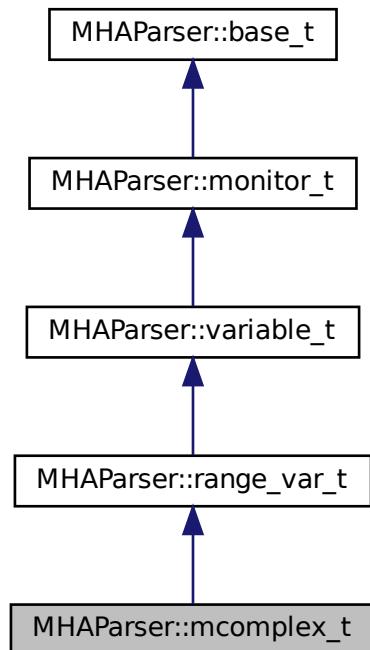
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.306 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

4.306.1 Detailed Description

Matrix variable with complex value.

4.306.2 Constructor & Destructor Documentation

```
4.306.2.1 mcomplex_t() MHAParser::mcomplex_t::mcomplex_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "")
```

4.306.3 Member Function Documentation

```
4.306.3.1 op_setval() std::string MHAParser::mcomplex_t::op_setval (
    expression_t & x) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1114).

```
4.306.3.2 query_type() std::string MHAParser::mcomplex_t::query_type (
    const std::string & s) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1033).

```
4.306.3.3 query_val() std::string MHAParser::mcomplex_t::query_val (
    const std::string & s) [protected], [virtual]
```

Reimplemented from **MHAParser::base_t** (p. 1033).

4.306.4 Member Data Documentation

4.306.4.1 data std::vector<std::vector<`mha_complex_t`> > MHAParser::mcomplex_t ←
 ::data

Data field.

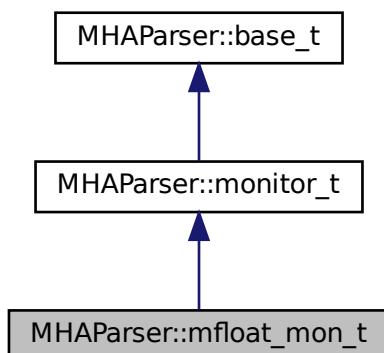
The documentation for this class was generated from the following files:

- [`mha_parser.hh`](#)
- [`mha_parser.cpp`](#)

4.307 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

4.307.1 Detailed Description

Matrix of floats monitor.

4.307.2 Constructor & Destructor Documentation

4.307.2.1 **mfloat_mon_t()** MHAParser::mfloat_mon_t::mfloat_mon_t (const std::string & *hlp*)

Create a matrix of floating point monitor values.

Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

4.307.3 Member Function Documentation

4.307.3.1 **query_val()** std::string MHAParser::mfloat_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.307.3.2 `query_type()` `std::string MHAParser::mfloat_mon_t::query_type (`
`const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.307.4 Member Data Documentation

4.307.4.1 `data` `std::vector< std::vector<float> > MHAParser::mfloat_mon_t::data`

Data field.

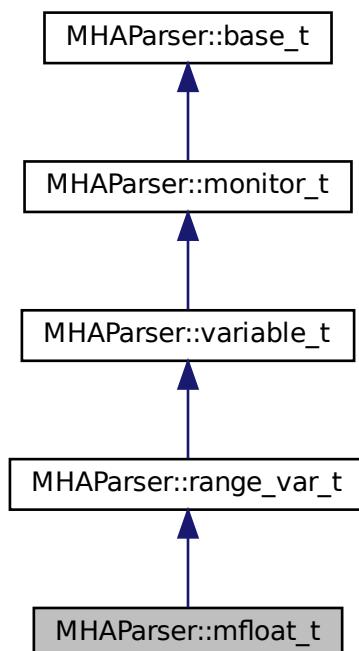
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.308 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

4.308.1 Detailed Description

Matrix variable with float value.

4.308.2 Constructor & Destructor Documentation

```
4.308.2.1 mfloat_t() MHAParser::mfloat_t::mfloat_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

Create a float matrix parser variable.

Parameters

<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.

4.308.3 Member Function Documentation

4.308.3.1 `op_setval()` `std::string MHAParser::mfloate_t::op_setval (expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.308.3.2 `query_type()` `std::string MHAParser::mfloate_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.308.3.3 `query_val()` `std::string MHAParser::mfloate_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.308.4 Member Data Documentation

4.308.4.1 `data` `std::vector<std::vector<float> > MHAParser::mfloate_t::data`

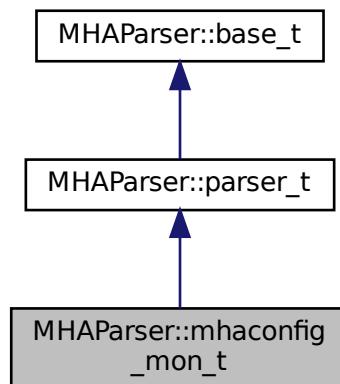
Data field.

The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.309 MHParse::mhaconfig_mon_t Class Reference

Inheritance diagram for MHParse::mhaconfig_mon_t:



Public Member Functions

- **mhaconfig_mon_t** (const std::string & **help**="")
- void **update** (const **mhaconfig_t** &cf)

Private Attributes

- **MHParse::int_mon_t channels**
Number of audio channels.
- **MHParse::string_mon_t domain**
Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)
- **MHParse::int_mon_t fragsize**
Fragment size of waveform data.
- **MHParse::int_mon_t wndlen**
Window length of spectral data.
- **MHParse::int_mon_t fftlen**
FFT length of spectral data.
- **MHParse::float_mon_t srate**
Sampling rate in Hz.

Additional Inherited Members

4.309.1 Constructor & Destructor Documentation

4.309.1.1 `mhaconfig_mon_t()` `MHAParser::mhaconfig_mon_t::mhaconfig_mon_t (`
`const std::string & help = "")`

4.309.2 Member Function Documentation

4.309.2.1 `update()` `void MHAParser::mhaconfig_mon_t::update (`
`const mhaconfig_t & cf)`

4.309.3 Member Data Documentation

4.309.3.1 `channels` `MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::channels`
[private]

Number of audio channels.

4.309.3.2 `domain` `MHAParser::string_mon_t MHAParser::mhaconfig_mon_t::domain` [private]

Signal domain (MHA_WAVEFORM or MHA_SPECTRUM)

4.309.3.3 `fragsize` `MHAParser::int_mon_t MHAParser::mhaconfig_mon_t::fragsize` [private]

Fragment size of waveform data.

4.309.3.4 wndlen `MHAParser::int_mon_t` MHAParser::mhaconfig_mon_t::wndlen [private]

Window length of spectral data.

4.309.3.5 fftlen `MHAParser::int_mon_t` MHAParser::mhaconfig_mon_t::fftlens [private]

FFT length of spectral data.

4.309.3.6 srate `MHAParser::float_mon_t` MHAParser::mhaconfig_mon_t::srate [private]

Sampling rate in Hz.

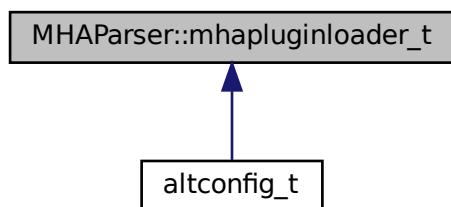
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.310 MHAParser::mhapluginloader_t Class Reference

Class to create a plugin loader in a parser, including the load logic.

Inheritance diagram for MHAParser::mhapluginloader_t:



Public Member Functions

- **mhapluginloader_t** (**MHAParser::parser_t** &parent, const **algo_comm_t** &ac, const std::string &plugname_name="plugin_name", const std::string &prefix="")
- ~**mhapluginloader_t** ()
- void **prepare** (**mhaconfig_t** &cf)
- void **release** ()
- void **process** (**mha_wave_t** *sIn, **mha_wave_t** **sOut)
- void **process** (**mha_spec_t** *sIn, **mha_spec_t** **sOut)
- void **process** (**mha_wave_t** *sIn, **mha_spec_t** **sOut)
- void **process** (**mha_spec_t** *sIn, **mha_wave_t** **sOut)
- **mhaconfig_t** **get_cfin** () const
- **mhaconfig_t** **get_cfout** () const
- const std::string & **get_last_name** () const

Protected Attributes

- **PluginLoader::mhapluginloader_t** * **plug**

Private Member Functions

- void **load_plug** ()

Private Attributes

- **MHAParser::parser_t** & **parent_**
- **MHAParser::string_t** **plugname**
- std::string **prefix_**
- **MHAEvents::connector_t< mhapluginloader_t >** **connector**
- **algo_comm_t** **ac_**
- std::string **last_name**
- std::string **plugname_name_**
- **mhaconfig_t** **cf_in_**
- **mhaconfig_t** **cf_out_**

Static Private Attributes

- static double **bookkeeping**

4.310.1 Detailed Description

Class to create a plugin loader in a parser, including the load logic.

4.310.2 Constructor & Destructor Documentation

4.310.2.1 mhaplugloader_t() MHParse::mhaplugloader_t::mhaplugloader_t (

```
    MHParse::parser_t & parent,
    const algo_comm_t & ac,
    const std::string & plugname_name = "plugin_name",
    const std::string & prefix = "" )
```

4.310.2.2 ~mhaplugloader_t() MHParse::mhaplugloader_t::~mhaplugloader_t (

```
)
```

4.310.3 Member Function Documentation

4.310.3.1 prepare() void MHParse::mhaplugloader_t::prepare (

```
    mhaconfig_t & cf )
```

4.310.3.2 release() void MHParse::mhaplugloader_t::release ()

4.310.3.3 process() [1/4] void MHParse::mhaplugloader_t::process (

```
    mha_wave_t * sIn,
    mha_wave_t ** sOut ) [inline]
```

4.310.3.4 process() [2/4] void MHParse::mhaplugloader_t::process (

```
    mha_spec_t * sIn,
    mha_spec_t ** sOut ) [inline]
```

4.310.3.5 process() [3/4] void MHAParser::mhapluginloader_t::process (
 mha_wave_t * sIn,
 mha_spec_t ** sOut) [inline]

4.310.3.6 process() [4/4] void MHAParser::mhapluginloader_t::process (
 mha_spec_t * sIn,
 mha_wave_t ** sOut) [inline]

4.310.3.7 get_cfin() mhaconfig_t MHAParser::mhapluginloader_t::get_cfin () const
[inline]

4.310.3.8 get_cfout() mhaconfig_t MHAParser::mhapluginloader_t::get_cfout () const
[inline]

4.310.3.9 get_last_name() const std::string& MHAParser::mhapluginloader_t::get_last_name () const [inline]

4.310.3.10 load_plug() void MHAParser::mhapluginloader_t::load_plug () [private]

4.310.4 Member Data Documentation

4.310.4.1 plug PluginLoader::mhapluginloader_t* MHAParser::mhapluginloader_t::plug
[protected]

4.310.4.2 parent_ `MHAParser::parser_t&` `MHAParser::mhaplugloader_t::parent_` \leftarrow
[private]

4.310.4.3 plugname `MHAParser::string_t` `MHAParser::mhaplugloader_t::plugname`
[private]

4.310.4.4 prefix_ `std::string` `MHAParser::mhaplugloader_t::prefix_` [private]

4.310.4.5 connector `MHAEevents::connector_t< mhaplugloader_t>` `MHAParser::mhaplugloader_t::connector` [private]

4.310.4.6 ac_ `algo_comm_t` `MHAParser::mhaplugloader_t::ac_` [private]

4.310.4.7 last_name `std::string` `MHAParser::mhaplugloader_t::last_name` [private]

4.310.4.8 plugname_name_ `std::string` `MHAParser::mhaplugloader_t::plugname_name_` \leftarrow
[private]

4.310.4.9 cf_in_ `mhaconfig_t` `MHAParser::mhaplugloader_t::cf_in_` [private]

4.310.4.10 cf_out_ `mhaconfig_t` `MHAParser::mhaplugloader_t::cf_out_` [private]

4.310.4.11 bookkeeping double MHAParser::mhapluginloader_t::bookkeeping [static],
[private]

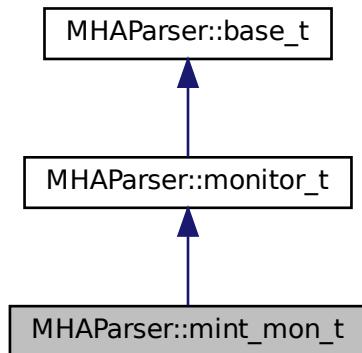
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

4.311 MHAParser::mint_mon_t Class Reference

Matrix of ints monitor.

Inheritance diagram for MHAParser::mint_mon_t:



Public Member Functions

- **mint_mon_t** (const std::string &hlp)
Create a matrix of integer monitor values.

Public Attributes

- std::vector< std::vector< int > > **data**
Data field.

Protected Member Functions

- std::string **query_val** (const std::string &)
- std::string **query_type** (const std::string &)

Additional Inherited Members

4.311.1 Detailed Description

Matrix of ints monitor.

4.311.2 Constructor & Destructor Documentation

4.311.2.1 mint_mon_t() MHAParser::mint_mon_t::mint_mon_t (const std::string & hlp)

Create a matrix of integer monitor values.

Parameters

<i>hlp</i>	A help text describing this monitor variable.
------------	---

4.311.3 Member Function Documentation

4.311.3.1 query_val() std::string MHAParser::mint_mon_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.311.3.2 query_type() std::string MHAParser::mint_mon_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.311.4 Member Data Documentation

4.311.4.1 **data** std::vector< std::vector<int> > MHParse::mint_mon_t::data

Data field.

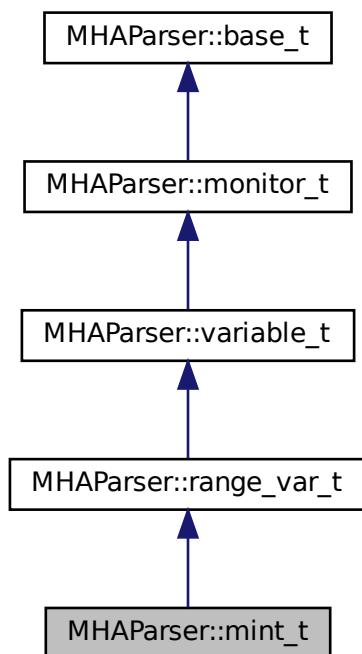
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.312 MHParse::mint_t Class Reference

Matrix variable with int value.

Inheritance diagram for MHParse::mint_t:



Public Member Functions

- **mint_t** (const std::string &, const std::string &, const std::string &="")

Create a int matrix parser variable.

Public Attributes

- std::vector< std::vector< int > > **data**

Data field.

Protected Member Functions

- std::string **op_setval** (**expression_t** &)
- std::string **query_type** (const std::string &)
- std::string **query_val** (const std::string &)

Additional Inherited Members

4.312.1 Detailed Description

Matrix variable with int value.

4.312.2 Constructor & Destructor Documentation

4.312.2.1 mint_t() MHAParser::mint_t::mint_t (

```
  const std::string & h,
  const std::string & v,
  const std::string & rg = "" )
```

Create a int matrix parser variable.

Parameters

<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.

4.312.3 Member Function Documentation

4.312.3.1 `op_setval()` `std::string MHAParser::mint_t::op_setval (expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.312.3.2 `query_type()` `std::string MHAParser::mint_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.312.3.3 `query_val()` `std::string MHAParser::mint_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.312.4 Member Data Documentation

4.312.4.1 `data` `std::vector<std::vector<int> > MHAParser::mint_t::data`

Data field.

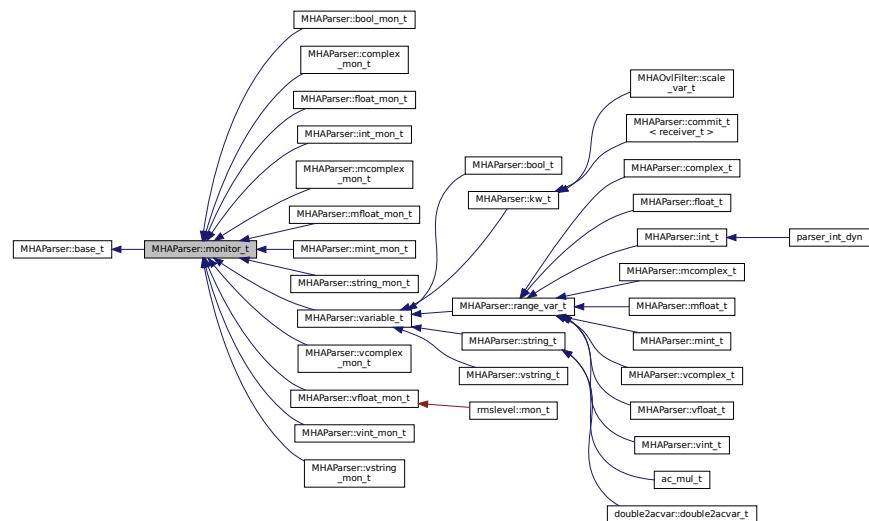
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.313 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

4.313.1 Detailed Description

Base class for monitors and variable nodes.

4.313.2 Constructor & Destructor Documentation

4.313.2.1 monitor_t() [1/2] `MHAParser::monitor_t::monitor_t (`
`const std::string & h)`

4.313.2.2 monitor_t() [2/2] `MHAParser::monitor_t::monitor_t (`
`const monitor_t & src)`

4.313.3 Member Function Documentation

4.313.3.1 op_query() `std::string MHAParser::monitor_t::op_query (`
`expression_t & x) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1032).

4.313.3.2 query_dump() `std::string MHAParser::monitor_t::query_dump (`
`const std::string & s) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1032).

4.313.3.3 query_perm() `std::string MHAParser::monitor_t::query_perm (`
`const std::string & s) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

Reimplemented in **MHAParser::variable_t** (p. 1114).

The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.314 MHParse::parser_t Class Reference

Parser node class.

Inherits **MHParse::base_t**.

Inherited by `alsa_dev_par_parser_t`, `AuditoryProfile::parser_t`, `AuditoryProfile::parser_t::ear_t`, `AuditoryProfile::parser_t::fmap_t`, `DynComp::dc_afterburn_vars_t`, `fw_t`, `io_alsa_t`, `io_asterisk_parser_t`, `io_file_t`, `io_parser_t`, `io_tcp_parser_t`, `MHAFilter::adapt_filter_t`, `MHAFilter::iir_filter_t`, `MHAIOJack::io_jack_t`, `MHAIOJackdb::io_jack_t`, `MHAIOPortAudio::device_info_t`, `MHAIOPortAudio::io_portaudio_t`, `MHParse::mhaconfig_mon_t`, `MHParse::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< acwriter_t >`, `MHAPlugIn::plugin_t< adaptive_feedback_canceller_config >`, `MHAPlugIn::plugin_t< adm_rtconfig_t >`, `MHAPlugIn::plugin_t< analysepath_t >`, `MHAPlugIn::plugin_t< cfg_t >`, `MHAPlugIn::plugin_t< char >`, `MHAPlugIn::plugin_t< cohfit_t >`, `MHAPlugIn::plugin_t< combc_t >`, `MHAPlugIn::plugin_t< cpupload_cfg_t >`, `MHAPlugIn::plugin_t< db_t >`, `MHAPlugIn::plugin_t< dbasync_t >`, `MHAPlugIn::plugin_t< dc_t >`, `MHAPlugIn::plugin_t< delaysum_t >`, `MHAPlugIn::plugin_t< delaysum_wave_t >`, `MHAPlugIn::plugin_t< doasvm_classification_config >`, `MHAPlugIn::plugin_t< doasvm_feature_extraction_config >`, `MHAPlugIn::plugin_t< example5_t >`, `MHAPlugIn::plugin_t< fftfb_plug_t >`, `MHAPlugIn::plugin_t< fftfbpow_t >`, `MHAPlugIn::plugin_t< fftfilter_t >`, `MHAPlugIn::plugin_t< fshift_config_t >`, `MHAPlugIn::plugin_t< gsc_adaptive_stage >`, `MHAPlugIn::plugin_t< gtfb_analyzer_cfg_t >`, `MHAPlugIn::plugin_t< gtfb_simd_cfg_t >`, `MHAPlugIn::plugin_t< gtfb_simple_rt_t >`, `MHAPlugIn::plugin_t< hilbert_shifter_t >`, `MHAPlugIn::plugin_t< int >`, `MHAPlugIn::plugin_t< level_matching_config_t >`, `MHAPlugIn::plugin_t< lpc_bl_predictor_config >`, `MHAPlugIn::plugin_t< lpc_burglattice_config >`, `MHAPlugIn::plugin_t< lpc_config >`, `MHAPlugIn::plugin_t< matlab_wrapper_rt_cfg_t >`, `MHAPlugIn::plugin_t< MHA_AC::spectrum_t >`, `MHAPlugIn::plugin_t< MHA_AC::waveform_t >`, `MHAPlugIn::plugin_t< mhachain::plugs_t >`, `MHAPlugIn::plugin_t< MHAFilter::partitioned_convolution_t >`, `MHAPlugIn::plugin_t< MHASignal::async_rmslevel_t >`, `MHAPlugIn::plugin_t< MHASignal::delay_t >`, `MHAPlugIn::plugin_t< MHASignal::waveform_t >`, `MHAPlugIn::plugin_t< MHAWindow::fun_t >`, `MHAPlugIn::plugin_t< noise_psd_estimator_t >`, `MHAPlugIn::plugin_t< overlapadd_t >`, `MHAPlugIn::plugin_t< plingploing_t >`, `MHAPlugIn::plugin_t< resampling_t >`, `MHAPlugIn::plugin_t< rmslevel_t >`, `MHAPlugIn::plugin_t< rohConfig >`, `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< smooth_cepstrum_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< wave2spec_t >`, `MHAPlugIn::plugin_t< wavewriter_t >`, `softclipper_variables_t`, `testplugin::ac_parser_t`, `testplugin::config_parser_t`, and `testplugin::signal_parser_t`.

Public Member Functions

- `parser_t (const std::string &help_text="")`

Construct detached node to be used in the configuration tree.

- **~parser_t ()**
- **void insert_item (const std::string &, base_t *)**
Register a parser item into this sub-parser.
- **void remove_item (const std::string &)**
Remove an item by name.
- **void force_remove_item (const std::string &)**
Remove an item by name.
- **void remove_item (const base_t *)**
Remove an item by address.

Protected Member Functions

- **std::string op_subparse (expression_t &)**
- **std::string op_setval (expression_t &)**
- **std::string op_query (expression_t &)**
- **std::string query_type (const std::string &)**
- **std::string query_dump (const std::string &)**
- **std::string query_entries (const std::string &)**
- **std::string query_readfile (const std::string &)**
- **std::string query_savefile (const std::string &)**
- **std::string query_savefile_compact (const std::string &)**
- **std::string query_savemons (const std::string &)**
- **std::string query_val (const std::string &)**
- **std::string query_listids (const std::string &)**
- **void set_id_string (const std::string &)**
- **bool has_entry (const std::string &)**

Private Attributes

- **entry_map_t entries**
- **std::string id_string**
identification string
- **std::string last_errormsg**

Additional Inherited Members

4.314.1 Detailed Description

Parser node class.

A **parser_t** (p. 1097) instance is a node in the configuration tree. A parser node can contain any number of other **parser_t** (p. 1097) instances or configuration language variables. These items are inserted into a parser node using the **parser_t::insert_item** (p. 1099) method.

4.314.2 Constructor & Destructor Documentation

4.314.2.1 parser_t() MHAParser::parser_t::parser_t (const std::string & help_text = "")

Construct detached node to be used in the configuration tree.

Parameters

<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.
------------------	--

4.314.2.2 ~parser_t() MHAParser::parser_t::~parser_t ()

4.314.3 Member Function Documentation

4.314.3.1 insert_item() void MHAParser::parser_t::insert_item (const std::string & *n*, MHAParser::base_t * *e*)

Register a parser item into this sub-parser.

This function registers an item under a given name into this sub-parser and makes it accessible to the parser interface.

Parameters

<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.

4.314.3.2 remove_item() [1/2] void MHAParser::parser_t::remove_item (const std::string & *n*)

Remove an item by name.

If the item does not exist, an error is being reported.

Parameters

<i>n</i>	Name of parser item to be removed from list.
----------	--

4.314.3.3 force_remove_item() void MHAParser::parser_t::force_remove_item (const std::string & *n*)

Remove an item by name.

Non-existing items are ignored.

Parameters

<i>n</i>	Name of parser item to be removed from list.
----------	--

4.314.3.4 remove_item() [2/2] void MHAParser::parser_t::remove_item (const **base_t** * *addr*)

Remove an item by address.

The item belonging to an address is being removed from the list of items.

Parameters

<i>addr</i>	Address of parser item to be removed.
-------------	---------------------------------------

4.314.3.5 op_subparse() std::string MHAParser::parser_t::op_subparse (**expression_t** & *x*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.314.3.6 op_setval() std::string MHAParser::parser_t::op_setval (
 expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.314.3.7 op_query() std::string MHAParser::parser_t::op_query (
 expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.314.3.8 query_type() std::string MHAParser::parser_t::query_type (
 const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.314.3.9 query_dump() std::string MHAParser::parser_t::query_dump (
 const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.314.3.10 query_entries() std::string MHAParser::parser_t::query_entries (
 const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1032).

4.314.3.11 query_readfile() std::string MHAParser::parser_t::query_readfile (const std::string & fname) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1034).

4.314.3.12 query_savefile() std::string MHAParser::parser_t::query_savefile (const std::string & fname) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1034).

4.314.3.13 query_savefile_compact() std::string MHAParser::parser_t::query_↔ savefile_compact (const std::string & fname) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1034).

4.314.3.14 query_savemons() std::string MHAParser::parser_t::query_savemons (const std::string & fname) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1034).

4.314.3.15 query_val() std::string MHAParser::parser_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.314.3.16 query_listids() std::string MHAParser::parser_t::query_listids (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1034).

4.314.3.17 set_id_string() void MHAParser::parser_t::set_id_string (const std::string & s) [protected]

4.314.3.18 has_entry() bool MHAParser::parser_t::has_entry (const std::string & s) [protected]

4.314.4 Member Data Documentation

4.314.4.1 entries entry_map_t MHAParser::parser_t::entries [private]

4.314.4.2 id_string std::string MHAParser::parser_t::id_string [private]

identification string

4.314.4.3 last_errormsg std::string MHAParser::parser_t::last_errormsg [private]

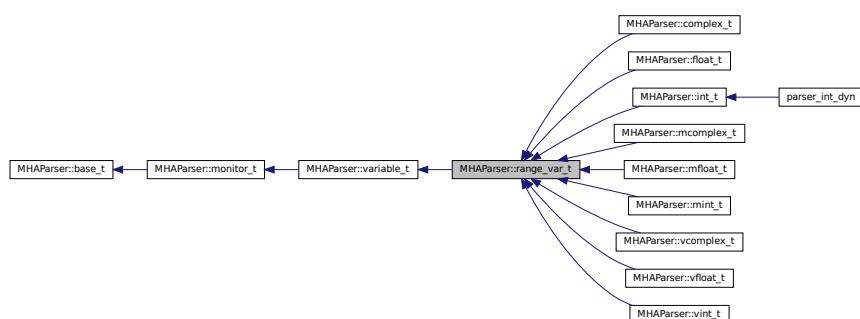
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.315 MHAParser::range_var_t Class Reference

Base class for all variables with a numeric value range.

Inheritance diagram for MHAParser::range_var_t:



Public Member Functions

- **range_var_t** (const std::string &, const std::string &="")
- **range_var_t** (const **range_var_t** &)
- std::string **query_range** (const std::string &)
- void **set_range** (const std::string &r)
Change the valid range of a variable.
- void **validate** (const int &)
- void **validate** (const float &)
- void **validate** (const **mha_complex_t** &)
- void **validate** (const std::vector< int > &)
- void **validate** (const std::vector< float > &)
- void **validate** (const std::vector< **mha_complex_t** > &)
- void **validate** (const std::vector< std::vector< int > > &)
- void **validate** (const std::vector< std::vector< float > > &)
- void **validate** (const std::vector< std::vector< **mha_complex_t** > > &)

Protected Attributes

- float **low_limit**
Lower limit of range.
- float **up_limit**
Upper limit of range.
- bool **low_incl**
Lower limit is included (or excluded) in range.
- bool **up_incl**
Upper limit is included (or excluded) in range.
- bool **check_low**
Check lower limit.
- bool **check_up**
Check upper limit.
- bool **check_range**
Range checking is active.

Additional Inherited Members

4.315.1 Detailed Description

Base class for all variables with a numeric value range.

4.315.2 Constructor & Destructor Documentation

4.315.2.1 range_var_t() [1/2] MHParse::range_var_t::range_var_t (

```
const std::string & h,  
const std::string & r = "")
```

4.315.2.2 range_var_t() [2/2] MHParse::range_var_t::range_var_t (

```
const range_var_t & src)
```

4.315.3 Member Function Documentation

4.315.3.1 query_range() std::string MHParse::range_var_t::query_range (

```
const std::string & s) [virtual]
```

Reimplemented from **MHParse::base_t** (p. 1033).

4.315.3.2 set_range() void MHParse::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)
----------	---

4.315.3.3 validate() [1/9] void MHParse::range_var_t::validate (

```
const int & v)
```

4.315.3.4 validate() [2/9] void MHAParser::range_var_t::validate (const float & v)

4.315.3.5 validate() [3/9] void MHAParser::range_var_t::validate (const mha_complex_t & v)

4.315.3.6 validate() [4/9] void MHAParser::range_var_t::validate (const std::vector< int > & v)

4.315.3.7 validate() [5/9] void MHAParser::range_var_t::validate (const std::vector< float > & v)

4.315.3.8 validate() [6/9] void MHAParser::range_var_t::validate (const std::vector< mha_complex_t > & v)

4.315.3.9 validate() [7/9] void MHAParser::range_var_t::validate (const std::vector< std::vector< int > > & v)

4.315.3.10 validate() [8/9] void MHAParser::range_var_t::validate (const std::vector< std::vector< float > > & v)

4.315.3.11 validate() [9/9] void MHAParser::range_var_t::validate (const std::vector< std::vector< mha_complex_t > > & v)

4.315.4 Member Data Documentation

4.315.4.1 low_limit float MHAParser::range_var_t::low_limit [protected]

Lower limit of range.

4.315.4.2 up_limit float MHAParser::range_var_t::up_limit [protected]

Upper limit of range.

4.315.4.3 low_incl bool MHAParser::range_var_t::low_incl [protected]

Lower limit is included (or excluded) in range.

4.315.4.4 up_incl bool MHAParser::range_var_t::up_incl [protected]

Upper limit is included (or excluded) in range.

4.315.4.5 check_low bool MHAParser::range_var_t::check_low [protected]

Check lower limit.

4.315.4.6 check_up bool MHAParser::range_var_t::check_up [protected]

Check upper limit.

4.315.4.7 check_range bool MHAParser::range_var_t::check_range [protected]

Range checking is active.

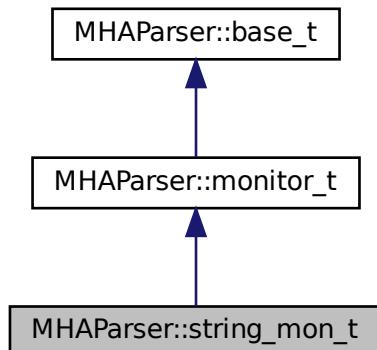
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.316 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

4.316.1 Detailed Description

Monitor with string value.

4.316.2 Constructor & Destructor Documentation

4.316.2.1 **string_mon_t()** 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.
------------	---

4.316.3 Member Function Documentation

4.316.3.1 **query_val()** std::string MHAParser::string_mon_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.316.3.2 **query_type()** std::string MHAParser::string_mon_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.316.4 Member Data Documentation

4.316.4.1 **data** std::string MHAParser::string_mon_t::data

Data field.

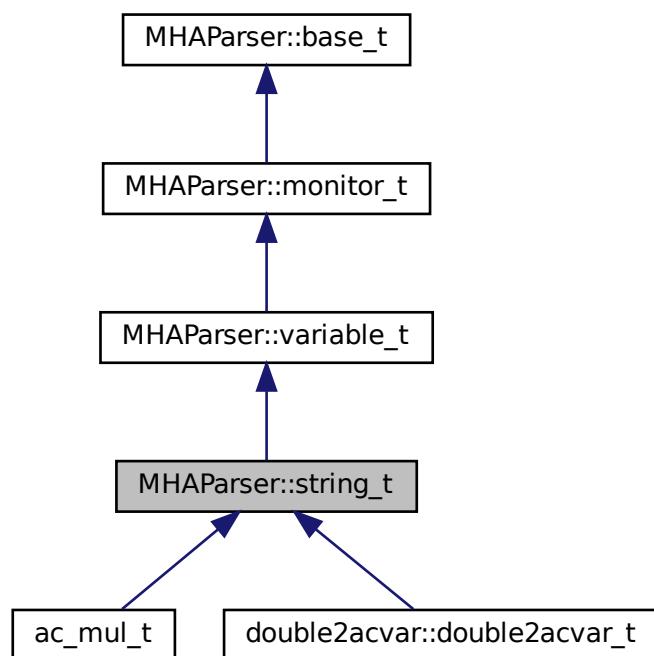
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.317 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

4.317.1 Detailed Description

Variable with a string value.

4.317.2 Constructor & Destructor Documentation

4.317.2.1 **string_t()** MHAParser::string_t::string_t (

```
const std::string & h,  
const std::string & v )
```

Constructor of a openMHA configuration variable for string values.

Parameters

<i>h</i>	A help string describing the purpose of this variable.
<i>v</i>	The initial string value

4.317.3 Member Function Documentation

4.317.3.1 `op_setval()` `std::string MHAParser::string_t::op_setval (expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.317.3.2 `query_type()` `std::string MHAParser::string_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.317.3.3 `query_val()` `std::string MHAParser::string_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.317.4 Member Data Documentation

4.317.4.1 `data` `std::string MHAParser::string_t::data`

Data field.

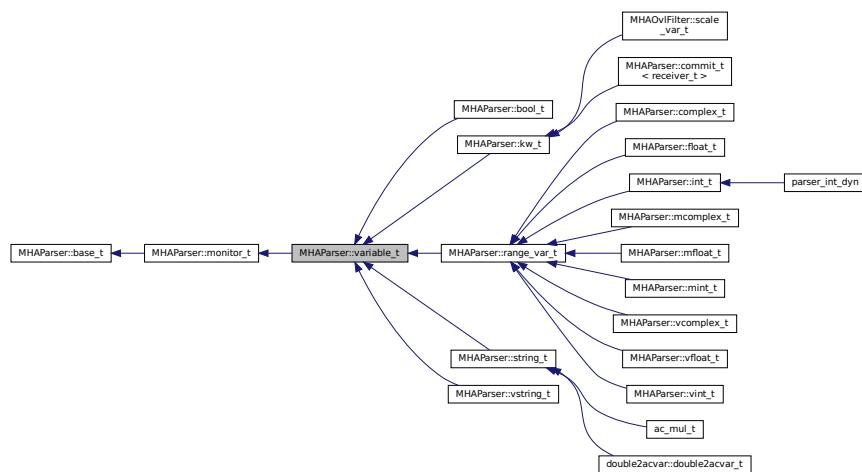
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.318 MHParse::variable_t Class Reference

Base class for variable nodes.

Inheritance diagram for MHParse::variable_t:



Public Member Functions

- **variable_t** (const std::string &)
- std::string **op_setval** (expression_t &)
- std::string **query_perm** (const std::string &)
- void **setlock** (const bool &)

Lock a variable against write access.

Private Attributes

- bool **locked**

Additional Inherited Members

4.318.1 Detailed Description

Base class for variable nodes.

4.318.2 Constructor & Destructor Documentation

4.318.2.1 variable_t() `MHAParser::variable_t::variable_t (const std::string & h)`

4.318.3 Member Function Documentation

4.318.3.1 op_setval() `std::string MHAParser::variable_t::op_setval (expression_t & x) [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1032).

Reimplemented in **MHAParser::mcomplex_t** (p. 1077), **MHAParser::mfloat_t** (p. 1082), **MHAParser::mint_t** (p. 1094), **MHAParser::vcomplex_t** (p. 1118), **MHAParser::vfloat_t** (p. 1124), **MHAParser::vint_t** (p. 1128), **MHAParser::complex_t** (p. 1053), **MHAParser::float_t** (p. 1060), **MHAParser::int_t** (p. 1065), **MHAParser::bool_t** (p. 1044), **MHAParser::vstring_t** (p. 1132), **MHAParser::string_t** (p. 1112), and **MHAParser::kw_t** (p. 1072).

4.318.3.2 query_perm() `std::string MHAParser::variable_t::query_perm (const std::string & s) [virtual]`

Reimplemented from **MHAParser::monitor_t** (p. 1096).

4.318.3.3 setlock() `void MHAParser::variable_t::setlock (const bool & b)`

Lock a variable against write access.

Parameters

<code>b</code>	Lock state
----------------	------------

4.318.4 Member Data Documentation

4.318.4.1 **locked** bool MHAParser::variable_t::locked [private]

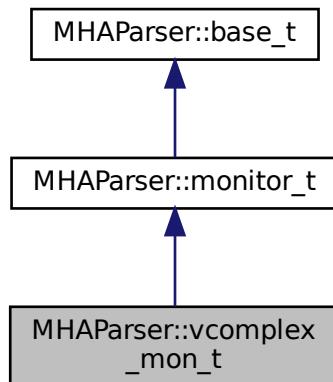
The documentation for this class was generated from the following files:

- `mha_parser.hh`
- `mha_parser.cpp`

4.319 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

4.319.1 Detailed Description

Monitor with vector of complex values.

4.319.2 Constructor & Destructor Documentation

4.319.2.1 vcomplex_mon_t() 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.
------------	---

4.319.3 Member Function Documentation

4.319.3.1 query_val() std::string MHAParser::vcomplex_mon_t::query_val (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.319.3.2 query_type() std::string MHAParser::vcomplex_mon_t::query_type (const std::string & *s*) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.319.4 Member Data Documentation

4.319.4.1 **data** std::vector< **mha_complex_t**> MHAParser::vcomplex_mon_t::data

Data field.

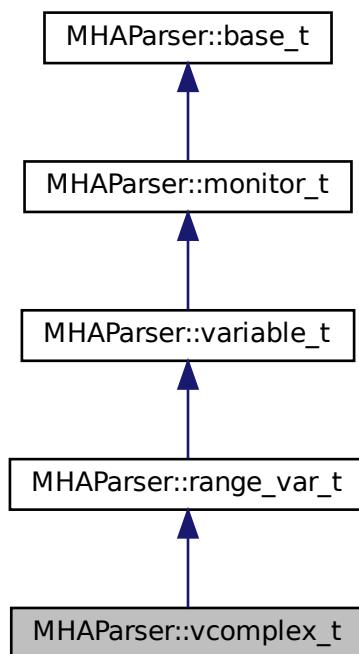
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.320 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

4.320.1 Detailed Description

Vector variable with complex value.

4.320.2 Constructor & Destructor Documentation

4.320.2.1 vcomplex_t() MHAParser::vcomplex_t::vcomplex_t (

```
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

4.320.3 Member Function Documentation

4.320.3.1 op_setval() std::string MHAParser::vcomplex_t::op_setval (expression_t & x) [protected], [virtual]

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.320.3.2 query_type() std::string MHAParser::vcomplex_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.320.3.3 query_val() std::string MHAParser::vcomplex_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.320.4 Member Data Documentation

4.320.4.1 data std::vector< mha_complex_t> MHAParser::vcomplex_t::data

Data field.

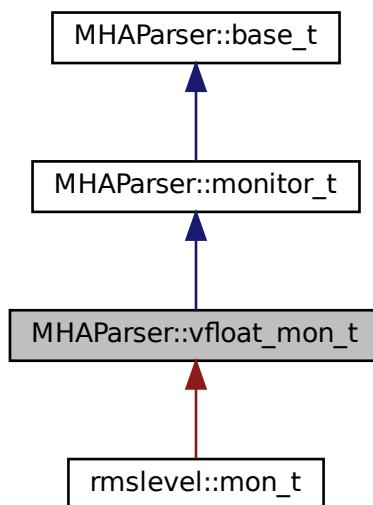
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.321 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

4.321.1 Detailed Description

Vector of floats monitor.

4.321.2 Constructor & Destructor Documentation

4.321.2.1 vfloat_mon_t() `MHAParser::vfloat_mon_t::vfloat_mon_t (const std::string & hlp)`

Create a vector of floating point monitor values.

Parameters

<code>hlp</code>	A help text describing this monitor variable.
------------------	---

4.321.3 Member Function Documentation

4.321.3.1 query_val() `std::string MHAParser::vfloat_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.321.3.2 query_type() `std::string MHAParser::vfloat_mon_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.321.4 Member Data Documentation

4.321.4.1 data std::vector<float> MHAParser::vfloat_mon_t::data

Data field.

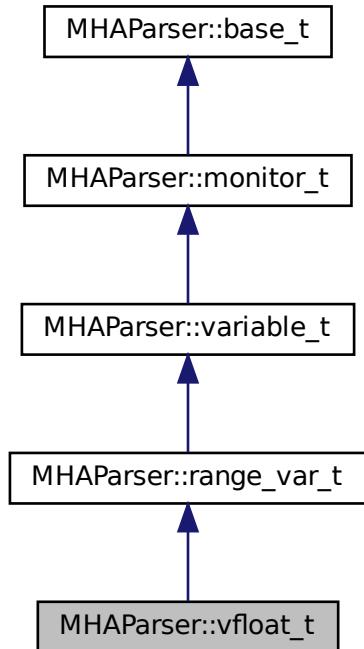
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.322 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

4.322.1 Detailed Description

Vector variable with float value.

4.322.2 Constructor & Destructor Documentation

```
4.322.2.1 vfloat_t() MHAParser::vfloat_t::vfloat_t (
    const std::string & h,
    const std::string & v,
    const std::string & rg = "" )
```

Create a float vector parser variable.

Parameters

<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.

•

4.322.3 Member Function Documentation

4.322.3.1 `op_setval()` `std::string MHAParser::vfloat_t::op_setval (expression_t & x) [protected], [virtual]`

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.322.3.2 `query_type()` `std::string MHAParser::vfloat_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.322.3.3 `query_val()` `std::string MHAParser::vfloat_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.322.4 Member Data Documentation

4.322.4.1 `data` `std::vector<float> MHAParser::vfloat_t::data`

Data field.

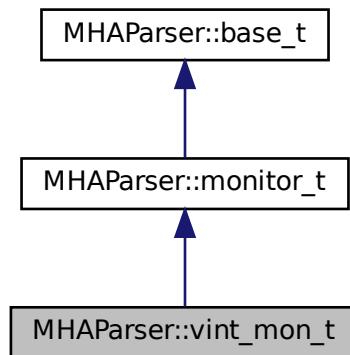
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.323 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

4.323.1 Detailed Description

Vector of ints monitor.

4.323.2 Constructor & Destructor Documentation

4.323.2.1 `vint_mon_t()` `MHAParser::vint_mon_t::vint_mon_t (const std::string & hlp)`

Create a vector of integer monitor values.

Parameters

<code>hlp</code>	A help text describing this monitor variable.
------------------	---

4.323.3 Member Function Documentation

4.323.3.1 `query_val()` `std::string MHAParser::vint_mon_t::query_val (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.323.3.2 `query_type()` `std::string MHAParser::vint_mon_t::query_type (const std::string & s) [protected], [virtual]`

Reimplemented from **MHAParser::base_t** (p. 1033).

4.323.4 Member Data Documentation

4.323.4.1 `data` `std::vector<int> MHAParser::vint_mon_t::data`

Data field.

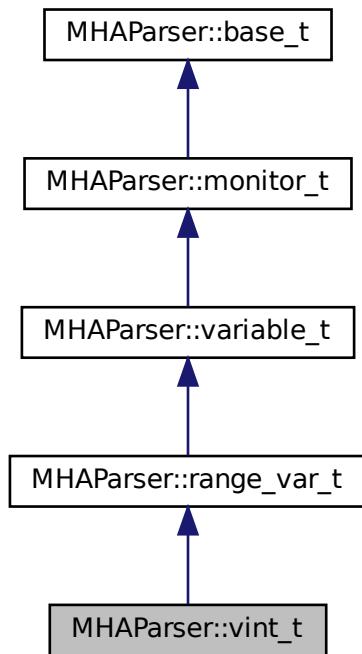
The documentation for this class was generated from the following files:

- [mha_parser.hh](#)
- [mha_parser.cpp](#)

4.324 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

4.324.1 Detailed Description

Variable with vector<int> value.

4.324.2 Constructor & Destructor Documentation

4.324.2.1 `vint_t()` MHAParser::vint_t::vint_t (

```
const std::string & h,
const std::string & v,
const std::string & rg = "")
```

Constructor.

Parameters

<code>h</code>	help string
<code>v</code>	initial value
<code>rg</code>	optional: range constraint for all elements

4.324.3 Member Function Documentation

4.324.3.1 `op_setval()` std::string MHAParser::vint_t::op_setval (

```
expression_t & x) [protected], [virtual]
```

Reimplemented from `MHAParser::variable_t` (p. 1114).

4.324.3.2 `query_type()` std::string MHAParser::vint_t::query_type (

```
const std::string & s) [protected], [virtual]
```

Reimplemented from `MHAParser::base_t` (p. 1033).

4.324.3.3 query_val() std::string MHAParser::vint_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.324.4 Member Data Documentation

4.324.4.1 data std::vector<int> MHAParser::vint_t::data

Data field.

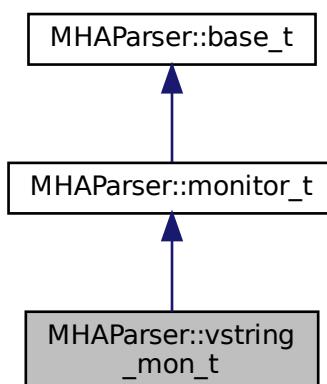
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.325 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

4.325.1 Detailed Description

Vector of monitors with string value.

4.325.2 Constructor & Destructor Documentation

4.325.2.1 **vstring_mon_t()** 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.
------------	---

4.325.3 Member Function Documentation

4.325.3.1 query_val() std::string MHAParser::vstring_mon_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.325.3.2 query_type() std::string MHAParser::vstring_mon_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.325.4 Member Data Documentation

4.325.4.1 data std::vector<std::string> MHAParser::vstring_mon_t::data

Data field.

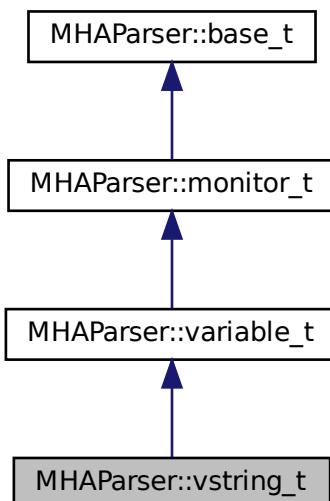
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.326 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

4.326.1 Detailed Description

Vector variable with string values.

4.326.2 Constructor & Destructor Documentation

4.326.2.1 **vstring_t()** MHAParser::vstring_t::vstring_t (

```
const std::string & h,  
const std::string & v )
```

4.326.3 Member Function Documentation

4.326.3.1 **op_setval()** std::string MHAParser::vstring_t::op_setval (

```
expression_t & x ) [protected], [virtual]
```

Reimplemented from **MHAParser::variable_t** (p. 1114).

4.326.3.2 query_type() std::string MHAParser::vstring_t::query_type (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.326.3.3 query_val() std::string MHAParser::vstring_t::query_val (const std::string & s) [protected], [virtual]

Reimplemented from **MHAParser::base_t** (p. 1033).

4.326.4 Member Data Documentation

4.326.4.1 data std::vector<std::string> MHAParser::vstring_t::data

Data field.

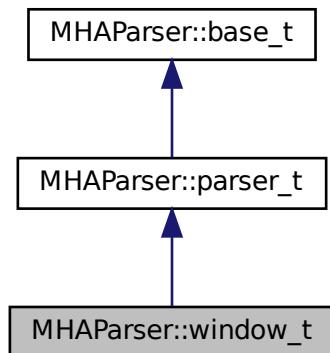
The documentation for this class was generated from the following files:

- **mha_parser.hh**
- **mha_parser.cpp**

4.327 MHAParser::window_t Class Reference

MHA configuration interface for a window function generator.

Inheritance diagram for MHAParser::window_t:



Public Types

- enum **wtype_t** {
 wnd_rect =0, **wnd_hann** =1, **wnd_hamming** =2, **wnd_blackman** =3,
wnd_bartlett =4, **wnd_user** =5 }

Public Member Functions

- **window_t** (const std::string & **help**="Window type configuration.")
Constructor to create parser class.
- **MHAWindow::base_t get_window** (unsigned int len) const
Create a window instance, use default parameters.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool min-included) const
Create a window instance.
- **MHAWindow::base_t get_window** (unsigned int len, float xmin, float xmax, bool min-included, bool maxincluded) const
Create a window instance.
- **MHAParser::window_t::wtype_t get_type** () const
Return currently selected window type.
- void **setlock** (bool b)

Private Attributes

- **MHAParser::kw_t wtype**
- **MHAParser::vfloat_t user**

Additional Inherited Members

4.327.1 Detailed Description

MHA configuration interface for a window function generator.

This class implements a configuration interface (sub-parser) for window type selection and user-defined window type. It provides member functions to generate an instance of **MHAWindow::base_t** (p. 1287) based on the values provided by the configuration interface.

The configuration interface is derived from **MHAParser::parser_t** (p. 1097) and can thus be inserted into the configuration tree using the **insert_item()** (p. 1099) method of the parent parser.

If one of the pre-defined window types is used, then the window is generated using the **MHAWindow::fun_t** (p. 1291) class constructor; for the user-defined type the values from the "user" variable are copied.

4.327.2 Member Enumeration Documentation

4.327.2.1 wtype_t enum MHAParser::window_t::wtype_t

Enumerator

wnd_rect	
wnd_hann	
wnd_hamming	
wnd_blackman	
wnd_bartlett	
wnd_user	

4.327.3 Constructor & Destructor Documentation

4.327.3.1 window_t() MHAParser::window_t::window_t (

```
const std::string & help = "Window type configuration." )
```

Constructor to create parser class.

4.327.4 Member Function Documentation

4.327.4.1 get_window() [1/5] MHAWindow::base_t MHAParser::window_t::get_window (

```
unsigned int len ) const
```

Create a window instance, use default parameters.

4.327.4.2 get_window() [2/5] `MHAWindow::base_t MHAParser::window_t::get_window (`
 `unsigned int len,`
 `float xmin) const`

Create a window instance.

4.327.4.3 get_window() [3/5] `MHAWindow::base_t MHAParser::window_t::get_window (`
 `unsigned int len,`
 `float xmin,`
 `float xmax) const`

Create a window instance.

4.327.4.4 get_window() [4/5] `MHAWindow::base_t MHAParser::window_t::get_window (`
 `unsigned int len,`
 `float xmin,`
 `float xmax,`
 `bool minincluded) const`

Create a window instance.

4.327.4.5 get_window() [5/5] `MHAWindow::base_t MHAParser::window_t::get_window (`
 `unsigned int len,`
 `float xmin,`
 `float xmax,`
 `bool minincluded,`
 `bool maxincluded) const`

Create a window instance.

4.327.4.6 get_type() `MHAParser::window_t::wtype_t MHAParser::window_t::get_type (`
 `) const`

Return currently selected window type.

```
4.327.4.7 setlock() void MHAParser::window_t::setlock (
    bool b ) [inline]
```

4.327.5 Member Data Documentation

4.327.5.1 wtype `MHAParser::kw_t` MHAParser::window_t::wtype [private]

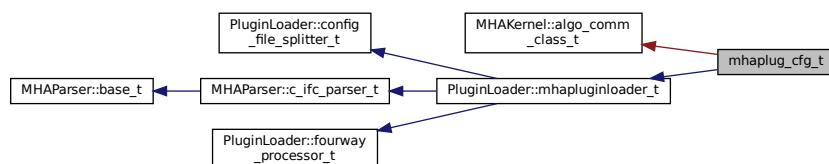
4.327.5.2 user `MHAParser::vfloat_t` MHAParser::window_t::user [private]

The documentation for this class was generated from the following files:

- `mha_windowparser.h`
- `mha_windowparser.cpp`

4.328 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

4.328.1 Constructor & Destructor Documentation

```
4.328.1.1 mhaplug_cfg_t() mhaplug_cfg_t::mhaplug_cfg_t (
    algo_comm_t iac,
    const std::string & libname,
    bool use_own_ac )
```

4.328.1.2 ~mhaplug_cfg_t() mhaplug_cfg_t::~mhaplug_cfg_t () throw () [inline]

The documentation for this class was generated from the following file:

- **altplugs.cpp**

4.329 MHAPlugin::cfg_node_t< runtime_cfg_t > Class Template Reference

A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.

Public Member Functions

- **cfg_node_t (runtime_cfg_t *runtime_cfg)**
Constructor for a singly linked list node.
- **~cfg_node_t ()**
Destructor of the singly linked list node.

Public Attributes

- std::atomic< **cfg_node_t< runtime_cfg_t > *next**
A pointer to the next node in the singly linked list.
- std::atomic< bool > **not_in_use**
Initially this data member is set to false by the constructor.
- runtime_cfg_t * **data**
A native pointer to the runtime configuration managed by this node.

4.329.1 Detailed Description

```
template<class runtime_cfg_t>
class MHAPlugin::cfg_node_t< runtime_cfg_t >
```

A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.

The singly linked list is designed for a single producer thread and a single consumer thread, where the producer is also responsible for destroying objects when they are no longer needed because the consumer is the signal processing thread that cannot afford memory allocation or deallocation operations.

4.329.2 Constructor & Destructor Documentation

```
4.329.2.1 cfg_node_t() template<class runtime_cfg_t >
MHAPlugin::cfg_node_t< runtime_cfg_t >:: cfg_node_t (
    runtime_cfg_t * runtime_cfg ) [explicit]
```

Constructor for a singly linked list node.

Parameters

<i>runtime_cfg</i>	Pointer to a runtime configuration object that was just created on the heap. The newly constructed cfg_node_t (p. 1138) object takes over object ownership of the pointed-to runtime configuration and will call delete on it in its destructor.
--------------------	---

```
4.329.2.2 ~cfg_node_t() template<class runtime_cfg_t >
MHAPlugin::cfg_node_t< runtime_cfg_t >::~ cfg_node_t
```

Destructor of the singly linked list node.

Will also delete the pointed-to data object (the runtime configuration)

4.329.3 Member Data Documentation

```
4.329.3.1 next template<class runtime_cfg_t >
std::atomic< cfg_node_t<runtime_cfg_t*>*> MHAPPlugin::cfg_node_t< runtime_cfg_t >::next
```

A pointer to the next node in the singly linked list.

On construction, this will be a NULL pointer. New objects can be appended to the singly linked list by writing the address to the next node into this data member. Since this pointer is std::atomic, writing to it is a release operation which means all threads seeing the new value of the next pointer will also see all other writes to memory that the thread doing this write has performed, which includes anything the constructor of the new runtime config has written and the assignment of the address of the new runtime config to the data pointer of the next node. This prevents making half-constructed runtime configuration objects visible to the consumer thread.

```
4.329.3.2 not_in_use template<class runtime_cfg_t >
std::atomic<bool> MHAPPlugin::cfg_node_t< runtime_cfg_t >::not_in_use
```

Initially this data member is set to false by the constructor.

It is set to true by the consumer thread when it no longer needs the run time configuration pointed to by this node's data member. This bool is atomic because this node and the runtime object it points to can be deleted by the configuration thread as soon as value true is stored in this bool, which happens right after the processing thread acquires a pointer to the next node in the singly linked list. The atomic ensures all threads agree on this "right after" ordering.

```
4.329.3.3 data template<class runtime_cfg_t >
runtime_cfg_t* MHAPPlugin::cfg_node_t< runtime_cfg_t >::data
```

A native pointer to the runtime configuration managed by this node.

The runtime configuration lives on the heap and is owned by this node. It is deleted in this node's destructor. The runtime configuration object must be created by client code with operator new before ownership is transferred to this node by passing a pointer to it as the constructor's parameter. This pointer does not need to be atomic, memory access ordering is ensured by the atomic next pointer and placing accesses to "data" in the correct places relative to accesses to "next".

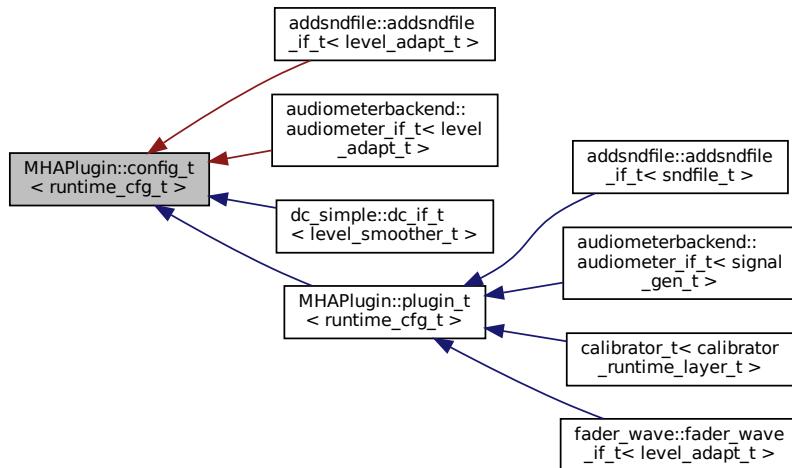
The documentation for this class was generated from the following file:

- [mha_plugin.hh](#)

4.330 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 * peek_config () const**
Receive the latest run time configuration without changing the configuration pointer.
- **void push_config (runtime_cfg_t *ncfg)**
Push a new run time configuration into the configuration fifo.
- **void cleanup_unused_cfg ()**
*To be called by the **push_config()** (p. 1145) for housekeeping.*
- **void remove_all_cfg ()**
To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in_use flag.

Protected Attributes

- runtime_cfg_t * **cfg**

Pointer to the runtime configuration currently used by the signal processing thread.

Private Attributes

- std::atomic< MHAPlugin::cfg_node_t< runtime_cfg_t > * > **cfg_root**
Start of a singly linked list of runtime configuration objects.
- MHAPlugin::cfg_node_t< runtime_cfg_t > * **cfg_node_current**
Pointer to the currently used plugin runtime configurations.

4.330.1 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. 122) 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. 1145). In the processing thread, the actual configuration can be received by a call of **poll_config()** (p. 1144).

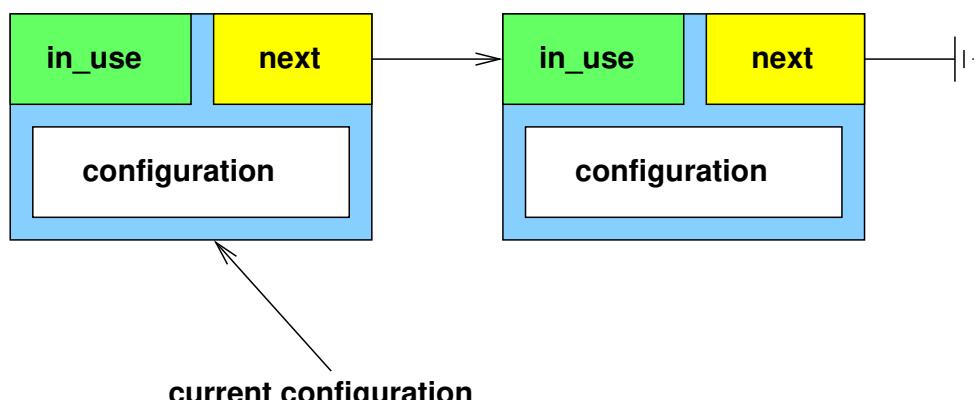


Figure 5 Schematic drawing of runtime configuration update: configuration updated, but not used yet.

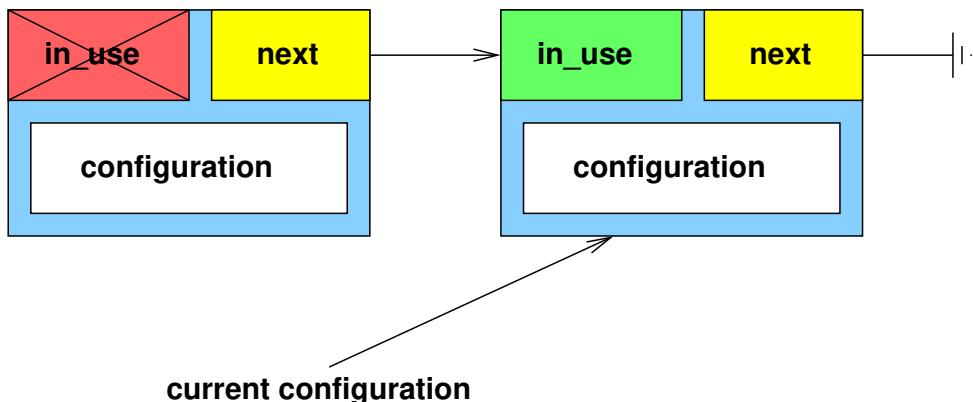


Figure 6 Schematic drawing of runtime configuration update: configuration in use.

To ensure lock-free thread safety, we use C++11 atomics and rely on the C++11 memory model. We only use store-release and load-acquire operations by using C++11 atomics with the default memory ordering. The semantics of these are:

The store-release operation atomically writes to an atomic variable, while the load-acquire operation atomically reads from an atomic variable.

The C++11 memory model guarantees that all previous writes to memory performed by the thread doing the store-release are visible to other threads when they see the new value in the shared atomic variable when that value is read by the other thread with a load-acquire operation.

An important precondition of this synchronization scheme is that there is only ever one audio thread and one configuration thread per plugin, i.e. there is only one thread doing the `push_config` and one thread doing the `poll_config` for each instance of `config_t` (p. 1141).

For more details on atomics, refer to the C++11 or later documentation, or to these conference talks by Sutter:

- Atomic Weapons, 2012
- Lock-Free Programming, 2014

4.330.2 Constructor & Destructor Documentation

4.330.2.1 config_t() template<class runtime_cfg_t > MHAPlugin::config_t< runtime_cfg_t >:: config_t

```
4.330.2.2 ~config_t() template<class runtime_cfg_t >
MHAPlugIn::config_t< runtime_cfg_t >::~ config_t
```

4.330.3 Member Function Documentation

```
4.330.3.1 poll_config() template<class runtime_cfg_t >
runtime_cfg_t * MHAPlugIn::config_t< runtime_cfg_t >::poll_config [protected]
```

Receive the latest run time configuration.

This function stores the latest run time configuration into the protected class member variable 'cfg'. If no configuration exists, then an exception will be thrown. If no changes occurred, then the value of 'cfg' will be untouched.

This function should be only called from the *processing* thread.

Should be called at the start of each process() callback to get the latest runtime configuration.

When this function finds newer run time configurations, it returns the newest and ensures the older run time configurations have their not_in_use flag set to true.

Returns

Pointer to the latest runtime configuration object (same pointer as stored by this function in data member 'cfg').

Exceptions

MHA_Error (p. 763)	if the resulting runtime configuration is NULL. This usually means that no push_config has occurred.
--	--

```
4.330.3.2 peek_config() template<class runtime_cfg_t >
runtime_cfg_t * MHAPlugIn::config_t< runtime_cfg_t >::peek_config [protected]
```

Receive the latest run time configuration without changing the configuration pointer.

This function retrieves the latest run time configuration. Returns a pointer to the latest runtime configuration without updating the data member cfg. For use in the configuration thread when

creation of a new runtime configuration object needs access to the previously created runtime configuration object. Should normally not be used because it introduces synchronization requirements between configuration thread and signal processing thread.

4.330.3.3 push_config() template<class runtime_cfg_t >
 void MHAPlugin::config_t< runtime_cfg_t >::push_config (
 runtime_cfg_t * ncfg) [protected]

Push a new run time configuration into the configuration fifo.

Should be called only by the configuration thread when a new runtime configuration object has been constructed in response to configuration changes, or during execution of the prepare() method to ensure that there is a valid runtime configuration for the signal processing which can start after prepare() returns.

For housekeeping, this method will also delete any runtime configuration objects that have previously been passed to **push_config()** (p. 1145) if they are no longer needed.

Parameters

<i>ncfg</i>	A pointer to the new runtime configuration object. This object must have been created on the heap by the configuration thread with operator new. By passing the pointer to this method, client code gives up ownership. The object will be deleted in a future invocation of push_config, or on destruction of this config_t (p. 1141) instance.
-------------	---

4.330.3.4 cleanup_unused_cfg() template<class runtime_cfg_t >
 void MHAPlugin::config_t< runtime_cfg_t >::cleanup_unused_cfg [protected]

To be called by the **push_config()** (p. 1145) for housekeeping.

Will delete any no longer used runtime configuration objects.

4.330.3.5 remove_all_cfg() template<class runtime_cfg_t >
 void MHAPlugin::config_t< runtime_cfg_t >::remove_all_cfg [protected]

To be called on Plugin destruction, will delete all runtime configuration list nodes and objects regardless of their in_use flag.

4.330.4 Member Data Documentation

4.330.4.1 cfg template<class runtime_cfg_t >
`runtime_cfg_t* MHAPlugin::config_t< runtime_cfg_t >::cfg [protected]`

Pointer to the runtime configuration currently used by the signal processing thread.

Should be used to access the current runtime configuration during signal processing. This pointer is updated as a side effect of calling `poll_config()` (p. 1144) on this object.

4.330.4.2 cfg_root template<class runtime_cfg_t >
`std::atomic< MHAPlugin::cfg_node_t<runtime_cfg_t> *> MHAPlugin::config_t< runtime_cfg_t >::cfg_root [private]`

Start of a singly linked list of runtime configuration objects.

`cfg_root` points to the oldest still existing node of that list. After object creation this pointer is updated by the configuration thread and then read by the signal processing thread. To ensure proper order of memory accesses for this transfer between threads, it needs to be atomic, this ensures that the start of the singly linked list of runtime configurations will be properly visible to the signal processing on startup.

4.330.4.3 cfg_node_current template<class runtime_cfg_t >
`MHAPlugin::cfg_node_t<runtime_cfg_t>* MHAPlugin::config_t< runtime_cfg_t >::cfg_node_current [private]`

Pointer to the currently used plugin runtime configurations.

Used as a hint for `poll_config` where to start looking for the newest node. This optimization allows `poll_config` to scale better with the number of nodes not yet cleaned up by `push_config`. Does not need to be atomic because it is only used within the signal processing thread.

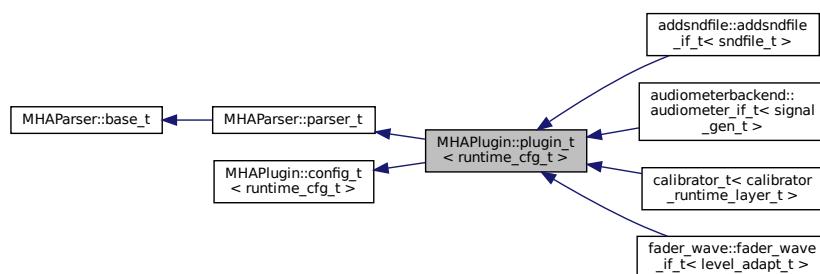
The documentation for this class was generated from the following file:

- `mha_plugin.hh`

4.331 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

4.331.1 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.
----------------------	-------------------------

This template class provides thread safe configuration handling and standard methods to be compatible to the C++ openMHA plugin wrapper macro **MHAPLUGIN_CALLBACKS** (p. 1618).

The template parameter *runtime_cfg_t* should be the runtime configuration of the plugin.

See **MHAPlugin::config_t** (p. 1141) for details on the thread safe communication update mechanism.

4.331.2 Constructor & Destructor Documentation

```
4.331.2.1 plugin_t() 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).

```
4.331.2.2 ~plugin_t() template<class runtime_cfg_t >
MHAPlugin::plugin_t< runtime_cfg_t >::~: plugin_t [virtual]
```

4.331.3 Member Function Documentation

```
4.331.3.1 prepare() template<class runtime_cfg_t >
virtual void MHAPlugin::plugin_t< runtime_cfg_t >::prepare (
    mhaconfig_t & ) [pure virtual]
```

Implemented in **calibrator_t** (p. 340), **dc::dc_if_t** (p. 382), **fftfbpow::fftfbpow_if_t** (p. 494), **save_spec_t** (p. 1418), **save_wave_t** (p. 1420), **wave2spec_if_t** (p. 1488), **windnoise::if_t** (p. 1509), **attenuate20_t** (p. 315), **matlab_wrapper::matlab_wrapper_t** (p. 699), **example3_t** (p. 474), **example4_t** (p. 478), **droptect_t** (p. 440), **example1_t** (p. 467), **example2_t** (p. 470), **mconv::MConv** (p. 718), **gtfb_simple_t** (p. 571), **plingploing::if_t** (p. 1339), **plugins::hoertech::acrec::acrec_t** (p. 1377), **wavrec_t** (p. 1498), **altconfig_t** (p. 297), **bbcalib_interface_t** (p. 335), **gtfb_simd_t** (p. 562), **addsndfile::addsndfile_if_t** (p. 251), **adm_if_t** (p. 273), **testplugin::if_t** (p. 1481), **analysispath_if_t** (p. 311), **osc2ac_t** (p. 1319), **dbasync_native::db_if_t** (p. 375), **ac2lsl::ac2lsl_t** (p. 163), **audiometerbackend::audiometer_if_t** (p. 317), **noise_psd_estimator::noise_psd_estimator_if_t** (p. 1310), **fftfilter::interface_t** (p. 501), **lsl2ac::lsl2ac_t** (p. 682), **rohBeam::rohBeam** (p. 1396), **smooth_cepstrum::smooth_cepstrum_if_t** (p. 1435), **multibandcompressor::interface_t** (p. 1301), **dc_simple::dc_if_t** (p. 394), **combc_if_t** (p. 357), **gtfb_analyzer::gtfb_analyzer_t** (p. 554), **rmslevel::rmslevel_if_t** (p. 1388), **coherence::cohflt_if_t** (p. 349), **plugin_interface_t** (p. 1352), **smoothgains_bridge::overlapadd_if_t** (p. 1449), **example6_t** (p. 482), **cpupload::cpupload_if_t** (p. 368), **noise_t** (p. 1316), **MHAPlugin_Resampling::resampling_if_t** (p. 1153), **shadowfilter_end::shadowfilter_end_t** (p. 1427), **ac2wave_if_t** (p. 187), **adaptive_feedback_canceller** (p. 242), **nlms_t** (p. 1306), **fshift_hilbert::frequency_translator_t** (p. 516), **spec2wave_if_t** (p. 1461), **level_matching::level_matching_t** (p. 655), **fader_wave::fader_wave_if_t** (p. 489), **mhachain::chain_base_t** (p. 843), **overlapadd::overlapadd_if_t** (p. 1329), **acsave::acsave_t** (p. 220), **complex_scale_channel_t** (p. 363), **doasvm_feature_extraction** (p. 427), **fshift::fshift_t** (p. 512), **shadowfilter_begin::shadowfilter_begin_t** (p. 1423), **lpc_bl_predictor** (p. 665), **lpc_burglattice** (p. 671), **sine_t** (p. 1431), **acPooling_wave** (p. 213), **steerbf** (p. 1470), **delaysum::delaysum_wave_if_t** (p. 412), **lpc** (p. 661), **db_if_t** (p. 371), **fader_if_t** (p. 487), **acConcat_wave** (p. 200), **acSteer** (p. 230), **gain::gain_if_t** (p. 534), **acTransform::wave** (p. 236), **doasvm_classification** (p. 421), **fftfilterbank::fftfb_interface_t** (p. 504), **matrixmixer::matmix_t** (p. 714), **route::interface_t** (p. 1409), **equalize::freqgains_t** (p. 464), **altplugs_t** (p. 301), **softclip_t** (p. 1454), **delaysum_spec::delaysum_spec_if_t** (p. 417), **ac_proc::interface_t** (p. 197), **ac2osc_t** (p. 182), **example7_t** (p. 485), **gsc::adaptive_stage::gsc_adaptive_stage_if** (p. 546), **acmon::acmon_t** (p. 208), **dropgen_t** (p. 437), **identity_t** (p. 576), **levelmeter_t** (p. 658), **delay::interface_t** (p. 410), **ds_t** (p. 444), and **us_t** (p. 1485).

```
4.331.3.2 release() template<class runtime_cfg_t >
void MHAPlugin::plugin_t< runtime_cfg_t >::release [virtual]
```

Reimplemented in **windnoise::if_t** (p. 1510), **attenuate20_t** (p. 315), **rohBeam::rohBeam** (p. 1396), **smooth_cepstrum::smooth_cepstrum_if_t** (p. 1435), **adaptive_feedback_canceller** (p. 242), **level_matching::level_matching_t** (p. 655), **doasvm_feature_extraction** (p. 428), **fshift::fshift_t** (p. 513), **example3_t** (p. 474), **example4_t** (p. 478), **lpc_bl_predictor** (p. 665), **lpc_burglattice** (p. 671), **acPooling_wave** (p. 213), **steerbf** (p. 1471), **droptect_t** (p. 441), **lpc** (p. 662), **acConcat_wave** (p. 201), **acSteer** (p. 231),

acTransform_wave (p. 236), **example2_t** (p. 470), **doasvm_classification** (p. 422), **example1_t** (p. 467), **gsc_adaptive_stage::gsc_adaptive_stage_if** (p. 546), **example7_t** (p. 485), **bbcalib_interface_t** (p. 335), **calibrator_t** (p. 340), **gtfb_simple_t** (p. 571), **addsndfile::addsndfile_if_t** (p. 252), **adm_if_t** (p. 273), **analysisispath_if_t** (p. 312), **ac2lsl::ac2lsl_t** (p. 164), **osc2ac_t** (p. 1319), **dbasync_native::db_if_t** (p. 376), **matlab_wrapper::matlab_wrapper_t** (p. 700), **lsl2ac::lsl2ac_t** (p. 682), **multibandcompressor::interface_t** (p. 1301), **wave2spec_if_t** (p. 1488), **dc_simple::dc_if_t** (p. 394), **plugins::hoertech::acrec::acrec_t** (p. 1377), **coherence::cohfilt_if_t** (p. 349), **smoothgains::bridge::overlapadd_if_t** (p. 1449), **MHAPlugIn_Resampling::resampling_if_t** (p. 1154), **ac2wave_if_t** (p. 187), **nlms_t** (p. 1306), **fshift_hilbert::frequency_translator_t** (p. 516), **spec2wave_if_t** (p. 1461), **fader_wave::fader_wave_if_t** (p. 489), **mhachain::chain_base_t** (p. 843), **overlapadd::overlapadd_if_t** (p. 1329), **acsave::acsave_t** (p. 221), **delaysum::delaysum_wave_if_t** (p. 413), **db_if_t** (p. 371), **wavrec_t** (p. 1498), **gain::gain_if_t** (p. 534), **altconfig_t** (p. 297), **fftfilterbank::fftfb_interface_t** (p. 504), **route::interface_t** (p. 1409), **mconv::MConv** (p. 718), **ac2osc_t** (p. 182), **altplugs_t** (p. 301), **ac_proc::interface_t** (p. 197), **acmon::acmon_t** (p. 209), **dropgen_t** (p. 437), **identity_t** (p. 576), **ds_t** (p. 444), and **us_t** (p. 1485).

4.331.3.3 prepare_() template<class runtime_cfg_t >
void MHAPlugIn::plugin_t< runtime_cfg_t >::prepare_(
 mhaconfig_t & cf)

4.331.3.4 release_() template<class runtime_cfg_t >
void MHAPlugIn::plugin_t< runtime_cfg_t >::release_

4.331.3.5 is_prepared() template<class runtime_cfg_t >
bool MHAPlugIn::plugin_t< runtime_cfg_t >::is_prepared () const [inline]

Flag, if the prepare method is successfully called (or currently evaluated)

4.331.3.6 input_cfg() template<class runtime_cfg_t >
mhaconfig_t MHAPlugIn::plugin_t< runtime_cfg_t >::input_cfg () const [inline]

Current input channel configuration.

```
4.331.3.7 output_cfg() template<class runtime_cfg_t >
mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::output_cfg( ) const [inline]
```

Current output channel configuration.

4.331.4 Member Data Documentation

```
4.331.4.1 tftype template<class runtime_cfg_t >
mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::tftype [protected]
```

Member for storage of plugin interface configuration.

This member is defined for convenience of the developer. Typically, the actual contents of **mhaconfig_t** (p. 850) are stored in this member in the **prepare()** (p. 1148) method.

Note

This member is likely to be removed in later versions, use **input_cfg()** (p. 1150) and **output_cfg()** (p. 1150) instead.

```
4.331.4.2 ac 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.

```
4.331.4.3 is_prepared_ template<class runtime_cfg_t >
bool MHAPlugin::plugin_t< runtime_cfg_t >::is_prepared_ [private]
```

```
4.331.4.4 input_cfg_ template<class runtime_cfg_t >
mhaconfig_t MHAPlugin::plugin_t< runtime_cfg_t >::input_cfg_ [private]
```

4.331.4.5 mhaconfig_cfg_ template<class runtime_cfg_t >
mhaconfig_t **MHAParser::plugin_t**< runtime_cfg_t >::mhaconfig_cfg_ [private]

4.331.4.6 mhaconfig_in template<class runtime_cfg_t >
MHAParser::mhaconfig_mon_t **MHAParser::plugin_t**< runtime_cfg_t >::mhaconfig_in [private]

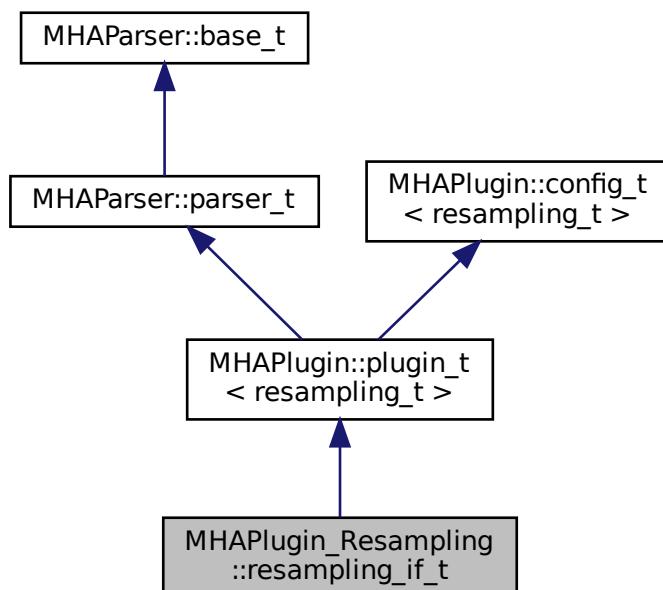
4.331.4.7 mhaconfig_out template<class runtime_cfg_t >
MHAParser::mhaconfig_mon_t **MHAParser::plugin_t**< runtime_cfg_t >::mhaconfig_out [private]

The documentation for this class was generated from the following file:

- **mha_plugin.hh**

4.332 MHAParser_Resampling::resampling_if_t Class Reference

Inheritance diagram for MHAParser_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::mhaplugloader_t plugloader**
- **std::string chain**
- **std::string algo**

Additional Inherited Members

4.332.1 Constructor & Destructor Documentation

```
4.332.1.1 resampling_if_t() MHAParser::resampling_if_t::resampling_if_t
(
    algo_comm_t iac,
    std::string th,
    std::string al )
```

4.332.2 Member Function Documentation

```
4.332.2.1 process() mha_wave_t * MHAParser::resampling_if_t::process (
    mha_wave_t * s )
```

4.332.2.2 `prepare()` `void MHAPlugIn_Resampling::resampling_if_t::prepare (mhaconfig_t & conf) [virtual]`

Implements `MHAPlugIn::plugin_t< resampling_t >` (p. 1148).

4.332.2.3 `release()` `void MHAPlugIn_Resampling::resampling_if_t::release () [virtual]`

Reimplemented from `MHAPlugIn::plugin_t< resampling_t >` (p. 1149).

4.332.3 Member Data Documentation

4.332.3.1 `srate` `MHAParser::float_t MHAPlugIn_Resampling::resampling_if_t::srate [private]`

4.332.3.2 `fragsize` `MHAParser::int_t MHAPlugIn_Resampling::resampling_if_t::fragsize [private]`

4.332.3.3 `nyquist_ratio` `MHAParser::float_t MHAPlugIn_Resampling::resampling_if_t::nyquist_ratio [private]`

4.332.3.4 `irslen_outer2inner` `MHAParser::float_t MHAPlugIn_Resampling::resampling_if_t::irslen_outer2inner [private]`

4.332.3.5 `irslen_inner2outer` `MHAParser::float_t MHAPlugIn_Resampling::resampling_if_t::irslen_inner2outer [private]`

4.332.3.6 plugloader `MHAParser::mhapluginloader_t` `MHAPlugin_Resampling::resampling_if_t::plugloader` [private]

4.332.3.7 chain `std::string` `MHAPlugin_Resampling::resampling_if_t::chain` [private]

4.332.3.8 algo `std::string` `MHAPlugin_Resampling::resampling_if_t::algo` [private]

The documentation for this class was generated from the following file:

- `resampling.cpp`

4.333 MHAPlugin_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`

4.333.1 Constructor & Destructor Documentation

4.333.1.1 `resampling_t()` `MHAPlugIn_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)`

4.333.2 Member Function Documentation

4.333.2.1 `process()` `mha_wave_t * MHAPlugIn_Resampling::resampling_t::process (`
 `mha_wave_t * s)`

4.333.3 Member Data Documentation

4.333.3.1 `outer_fragsize` `unsigned MHAPlugIn_Resampling::resampling_t::outer_fragsize`
[private]

4.333.3.2 `inner_fragsize` `unsigned MHAPlugIn_Resampling::resampling_t::inner_fragsize`
[private]

4.333.3.3 `outer_srate` `float MHAPlugIn_Resampling::resampling_t::outer_srate` [private]

4.333.3.4 `inner_srate` `float MHAPlugIn_Resampling::resampling_t::inner_srate` [private]

4.333.3.5 nchannels_in `unsigned MHAPlugin_Resampling::resampling_t::nchannels_in [private]`

4.333.3.6 nchannels_out `unsigned MHAPlugin_Resampling::resampling_t::nchannels_out [private]`

4.333.3.7 outer2inner_resampling `MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugin_Resampling::resampling_t::outer2inner_resampling [private]`

4.333.3.8 inner2outer_resampling `MHAFilter::blockprocessing_polyphase_resampling_t MHAPlugin_Resampling::resampling_t::inner2outer_resampling [private]`

4.333.3.9 plugloader `MHAParser::mhapluginloader_t& MHAPlugin_Resampling::resampling_t::plugloader [private]`

4.333.3.10 inner_signal `MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::inner_signal [private]`

4.333.3.11 output_signal `MHASignal::waveform_t MHAPlugin_Resampling::resampling_t::output_signal [private]`

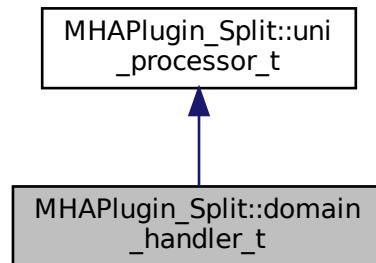
The documentation for this class was generated from the following file:

- **resampling.cpp**

4.334 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.

4.334.1 Detailed Description

Handles domain-specific partial input and output signal.

4.334.2 Constructor & Destructor Documentation

```
4.334.2.1 domain_handler_t() [1/2] MHAPlugin_Split::domain_handler_t::domain_<→
handler_t (
    const domain_handler_t & ) [private]
```

Disallow copy constructor.

```
4.334.2.2 domain_handler_t() [2/2] MHAPlugin_Split::domain_handler_t::domain_←
handler_t (
    const mhaconfig_t & settings_in,
    const mhaconfig_t & settings_out,
    PluginLoader::fourway_processor_t * processor ) [inline]
```

Construct a new domain handler once the domains and dimensions of input and output signal of one of the child plugins of split are known.

```
4.334.2.3 ~domain_handler_t() virtual MHAPlugin_Split::domain_handler_t::~domain_←
_handler_t ( ) [inline], [virtual]
```

Deallocation of signal holders.

4.334.3 Member Function Documentation

```
4.334.3.1 operator=() domain_handler_t& MHAPlugin_Split::domain_handler_t::operator=
(
    const domain_handler_t & ) [private]
```

Disallow assignment operator.

```
4.334.3.2 set_input_domain() void MHAPlugin_Split::domain_handler_t::set_input_←
domain (
    const mhaconfig_t & settings_in ) [inline]
```

Set parameters of input signal.

Parameters

<i>settings_← _in</i>	domain and dimensions of partial input signal
---------------------------	---

```
4.334.3.3 set_output_domain() void MHAPlugin_Split::domain_handler_t::set_output<-
_domain (
    const mhaconfig_t & settings_out ) [inline]
```

Set output signal parameters.

Parameters

<i>settings_out</i>	domain and dimensions of partial output signal
---------------------	--

```
4.334.3.4 deallocate_domains() void MHAPlugin_Split::domain_handler_t::deallocate<-
_domains ( ) [inline]
```

Deallocate domain indicators and signal holders.

```
4.334.3.5 put_signal() [1/2] unsigned MHAPlugin_Split::domain_handler_t::put_signal<-
(
    mha_wave_t * s_in,
    unsigned start_channel ) [inline]
```

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **wave_in** (p. 1163).

Parameters

<i>s_in</i>	The combined waveform 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

```
4.334.3.6 put_signal() [2/2] unsigned MHAPlugin_Split::domain_handler_t::put_signal
(
    mha_spec_t * s_in,
    unsigned start_channel ) [inline]
```

Store the relevant channels from the input signal for processing.

The number of channels to store is taken from the dimensions of the partial input signal holder **spec_in** (p. 1164).

Parameters

<i>s_in</i>	The combined spectrum 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

```
4.334.3.7 get_signal() [1/2] unsigned MHAPlugin_Split::domain_handler_t::get_signal
(
```

```
    MHASignal::waveform_t * s_out,
    unsigned start_channel ) [inline]
```

Store all partial signal output channels in the combined waveform signal with the given channel offset.

All channels present in **wave_out** (p. 1163) 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 <i>s_out</i> .

Returns

The number of channels that were copied to the output signal

```
4.334.3.8 get_signal() [2/2] unsigned MHAPlugin_Split::domain_handler_t::get_signal
(
    MHASignal::spectrum_t * s_out,
    unsigned start_channel ) [inline]
```

Store all partial signal output channels in the combined spectrum signal with the given channel offset.

All channels present in **spec_out** (p. 1164) 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 <i>s_out</i> .

Returns

The number of channels that were copied to the output signal

4.334.3.9 process() 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. 1161) before this method may be called.

Implements **MHAPlugin_Split::uni_processor_t** (p. 1188).

4.334.4 Member Data Documentation

4.334.4.1 wave_in MHASignal::waveform_t* MHAPlugin_Split::domain_handler_t::wave_in

Partial wave input signal.

4.334.4.2 wave_out `mha_wave_t** MHAPlugin_Split::domain_handler_t::wave_out`

Partial wave output signal.

4.334.4.3 spec_in `MHASignal::spectrum_t* MHAPlugin_Split::domain_handler_t::spec_in`

Partial spec input signal.

4.334.4.4 spec_out `mha_spec_t** MHAPlugin_Split::domain_handler_t::spec_out`

Partial spec input signal.

4.334.4.5 processor `PluginLoader::fourway_processor_t* MHAPlugin_Split::domain_handler_t::processor`

The domain-specific signal processing methods are implemented here.

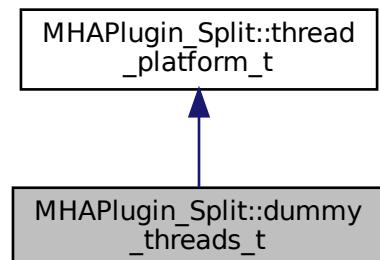
The documentation for this class was generated from the following file:

- `split.cpp`

4.335 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 dummy_threads_t::kick_thread.
- **dummy_threads_t (uni_processor_t *proc, const std::string &thread_scheduler, int thread_priority)**
Constructor.

Additional Inherited Members

4.335.1 Detailed Description

Dummy specification of a thread platform: This class implements everything in a single thread.

4.335.2 Constructor & Destructor Documentation

4.335.2.1 dummy_threads_t() MHAPlugin_Split::dummy_threads_t::dummy_threads_t (
`uni_processor_t * proc,`
`const std::string & thread_scheduler,`
`int thread_priority) [inline]`

Constructor.

Parameters

<code>proc</code>	Pointer to the associated plugin loader
<code>thread_scheduler</code>	Unused in dummy thread platform
<code>thread_priority</code>	Unused in dummy thread platform

4.335.3 Member Function Documentation

4.335.3.1 kick_thread() void MHAPlugIn_Split::dummy_threads_t::kick_thread () [inline], [virtual]

perform signal processing immediately (no multiple threads in this dummy class)

Implements **MHAPlugIn_Split::thread_platform_t** (p. 1186).

4.335.3.2 catch_thread() void MHAPlugIn_Split::dummy_threads_t::catch_thread () [inline], [virtual]

No implementation needed: Processing has been completed during ummy_threads_t::kick_thread.

Implements **MHAPlugIn_Split::thread_platform_t** (p. 1186).

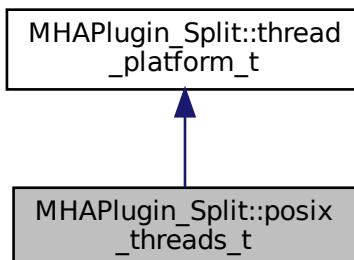
The documentation for this class was generated from the following file:

- **split.cpp**

4.336 MHAPlugIn_Split::posix_threads_t Class Reference

Posix threads specification of thread platform.

Inheritance diagram for MHAPlugIn_Split::posix_threads_t:



Public Member Functions

- void **kick_thread** ()
Start signal processing in separate thread.
- void **catch_thread** ()
Wait for signal processing to finish.
- **posix_threads_t** (**uni_processor_t** *proc, const std::string &thread_scheduler, int thread_priority)
Constructor.
- **~posix_threads_t** ()
Terminate thread.
- void **main** ()
Thread main loop. Wait for process/termination trigger, then act.

Static Public Member Functions

- static void * **thread_start** (void *thr)
Thread start function.
- static std::string **current_thread_scheduler** ()
- static int **current_thread_priority** ()

Private Attributes

- pthread_mutex_t **mutex**
The mutex.
- pthread_cond_t **kick_condition**
The condition for signalling the kicking and termination.
- pthread_cond_t **catch_condition**
The condition for signalling the processing is finished.
- pthread_attr_t **attr**
Thread attributes.
- struct sched_param **priority**
Thread scheduling priority.
- int **scheduler**
- pthread_t **thread**
The thread object.
- bool **kicked**
A flag that is set to true by kick_thread and to false by the thread after it has woken up from the kicking.
- bool **processing_done**
A flag that is set to true by the thread when it returns from processing and to false by catch_thread after it has waited for that return.
- bool **termination_request**
Set to true by the destructor.

Additional Inherited Members

4.336.1 Detailed Description

Posix threads specification of thread platform.

4.336.2 Constructor & Destructor Documentation

```
4.336.2.1 posix_threads_t() MHAPlugin_Split::posix_threads_t::posix_threads_t (
    uni_processor_t * proc,
    const std::string & thread_scheduler,
    int thread_priority ) [inline]
```

Constructor.

Parameters

<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.

```
4.336.2.2 ~posix_threads_t() MHAPlugin_Split::posix_threads_t::~posix_threads_t (
) [inline]
```

Terminate thread.

4.336.3 Member Function Documentation

```
4.336.3.1 kick_thread() void MHAPlugin_Split::posix_threads_t::kick_thread ( ) [inline],
[virtual]
```

Start signal processing in separate thread.

Implements **MHAPlugin_Split::thread_platform_t** (p. 1186).

4.336.3.2 catch_thread() void MHAPlugin_Split::posix_threads_t::catch_thread ()
[inline], [virtual]

Wait for signal processing to finish.

Implements **MHAPlugin_Split::thread_platform_t** (p. 1186).

4.336.3.3 thread_start() static void* MHAPlugin_Split::posix_threads_t::thread_start
(
void * thr) [inline], [static]

Thread start function.

4.336.3.4 main() void MHAPlugin_Split::posix_threads_t::main () [inline]

Thread main loop. Wait for process/termination trigger, then act.

4.336.3.5 current_thread_scheduler() static std::string MHAPlugin_Split::posix_←
threads_t::current_thread_scheduler () [inline], [static]

4.336.3.6 current_thread_priority() static int MHAPlugin_Split::posix_threads_t←
::current_thread_priority () [inline], [static]

4.336.4 Member Data Documentation

4.336.4.1 mutex pthread_mutex_t MHAPlugin_Split::posix_threads_t::mutex [private]

The mutex.

4.336.4.2 kick_condition pthread_cond_t MHAPlugIn_Split::posix_threads_t::kick_condition [private]

The condition for signalling the kicking and termination.

4.336.4.3 catch_condition pthread_cond_t MHAPlugIn_Split::posix_threads_t::catch_condition [private]

The condition for signalling the processing is finished.

4.336.4.4 attr pthread_attr_t MHAPlugIn_Split::posix_threads_t::attr [private]

Thread attributes.

4.336.4.5 priority struct sched_param MHAPlugIn_Split::posix_threads_t::priority [private]

Thread scheduling priority.

4.336.4.6 scheduler int MHAPlugIn_Split::posix_threads_t::scheduler [private]

4.336.4.7 thread pthread_t MHAPlugIn_Split::posix_threads_t::thread [private]

The thread object.

4.336.4.8 kicked bool MHAPlugIn_Split::posix_threads_t::kicked [private]

A flag that is set to true by kick_thread and to false by the thread after it has woken up from the kicking.

4.336.4.9 processing_done bool MHAPlugin_Split::posix_threads_t::processing_done
[private]

A flag that is set to true by the thread when it returns from processing and to false by catch_↳
thread after it has waited for that return.

4.336.4.10 termination_request bool MHAPlugin_Split::posix_threads_t::termination↳
_request [private]

Set to true by the destructor.

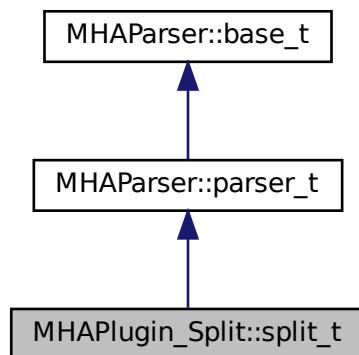
The documentation for this class was generated from the following file:

- **split.cpp**

4.337 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

4.337.1 Detailed Description

Implements split plugin.

An instance of class **split_t** (p. 1171) implements the split plugin functionality: The audio channels are splitted and groups of audio channels are processed by different plugins in parallel.

4.337.2 Constructor & Destructor Documentation

4.337.2.1 **split_t()** MHAPlugin_Split::split_t::split_t (

```
    algo_comm_t iac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Plugin constructor.

4.337.2.2 ~split_t() `MHAPlugin_Split::split_t::~split_t ()`

Plugin destructor. Unloads nested plugins.

4.337.3 Member Function Documentation**4.337.3.1 prepare_()** `void MHAPlugin_Split::split_t::prepare_ (mhaconfig_t & signal_parameters)`

Check signal parameters, prepare chains, and allocate output signal holders.

4.337.3.2 release_() `void MHAPlugin_Split::split_t::release_ ()`

Delete output signal holder and release chains.

4.337.3.3 process() `template<class SigTypeIn , class SigTypeOut >`
`void MHAPlugin_Split::split_t::process (`
`SigTypeIn * s_in,`
`SigTypeOut ** s_out)`

Let the parallel plugins process channel groups of the input signal.

4.337.3.4 update() `void MHAPlugin_Split::split_t::update () [private]`

Load plugins in response to a value change in the algos variable.

4.337.3.5 clear_chains() `void MHAPlugin_Split::split_t::clear_chains () [private]`

Unload the plugins.

4.337.3.6 copy_output_wave() `mha_wave_t* MHAPlugin_Split::split_t::copy_output_<wave () [private]`

4.337.3.7 copy_output_spec() `mha_spec_t* MHAPlugin_Split::split_t::copy_output_<spec () [private]`

4.337.3.8 trigger_processing() `template<class SigType >
void MHAPlugin_Split::split_t::trigger_processing (`
`SigType * s_in) [private]`

Split the argument input signal to groups of channels for the plugins and initiate signal processing.

4.337.3.9 collect_result() `template<class SigType >
void MHAPlugin_Split::split_t::collect_result (`
`SigType * s_out) [private]`

Combine the output signal from the plugins.

4.337.3.10 signal_out() [1/2] `MHASignal::waveform_t* MHAPlugin_Split::split_t<::signal_out (`
`mha_wave_t **) [inline], [private]`

Waveform domain output signal structure accessor.

Parameter is only for domain disambiguation and is ignored.

4.337.3.11 signal_out() [2/2] `MHASignal::spectrum_t* MHAPlugin_Split::split_t<::signal_out (`
`mha_spec_t **) [inline], [private]`

Spectrum domain output signal structure. Parameter is ignored.

4.337.4 Member Data Documentation

4.337.4.1 patchbay `MHAEEvents::patchbay_t< split_t> MHAParser::patchbay [private]`

Reload plugins when the algos variable changes.

4.337.4.2 algos `MHAParser::vstring_t MHAParser::algos [private]`

Vector of plugins to load in parallel.

4.337.4.3 channels `MHAParser::vint_t MHAParser::channels [private]`

Number of channels to route through each plugin.

4.337.4.4 thread_platform `MHAParser::kw_t MHAParser::thread_platform [private]`

Thread platform chooser.

4.337.4.5 worker_thread_scheduler `MHAParser::kw_t MHAParser::worker_thread_scheduler [private]`

Scheduler used for worker threads.

4.337.4.6 worker_thread_priority `MHAParser::int_t MHAPlugin_Split::split_t::worker_thread_priority [private]`

Priority of worker threads.

4.337.4.7 framework_thread_scheduler `MHAParser::string_mon_t MHAPlugin_Split::split_t::framework_thread_scheduler [private]`

Scheduler of the signal processing thread.

4.337.4.8 framework_thread_priority `MHAParser::int_mon_t MHAPlugin_Split::split_t::framework_thread_priority [private]`

Priority of signal processing thread.

4.337.4.9 delay `MHAParser::bool_t MHAPlugin_Split::split_t::delay [private]`

Switch to activate parallel processing of plugins at the cost of one block of additional delay.

4.337.4.10 chains `std::vector< splitted_part_t*> MHAPlugin_Split::split_t::chains [private]`

Interfaces to parallel plugins.

4.337.4.11 wave_out `MHASignal::waveform_t* MHAPlugin_Split::split_t::wave_out [private]`

Combined output waveforms structure.

4.337.4.12 spec_out `MHASignal::spectrum_t* MHAPlugin_Split::split_t::spec_out`
 [private]

Combined output spectra structure.

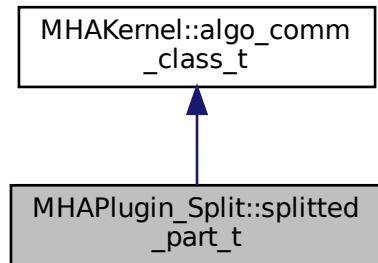
The documentation for this class was generated from the following file:

- `split.cpp`

4.338 MHAPlugin_Split::splitted_part_t Class Reference

The `splitted_part_t` (p. 1178) instance manages the plugin that performs processing on the reduced set of channels.

Inheritance diagram for MHAPlugin_Split::splitted_part_t:



Public Member Functions

- `splitted_part_t (const std::string &pluginname, 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. 1183).*
- `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. 1158) instance.*
- `void release ()`

*Delegates the release method to the plugin and deletes the **MHAPlugin_Split::domain_handler_t** (p. 1158) instance.*

- std::string **parse** (const std::string &str)
Delegates parser invocation 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

4.338.1 Detailed Description

The **splitted_part_t** (p. 1178) 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.

4.338.2 Constructor & Destructor Documentation

4.338.2.1 `splitted_part_t()` [1/3] `MHAPlugIn_Split::splitted_part_t::splitted_part_t (const splitted_part_t &)` [private]

Disallow copy constructor.

4.338.2.2 `splitted_part_t()` [2/3] `MHAPlugIn_Split::splitted_part_t::splitted_part_t (const std::string & plugname, MHParseR::parser_t * parent)`

Load the plugin for this partial signal path.

Loads the MHA plugin for a signal path of these audio channels.

Parameters

<code>plugname</code>	The name of the MHA plugin, optionally followed by a colon and the algorithm name.
<code>parent</code>	The parser node where the configuration of the new plugin is inserted. The plugin's parser name is the configured name (colon syntax).

4.338.2.3 `splitted_part_t()` [3/3] `MHAPlugIn_Split::splitted_part_t::splitted_part_t (PluginLoader::fourway_processor_t * plugin)`

Create the handler for the partial signal.

The plugin is loaded by the caller, but it will be deleted by the destructor of this class. This constructor exists solely for testing purposes.

Parameters

<code>plugin</code>	The plugin used for processing the signal. The new <code>splitted_part_t</code> (p. 1179) instance will take ownership of this instance and release it in the destructor.
---------------------	---

4.338.2.4 `~splitted_part_t()` `MHAPlugIn_Split::splitted_part_t::~splitted_part_t () throw ()`

Destructor. Deletes the plugin `plug` (p. 1183).

4.338.3 Member Function Documentation

```
4.338.3.1 operator=() splitted_part_t& MHAPlugin_Split::splitted_part_t::operator=
(
    const splitted_part_t & ) [private]
```

Disallow assignment operator.

```
4.338.3.2 prepare() void MHAPlugin_Split::splitted_part_t::prepare (
    mhaconfig_t & signal_parameters,
    const std::string & thread_platform,
    const std::string & thread_scheduler,
    int thread_priority )
```

Delegates the prepare method to the plugin and allocates a suitable **MHAPlugin_Split**::**domain_handler_t** (p. 1158) 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.

```
4.338.3.3 release() void MHAPlugin_Split::splitted_part_t::release ( )
```

Delegates the release method to the plugin and deletes the **MHAPlugin_Split**::**domain_handler_t** (p. 1158) instance.

Release the loaded plugin.

Plugin release.

```
4.338.3.4 parse() std::string MHAPlugin_Split::splitted_part_t::parse (
    const std::string & str ) [inline]
```

Delegates parser invocation to plugin.

```
4.338.3.5 trigger_processing() template<class SigType >
unsigned MHAPlugin_Split::splitted_part_t::trigger_processing (
    SigType * s_in,
    unsigned start_channel ) [inline]
```

The domain handler copies the input signal channels.

Then, processing is initiated.

Parameters

<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

```
4.338.3.6 collect_result() template<class SigType >
unsigned MHAPlugin_Split::splitted_part_t::collect_result (
    SigType * s_out,
    unsigned start_channel ) [inline]
```

Wait until processing is finished, then copy the output data.

Parameters

<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

4.338.4 Member Data Documentation

4.338.4.1 plug `PluginLoader::fourway_processor_t* MHAPlugin_Split::splitted_part_t::plug` [private]

The plugin that performs the signal processing on the prepared channels.

4.338.4.2 domain `domain_handler_t* MHAPlugin_Split::splitted_part_t::domain` [private]

The domain specific signal handler, allocated from prepare when input and output domains and signal parameters are known.

4.338.4.3 thread `thread_platform_t* MHAPlugin_Split::splitted_part_t::thread` [private]

The platform-dependent thread synchronization implementation.

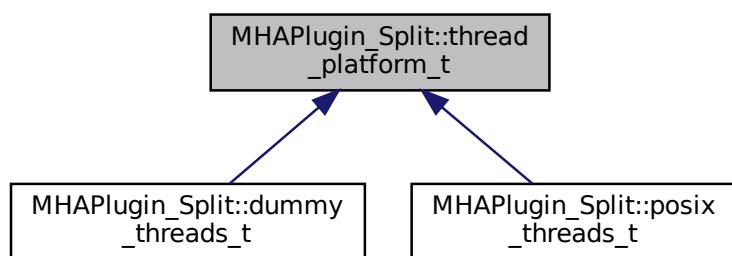
The documentation for this class was generated from the following file:

- `split.cpp`

4.339 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.

4.339.1 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.

4.339.2 Constructor & Destructor Documentation

```
4.339.2.1 thread_platform_t() [1/2] MHAPlugIn_Split::thread_platform_t::thread_<--  

platform_t (
```

```
    const thread_platform_t & ) [private]
```

Disallow copy constructor.

```
4.339.2.2 thread_platform_t() [2/2] MHAPlugin_Split::thread_platform_t::thread_←
platform_t (
    uni_processor_t * proc ) [inline]
```

Constructor.

Derived classes create the thread in the constructor.

Parameters

<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. 1178) instance.
-------------	--

4.339.2.3 ~thread_platform_t() virtual MHAPlugIn_Split::thread_platform_t::~thread_platform_t () [inline], [virtual]

Make derived classes destructable via pointer to this base class.

Derived classes' destructors notify the thread that it should terminate itself, and wait for the termination to occur.

4.339.3 Member Function Documentation

4.339.3.1 operator=() thread_platform_t& MHAPlugIn_Split::thread_platform_t::operator= (const thread_platform_t &) [private]

Disallow assignment operator.

4.339.3.2 kick_thread() virtual void MHAPlugIn_Split::thread_platform_t::kick_thread () [pure virtual]

Derived classes notify their processing thread that it should call processor->process().

Implemented in **MHAPlugIn_Split::posix_threads_t** (p. 1168), and **MHAPlugIn_Split::dummy_threads_t** (p. 1165).

4.339.3.3 catch_thread() virtual void MHAPlugin_Split::thread_platform_t::catch_thread () [pure virtual]

Derived classes wait for their signal processing thread to return from the call to part->process().

Implemented in **MHAPlugin_Split::posix_threads_t** (p. 1168), and **MHAPlugin_Split::dummy_threads_t** (p. 1166).

4.339.4 Member Data Documentation

4.339.4.1 processor uni_processor_t* MHAPlugin_Split::thread_platform_t::processor [protected]

A pointer to the plugin loader that processes the sound data in the channels for which this thread was created.

Using the **MHAPlugin_Split::uni_processor_t** (p. 1187) interface instead of the mhaplugin-loader class directly for testability (no need to load real plugins for testing the thread platform).

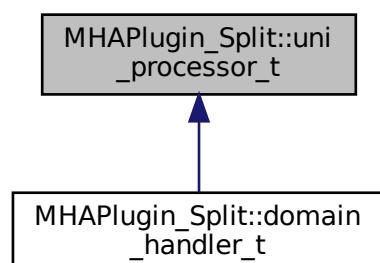
The documentation for this class was generated from the following file:

- **split.cpp**

4.340 MHAPlugin_Split::uni_processor_t Class Reference

An interface to a class that sports a process method with no parameters and no return value.

Inheritance diagram for MHAPlugin_Split::uni_processor_t:



Public Member Functions

- virtual void **process ()=0**
This method uses some input signal, performs processing and stores the output signal somewhere.
- virtual ~**uni_processor_t ()**
Classes containing virtual methods need virtual destructors.

4.340.1 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.

4.340.2 Constructor & Destructor Documentation

4.340.2.1 ~uni_processor_t() virtual MHAPlugin_Split::uni_processor_t::~uni_processor_t () [inline], [virtual]

Classes containing virtual methods need virtual destructors.

4.340.3 Member Function Documentation

4.340.3.1 process() virtual void MHAPlugin_Split::uni_processor_t::process () [pure virtual]

This method uses some input signal, performs processing and stores the output signal somewhere.

This method also has to dispatch the process call based on the configured domains.

Signal transfer and domain configuration have to be done in derived class in different methods.

Implemented in **MHAPlugin_Split::domain_handler_t** (p. [1163](#)).

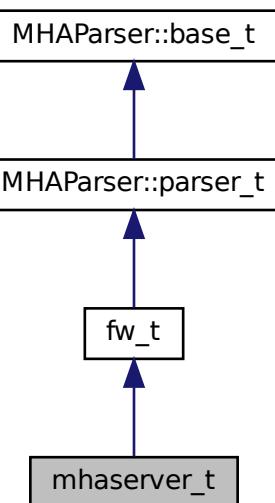
The documentation for this class was generated from the following file:

- **split.cpp**

4.341 mhaserver_t Class Reference

MHA Framework listening on TCP port for commands.

Inheritance diagram for mhaserver_t:



Classes

- class **tcp_server_t**

Public Member Functions

- **mhaserver_t** (const std::string &ao, const std::string &af, const std::string &lf, bool b_← interactive_)
- **~mhaserver_t ()**
- virtual std::string **on_received_line** (const std::string &line)

A line of text was received from network client.
- virtual void **acceptor_started ()**

Notification: "TCP port is open".
- virtual void **send_port_announcement ()**

sends an announcement which port this MHA is listening on to the creator of the process.
- virtual void **start_stdin_thread ()**

Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.

- virtual void **set_announce_port** (unsigned short **announce_port**)
If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.
- void **logstring** (const std::string &)
Log a message to log file.
- int **run** (unsigned short **port**, const std::string &_interface)
Accept network connections and act on commands.

Public Attributes

- **MHAParser::int_t port**

Private Attributes

- std::shared_ptr< **tcp_server_t** > **tcpserver**
- std::string **ack_ok**
- std::string **ack_fail**
- std::string **logfile**
- unsigned short **announce_port**
- bool **b_interactive**
- **MHAParser::int_mon_t pid_mon**

Additional Inherited Members

4.341.1 Detailed Description

MHA Framework listening on TCP port for commands.

4.341.2 Constructor & Destructor Documentation

4.341.2.1 **mhaserver_t()** `mhaserver_t::mhaserver_t (`

```
    const std::string & ao,
    const std::string & af,
    const std::string & lf,
    bool b_interactive_ )
```

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.

4.341.2.2 ~mhaserver_t() mhaserver_t::~mhaserver_t ()

4.341.3 Member Function Documentation

4.341.3.1 on_received_line() std::string mhaserver_t::on_received_line (const std::string & line) [virtual]

A line of text was received from network client.

4.341.3.2 acceptor_started() void mhaserver_t::acceptor_started () [virtual]

Notification: "TCP port is open".

4.341.3.3 send_port_announcement() void mhaserver_t::send_port_announcement () [virtual]

sends an announcement which port this MHA is listening on to the creator of the process.

See command line option –announce

4.341.3.4 start_stdin_thread() void mhaserver_t::start_stdin_thread () [virtual]

Starts a separate thread that reads lines from stdin and forwards these lines over TCP to the MHA configuration thread which multiplexes multiple TCP connections.

Enables users to type mha configuration language commands directly into the terminal where MHA was started, without the need to use third-party tools like nc or putty.

4.341.3.5 `set_announce_port()` `void mhaserver_t::set_announce_port (`
`unsigned short announce_port) [virtual]`

If set to nonzero, the spawning process has asked to be notified of the TCP port used by this process.

4.341.3.6 `logstring()` `void mhaserver_t::logstring (`
`const std::string & s) [inline]`

Log a message to log file.

4.341.3.7 `run()` `int mhaserver_t::run (`
`unsigned short port,`
`const std::string & _interface)`

Accept network connections and act on commands.

Calls **acceptor_started()** (p. 1191) when the TCP port is opened. Calls `on_received_line` for every line received.

Returns

exit code that can be used as process exit code

4.341.4 Member Data Documentation

4.341.4.1 `tcpserver` `std::shared_ptr< tcp_server_t> mhaserver_t::tcpserver [private]`

4.341.4.2 `ack_ok` `std::string mhaserver_t::ack_ok [private]`

4.341.4.3 ack_fail std::string mhaserver_t::ack_fail [private]

4.341.4.4 logfile std::string mhaserver_t::logfile [private]

4.341.4.5 announce_port unsigned short mhaserver_t::announce_port [private]

4.341.4.6 b_interactive bool mhaserver_t::b_interactive [private]

4.341.4.7 pid_mon MHAParser::int_mon_t mhaserver_t::pid_mon [private]

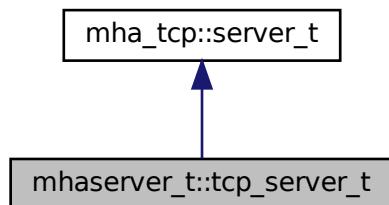
4.341.4.8 port MHAParser::int_t mhaserver_t::port

The documentation for this class was generated from the following file:

- **mhamain.cpp**

4.342 mhaserver_t::tcp_server_t Class Reference

Inheritance diagram for mhaserver_t::tcp_server_t:



Public Member Functions

- **tcp_server_t** (const std::string &interface, uint16_t **port**, **mhaserver_t** * **mha**)
- virtual bool **on_received_line** (std::shared_ptr< **mha_tcp::buffered_socket_t** > **c**, const std::string &**l**) override

This method is invoked when a line of text is received on one of the accepted connections.

Private Attributes

- **mhaserver_t** * **mha**

4.342.1 Constructor & Destructor Documentation

```
4.342.1.1 tcp_server_t() mhaserver_t::tcp_server_t::tcp_server_t (
    const std::string & interface,
    uint16_t port,
    mhaserver_t * mha ) [inline]
```

4.342.2 Member Function Documentation

```
4.342.2.1 on_received_line() virtual bool mhaserver_t::tcp_server_t::on_received_←
line (
    std::shared_ptr< mha_tcp::buffered_socket_t > c,
    const std::string & l ) [inline], [override], [virtual]
```

This method is invoked when a line of text is received on one of the accepted connections.

Override this method to process the communication with the client.

Parameters

c	the connection that has received this line
l	the line that has been received, without the line ending

Returns

client should return true when client wants to read another line of text, else false.

Reimplemented from [mha_tcp::server_t](#) (p. 821).

4.342.3 Member Data Documentation

4.342.3.1 mha_mhaserver_t* mhaserver_t::tcp_server_t::mha [private]

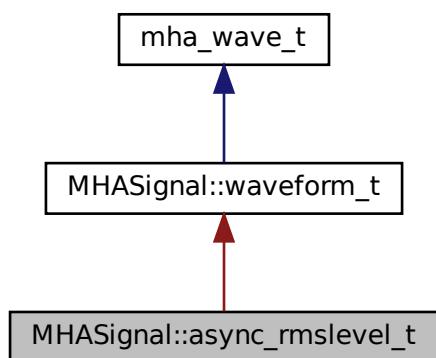
The documentation for this class was generated from the following file:

- [mhamain.cpp](#)

4.343 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

4.343.1 Detailed Description

Class for asynchronous level metering.

4.343.2 Constructor & Destructor Documentation

4.343.2.1 **async_rmslevel_t()** MHASignal::async_rmslevel_t::async_rmslevel_t (

```
    unsigned int frames,
    unsigned int channels )
```

Constructor for level metering class.

Allocate memory for metering. The RMS integration time corresponds to the number of frames in the buffer.

Parameters

<i>frames</i>	Number of frames to integrate.
<i>channels</i>	Number of channels used for level-metering.

4.343.3 Member Function Documentation

4.343.3.1 rmslevel() `std::vector< float > MHASignal::async_rmslevel_t::rmslevel () const`

Read-only function for querying the current RMS level.

Returns

Vector of floats, one value for each channel, containing the RMS level in dB (SPL if calibrated properly).

4.343.3.2 peaklevel() `std::vector< float > MHASignal::async_rmslevel_t::peaklevel () const`

Read-only function for querying the current peak level.

Returns

Vector of floats, one value for each channel, containing the peak level in dB (SPL if calibrated properly).

4.343.3.3 process() `void MHASignal::async_rmslevel_t::process (mha_wave_t * s)`

Function to store a chunk of audio in the level meter.

Parameters

<code>s</code>	Audio chunk (same number of channels required as given in the constructor).
----------------	---

4.343.4 Member Data Documentation

4.343.4.1 pos unsigned int MHASignal::async_rmslevel_t::pos [private]

4.343.4.2 filled unsigned int MHASignal::async_rmslevel_t::filled [private]

The documentation for this class was generated from the following files:

- [mha_signal.hh](#)
- [mha_signal.cpp](#)

4.344 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**

4.344.1 Constructor & Destructor Documentation

4.344.1.1 delay_spec_t() MHASignal::delay_spec_t::delay_spec_t (

```
    unsigned int delay,
    unsigned int frames,
    unsigned int channels )
```

4.344.1.2 ~delay_spec_t() MHASignal::delay_spec_t::~delay_spec_t ()

4.344.2 Member Function Documentation

4.344.2.1 process() `mha_spec_t * MHASignal::delay_spec_t::process (mha_spec_t * s)`

4.344.3 Member Data Documentation

4.344.3.1 delay `unsigned int MHASignal::delay_spec_t::delay [private]`

4.344.3.2 buffer `MHASignal::spectrum_t** MHASignal::delay_spec_t::buffer [private]`

4.344.3.3 pos `unsigned int MHASignal::delay_spec_t::pos [private]`

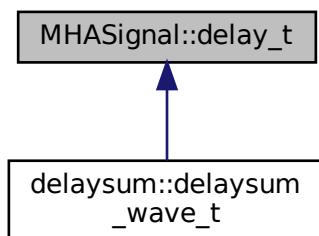
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.345 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`

4.345.1 Detailed Description

Class to realize a simple delay of waveform streams.

4.345.2 Constructor & Destructor Documentation

4.345.2.1 `delay_t()` `MHASignal::delay_t::delay_t (`
`std::vector< int > delays,`
`unsigned int channels)`

Constructor.

Parameters

<code>delays</code>	Vector of delays, one entry for each channel.
<code>channels</code>	Number of channels expected.

4.345.2.2 `~delay_t()` `MHASignal::delay_t::~delay_t ()`

4.345.3 Member Function Documentation

4.345.3.1 process() `mha_wave_t * MHASignal::delay_t::process (mha_wave_t * s)`

Processing method.

Parameters

<code>s</code>	Input waveform fragment, with number of channels provided in constructor.
----------------	---

Returns

Output waveform fragment.

4.345.3.2 inspect() `std::string MHASignal::delay_t::inspect () const [inline]`

4.345.4 Member Data Documentation

4.345.4.1 channels `unsigned int MHASignal::delay_t::channels [private]`

4.345.4.2 delays `unsigned int* MHASignal::delay_t::delays [private]`

4.345.4.3 pos `unsigned int* MHASignal::delay_t::pos [private]`

4.345.4.4 buffer `mha_real_t** MHASignal::delay_t::buffer [private]`

The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.346 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`

4.346.1 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).

4.346.2 Constructor & Destructor Documentation**4.346.2.1 delay_wave_t()** `MHASignal::delay_wave_t::delay_wave_t (`
`unsigned int delay,`
`unsigned int frames,`
`unsigned int channels)`

4.346.2.2 ~delay_wave_t() MHASignal::delay_wave_t::~delay_wave_t ()

4.346.3 Member Function Documentation

4.346.3.1 process() mha_wave_t * MHASignal::delay_wave_t::process (mha_wave_t * s)

4.346.4 Member Data Documentation

4.346.4.1 delay unsigned int MHASignal::delay_wave_t::delay [private]

4.346.4.2 buffer MHASignal::waveform_t** MHASignal::delay_wave_t::buffer [private]

4.346.4.3 pos unsigned int MHASignal::delay_wave_t::pos [private]

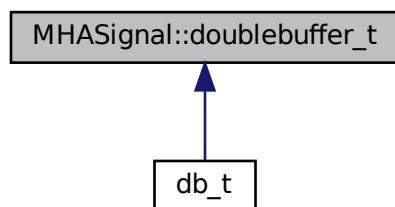
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

4.347 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**

4.347.1 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. 1205) is called whenever enough data is available.

4.347.2 Constructor & Destructor Documentation

```
4.347.2.1 doublebuffer_t() MHASignal::doublebuffer_t::doublebuffer_t (
    unsigned int nchannels_in,
    unsigned int nchannels_out,
    unsigned int outer_fragsize,
    unsigned int inner_fragsize )
```

Constructor of double buffer.

Parameters

<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)

4.347.2.2 ~doublebuffer_t() `MHASignal::doublebuffer_t::~doublebuffer_t () [virtual]`**4.347.3 Member Function Documentation****4.347.3.1 outer_process()** `mha_wave_t * MHASignal::doublebuffer_t::outer_process (mha_wave_t * s)`

Method to pass audio fragments into the inner layer.

Parameters

<i>s</i>	Pointer to input waveform fragment.
----------	-------------------------------------

Returns

Pointer to output waveform fragment.

4.347.3.2 inner_process() `virtual mha_wave_t* MHASignal::doublebuffer_t::inner_process (mha_wave_t * s) [protected], [pure virtual]`

Method to realize inner processing callback.

To be overwritten by derived classes.

Parameters

s	Pointer to input waveform fragment.
----------	-------------------------------------

Returns

Pointer to output waveform fragment.

Implemented in **db_t** (p. 373).

4.347.3.3 min() `unsigned int MHASignal::doublebuffer_t::min (`
`unsigned int a,`
`unsigned int b) [inline], [private]`

4.347.4 Member Data Documentation

4.347.4.1 outer_out `waveform_t MHASignal::doublebuffer_t::outer_out [private]`

4.347.4.2 this_outer_out `mha_wave_t MHASignal::doublebuffer_t::this_outer_out [private]`

4.347.4.3 inner_in `waveform_t MHASignal::doublebuffer_t::inner_in [private]`

4.347.4.4 inner_out `waveform_t MHASignal::doublebuffer_t::inner_out [private]`

4.347.4.5 k_inner unsigned int MHASignal::doublebuffer_t::k_inner [private]

4.347.4.6 k_outer unsigned int MHASignal::doublebuffer_t::k_outer [private]

4.347.4.7 ch unsigned int MHASignal::doublebuffer_t::ch [private]

The documentation for this class was generated from the following files:

- [mha_signal.hh](#)
- [mha_signal.cpp](#)

4.348 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 **rfftw_plan_wave2spec**
- rfftw_plan **rfftw_plan_spec2wave**
- fftw_plan **fftw_plan_fft**
- fftw_plan **fftw_plan_ifft**

4.348.1 Constructor & Destructor Documentation

4.348.1.1 `fft_t()` MHASignal::fft_t::fft_t (const unsigned int & n)

4.348.1.2 `~fft_t()` MHASignal::fft_t::~fft_t ()

4.348.2 Member Function Documentation

4.348.2.1 `wave2spec()` void MHASignal::fft_t::wave2spec (const **mha_wave_t** * wave, **mha_spec_t** * spec, bool swap)

fast fourier transform.

if swap is set, the buffer halves of the wave signal are exchanged before computing the fft.

```
4.348.2.2 spec2wave() [1/2] void MHASignal::fft_t::spec2wave (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

```
4.348.2.3 spec2wave() [2/2] void MHASignal::fft_t::spec2wave (
    const mha_spec_t * spec,
    mha_wave_t * wave,
    unsigned int offset )
```

wave may have fewer number of frames than needed for a complete iFFT.

Only as many frames are written into wave as fit, starting with offset offset of the complete iFFT.

```
4.348.2.4 forward() void MHASignal::fft_t::forward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

```
4.348.2.5 backward() void MHASignal::fft_t::backward (
    mha_spec_t * sIn,
    mha_spec_t * sOut )
```

```
4.348.2.6 wave2spec_scale() void MHASignal::fft_t::wave2spec_scale (
    const mha_wave_t * wave,
    mha_spec_t * spec,
    bool swap )
```

```
4.348.2.7 spec2wave_scale() void MHASignal::fft_t::spec2wave_scale (
    const mha_spec_t * spec,
    mha_wave_t * wave )
```

4.348.2.8 forward_scale() void MHASignal::fft_t::forward_scale (

```
mha_spec_t * sIn,
mha_spec_t * sOut )
```

4.348.2.9 backward_scale() void MHASignal::fft_t::backward_scale (

```
mha_spec_t * sIn,
mha_spec_t * sOut )
```

4.348.2.10 sort_fftw2spec() void MHASignal::fft_t::sort_fftw2spec (

```
fftw_real * s_fftw,
mha_spec_t * s_spec,
unsigned int ch ) [private]
```

Arrange the order of an fftw spectrum to the internal order.

The fftw spectrum is arranged [r0 r1 r2 ... rn-1 in in-1 ... i1], while the internal order is [r0 – r1 i1 r2 i2 ... rn-1 in-1 rn –].

4.348.2.11 sort_spec2fftw() void MHASignal::fft_t::sort_spec2fftw (

```
fftw_real * s_fftw,
const mha_spec_t * s_spec,
unsigned int ch ) [private]
```

Arrange the order of an internal spectrum to the fftw order.

4.348.3 Member Data Documentation

4.348.3.1 nfft unsigned int MHASignal::fft_t::nfft [private]

4.348.3.2 n_re unsigned int MHASignal::fft_t::n_re [private]

4.348.3.3 n_im unsigned int MHASignal::fft_t::n_im [private]

4.348.3.4 scale mha_real_t MHASignal::fft_t::scale [private]

4.348.3.5 buf_in mha_real_t* MHASignal::fft_t::buf_in [private]

4.348.3.6 buf_out mha_real_t* MHASignal::fft_t::buf_out [private]

4.348.3.7 fftw_plan_wave2spec rffftw_plan MHASignal::fft_t::fftw_plan_wave2spec [private]

4.348.3.8 fftw_plan_spec2wave rffftw_plan MHASignal::fft_t::fftw_plan_spec2wave [private]

4.348.3.9 fftw_plan_fft fftw_plan MHASignal::fft_t::fftw_plan_fft [private]

4.348.3.10 fftw_plan_ifft fftw_plan MHASignal::fft_t::fftw_plan_ifft [private]

The documentation for this class was generated from the following files:

- **mha_signal_fft.h**
- **mha_signal.cpp**

4.349 MHASignal::hilbert_fftw_t Class Reference

Public Member Functions

- **hilbert_fftw_t** (unsigned int len)
C'tor of hilbert_fftw_t (p. 1212).
- **~hilbert_fftw_t ()**
D'tor of hilbert_fftw_t (p. 1212).
- 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**

4.349.1 Constructor & Destructor Documentation

4.349.1.1 hilbert_fftw_t() MHASignal::hilbert_fftw_t::hilbert_fftw_t (unsigned int len)

C'tor of **hilbert_fftw_t** (p. 1212).

Parameters

<i>len</i>	fft length
------------	------------

4.349.1.2 ~hilbert_fftw_t() MHASignal::hilbert_fftw_t::~hilbert_fftw_t ()

D'tor of **hilbert_fftw_t** (p. 1212).

4.349.2 Member Function Documentation

4.349.2.1 hilbert() void MHASignal::hilbert_fftw_t::hilbert (const mha_wave_t * s_in, mha_wave_t * s_out)

4.349.3 Member Data Documentation

4.349.3.1 n unsigned int MHASignal::hilbert_fftw_t::n [private]

4.349.3.2 p1 rfftw_plan MHASignal::hilbert_fftw_t::p1 [private]

4.349.3.3 p2 fftw_plan MHASignal::hilbert_fftw_t::p2 [private]

4.349.3.4 buf_r_in fftw_real* MHASignal::hilbert_fftw_t::buf_r_in [private]

4.349.3.5 buf_r_out fftw_real* MHASignal::hilbert_fftw_t::buf_r_out [private]

4.349.3.6 buf_c_in fftw_complex* MHASignal::hilbert_fftw_t::buf_c_in [private]

4.349.3.7 buf_c_out fftw_complex* MHASignal::hilbert_fftw_t::buf_c_out [private]

4.349.3.8 SC mha_real_t MHASignal::hilbert_fftw_t::sc [private]

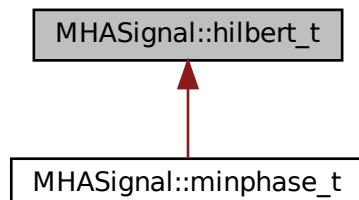
The documentation for this class was generated from the following file:

- [mha_signal.cpp](#)

4.350 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**

4.350.1 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.

4.350.2 Constructor & Destructor Documentation

4.350.2.1 hilbert_t() MHASignal::hilbert_t::hilbert_t (unsigned int len)

Parameters

<i>len</i>	Length of waveform segment
------------	----------------------------

4.350.2.2 ~hilbert_t() MHASignal::hilbert_t::~hilbert_t ()

4.350.3 Member Function Documentation

4.350.3.1 operator() void MHASignal::hilbert_t::operator() (const mha_wave_t * s_in, mha_wave_t * s_out)

Apply Hilbert transformation on a waveform segment.

4.350.4 Member Data Documentation

4.350.4.1 **h** void* MHASignal::hilbert_t::h [private]

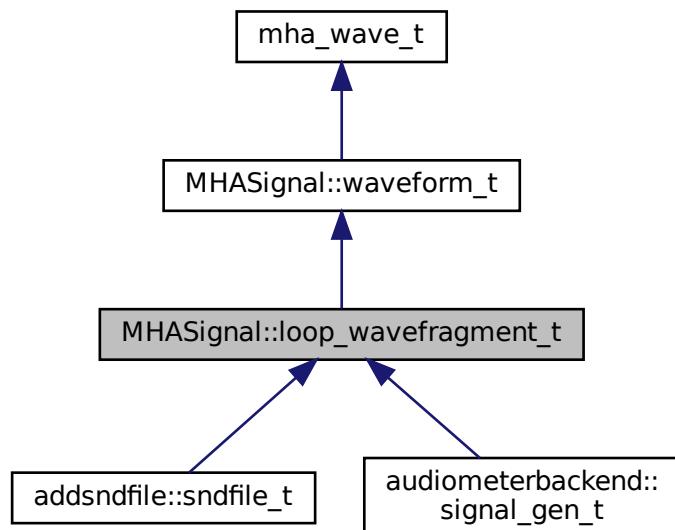
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

4.351 MHASignal::loop_wavefragment_t Class Reference

Copy a fixed waveform fragment to a series of waveform fragments of other size.

Inheritance diagram for MHASignal::loop_wavefragment_t:



Public Types

- enum **level_mode_t** { **relative**, **peak**, **rms**, **rms_limit40** }
Switch for playback level mode.
- enum **playback_mode_t** { **add**, **replace**, **input**, **mute** }
Switch for playback mode.

Public Member Functions

- **loop_wavefragment_t** (const **mha_wave_t** &src, bool loop, **level_mode_t** level_mode, std::vector< int > **channels**, unsigned int startpos=0)
*Constructor to create an instance of **loop_wavefragment_t** (p. 1216) 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

4.351.1 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.

4.351.2 Member Enumeration Documentation

4.351.2.1 **level_mode_t** enum **MHASignal::loop_wavefragment_t::level_mode_t**

Switch for playback level mode.

Enumerator

relative	The nominal level is applied as a gain to the source signal.
peak	The nominal level is the peak level of source signal in Pascal.
rms	The nominal level is the RMS level of the source signal in Pascal.
rms_limit40	

4.351.2.2 playback_mode_t enum MHASignal::loop_wavefragment_t::playback_mode_t

Switch for playback mode.

Enumerator

add	Add source signal to output stream.
replace	Replace output stream by source signal.
input	Do nothing, keep output stream (source position is unchanged).
mute	Mute output stream (source position is unchanged).

4.351.3 Constructor & Destructor Documentation

```
4.351.3.1 loop_wavefragment_t() MHASignal::loop_wavefragment_t::loop_wavefragment_t (
    const mha_wave_t & src,
    bool loop,
    level_mode_t level_mode,
    std::vector< int > channels,
    unsigned int startpos = 0 )
```

Constructor to create an instance of **loop_wavefragment_t** (p. 1216) 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. 1217) for details)
<i>channels</i>	Mapping of input to output channels.
<i>startpos</i>	Starting position

4.351.4 Member Function Documentation

4.351.4.1 get_mapping() `std::vector< int > MHASignal::loop_wavefragment_t::get_mapping (unsigned int channels)`

4.351.4.2 playback() [1/3] `void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode, mha_wave_t * level_pa, const std::vector< int > & channels)`

Add source waveform block to an output block.

Parameters

<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

4.351.4.3 playback() [2/3] `void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode, mha_wave_t * level_pa)`

Add source waveform block to an output block.

Parameters

<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.

4.351.4.4 `playback()` [3/3] `void MHASignal::loop_wavefragment_t::playback (mha_wave_t * s, playback_mode_t pmode)`

Add source waveform block to an output block.

Parameters

<code>s</code>	Output block (streamed signal).
<code>pmode</code>	Playback mode (add, replace, input, mute).

4.351.4.5 `set_level_lin()` `void MHASignal::loop_wavefragment_t::set_level_lin (mha_real_t l)`

4.351.4.6 `set_level_db()` `void MHASignal::loop_wavefragment_t::set_level_db (mha_real_t l)`

4.351.4.7 `rewind()` `void MHASignal::loop_wavefragment_t::rewind ()` [inline]

4.351.4.8 `locate_end()` `void MHASignal::loop_wavefragment_t::locate_end ()` [inline]

4.351.4.9 `is_playback_active()` `bool MHASignal::loop_wavefragment_t::is_playback_active () const` [inline]

4.351.5 Member Data Documentation

4.351.5.1 playback_channels std::vector<int> MHASignal::loop_wavefragment_t::playback_channels [private]

4.351.5.2 b_loop bool MHASignal::loop_wavefragment_t::b_loop [private]

4.351.5.3 pos unsigned int MHASignal::loop_wavefragment_t::pos [private]

4.351.5.4 intern_level MHASignal::waveform_t MHASignal::loop_wavefragment_t::intern_level [private]

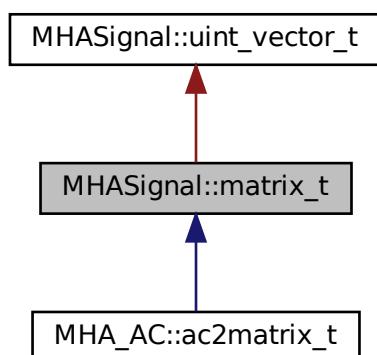
The documentation for this class was generated from the following files:

- [mha_signal.hh](#)
- [mha_signal.cpp](#)

4.352 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

4.352.1 Detailed Description

n-dimensional matrix with real or complex floating point values.

Warning

The member functions **imag()** (p. 1228) and **operator()** should only be called if the matrix is defined to hold complex values.

4.352.2 Constructor & Destructor Documentation

4.352.2.1 matrix_t() [1/5] MHASignal::matrix_t::matrix_t (

```
unsigned int nRows,
unsigned int nCols,
bool b_is_complex = true )
```

Create a two-dimensional matrix.

Parameters

<i>nrows</i>	Number of rows
<i>ncols</i>	Number of columns
<i>b_is_complex</i>	Add space for complex values

4.352.2.2 matrix_t() [2/5] `MHASignal::matrix_t::matrix_t (`
`const mha_spec_t & spec)`

Create a two-dimensional matrix from a spectrum, copy values.

Parameters

<i>spec</i>	Source spectrum structure
-------------	---------------------------

4.352.2.3 matrix_t() [3/5] `MHASignal::matrix_t::matrix_t (`
`const MHASignal::uint_vector_t & size,`
`bool b_is_complex = true)`

Create n-dimensional matrix, described by size argument.

Parameters

<i>size</i>	Size vector
<i>b_is_complex</i>	Add space for complex values

4.352.2.4 matrix_t() [4/5] `MHASignal::matrix_t::matrix_t (`
`const MHASignal::matrix_t & src)`

4.352.2.5 matrix_t() [5/5] `MHASignal::matrix_t::matrix_t (`
`const uint8_t * buf,`
`unsigned int len)`

Construct from memory area.

Warning

This constructor is not real time safe

4.352.2.6 ~matrix_t() `MHASignal::matrix_t::~matrix_t ()`

4.352.3 Member Function Documentation

4.352.3.1 operator=() [1/2] `matrix_t & MHASignal::matrix_t::operator= (const MHASignal::matrix_t & src)`

4.352.3.2 operator=() [2/2] `MHASignal::matrix_t & MHASignal::matrix_t::operator= (const comm_var_t & v)`

Fill matrix with data of an AC variable object.

Parameters

<code>v</code>	Source AC variable (comm_var_t (p. 360))
----------------	--

Note

The type and dimension of the AC variable must match the type and dimension of the matrix.

4.352.3.3 get_comm_var() `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 \(p. 360\)](#)), valid for the life time of the matrix.

4.352.3.4 dimension() `unsigned int MHASignal::matrix_t::dimension () const [inline]`

Return the dimension of the matrix.

Returns

Dimension of the matrix

4.352.3.5 size() `unsigned int MHASignal::matrix_t::size (unsigned int k) const [inline]`

Return the size of the matrix.

Parameters

<code>k</code>	Dimension
----------------	-----------

Returns

Size of the matrix in dimension `k`

4.352.3.6 get_nelements() `unsigned int MHASignal::matrix_t::get_nelements () const`

Return total number of elements.

4.352.3.7 is_same_size() `bool MHASignal::matrix_t::is_same_size (const MHASignal::matrix_t & src)`

Test if matrix has same size as other.

4.352.3.8 iscomplex() `bool MHASignal::matrix_t::iscomplex () const [inline]`

Return information about complexity.

4.352.3.9 real() [1/4] `mha_real_t& MHASignal::matrix_t::real (const MHASignal::uint_vector_t & index) [inline]`

Access real part of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

4.352.3.10 `imag()` [1/4] `mha_real_t& MHASignal::matrix_t::imag (const MHASignal::uint_vector_t & index)` [inline]

Access imaginary part of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

4.352.3.11 `operator()()` [1/4] `mha_complex_t& MHASignal::matrix_t::operator() (const MHASignal::uint_vector_t & index)` [inline]

Access complex value of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

4.352.3.12 `real()` [2/4] `const mha_real_t& MHASignal::matrix_t::real (const MHASignal::uint_vector_t & index) const` [inline]

Access real part of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

4.352.3.13 `imag()` [2/4] `const mha_real_t& MHASignal::matrix_t::imag (const MHASignal::uint_vector_t & index) const` [inline]

Access imaginary part of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

```
4.352.3.14 operator() [2/4] const mha_complex_t& MHASignal::matrix_t::operator()
(
    const MHASignal::uint_vector_t & index ) const [inline]
```

Access complex value of an element in a n-dimensional matrix.

Parameters

<i>index</i>	Index vector
--------------	--------------

```
4.352.3.15 real() [3/4] mha_real_t& MHASignal::matrix_t::real (
    unsigned int row,
    unsigned int col ) [inline]
```

Access real part of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

```
4.352.3.16 imag() [3/4] mha_real_t& MHASignal::matrix_t::imag (
    unsigned int row,
    unsigned int col ) [inline]
```

Access imaginary part of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

4.352.3.17 operator()() [3/4] `mha_complex_t& MHASignal::matrix_t::operator() (`
 `unsigned int row,`
 `unsigned int col) [inline]`

Access complex value of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

4.352.3.18 real() [4/4] `const mha_real_t& MHASignal::matrix_t::real (`
 `unsigned int row,`
 `unsigned int col) const [inline]`

Access real part of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

4.352.3.19 imag() [4/4] `const mha_real_t& MHASignal::matrix_t::imag (`
 `unsigned int row,`
 `unsigned int col) const [inline]`

Access imaginary part of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

```
4.352.3.20 operator() [4/4] const mha_complex_t& MHASignal::matrix_t::operator()
(
    unsigned int row,
    unsigned int col ) const [inline]
```

Access complex value of an element in a two-dimensional matrix.

Parameters

<i>row</i>	Row number of element
<i>col</i>	Column number of element

```
4.352.3.21 get_nreals() unsigned int MHASignal::matrix_t::get_nreals () const [inline]
```

```
4.352.3.22 get_index() [1/2] unsigned int MHASignal::matrix_t::get_index (
    unsigned int row,
    unsigned int col ) const
```

```
4.352.3.23 get_index() [2/2] unsigned int MHASignal::matrix_t::get_index (
    const MHASignal::uint_vector_t & index ) const
```

```
4.352.3.24 nbytes() unsigned int MHASignal::matrix_t::nbytes () const
```

Return number of bytes needed to store into memory.

```
4.352.3.25 write() unsigned int MHASignal::matrix_t::write (
    uint8_t * buf,
    unsigned int len ) const
```

Copy to memory area.

4.352.3.26 `get_rdata()` const `mha_real_t*` MHASignal::matrix_t::get_rdata () const [inline]

Return pointer of real data.

4.352.3.27 `get_cdata()` const `mha_complex_t*` MHASignal::matrix_t::get_cdata () const [inline]

Return pointer of complex data.

4.352.4 Member Data Documentation

4.352.4.1 `complex_ofs` uint32_t MHASignal::matrix_t::complex_ofs [private]

4.352.4.2 `nelements` uint32_t MHASignal::matrix_t::nelements [private]

4.352.4.3 `rdata` `mha_real_t*` MHASignal::matrix_t::rdata

4.352.4.4 `cdata` `mha_complex_t*` MHASignal::matrix_t::cdata

4.352.4.5 "@1 union { ... } [private]

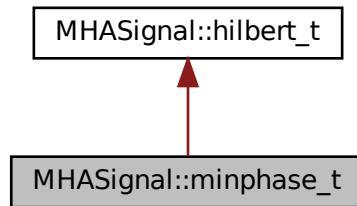
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.353 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

4.353.1 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 $\mathcal{H}\{\dots\}$.

4.353.2 Constructor & Destructor Documentation

4.353.2.1 minphase_t() `MHASignal::minphase_t::minphase_t (`
`unsigned int fftlen,`
`unsigned int ch)`

Constructor.

Parameters

<i>ffflen</i>	FFT length
<i>ch</i>	Number of channels

4.353.3 Member Function Documentation

4.353.3.1 operator() `void MHASignal::minphase_t::operator() (mha_spec_t * s)`

Transform input spectrum to a minimal-phase spectrum, discarding the original phase.

Parameters

<i>s</i>	Spectrum to operate on.
----------	-------------------------

4.353.4 Member Data Documentation

4.353.4.1 phase `MHASignal::waveform_t MHASignal::minphase_t::phase [private]`

The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.354 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`

4.354.1 Detailed Description

Simple simulation of fixpoint quantization.

4.354.2 Constructor & Destructor Documentation

4.354.2.1 `quantizer_t()` `MHASignal::quantizer_t::quantizer_t (` `unsigned int num_bits)`

Constructor.

Parameters

<code>num_bits</code>	Number of bits to simulate, or zero for limiting to [-1,1] only.
-----------------------	--

4.354.3 Member Function Documentation

4.354.3.1 `operator()()` `void MHASignal::quantizer_t::operator() (` `mha_wave_t & s)`

Quantization of a waveform fragment.

Parameters

<code>s</code>	Waveform fragment to be quantized.
----------------	------------------------------------

4.354.4 Member Data Documentation

4.354.4.1 limit `bool MHASignal::quantizer_t::limit` [private]

4.354.4.2 upscale `mha_real_t MHASignal::quantizer_t::upscale` [private]

4.354.4.3 downscale `mha_real_t MHASignal::quantizer_t::downscale` [private]

4.354.4.4 up_limit `mha_real_t MHASignal::quantizer_t::up_limit` [private]

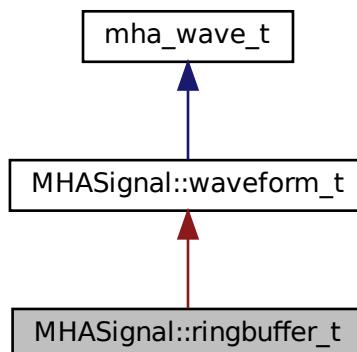
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.355 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

4.355.1 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. 1239) method. Individual audio samples can be accessed and altered using the **value** (p. 1238) method. Blocks of audio data can be deleted from the ringbuffer using the **discard** (p. 1239) method.

4.355.2 Constructor & Destructor Documentation

4.355.2.1 **ringbuffer_t()** `ringbuffer_t::ringbuffer_t (` `unsigned frames,` `unsigned channels,` `unsigned prefilled_frames)`

Creates new ringbuffer for time domain signal.

Constructor allocates enough storage so that *frames* audio samples can be stored in the ringbuffer.

Parameters

<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

MHA_Error (p. 763) if *prefilled_frames > frames*

4.355.3 Member Function Documentation

4.355.3.1 contained_frames() `unsigned MHASignal::ringbuffer_t::contained_frames () const [inline]`

number of currently contained frames

4.355.3.2 value() `mha_real_t& MHASignal::ringbuffer_t::value (unsigned frame, unsigned channel) [inline]`

Access to value stored in ringbuffer.

frame index is relative to the oldest frame stored in the ringbuffer, therefore, the meaning of the *frame* changes when the **discard** (p. 1239) 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. 763)	if channel or frame out of bounds.
---------------------------	------------------------------------

4.355.3.3 discard() void MHASignal::ringbuffer_t::discard (unsigned *frames*) [inline]

Discards the oldest frames.

Makes room for new **write** (p. 1239), alters base frame index for **value** (p. 1238)

Parameters

<i>frames</i>	how many frames to discard.
---------------	-----------------------------

Exceptions

MHA_Error (p. 763)	if frames > contained_frames (p. 1238)
---------------------------	---

4.355.3.4 write() void MHASignal::ringbuffer_t::write (**mha_wave_t** & *signal*) [inline]

Copies the contents of the signal into the ringbuffer if there is enough space.

Parameters

<i>signal</i>	New signal to be appended to the signal already present in the ringbuffer
---------------	---

Exceptions

MHA_Error (p. 763)	if there is not enough space or if the channel count mismatches. Nothing is copied if the space is insufficient.
---------------------------	---

4.355.4 Member Data Documentation

4.355.4.1 next_read_frame_index `unsigned MHASignal::ringbuffer_t::next_read_<→frame_index [private]`

Index of oldest frame in underlying storage for the ringbuffer.

This value is added to the frame parameter of the **value** (p. 1238) method, and this value is altered when **discard** (p. 1239) is called.

4.355.4.2 next_write_frame_index `unsigned MHASignal::ringbuffer_t::next_write_<→frame_index [private]`

Index of first free frame in underlying storage.

Next frame to be stored will be placed here.

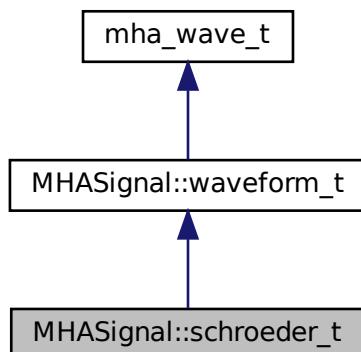
The documentation for this class was generated from the following files:

- [mha_signal.hh](#)
- [mha_signal.cpp](#)

4.356 MHASignal::schroeder_t Class Reference

Schroeder tone complex class.

Inheritance diagram for MHASignal::schroeder_t:



Public Types

- enum **sign_t** { **up**, **down** }
Enumerator for sign of Schroeder tone complex sweep direction.
- typedef float(*) **groupdelay_t** (float f, float fmin, float fmax)
Function type for group delay definition.

Public Member Functions

- schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::sign_t** sign=**up**, **mha_real_t** speed=1)
Constructor.
- schroeder_t** (unsigned int len, unsigned int **channels**=1, **schroeder_t::groupdelay_t** freqfun= **MHASignal::schroeder_t::identity**, float fmin=0, float fmax=1, float eps=1e-10)
Construct create Schroeder tone complex from a given frequency function.

Static Public Member Functions

- static float **identity** (float x, float, float)
- static float **log_up** (float x, float fmin, float fmax)
- static float **log_down** (float x, float fmin, float fmax)

Additional Inherited Members

4.356.1 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.

4.356.2 Member Typedef Documentation

4.356.2.1 **groupdelay_t** typedef float(*) MHASignal::schroeder_t::groupdelay_t (float f, float fmin, float fmax)

Function type for group delay definition.

Parameters

<i>f</i>	Frequency relative to Nyquist frequency.
<i>fmin</i>	Minimum frequency relative to Nyquist frequency.
<i>fmax</i>	Maximum frequency relative to Nyquist frequency.

4.356.3 Member Enumeration Documentation

4.356.3.1 `sign_t` enum `MHASignal::schroeder_t::sign_t`

Enumerator for sign of Schroeder tone complex sweep direction.

Enumerator

<code>up</code>	Sweep from zero to Nyquist frequency ($\sigma = -1$)
<code>down</code>	Sweep from Nyquist frequency to zero ($\sigma = +1$)

4.356.4 Constructor & Destructor Documentation

4.356.4.1 `schroeder_t()` [1/2] `MHASignal::schroeder_t::schroeder_t (`

```
unsigned int len,
unsigned int channels = 1,
schroeder_t::sign_t sign = up,
mha_real_t speed = 1 )
```

Constructor.

Parameters of the Schroeder tone complex are configured in the constructor.

Parameters

<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)

4.356.4.2 schroeder_t() [2/2] `MHASignal::schroeder_t::schroeder_t (`

```
    unsigned int len,
    unsigned int channels = 1,
    schroeder_t::groupdelay_t freqfun = MHASignal::schroeder_t::identity,
    float fmin = 0,
    float fmax = 1,
    float eps = 1e-10 )
```

Construct create Schroeder tone complex from a given frequency function.

The frequency function $g(f)$ defines the sweep speed and sign (based on the group delay). It must be defined in the interval $[0,1]$ and should return values in the interval $[0,1]$.

$$\Phi(f) = -4\pi\tau \int_0^{\tau} g(f) \, df, \quad S(f) = e^{i\Phi(f)}$$

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.

4.356.5 Member Function Documentation**4.356.5.1 identity()** `static float MHASignal::schroeder_t::identity (`

```
    float x,
    float ,
    float ) [inline], [static]
```

4.356.5.2 `log_up()` static float MHASignal::schroeder_t::log_up (float *x*, float *fmin*, float *fmax*) [inline], [static]

4.356.5.3 `log_down()` static float MHASignal::schroeder_t::log_down (float *x*, float *fmin*, float *fmax*) [inline], [static]

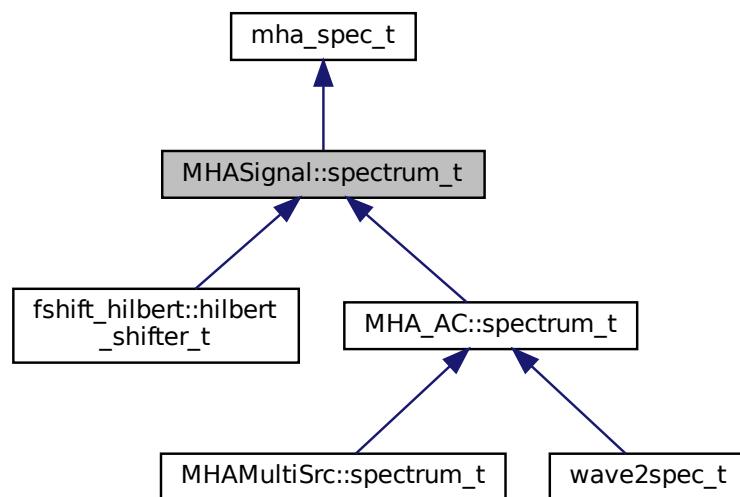
The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.357 MHASignal::spectrum_t Class Reference

a signal processing class for spectral data (based on `mha_spec_t` (p. 793))

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

4.357.1 Detailed Description

a signal processing class for spectral data (based on **mha_spec_t** (p. 793))

4.357.2 Constructor & Destructor Documentation

4.357.2.1 **spectrum_t()** [1/4] `spectrum_t::spectrum_t (` `const unsigned int & frames,` `const unsigned int & channels)`

constructor of spectrum class

Allocates buffers and initializes memory to zeros.

Parameters

<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

4.357.2.2 `spectrum_t()` [2/4] `spectrum_t::spectrum_t (const mha_spec_t & src)` [explicit]

Copy constructor.

4.357.2.3 `spectrum_t()` [3/4] `spectrum_t::spectrum_t (const MHASignal::spectrum_t & src)`

Copy constructor.

4.357.2.4 `spectrum_t()` [4/4] `spectrum_t::spectrum_t (const std::vector< mha_complex_t > & src)`

4.357.2.5 `~spectrum_t()` `spectrum_t::~spectrum_t (void)` [virtual]

Reimplemented in **MHA_AC::spectrum_t** (p. 735).

4.357.3 Member Function Documentation

4.357.3.1 `operator()()` `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

4.357.3.2 operator[]() *mha_complex_t&* MHASignal::spectrum_t::operator[] (unsigned int *k*) [inline]

Access to a single element, direct index into data buffer.

Parameters

<i>k</i>	Buffer index
----------	--------------

Returns

Reference to element

4.357.3.3 value() *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

4.357.3.4 `copy()` `void spectrum_t::copy (`
`const mha_spec_t & src)`

copy all elements from a spectrum

Parameters

<i>src</i>	input spectrum
------------	----------------

4.357.3.5 `copy_channel()` `void spectrum_t::copy_channel (`
`const mha_spec_t & s,`
`unsigned sch,`
`unsigned dch)`

Copy one channel of a given spectrum signal to a target channel.

Parameters

<i>s</i>	Input spectrum signal
<i>sch</i>	Channel index in source signal
<i>dch</i>	Channel index in destination (this) signal

4.357.3.6 `export_to()` `void spectrum_t::export_to (`
`mha_spec_t & dest)`

copy elements to spectrum structure

Parameters

<i>dest</i>	destination spectrum structure
-------------	--------------------------------

4.357.3.7 `scale()` `void spectrum_t::scale (`
`const unsigned int & a,`
`const unsigned int & b,`
`const unsigned int & ch,`
`const mha_real_t & val)`

scale section [a,b) in channel "ch" by "val"

Parameters

<i>a</i>	starting frame
<i>b</i>	end frame (excluded)
<i>ch</i>	channel number
<i>val</i>	scale factor

```
4.357.3.8 scale_channel() void spectrum_t::scale_channel (
    const unsigned int & ch,
    const mha_real_t & src )
```

scale all elements in one channel

Parameters

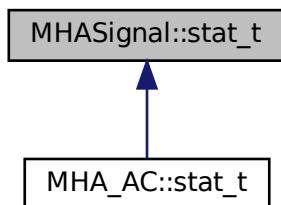
<i>ch</i>	channel number
<i>src</i>	scale factor

The documentation for this class was generated from the following files:

- [mha_signal.hh](#)
- [mha_signal.cpp](#)

4.358 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**

4.358.1 Constructor & Destructor Documentation

4.358.1.1 stat_t() MHASignal::stat_t::stat_t (

```
    const unsigned int & frames,
    const unsigned int & channels )
```

4.358.2 Member Function Documentation

4.358.2.1 mean() void MHASignal::stat_t::mean (

```
    mha_wave_t & m )
```

4.358.2.2 mean_std() void MHASignal::stat_t::mean_std (

```
    mha_wave_t & m,
    mha_wave_t & s )
```

4.358.2.3 `push()` [1/2] `void MHASignal::stat_t::push (`
`const mha_wave_t & x)`

4.358.2.4 `push()` [2/2] `void MHASignal::stat_t::push (`
`const mha_real_t & x,`
`const unsigned int & k,`
`const unsigned int & ch)`

4.358.3 Member Data Documentation

4.358.3.1 `n` `MHASignal::waveform_t MHASignal::stat_t::n` [private]

4.358.3.2 `sum` `MHASignal::waveform_t MHASignal::stat_t::sum` [private]

4.358.3.3 `sum2` `MHASignal::waveform_t MHASignal::stat_t::sum2` [private]

The documentation for this class was generated from the following files:

- `mha_signal.hh`
- `mha_signal.cpp`

4.359 `MHASignal::subsample_delay_t` Class Reference

implements subsample delay in spectral domain.

Public Member Functions

- **`subsample_delay_t`** (`const std::vector< float > &subsample_delay, unsigned fftlen)`
Constructor computes complex phase factors to apply to achieve subsample delay.
- **`void process (mha_spec_t *s)`**
Apply the phase_gains to s to achieve the subsample delay.
- **`void process (mha_spec_t *s, unsigned idx)`**
Apply the pase gains to channel idx in s to achieve the subsample delay in channel idx.

Public Attributes

- **spectrum_t phase_gains**

The complex factors to apply to achieve the necessary phase shift.

Private Attributes

- unsigned **last_complex_bin**

index of the last complex fft bin for the used fft length.

4.359.1 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.

4.359.2 Constructor & Destructor Documentation

4.359.2.1 subsample_delay_t() `MHASignal::subsample_delay_t::subsample_delay_t (`
`const std::vector< float > & subsample_delay,`
`unsigned fftlen)`

Constructor computes complex phase factors to apply to achieve subsample delay.

Parameters

<code>subsample_delay</code>	The subsample delay to apply. $-0.5 \leq \text{subsample_delay} \leq 0.5$
<code>fftlen</code>	FFT length

Exceptions

MHA_Error (p. 763) if the parameters are out of range

4.359.3 Member Function Documentation

4.359.3.1 process() [1/2] void MHASignal::subsample_delay_t::process (
`mha_spec_t * s)`

Apply the phase_gains to s to achieve the subsample delay.

4.359.3.2 process() [2/2] void MHASignal::subsample_delay_t::process (
`mha_spec_t * s,`
`unsigned idx)`

Apply the pase gains to channel idx in s to achieve the subsample delay in channel idx.

Parameters

<i>s</i>	signal
<i>idx</i>	channel index, 0-based

Exceptions

MHA_Error (p. 763)	if idx >= s->num_channels
---------------------------	---------------------------

4.359.4 Member Data Documentation

4.359.4.1 phase_gains `spectrum_t` MHASignal::subsample_delay_t::phase_gains

The complex factors to apply to achieve the necessary phase shift.

4.359.4.2 last_complex_bin `unsigned` MHASignal::subsample_delay_t::last_complex_bin
[private]

index of the last complex fft bin for the used fft length.

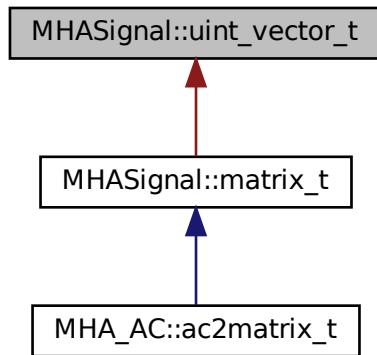
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

4.360 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. 1255).*
- 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`

4.360.1 Detailed Description

Vector of unsigned values, used for size and index description of n-dimensional matrixes.

4.360.2 Constructor & Destructor Documentation

4.360.2.1 `uint_vector_t()` [1/3] `MHASignal::uint_vector_t::uint_vector_t (`
`unsigned int len)`

Constructor, initializes all elements to zero.

Parameters

<code>len</code>	Length of vector.
------------------	-------------------

4.360.2.2 `uint_vector_t()` [2/3] `MHASignal::uint_vector_t::uint_vector_t (`
`const uint_vector_t & src)`

4.360.2.3 `uint_vector_t()` [3/3] `MHASignal::uint_vector_t::uint_vector_t (`
`const uint8_t * buf,`
`unsigned int len)`

Construct from memory area.

Warning

This constructor is not real time safe

4.360.2.4 ~uint_vector_t() MHASignal::uint_vector_t::~uint_vector_t ()**4.360.3 Member Function Documentation****4.360.3.1 operator==()** bool MHASignal::uint_vector_t::operator== (const uint_vector_t & src) const

Check for equality.

4.360.3.2 operator=() uint_vector_t & MHASignal::uint_vector_t::operator= (const uint_vector_t & src)

Assign from other **uint_vector_t** (p. 1255).

Warning

This assignment will fail if the lengths mismatch.

4.360.3.3 get_length() unsigned int MHASignal::uint_vector_t::get_length () const [inline]

Return the length of the vector.

4.360.3.4 operator[]() [1/2] const uint32_t& MHASignal::uint_vector_t::operator[] (unsigned int k) const [inline]

Read-only access to elements.

4.360.3.5 operator[]() [2/2] `uint32_t& MHASignal::uint_vector_t::operator[] (unsigned int k) [inline]`

Access to elements.

4.360.3.6 nbytes() `unsigned int MHASignal::uint_vector_t::nbytes () const`

Return number of bytes needed to store into memory.

4.360.3.7 write() `unsigned int MHASignal::uint_vector_t::write (uint8_t * buf, unsigned int len) const`

Copy to memory area.

4.360.3.8 getdata() `const uint32_t* MHASignal::uint_vector_t::getdata () const [inline]`

Return pointer to the data field.

4.360.4 Member Data Documentation

4.360.4.1 length `uint32_t MHASignal::uint_vector_t::length [protected]`

4.360.4.2 data `uint32_t* MHASignal::uint_vector_t::data [protected]`

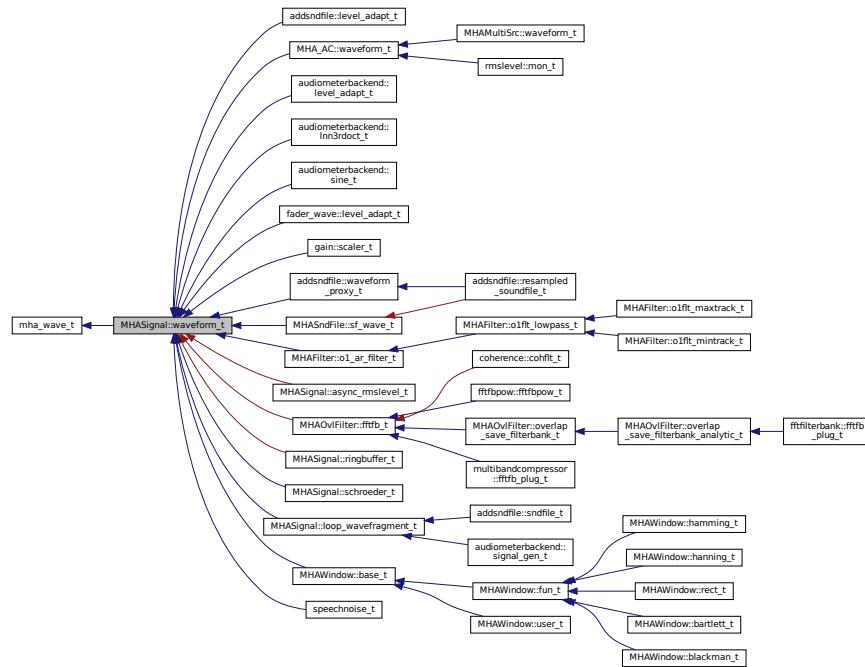
The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

4.361 MHASignal::waveform_t Class Reference

signal processing class for waveform data (based on `mha_wave_t` (p. 839))

Inheritance diagram for MHASignal::waveform_t:



Public Member Functions

- **waveform_t** (const unsigned int &frames, const unsigned int & **channels**)
constructor of waveform_t (p. 1259)
 - **waveform_t** (const **mhaconfig_t** &cf)
Constructor to create a waveform from plugin configuration.
 - **waveform_t** (const **mha_wave_t** &src)
Copy contructor for mha_wave_t (p. 839) source.
 - **waveform_t** (const **MHASignal::waveform_t** &src)
Copy contructor.
 - **waveform_t** (const std::vector< **mha_real_t** > &src)
Copy contructor for std::vector<mha_real_t> source.
 - virtual ~**waveform_t** (void)
 - std::vector< **mha_real_t** > **flatten** () const
 - **operator std::vector< mha_real_t >** () const
 - void **operator=** (const **mha_real_t** &v)
 - **mha_real_t** & **operator[]** (unsigned int k)
 - const **mha_real_t** & **operator[]** (unsigned int k) const
 - **mha_real_t** & **value** (unsigned int t, unsigned int ch)

- Element accessor.*
- **mha_real_t & operator()** (unsigned int t, unsigned int ch)

Element accessor.
 - const **mha_real_t & value** (unsigned int t, unsigned int ch) const

Constant element accessor.
 - const **mha_real_t & operator()** (unsigned int t, unsigned int ch) const

Constant element accessor.
 - **mha_real_t sum** (const unsigned int &a, const unsigned int &b)

sum of all elements between [a,b] in all channels
 - **mha_real_t sum** (const unsigned int &a, const unsigned int &b, const unsigned int &ch)

sum of all elements between [a,b] in channel ch
 - **mha_real_t sum ()**

sum of all elements
 - **mha_real_t sumsqr ()**

sum of square of all elements
 - **mha_real_t sum_channel** (const unsigned int &)

return sum of all elements in one channel
 - void **assign** (const unsigned int &k, const unsigned int &ch, const **mha_real_t** &val)

set frame "k" in channel "ch" to value "val"
 - void **assign** (const **mha_real_t** &)

set all elements to value
 - void **assign_frame** (const unsigned int &k, const **mha_real_t** &val)

assign value "val" to frame k in all channels
 - void **assign_channel** (const unsigned int &c, const **mha_real_t** &val)

assign value "val" to channel ch in all frames
 - void **copy** (const std::vector< **mha_real_t** > &v)
 - void **copy** (const **mha_wave_t** &)

copy data from source into current waveform
 - void **copy** (const **mha_wave_t** *)
 - void **copy_channel** (const **mha_wave_t** &, unsigned int, unsigned int)

Copy one channel of a given waveform signal to a target channel.
 - void **copy_from_at** (unsigned int, unsigned int, const **mha_wave_t** &, unsigned int)

Copy part of the source signal into part of this waveform object.
 - void **export_to** (**mha_wave_t** &)

*copy data into allocated **mha_wave_t** (p. 839) 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

4.361.1 Detailed Description

signal processing class for waveform data (based on **mha_wave_t** (p. 839))

4.361.2 Constructor & Destructor Documentation

4.361.2.1 waveform_t() [1/5] waveform_t::waveform_t (

const unsigned int & <i>frames</i> ,	
const unsigned int & <i>channels</i>)	

constructor of **waveform_t** (p. 1259)

Allocates buffer memory and initializes values to zero.

Parameters

<i>frames</i>	number of frames in each channel
<i>channels</i>	number of channels

4.361.2.2 waveform_t() [2/5] waveform_t::waveform_t (

const mhaconfig_t & <i>cf</i>) [explicit]	
---	--

Constructor to create a waveform from plugin configuration.

Parameters

<i>cf</i>	Plugin configuration
-----------	----------------------

4.361.2.3 waveform_t() [3/5] `waveform_t::waveform_t (`
 `const mha_wave_t & src)` [explicit]

Copy contructor for `mha_wave_t` (p. [839](#)) source.

4.361.2.4 waveform_t() [4/5] `waveform_t::waveform_t (`
 `const MHASignal::waveform_t & src)`

Copy contructor.

4.361.2.5 waveform_t() [5/5] `waveform_t::waveform_t (`
 `const std::vector< mha_real_t > & src)`

Copy contructor for `std::vector<mha_real_t>` source.

A waveform structure with a single channel is created, the length is equal to the number of elements in the source vector.

4.361.2.6 ~waveform_t() `waveform_t::~waveform_t (`
 `void)` [virtual]

Reimplemented in `MHA_AC::waveform_t` (p. [739](#)).

4.361.3 Member Function Documentation

4.361.3.1 flatten() `std::vector< mha_real_t > waveform_t::flatten () const`

4.361.3.2 operator std::vector< mha_real_t >() MHASignal::waveform_t::operator std::vector< mha_real_t > () const [explicit]

4.361.3.3 operator=() void MHASignal::waveform_t::operator= (const mha_real_t & v) [inline]

4.361.3.4 operator[](1/2) mha_real_t& MHASignal::waveform_t::operator[] (unsigned int k) [inline]

4.361.3.5 operator[](2/2) const mha_real_t& MHASignal::waveform_t::operator[] (unsigned int k) const [inline]

4.361.3.6 value() [1/2] 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

4.361.3.7 operator()() [1/2] 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

4.361.3.8 `value()` [2/2] const `mha_real_t&` MHASignal::waveform_t::value (

```
unsigned int t,
unsigned int ch) const [inline]
```

Constant element accessor.

Parameters

<i>t</i>	Frame number
<i>ch</i>	Channel number

Returns

Reference to element

4.361.3.9 `operator()()` [2/2] const `mha_real_t&` MHASignal::waveform_t::operator() (

```
unsigned int t,
unsigned int ch) const [inline]
```

Constant element accessor.

Parameters

<i>t</i>	Frame number
<i>ch</i>	Channel number

Returns

Reference to element

4.361.3.10 sum() [1/3] `mha_real_t waveform_t::sum (`

```
const unsigned int & a,  
const unsigned int & b )
```

sum of all elements between [a,b) in all channels

Parameters

<i>a</i>	starting frame
<i>b</i>	end frame (excluded)

Returns

sum

4.361.3.11 sum() [2/3] `mha_real_t waveform_t::sum (`

```
const unsigned int & a,  
const unsigned int & b,  
const unsigned int & ch )
```

sum of all elements between [a,b) in channel ch

Parameters

<i>a</i>	starting frame
<i>b</i>	end frame (excluded)
<i>ch</i>	channel number

Returns

sum

4.361.3.12 sum() [3/3] `mha_real_t waveform_t::sum ()`

sum of all elements

Returns

sum of all elements

4.361.3.13 sumsqr() `mha_real_t waveform_t::sumsqr ()`

sum of square of all elements

Returns

sum of square of all elements

4.361.3.14 sum_channel() `mha_real_t waveform_t::sum_channel (const unsigned int & ch)`

return sum of all elements in one channel

Parameters

<code>ch</code>	channel number
-----------------	----------------

Returns

sum

4.361.3.15 assign() [1/2] `void waveform_t::assign (const unsigned int & k, const unsigned int & ch, const mha_real_t & val)`

set frame "k" in channel "ch" to value "val"

Parameters

<code>k</code>	frame number
<code>ch</code>	channel number
<code>val</code>	new value

4.361.3.16 assign() [2/2] void waveform_t::assign (const mha_real_t & val)

set all elements to value

Parameters

val	new value
-----	-----------

4.361.3.17 assign_frame() void waveform_t::assign_frame (const unsigned int & k, const mha_real_t & val)

assign value "val" to frame k in all channels

Parameters

k	frame number
val	new value

4.361.3.18 assign_channel() void waveform_t::assign_channel (const unsigned int & ch, const mha_real_t & val)

assign value "val" to channel ch in all frames

Parameters

ch	channel number
val	new value

4.361.3.19 copy() [1/3] void waveform_t::copy (const std::vector< mha_real_t > & v)

4.361.3.20 copy() [2/3] `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)
------------	---

4.361.3.21 copy() [3/3] `void waveform_t::copy (`
`const mha_wave_t * src)`

4.361.3.22 copy_channel() `void waveform_t::copy_channel (`
`const mha_wave_t & src,`
`unsigned int src_channel,`
`unsigned int dest_channel)`

Copy one channel of a given waveform signal to a target channel.

Parameters

<i>src</i>	Input waveform signal
<i>src_channel</i>	Channel in source signal
<i>dest_channel</i>	Channel number in destination signal

4.361.3.23 copy_from_at() `void waveform_t::copy_from_at (`
`unsigned int to_pos,`
`unsigned int len,`
`const mha_wave_t & src,`
`unsigned int from_pos)`

Copy part of the source signal into part of this waveform object.

Source and target have to have the same number of channels.

Parameters

<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

4.361.3.24 export_to() `void waveform_t::export_to (mha_wave_t & dest)`

copy data into allocated **mha_wave_t** (p. 839) structure

Parameters

<i>dest</i>	destination structure
-------------	-----------------------

4.361.3.25 limit() `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

4.361.3.26 power() `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)
------------	--------------------------------

4.361.3.27 powspec() 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)
------------	-------------------------------------

4.361.3.28 scale() [1/2] void waveform_t::scale (const unsigned int & a, const unsigned int & b, const unsigned int & ch, const mha_real_t & val)

scale section [a,b) in channel "ch" by "val"

Parameters

<i>a</i>	starting frame
<i>b</i>	end frame (excluded)
<i>ch</i>	channel number
<i>val</i>	scale factor

4.361.3.29 scale() [2/2] void waveform_t::scale (const unsigned int & k, const unsigned int & ch, const mha_real_t & val)

scale one element

Parameters

<i>k</i>	frame number
<i>ch</i>	channel number
<i>val</i>	scale factor

```
4.361.3.30 scale_channel() void waveform_t::scale_channel (
    const unsigned int & ch,
    const mha_real_t & src )
```

scale one channel of target with a scalar

Parameters

<i>ch</i>	channel number
<i>src</i>	factor

```
4.361.3.31 scale_frame() void waveform_t::scale_frame (
    const unsigned int & frame,
    const mha_real_t & val )
```

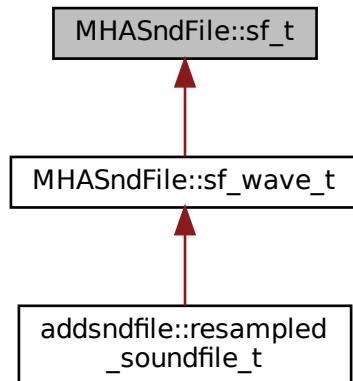
```
4.361.3.32 get_size() unsigned int MHASignal::waveform_t::get_size ( ) const [inline]
```

The documentation for this class was generated from the following files:

- **mha_signal.hh**
- **mha_signal.cpp**

4.362 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**

4.362.1 Constructor & Destructor Documentation

4.362.1.1 sf_t() MHASndFile::sf_t::sf_t (const std::string & fname)

4.362.1.2 ~sf_t() MHASndFile::sf_t::~sf_t ()

4.362.2 Member Data Documentation

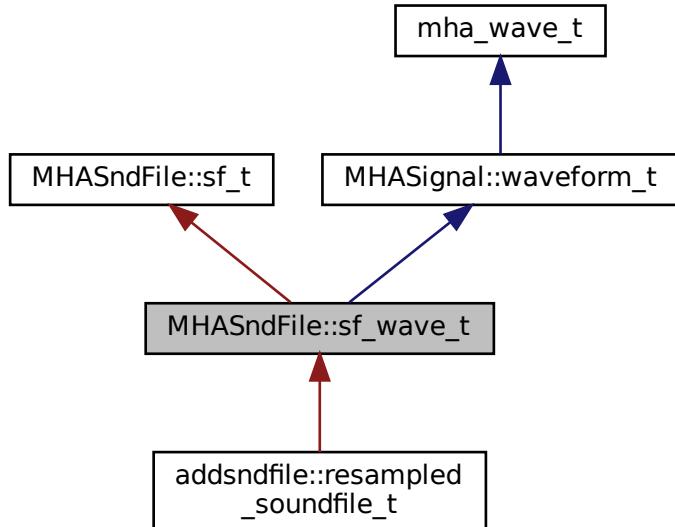
4.362.2.1 sf SNDFILE* MHASndFile::sf_t::sf

The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

4.363 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

4.363.1 Constructor & Destructor Documentation

```
4.363.1.1 sf_wave_t() MHASndFile::sf_wave_t::sf_wave_t (
    const std::string & fname,
    mha_real_t peaklevel_db,
    unsigned int maxlen = std::numeric_limits<unsigned int>:: max(),
    unsigned int startpos = 0,
    std::vector< int > channel_map = std::vector<int>() )
```

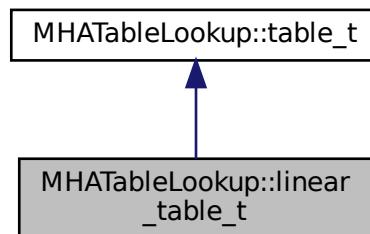
The documentation for this class was generated from the following files:

- **mhasndfile.h**
- **mhasndfile.cpp**

4.364 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. 1274) 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

4.364.1 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.

4.364.2 Constructor & Destructor Documentation

4.364.2.1 linear_table_t() `linear_table_t::linear_table_t (void)`

contructor creates an empty **linear_table_t** (p. 1274) 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.

4.364.2.2 ~linear_table_t() `linear_table_t::~linear_table_t (void)`

destructor

4.364.3 Member Function Documentation

4.364.3.1 lookup() `mha_real_t linear_table_t::lookup (mha_real_t x) const [virtual]`

look up the y value that is stored for the mesh point where x is lower than or equal to the x value given here.

This method does not extrapolate, so for $x < \text{xmin}$, the y value for xmin is returned. For all x greater than the x of the last mesh point, the y value of the last mesh point is returned.

Precondition

`prepare` must have been called before `lookup` may be called.

Implements **MHATableLookup::table_t** (p. 1280).

4.364.3.2 interp() `mha_real_t linear_table_t::interp (mha_real_t x) const [virtual]`

interpolate y value for the given x value.

The y values for the neighbouring mesh points are looked up and linearly interpolated. For x values outside the range of mesh points, the y value is extrapolated from the nearest two mesh points.

Precondition

prepare must have been called before interp may be called.

Implements **MHATableLookup::table_t** (p. [1280](#)).

4.364.3.3 set_xmin() `void linear_table_t::set_xmin (mha_real_t xmin)`

set the x value for the first mesh point.

Must be called before prepare can be called.

4.364.3.4 add_entry() `void linear_table_t::add_entry (mha_real_t y)`

set the y value for the next mesh point.

Must be called at least twice before prepare can be called.

4.364.3.5 set_xmax() `void linear_table_t::set_xmax (mha_real_t xmax)`

this sets the x value for a past-the-end, not added mesh point.

Example:

```
t.set_xmin(100);
t.add_entry(0); // mesh point {100,0}
t.add_entry(1); // mesh point {110,1}
// the next mesh point would be at x=120, but we do not add this
t.set_xmax(120); // the x where the next mesh point would be
t.prepare();
```

now, t.interp(100) == 0; t.interp(110) == 1; t.interp(105) == 0.5;

4.364.3.6 `prepare()` `void linear_table_t::prepare (`
`void)`

`prepare` computes the x distance of the mesh points based on the values given to `set_xmin`, `set_xmax`, and the number of times that `add_entry` was called.

Precondition

`set_xmin`, `set_xmax`, `add_entry` functions must have been called before calling `prepare`, `add_entry` must have been called at least twice.

Only after this method has been called, `interp` or `lookup` may be called.

4.364.3.7 `clear()` `void linear_table_t::clear (`
`void) [virtual]`

`clear` resets the state of this object to the state directly after construction.

mesh entries and x range are deleted.

`interp` and `lookup` may not be called after this function has been called unless `prepare` and before that its precondition methods are called again.

Implements `MHATableLookup::table_t` (p. 1280).

4.364.4 Member Data Documentation

4.364.4.1 `vy` `mha_real_t* MHATableLookup::linear_table_t::vy [protected]`

4.364.4.2 `len` `unsigned int MHATableLookup::linear_table_t::len [protected]`

4.364.4.3 `vec_y` `vector< mha_real_t> MHATableLookup::linear_table_t::vec_y [private]`

4.364.4.4 xmin `mha_real_t` MHATableLookup::linear_table_t::xmin [private]

4.364.4.5 xmax `mha_real_t` MHATableLookup::linear_table_t::xmax [private]

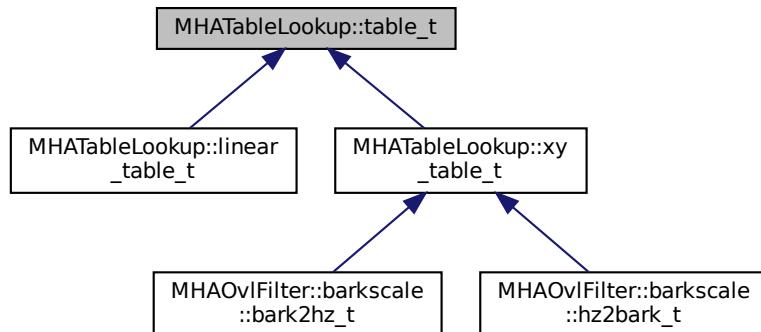
4.364.4.6 scalefac `mha_real_t` MHATableLookup::linear_table_t::scalefac [private]

The documentation for this class was generated from the following files:

- `mha_tablelookup.hh`
- `mha_tablelookup.cpp`

4.365 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

4.365.1 Constructor & Destructor Documentation

4.365.1.1 `table_t()` `table_t::table_t (`
 `void)`

4.365.1.2 `~table_t()` `table_t::~table_t (`
 `void) [virtual]`

4.365.2 Member Function Documentation

4.365.2.1 `lookup()` `virtual mha_real_t MHATableLookup::table_t::lookup (`
 `mha_real_t) const [pure virtual]`

Implemented in **MHATableLookup::xy_table_t** (p. 1282), and **MHATableLookup::linear_table_t** (p. 1276).

4.365.2.2 `interp()` `virtual mha_real_t MHATableLookup::table_t::interp (`
 `mha_real_t) const [pure virtual]`

Implemented in **MHATableLookup::xy_table_t** (p. 1283), and **MHATableLookup::linear_table_t** (p. 1276).

```
4.365.2.3 clear() virtual void MHATableLookup::table_t::clear (
    void ) [protected], [pure virtual]
```

Implemented in [MHATableLookup::linear_table_t](#) (p. 1278), and [MHATableLookup::xy_table_t](#) (p. 1284).

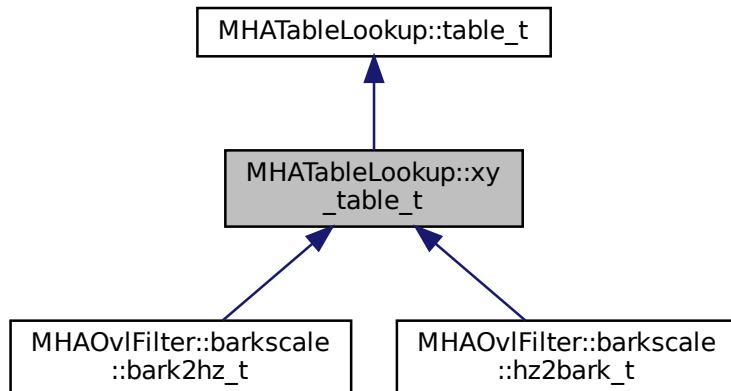
The documentation for this class was generated from the following files:

- [mha_tablelookup.hh](#)
- [mha_tablelookup.cpp](#)

4.366 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

4.366.1 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. 1283) and **xy_table_t::lookup()** (p. 1282). 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.

4.366.2 Constructor & Destructor Documentation

4.366.2.1 **xy_table_t()** `xy_table_t::xy_table_t ()`

4.366.3 Member Function Documentation

4.366.3.1 **lookup()** `mha_real_t xy_table_t::lookup (` `mha_real_t x) const [virtual]`

Return the y-value at the position of the nearest x value below input.

Parameters

x	Input value
---	-------------

Returns

y value at nearest x value below input.

Implements **MHATableLookup::table_t** (p. 1280).

4.366.3.2 interp() `mha_real_t xy_table_t::interp (mha_real_t x) const [virtual]`

Linear interpolation function.

Parameters

x	x value
---	---------

Returns

interpolated y value

Implements **MHATableLookup::table_t** (p. 1280).

4.366.3.3 add_entry() [1/2] `void xy_table_t::add_entry (mha_real_t x, mha_real_t y)`

Add a single x-y pair entry.

Parameters

x	x value
y	corresponding y value

4.366.3.4 add_entry() [2/2] void xy_table_t::add_entry (

```
mha_real_t * pVX,
mha_real_t * pVY,
unsigned int uLength )
```

Add multiple entries at once.

Parameters

<i>pVX</i>	array of x values
<i>pVY</i>	array of y values
<i>uLength</i>	Length of x and y arrays

4.366.3.5 clear() void xy_table_t::clear () [virtual]

Clear the table and transformation functions.

Implements **MHATableLookup::table_t** (p. 1280).

4.366.3.6 set_xfun() void xy_table_t::set_xfun (

```
float(*)(float) fun )
```

Set transformation function for x values.

Parameters

<i>fun</i>	Transformation function.
------------	--------------------------

4.366.3.7 set_yfun() void xy_table_t::set_yfun (

```
float(*)(float) fun )
```

Set transformation function for y values during insertion.

Parameters

<i>fun</i>	Transformation function.
------------	--------------------------

4.366.3.8 set_xyfun() void xy_table_t::set_xyfun (float(*)(float, float) fun)

Set transformation function for y values during insertion, based on x and y values.

Parameters

<i>fun</i>	Transformation function.
------------	--------------------------

4.366.3.9 get_xlimits() std::pair< mha_real_t, mha_real_t> MHATableLookup::xy_table_t::get_xlimits () const [inline]

returns the min and max x of all mesh points that are stored in the lookup table, i.e.

after transformation with xfun, if any. Not real-time safe

4.366.4 Member Data Documentation

4.366.4.1 mXY std::map< mha_real_t, mha_real_t> MHATableLookup::xy_table_t::mXY [private]

4.366.4.2 xfun float(* MHATableLookup::xy_table_t::xfun) (float) [private]

4.366.4.3 yfun float(* MHATableLookup::xy_table_t::yfun) (float) [private]

4.366.4.4 xyfun float(* MHATableLookup::xy_table_t::xyfun) (float, float) [private]

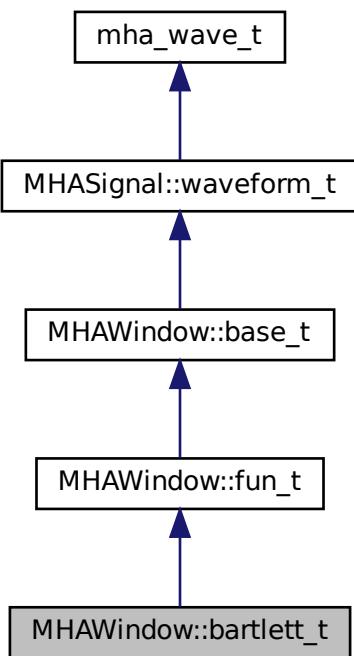
The documentation for this class was generated from the following files:

- **mha_tablelookup.hh**
- **mha_tablelookup.cpp**

4.367 MHAWindow::bartlett_t Class Reference

Bartlett window.

Inheritance diagram for MHAWindow::bartlett_t:



Public Member Functions

- **bartlett_t** (unsigned int n)

Additional Inherited Members

4.367.1 Detailed Description

Bartlett window.

4.367.2 Constructor & Destructor Documentation

4.367.2.1 bartlett_t() `MHAWindow::bartlett_t::bartlett_t (unsigned int n) [inline]`

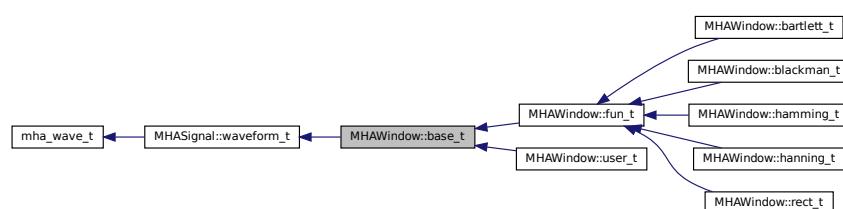
The documentation for this class was generated from the following file:

- `mha_windowparser.h`

4.368 MHAWindow::base_t Class Reference

Common base for window types.

Inheritance diagram for MHAWindow::base_t:



Public Member Functions

- **base_t** (unsigned int len)
Constructor.
- **base_t** (const `MHAWindow::base_t` &src)
Copy constructor.
- void **operator()** (`mha_wave_t` &) const
Apply window to waveform segment (reference)
- void **operator()** (`mha_wave_t` *) const
Apply window to waveform segment (pointer)
- void **ramp_begin** (`mha_wave_t` &) const
Apply a ramp at the begining.
- void **ramp_end** (`mha_wave_t` &) const
Apply a ramp at the end.

Additional Inherited Members

4.368.1 Detailed Description

Common base for window types.

4.368.2 Constructor & Destructor Documentation

4.368.2.1 `base_t()` [1/2] `MHAWindow::base_t::base_t (`
`unsigned int len)`

Constructor.

Parameters

<code>len</code>	Window length in samples.
------------------	---------------------------

4.368.2.2 `base_t()` [2/2] `MHAWindow::base_t::base_t (`
`const MHAWindow::base_t & src)`

Copy constructor.

Parameters

<code>src</code>	Source to be copied
------------------	---------------------

4.368.3 Member Function Documentation

4.368.3.1 `operator()()` [1/2] `void MHAWindow::base_t::operator() (`
`mha_wave_t & s) const`

Apply window to waveform segment (reference)

4.368.3.2 operator()() [2/2] void MHAWindow::base_t::operator() (
 mha_wave_t * s) const

Apply window to waveform segment (pointer)

4.368.3.3 ramp_begin() void MHAWindow::base_t::ramp_begin (
 mha_wave_t & s) const

Apply a ramp at the begining.

4.368.3.4 ramp_end() void MHAWindow::base_t::ramp_end (
 mha_wave_t & s) const

Apply a ramp at the end.

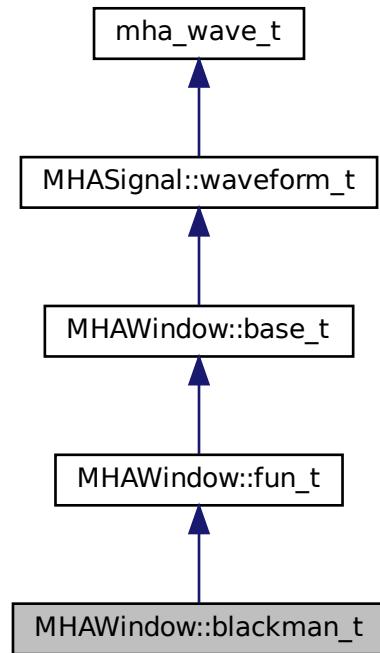
The documentation for this class was generated from the following files:

- mha_windowparser.h
- mha_windowparser.cpp

4.369 MHAWindow::blackman_t Class Reference

Blackman window.

Inheritance diagram for MHAWindow::blackman_t:



Public Member Functions

- **blackman_t** (unsigned int n)

Additional Inherited Members

4.369.1 Detailed Description

Blackman window.

4.369.2 Constructor & Destructor Documentation

4.369.2.1 blackman_t() MHAWindow::blackman_t::blackman_t (unsigned int n) [inline]

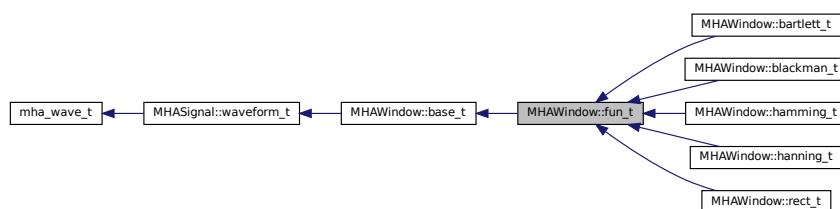
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

4.370 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

4.370.1 Detailed Description

Generic window based on a generator function.

The generator function should return a valid window function in the interval [-1,1[.

4.370.2 Constructor & Destructor Documentation

4.370.2.1 fun_t() MHAWindow::fun_t::fun_t (unsigned int n, float(*)(float) fun, float xmin = -1, float xmax = 1, bool min_included = true, bool max_included = false)

Constructor.

Parameters

<i>n</i>	Window length
<i>fun</i>	Generator function, i.e. MHAWindow::hanning() (p. 155)
<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

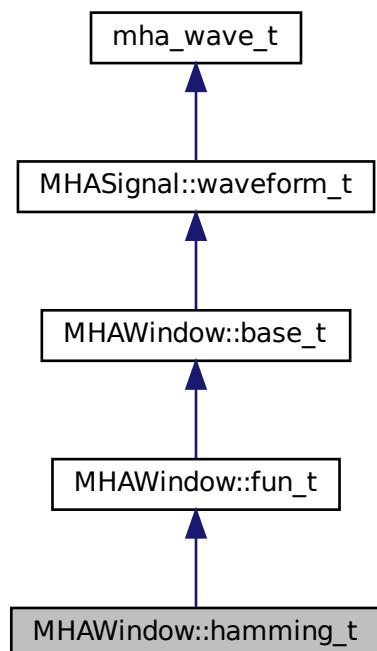
The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

4.371 MHAWindow::hamming_t Class Reference

Hamming window.

Inheritance diagram for MHAWindow::hamming_t:



Public Member Functions

- **hamming_t** (unsigned int n)

Additional Inherited Members

4.371.1 Detailed Description

Hamming window.

4.371.2 Constructor & Destructor Documentation

4.371.2.1 **hamming_t()** MHAWindow::hanning_t::hamming_t (

```
unsigned int n ) [inline]
```

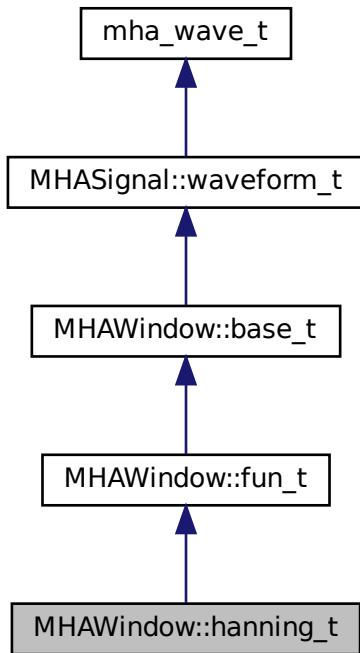
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

4.372 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

4.372.1 Detailed Description

von-Hann window

4.372.2 Constructor & Destructor Documentation

4.372.2.1 hanning_t() `MHAWindow::hanning_t::hanning_t (unsigned int n) [inline]`

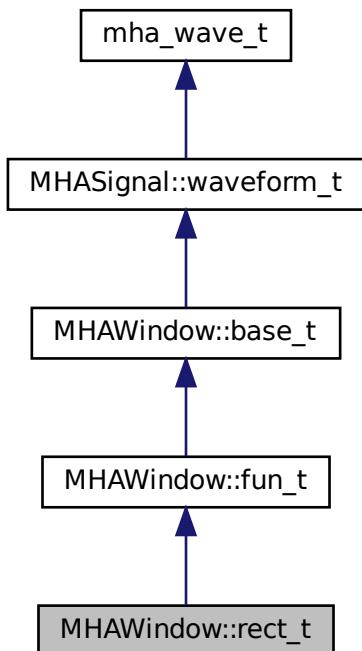
The documentation for this class was generated from the following file:

- `mha_windowparser.h`

4.373 MHAWindow::rect_t Class Reference

Rectangular window.

Inheritance diagram for MHAWindow::rect_t:



Public Member Functions

- `rect_t (unsigned int n)`

Additional Inherited Members

4.373.1 Detailed Description

Rectangular window.

4.373.2 Constructor & Destructor Documentation

4.373.2.1 rect_t() MHAWindow::rect_t::rect_t (unsigned int n) [inline]

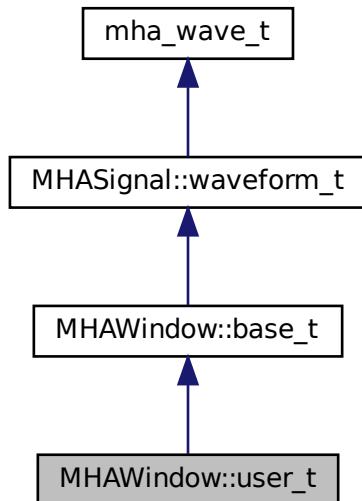
The documentation for this class was generated from the following file:

- **mha_windowparser.h**

4.374 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

4.374.1 Detailed Description

User defined window.

4.374.2 Constructor & Destructor Documentation

4.374.2.1 user_t() MHAWindow::user_t::user_t (

```
const std::vector< mha_real_t > & wnd )
```

Constructor.

Parameters

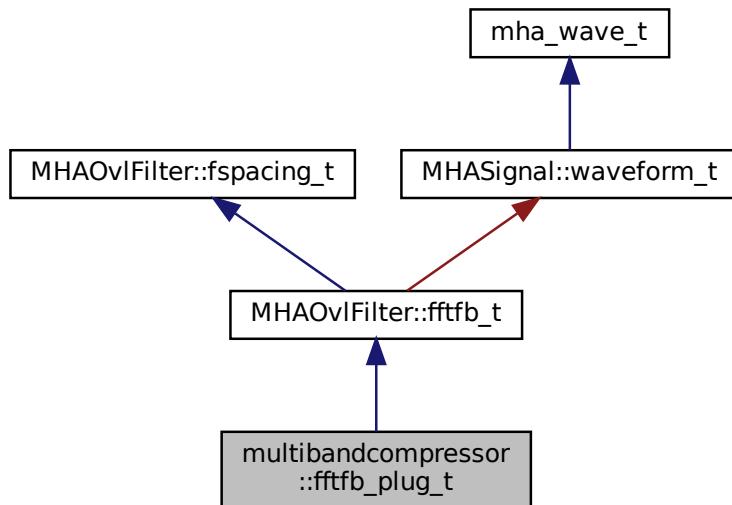
<i>wnd</i>	User defined window
------------	---------------------

The documentation for this class was generated from the following files:

- **mha_windowparser.h**
- **mha_windowparser.cpp**

4.375 multibandcompressor::fftfb_plug_t Class Reference

Inheritance diagram for multibandcompressor::fftfb_plug_t:



Public Member Functions

- **fftfb_plug_t** (**MHAOvIFilter::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

4.375.1 Constructor & Destructor Documentation

```
4.375.1.1 fftfb_plug_t() multibandcompressor::fftfb_plug_t::fftfb_plug_t (
    MHAOvlFilter::fftfb_vars_t & vars,
    const mhaconfig_t & cfg,
    algo_comm_t ac,
    std::string alg )
```

4.375.2 Member Function Documentation

4.375.2.1 insert() void multibandcompressor::fftfb_plug_t::insert ()

4.375.3 Member Data Documentation

4.375.3.1 cfv MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::cfv [private]

vector of nominal center frequencies / Hz

4.375.3.2 efv MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::efv [private]

vector of edge frequencies / Hz

4.375.3.3 bwv MHA_AC::waveform_t multibandcompressor::fftfb_plug_t::bwv [private]

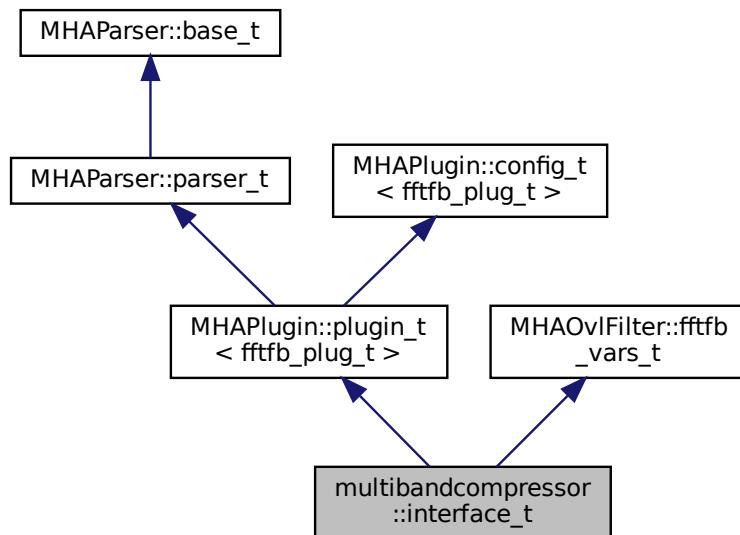
vector of band-weights (sum of squared fft-bin-weights)/num_frames

The documentation for this class was generated from the following file:

- **multibandcompressor.cpp**

4.376 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

- **MHA_AC::int_t num_channels**
- **DynComp::dc_afterburn_t burn**
- **MHAEvents::patchbay_t< interface_t > patchbay**
- std::string **algo**
- **MHParse::mhapluginloader_t plug**
- **plugin_signals_t** * **plug_sigs**

Additional Inherited Members

4.376.1 Constructor & Destructor Documentation

4.376.1.1 interface_t() `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

<code>ac_</code>	algorithm communication handle
<code>th</code>	chain name
<code>al</code>	algorithm name

4.376.2 Member Function Documentation

4.376.2.1 prepare() `void multibandcompressor::interface_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements **MHAPlugin::plugin_t<fftfb_plug_t>** (p. 1148).

4.376.2.2 release() `void multibandcompressor::interface_t::release () [virtual]`

Reimplemented from **MHAPlugin::plugin_t<fftfb_plug_t>** (p. 1149).

4.376.2.3 process() `mha_spec_t * multibandcompressor::interface_t::process (`
 `mha_spec_t * s)`

4.376.2.4 update_cfg() void multibandcompressor::interface_t::update_cfg () [private]

4.376.3 Member Data Documentation

4.376.3.1 num_channels MHA_AC::int_t multibandcompressor::interface_t::num_channels [private]

4.376.3.2 burn DynComp::dc_afterburn_t multibandcompressor::interface_t::burn [private]

4.376.3.3 patchbay MHAEVENTS::patchbay_t< interface_t> multibandcompressor::interface_t::patchbay [private]

4.376.3.4 algo std::string multibandcompressor::interface_t::algo [private]

4.376.3.5 plug MHAPARSER::mhapluginloader_t multibandcompressor::interface_t::plug [private]

4.376.3.6 plug_sigs plugin_signals_t* multibandcompressor::interface_t::plug_sigs [private]

The documentation for this class was generated from the following file:

- **multibandcompressor.cpp**

4.377 multibandcompressor::plugin_signals_t Class Reference

Public Member Functions

- **plugin_signals_t** (unsigned int **channels**, unsigned int bands)
- void **update_levels** (**MHAOvlFilter::fftfb_t** *, **mha_spec_t** ***s_in**)
- void **apply_gains** (**MHAOvlFilter::fftfb_t** *, **DynComp::dc_afterburn_t** &**burn**, **mha_spec_t** ***s_out**)

Public Attributes

- **mha_wave_t** * **plug_output**

Private Attributes

- **MHASignal::waveform_t** **plug_level**
- **MHASignal::waveform_t** **gain**

4.377.1 Constructor & Destructor Documentation

```
4.377.1.1 plugin_signals_t() multibandcompressor::plugin_signals_t::plugin_signals_t ( unsigned int channels, unsigned int bands )
```

4.377.2 Member Function Documentation

```
4.377.2.1 update_levels() void multibandcompressor::plugin_signals_t::update_levels ( MHAOvlFilter::fftfb_t * pFb, mha_spec_t * s_in )
```

```
4.377.2.2 apply_gains() void multibandcompressor::plugin_signals_t::apply_gains (  
    MHAOvlFilter::fftfb_t * pFb,  
    DynComp::dc_afterburn_t & burn,  
    mha_spec_t * s_out )
```

4.377.3 Member Data Documentation

4.377.3.1 plug_level MHASignal::waveform_t multibandcompressor::plugin_signals_t::plug_level [private]

4.377.3.2 gain MHASignal::waveform_t multibandcompressor::plugin_signals_t::gain [private]

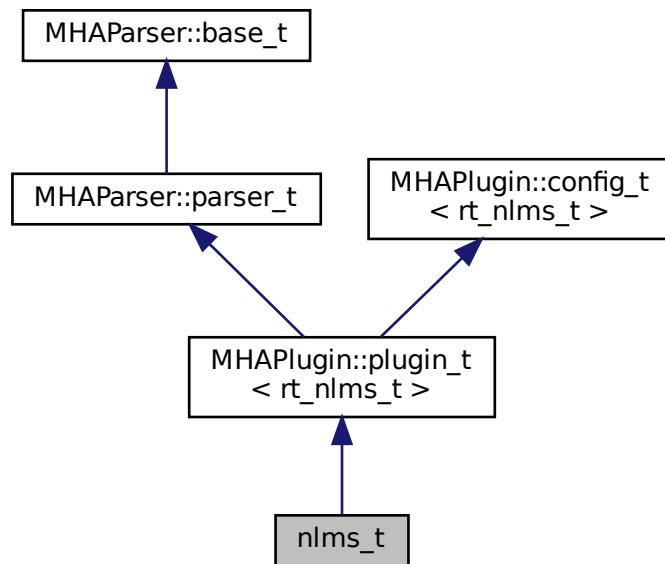
4.377.3.3 plug_output mha_wave_t* multibandcompressor::plugin_signals_t::plug_output

The documentation for this class was generated from the following file:

- **multibandcompressor.cpp**

4.378 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

- `MHPParser::float_t rho`
- `MHPParser::float_t c`
- `MHPParser::int_t ntaps`
- `MHPParser::string_t name_u`
- `MHPParser::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

4.378.1 Constructor & Destructor Documentation

4.378.1.1 nlms_t() nlms_t::nlms_t (

```
    algo_comm_t ac,
    const char * ,
    const char * ialg )
```

4.378.2 Member Function Documentation

4.378.2.1 prepare() void nlms_t::prepare (

```
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAParser::MHAPlugin::plugin_t< rt_nlms_t >** (p. 1148).

4.378.2.2 release() void nlms_t::release () [virtual]

Reimplemented from **MHAParser::MHAPlugin::plugin_t< rt_nlms_t >** (p. 1149).

4.378.2.3 process() `mha_wave_t * nlms_t::process (`
`mha_wave_t * s)`

4.378.2.4 update() `void nlms_t::update () [private]`

4.378.3 Member Data Documentation

4.378.3.1 rho `MHAParser::float_t nlms_t::rho [private]`

4.378.3.2 c `MHAParser::float_t nlms_t::c [private]`

4.378.3.3 ntaps `MHAParser::int_t nlms_t::ntaps [private]`

4.378.3.4 name_u `MHAParser::string_t nlms_t::name_u [private]`

4.378.3.5 name_d `MHAParser::string_t nlms_t::name_d [private]`

4.378.3.6 normtype `MHAParser::kw_t nlms_t::normtype [private]`

4.378.3.7 estimtype `MHAParser::kw_t nlms_t::estimtype [private]`

4.378.3.8 lambda_smoothing_power `MHAParser::float_t nlms_t::lambda_smoothing←
_power [private]`

4.378.3.9 name_e `MHAParser::string_t nlms_t::name_e [private]`

4.378.3.10 name_f `MHAParser::string_t nlms_t::name_f [private]`

4.378.3.11 n_no_update `MHAParser::int_t nlms_t::n_no_update [private]`

4.378.3.12 algo `std::string nlms_t::algo [private]`

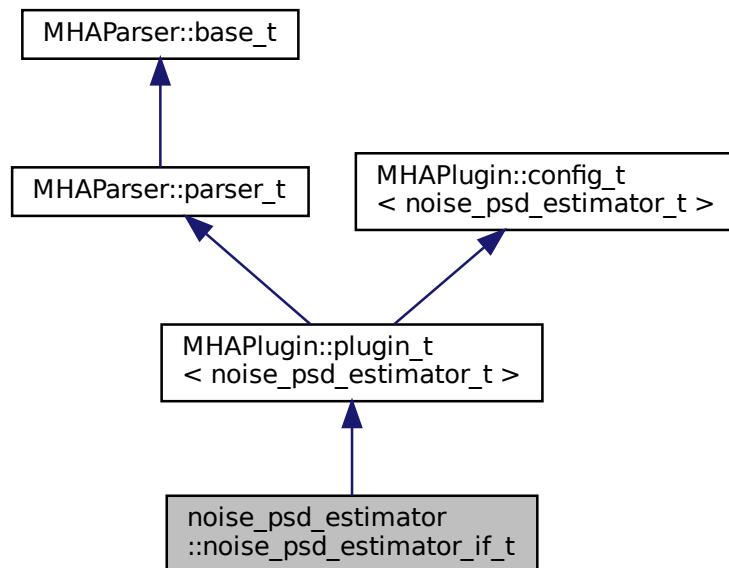
4.378.3.13 patchbay `MHAEVENTS::patchbay_t< nlms_t> nlms_t::patchbay [private]`

The documentation for this class was generated from the following file:

- `nlms_wave.cpp`

4.379 noise_psd_estimator::noise_psd_estimator_if_t Class Reference

Inheritance diagram for noise_psd_estimator::noise_psd_estimator_if_t:



Public Member Functions

- `noise_psd_estimator_if_t (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< noise_psd_estimator_if_t > patchbay`

Additional Inherited Members

4.379.1 Constructor & Destructor Documentation

```
4.379.1.1 noise_psd_estimator_if_t() noise_psd_estimator::noise_psd_estimator_if_t::noise_psd_estimator_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & iname )
```

4.379.2 Member Function Documentation

```
4.379.2.1 process() mha_spec_t * noise_psd_estimator::noise_psd_estimator_if_t::process (
    mha_spec_t * s )
```

```
4.379.2.2 prepare() void noise_psd_estimator::noise_psd_estimator_if_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< noise_psd_estimator_t >** (p. [1148](#)).

```
4.379.2.3 update_cfg() void noise_psd_estimator::noise_psd_estimator_if_t::update_cfg (
    ) [private]
```

4.379.3 Member Data Documentation

4.379.3.1 alphaPH1mean `MHAParser::float_t noise_psd_estimator::noise_psd_<estimator_if_t::alphaPH1mean [private]`

4.379.3.2 alphaPSD `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_<if_t::alphaPSD [private]`

4.379.3.3 q `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_if_t::q [private]`

4.379.3.4 xiOptDb `MHAParser::float_t noise_psd_estimator::noise_psd_estimator_<if_t::xiOptDb [private]`

4.379.3.5 name `std::string noise_psd_estimator::noise_psd_estimator_if_t::name [private]`

4.379.3.6 patchbay `MHAEvents::patchbay_t< noise_psd_estimator_if_t> noise_psd_<estimator::noise_psd_estimator_if_t::patchbay [private]`

The documentation for this class was generated from the following file:

- `noise_psd_estimator.cpp`

4.380 noise_psd_estimator::noise_psd_estimator_t Class Reference

Public Member Functions

- `noise_psd_estimator_t (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`

4.380.1 Constructor & Destructor Documentation

```
4.380.1.1 noise_psd_estimator_t() noise_psd_estimator::noise_psd_estimator_t<-
::noise_psd_estimator_t (
    const mhaconfig_t & cf,
    algo_comm_t ac,
    const std::string & name,
    float alphaPH1mean,
    float alphaPSD,
    float q,
    float xiOptDb )
```

4.380.2 Member Function Documentation

```
4.380.2.1 process() void noise_psd_estimator::noise_psd_estimator_t::process (
    mha_spec_t * noisyDftFrame )
```

4.380.2.2 insert() void noise_psd_estimator::noise_psd_estimator_t::insert () [inline]

4.380.3 Member Data Documentation

4.380.3.1 noisyPer `MHASignal::waveform_t` noise_psd_estimator::noise_psd_estimator_t::noisyPer [private]

4.380.3.2 PH1mean `MHASignal::waveform_t` noise_psd_estimator::noise_psd_estimator_t::PH1mean [private]

4.380.3.3 noisePow `MHA_AC::waveform_t` noise_psd_estimator::noise_psd_estimator_t::noisePow [private]

4.380.3.4 inputPow `MHA_AC::waveform_t` noise_psd_estimator::noise_psd_estimator_t::inputPow [private]

4.380.3.5 snrPost1Debug `MHA_AC::waveform_t` noise_psd_estimator::noise_psd_estimator_t::snrPost1Debug [private]

4.380.3.6 GLRDebug `MHA_AC::waveform_t` noise_psd_estimator::noise_psd_estimator_t::GLRDebug [private]

4.380.3.7 PH1Debug `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::PH1Debug` [private]

4.380.3.8 estimateDebug `MHA_AC::waveform_t` `noise_psd_estimator::noise_psd_estimator_t::estimateDebug` [private]

4.380.3.9 inputSpec `MHA_AC::spectrum_t` `noise_psd_estimator::noise_psd_estimator_t::inputSpec` [private]

4.380.3.10 alphaPH1mean_ `float` `noise_psd_estimator::noise_psd_estimator_t::alphaPH1mean_` [private]

4.380.3.11 alphaPSD_ `float` `noise_psd_estimator::noise_psd_estimator_t::alphaPSD_` [private]

4.380.3.12 priorFact `float` `noise_psd_estimator::noise_psd_estimator_t::priorFact` [private]

4.380.3.13 xiOpt `float` `noise_psd_estimator::noise_psd_estimator_t::xiOpt` [private]

4.380.3.14 logGLRFact `float` `noise_psd_estimator::noise_psd_estimator_t::logGLRFact` [private]

4.380.3.15 GLRexp float noise_psd_estimator::noise_psd_estimator_t::GLRexp [private]

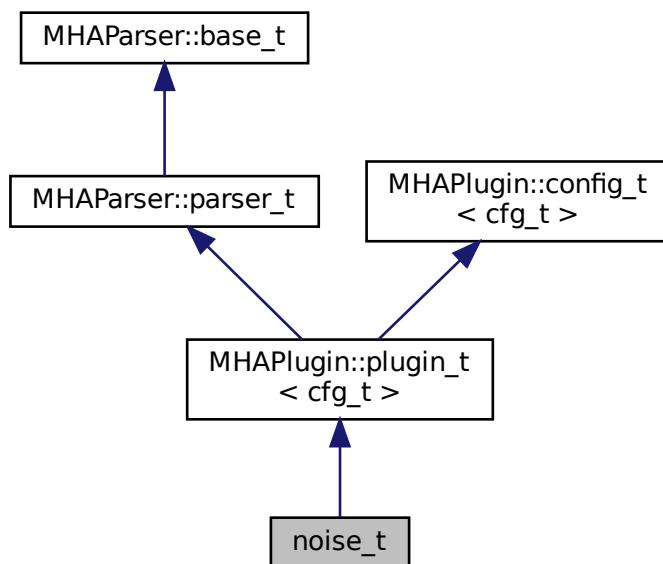
4.380.3.16 frameno int noise_psd_estimator::noise_psd_estimator_t::frameno [private]

The documentation for this class was generated from the following file:

- `noise_psd_estimator.cpp`

4.381 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`
- `MHAParser::int_t seed`
- `MHAEvents::patchbay_t< noise_t > patchbay`

Additional Inherited Members

4.381.1 Constructor & Destructor Documentation

4.381.1.1 `noise_t()` `noise_t::noise_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string &)`

4.381.2 Member Function Documentation

4.381.2.1 `process() [1/2]` `mha_wave_t * noise_t::process (`
 `mha_wave_t * s)`

4.381.2.2 `process() [2/2]` `mha_spec_t * noise_t::process (`
 `mha_spec_t * s)`

4.381.2.3 `prepare()` `void noise_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements `MHAParser::MHAPlugin< plugin_t >` (p. 1148).

4.381.2.4 update_cfg() void noise_t::update_cfg ()

4.381.3 Member Data Documentation

4.381.3.1 lev `MHAParser::float_t` noise_t::lev [private]

4.381.3.2 mode `MHAParser::kw_t` noise_t::mode [private]

4.381.3.3 frozennoise_length `MHAParser::float_t` noise_t::frozennoise_length [private]

4.381.3.4 seed `MHAParser::int_t` noise_t::seed [private]

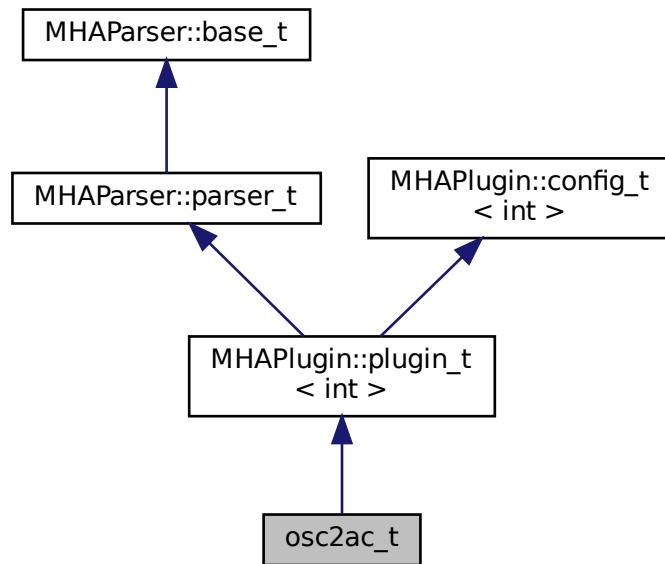
4.381.3.5 patchbay `MHAEVENTS::patchbay_t< noise_t>` noise_t::patchbay [private]

The documentation for this class was generated from the following file:

- `noise.cpp`

4.382 osc2ac_t Class Reference

Inheritance diagram for osc2ac_t:



Public Member Functions

- `osc2ac_t (algo_comm_t iac, const char *chain, const char *algo)`
- `void prepare (mhaconfig_t &)`
- `void release ()`
- `mha_wave_t * process (mha_wave_t *s)`
- `mha_spec_t * process (mha_spec_t *s)`
- `void process ()`

Private Member Functions

- `void setlock (bool b)`

Private Attributes

- `MHAParser::string_t host`
- `MHAParser::string_t port`
- `MHAParser::vstring_t vars`
- `MHAParser::vint_t size`
- `MHAEvents::patchbay_t< osc2ac_t > patchbay`
- `std::unique_ptr< osc_server_t > srv`

Additional Inherited Members

4.382.1 Constructor & Destructor Documentation

4.382.1.1 osc2ac_t() `osc2ac_t::osc2ac_t (`
 `algo_comm_t iac,`
 `const char * chain,`
 `const char * algo)`

4.382.2 Member Function Documentation

4.382.2.1 prepare() `void osc2ac_t::prepare (`
 `mhaconfig_t & cf) [virtual]`

Implements `MHAPlugin::plugin_t< int >` (p. [1148](#)).

4.382.2.2 release() `void osc2ac_t::release () [virtual]`

Reimplemented from `MHAPlugin::plugin_t< int >` (p. [1149](#)).

4.382.2.3 process() [1/3] `mha_wave_t* osc2ac_t::process (`
 `mha_wave_t * s) [inline]`

4.382.2.4 process() [2/3] `mha_spec_t* osc2ac_t::process (`
 `mha_spec_t * s) [inline]`

4.382.2.5 process() [3/3] void osc2ac_t::process ()

4.382.2.6 setlock() void osc2ac_t::setlock (bool b) [private]

4.382.3 Member Data Documentation

4.382.3.1 host MHAParser::string_t osc2ac_t::host [private]

4.382.3.2 port MHAParser::string_t osc2ac_t::port [private]

4.382.3.3 vars MHAParser::vstring_t osc2ac_t::vars [private]

4.382.3.4 size MHAParser::vint_t osc2ac_t::size [private]

4.382.3.5 patchbay MHAEEvents::patchbay_t< osc2ac_t> osc2ac_t::patchbay [private]

4.382.3.6 srv std::unique_ptr< osc_server_t> osc2ac_t::srv [private]

The documentation for this class was generated from the following file:

- osc2ac.cpp

4.383 osc_server_t Class Reference

OSC receiver implemented using liblo.

Public Member Functions

- `osc_server_t (const std::string &multicast_addr, const std::string &port)`
- `~osc_server_t ()`
- `void server_stop ()`
- `void server_start ()`
- `void insert_variable (const std::string &name, unsigned int size, algo_comm_t hAC)`
- `void sync_osc2ac ()`
- `void ac_insert ()`

Static Public Member Functions

- `static void error_h (int num, const char *msg, const char *path)`

Private Attributes

- `std::vector< std::unique_ptr< osc_variable_t > > pVars`
- `lo_server_thread lost`
- `bool is_running`

4.383.1 Detailed Description

OSC receiver implemented using liblo.

4.383.2 Constructor & Destructor Documentation

4.383.2.1 osc_server_t() `osc_server_t::osc_server_t (`
`const std::string & multicast_addr,`
`const std::string & port)`

4.383.2.2 ~osc_server_t() `osc_server_t::~osc_server_t ()`

4.383.3 Member Function Documentation

4.383.3.1 server_stop() `void osc_server_t::server_stop ()`

4.383.3.2 server_start() `void osc_server_t::server_start ()`

4.383.3.3 insert_variable() `void osc_server_t::insert_variable (`
 `const std::string & name,`
 `unsigned int size,`
 `algo_comm_t hAC)`

4.383.3.4 sync_osc2ac() `void osc_server_t::sync_osc2ac ()`

4.383.3.5 ac_insert() `void osc_server_t::ac_insert ()`

4.383.3.6 error_h() `void osc_server_t::error_h (`
 `int num,`
 `const char * msg,`
 `const char * path) [static]`

4.383.4 Member Data Documentation

4.383.4.1 pVars std::vector<std::unique_ptr< osc_variable_t > > osc_server_t::pVars [private]

4.383.4.2 lost lo_server_thread osc_server_t::lost [private]

4.383.4.3 is_running bool osc_server_t::is_running [private]

The documentation for this class was generated from the following file:

- **osc2ac.cpp**

4.384 osc_variable_t Class Reference

Class for converting messages received at a single osc address to a single AC variable.

Public Member Functions

- **osc_variable_t** (const osc_variable_t &) = delete
An instance of this class cannot safely be copied.
- **osc_variable_t** (const std::string &name, unsigned int size, algo_comm_t hAC, lo_server_thread lost)
Constructor.
- **void sync_osc2ac ()**
Copies the latest OSC data from the OSC storage to the AC storage.
- **void ac_insert ()**
Insert/Re-insert the AC variable into AC space.
- **int handler** (const char *types, lo_arg **argv, int argc)
Callback function called by network thread managed by liblo when a new OSC message has been received.

Static Public Member Functions

- **static int handler** (const char *path, const char *types, lo_arg **argv, int argc, lo_message msg, void *user_data)
Callback function called by network thread managed by liblo when a new OSC message has been received.

Private Attributes

- std::string **acname**
Name of the ac variable.
- std::string **oscaddr**
OSC address.
- **MHA_AC::waveform_t ac_data**
AC variable storage.
- **MHASignal::waveform_t osc_data**
OSC variable storage.
- std::string **name_**
Name of AC variable and OSC address without the initial slash.

4.384.1 Detailed Description

Class for converting messages received at a single osc address to a single AC variable.

OSC variables are received asynchronously in a network thread and must not modify their AC variables directly, because MHA plugins may only access their AC variables while executing their prepare, release, or process callbacks.

One osc2ac plugin uses multiple instances of **osc_variable_t** (p. 1323), one for each mapping of an OSC address to an AC variable.

4.384.2 Constructor & Destructor Documentation

4.384.2.1 **osc_variable_t()** [1/2] `osc_variable_t::osc_variable_t (` `const osc_variable_t &) [delete]`

An instance of this class cannot safely be copied.

4.384.2.2 **osc_variable_t()** [2/2] `osc_variable_t::osc_variable_t (` `const std::string & name,` `unsigned int size,` `algo_comm_t hac,` `lo_server_thread lost)`

Constructor.

Allocates memory.

Parameters

<i>name</i>	The name of the AC variable that stores the latest value.
<i>size</i>	Number of elements to copy from OSC message to AC variable.
<i>hAC</i>	Handle of Algorithm Communication Variable space.
<i>lost</i>	libLO Server Thread.

4.384.3 Member Function Documentation

4.384.3.1 sync_osc2ac() void osc_variable_t::sync_osc2ac () [inline]

Copies the latest OSC data from the OSC storage to the AC storage.

To be executed during process callback of osc2ac plugin.

4.384.3.2 ac_insert() void osc_variable_t::ac_insert () [inline]

Insert/Re-insert the AC variable into AC space.

Should be done in each process callback.

4.384.3.3 handler() [1/2] int osc_variable_t::handler (

```
const char * path,
const char * types,
lo_arg ** argv,
int argc,
lo_message msg,
void * user_data ) [static]
```

Callback function called by network thread managed by liblo when a new OSC message has been received.

This static method forwards to the instance method by casting user_data to osc_variable_t*.

Parameters

<i>path</i>	Unused.
<i>types</i>	The OSC data type indicator of the received message.
<i>argv</i>	Array of received OSC data.
<i>argc</i>	Number of elements in array of received OSC data.
<i>user_data</i>	Pointer to osc_variable_t (p. 1323) instance.

Returns

1 if the message was accepted, 0 if not.

```
4.384.3.4 handler() [2/2] int osc_variable_t::handler (
    const char * types,
    lo_arg ** argv,
    int argc )
```

Callback function called by network thread managed by liblo when a new OSC message has been received.

This instance method checks if the received data is of expected length and contains only floats, and if yes, copies the data into the buffer osc_data where the latest received data for this osc address is stored until it is either overwritten by the next data for the same osc address or copied to an AC variable.

Parameters

<i>types</i>	The OSC data type indicator of the received message.
<i>argv</i>	Array of received OSC data.
<i>argc</i>	Number of elements in array of received OSC data.

Returns

1 if the message had correct length and contained only floats, 0 if not.

4.384.4 Member Data Documentation

4.384.4.1 acname std::string osc_variable_t::acname [private]

Name of the ac variable.

4.384.4.2 oscaddr std::string osc_variable_t::oscaddr [private]

OSC address.

4.384.4.3 ac_data `MHA_AC::waveform_t` `osc_variable_t::ac_data` [private]

AC variable storage.

4.384.4.4 osc_data `MHASignal::waveform_t` `osc_variable_t::osc_data` [private]

OSC variable storage.

4.384.4.5 name_ `std::string` `osc_variable_t::name_` [private]

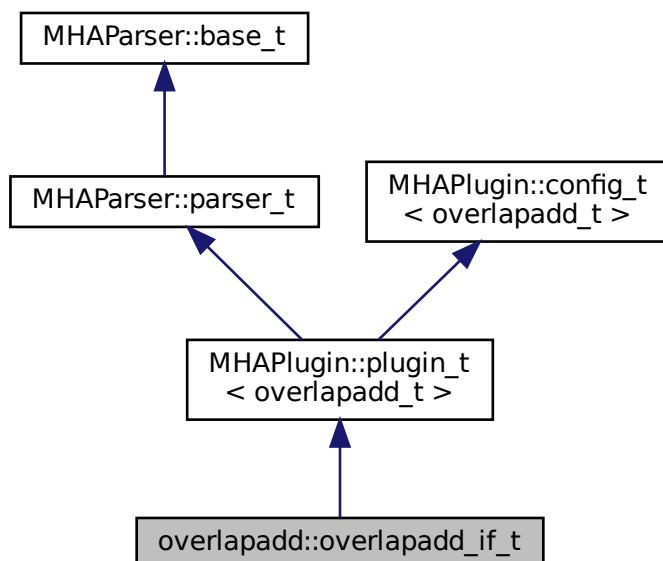
Name of AC variable and OSC address without the initial slash.

The documentation for this class was generated from the following file:

- `osc2ac.cpp`

4.385 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 ()**=default
- void **prepare** (**mhaconfig_t** &)
- void **release** ()
- **mha_wave_t * process** (**mha_wave_t** *)

Private Member Functions

- void **update** ()
 - void **setlock** (bool b)
- Lock/Unlock all configuration variables.*

Private Attributes

- **MHAParser::int_t nfft**
FFT length to be used, zero-padding is FFT length-wndlength.
- **MHAParser::int_t nwnd**
Window length to be used (overlap is 1-fragsize/wndlength)
- **MHAParser::float_t wndpos**
Relative position of zero padding (0 end, 0.5 center, 1 start)
- **MHAParser::window_t window**
- **MHAParser::float_t wndexp**
- **MHAParser::window_t zerowindow**
- **MHAParser::bool_t strict_window_ratio**
Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.
- **MHAParser::mhapluginloader_t plugloader**
- **MHAParser::float_mon_t prescale**
- **MHAParser::float_mon_t postscale**
- std::string **algo**
- **mhaconfig_t cf_in**
- **mhaconfig_t cf_out**

Additional Inherited Members

4.385.1 Constructor & Destructor Documentation

```
4.385.1.1 overlapadd_if_t() overlapadd::overlapadd_if_t::overlapadd_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & ialg )
```

```
4.385.1.2 ~overlapadd_if_t() overlapadd::overlapadd_if_t::~overlapadd_if_t ( )
[default]
```

4.385.2 Member Function Documentation

```
4.385.2.1 prepare() void overlapadd::overlapadd_if_t::prepare (
    mhaconfig_t & t ) [virtual]
```

Implements **MHAPlugin::plugin_t< overlapadd_t >** (p. 1148).

```
4.385.2.2 release() void overlapadd::overlapadd_if_t::release ( ) [virtual]
```

Reimplemented from **MHAPlugin::plugin_t< overlapadd_t >** (p. 1149).

```
4.385.2.3 process() mha_wave_t * overlapadd::overlapadd_if_t::process (
    mha_wave_t * wave_in )
```

```
4.385.2.4 update() void overlapadd::overlapadd_if_t::update ( ) [private]
```

```
4.385.2.5 setlock() void overlapadd::overlapadd_if_t::setlock (
    bool b ) [inline], [private]
```

Lock/Unlock all configuration variables.

Parameters

<i>b</i>	Desired lock state
----------	--------------------

4.385.3 Member Data Documentation**4.385.3.1 nfft `MHAParser::int_t overlapadd::overlapadd_if_t::nfft` [private]**

FFT length to be used, zero-padding is FFT length-wndlength.

4.385.3.2 nwnd `MHAParser::int_t overlapadd::overlapadd_if_t::nwnd` [private]

Window length to be used (overlap is 1-fragsize/wndlength)

4.385.3.3 wndpos `MHAParser::float_t overlapadd::overlapadd_if_t::wndpos` [private]

Relative position of zero padding (0 end, 0.5 center, 1 start)

4.385.3.4 window `MHAParser::window_t overlapadd::overlapadd_if_t::window` [private]**4.385.3.5 wndexp `MHAParser::float_t overlapadd::overlapadd_if_t::wndexp` [private]****4.385.3.6 zerowindow `MHAParser::window_t overlapadd::overlapadd_if_t::zerowindow` [private]**

4.385.3.7 strict_window_ratio `MHAParser::bool_t overlapadd::overlapadd_if_t::strict_window_ratio [private]`

Disallow window sizes that are not a multiple of the hop size ("fragsize" in MHA) a by power of two.

4.385.3.8 plugloader `MHAParser::mhaplugloader_t overlapadd::overlapadd_if_t::plugloader [private]`

4.385.3.9 prescale `MHAParser::float_mon_t overlapadd::overlapadd_if_t::prescale [private]`

4.385.3.10 postscale `MHAParser::float_mon_t overlapadd::overlapadd_if_t::postscale [private]`

4.385.3.11 algo `std::string overlapadd::overlapadd_if_t::algo [private]`

4.385.3.12 cf_in `mhaconfig_t overlapadd::overlapadd_if_t::cf_in [private]`

4.385.3.13 cf_out `mhaconfig_t overlapadd::overlapadd_if_t::cf_out [private]`

The documentation for this class was generated from the following files:

- `overlapadd.hh`
- `overlapadd.cpp`

4.386 overlapadd::overlapadd_t Class Reference

Public Member Functions

- `overlapadd_t (mhaconfig_t spar_in, mhaconfig_t spar_out, float wexp, float wndpos, const MHAParser::window_t &window, const MHAParser::window_t &zerowindow, float &prescale_fac, float &postscale_fac)`
- `~overlapadd_t ()`
- `mha_spec_t * wave2spec (mha_wave_t *)`
- `mha_wave_t * spec2wave (mha_spec_t *)`

Private Member Functions

- `void wave2spec_hop_forward (mha_wave_t *)`
- `void wave2spec_apply_window (void)`
- `mha_spec_t * wave2spec_compute_fft (void)`

Private Attributes

- `mha_fft_t fft`
- `MHAWindow::base_t prewnd`
- `MHAWindow::base_t postwnd`
- `MHASignal::waveform_t wave_in1`
- `MHASignal::waveform_t wave_out1`
- `MHASignal::spectrum_t spec_in`
- `MHASignal::waveform_t calc_out`
- `MHASignal::waveform_t out_buf`
- `MHASignal::waveform_t write_buf`
- `unsigned int n_zero`
- `unsigned int n_pad1`
- `unsigned int n_pad2`

4.386.1 Constructor & Destructor Documentation

4.386.1.1 overlapadd_t() overlapadd::overlapadd_t::overlapadd_t (

```
    mhaconfig_t spar_in,
    mhaconfig_t spar_out,
    float wexp,
    float wndpos,
    const MHAParser::window_t & window,
    const MHAParser::window_t & zerowindow,
    float & prescale_fac,
    float & postscale_fac )
```

4.386.1.2 ~overlapadd_t() overlapadd::overlapadd_t::~overlapadd_t ()

4.386.2 Member Function Documentation

4.386.2.1 wave2spec() mha_spec_t * overlapadd::overlapadd_t::wave2spec (mha_wave_t * s)

4.386.2.2 spec2wave() mha_wave_t * overlapadd::overlapadd_t::spec2wave (mha_spec_t * s)

4.386.2.3 wave2spec_hop_forward() void overlapadd::overlapadd_t::wave2spec_hop_forward (mha_wave_t * s) [private]

4.386.2.4 wave2spec_apply_window() void overlapadd::overlapadd_t::wave2spec_apply_window (void) [private]

4.386.2.5 wave2spec_compute_fft() mha_spec_t * overlapadd::overlapadd_t::wave2spec_compute_fft (void) [private]

4.386.3 Member Data Documentation

4.386.3.1 fft mha_fft_t overlapadd::overlapadd_t::fft [private]

4.386.3.2 prewnd `MHAWindow::base_t overlapadd::overlapadd_t::prewnd` [private]

4.386.3.3 postwnd `MHAWindow::base_t overlapadd::overlapadd_t::postwnd` [private]

4.386.3.4 wave_in1 `MHASignal::waveform_t overlapadd::overlapadd_t::wave_in1` [private]

4.386.3.5 wave_out1 `MHASignal::waveform_t overlapadd::overlapadd_t::wave_out1` [private]

4.386.3.6 spec_in `MHASignal::spectrum_t overlapadd::overlapadd_t::spec_in` [private]

4.386.3.7 calc_out `MHASignal::waveform_t overlapadd::overlapadd_t::calc_out` [private]

4.386.3.8 out_buf `MHASignal::waveform_t overlapadd::overlapadd_t::out_buf` [private]

4.386.3.9 write_buf `MHASignal::waveform_t overlapadd::overlapadd_t::write_buf` [private]

4.386.3.10 n_zero `unsigned int overlapadd::overlapadd_t::n_zero` [private]

4.386.3.11 n_pad1 unsigned int overlapadd::overlapadd_t::n_pad1 [private]

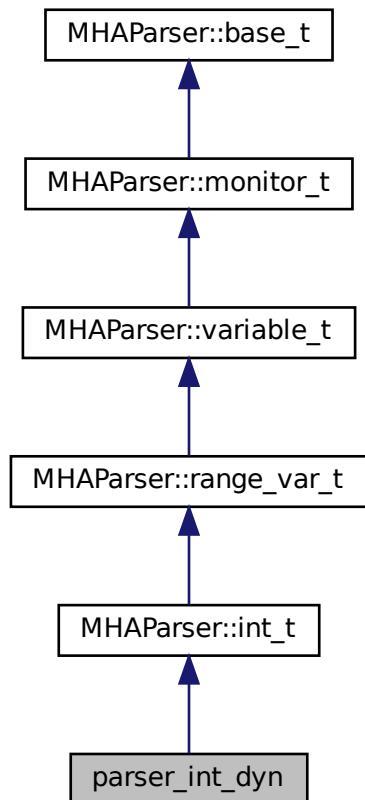
4.386.3.12 n_pad2 unsigned int overlapadd::overlapadd_t::n_pad2 [private]

The documentation for this class was generated from the following files:

- **overlapadd.hh**
- **overlapadd.cpp**

4.387 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

4.387.1 Constructor & Destructor Documentation

```
4.387.1.1 parser_int_dyn() parser_int_dyn::parser_int_dyn (
    const std::string & help_text,
    const std::string & initial_value,
    const std::string & range ) [inline]
```

4.387.2 Member Function Documentation

```
4.387.2.1 set_max_angle_ind() void parser_int_dyn::set_max_angle_ind (
    unsigned int max_ind ) [inline]
```

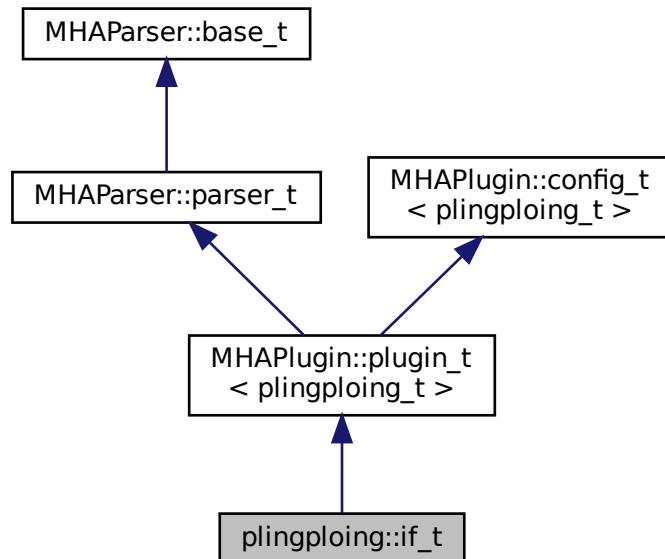
The documentation for this class was generated from the following file:

- **steerbf.h**

4.388 `plingploing::if_t` Class Reference

Plugin class of the plingploing music generator.

Inheritance diagram for `plingploing::if_t`:



Public Member Functions

- `if_t (algo_comm_t, const char *, const char *)`
- `mha_wave_t * process (mha_wave_t *)`
- `void prepare (mhaconfig_t &cf)`

Private Member Functions

- `void update ()`

Private Attributes

- **MHAEvents::patchbay_t < if_t > patchbay**
- **MHAParser::float_t level**
Output level in dB SPL.
- **MHAParser::float_t pitch**
Bass pitch in Hz.
- **MHAParser::float_t fun1_key**
Key1.
- **MHAParser::float_t fun1_range**
Range1.
- **MHAParser::float_t fun2_key**
Key 2.
- **MHAParser::float_t fun2_range**
Range 2.
- **MHAParser::float_t bpm**
Speed in beats per minute (bpm)
- **MHAParser::float_t minlen**
Minimum note length in beats.
- **MHAParser::float_t maxlen**
Maximum note length in beats.
- **MHAParser::float_t bassmod**
Bass key modulation depth.
- **MHAParser::float_t bassperiod**
Bass key modulation period.

Additional Inherited Members

4.388.1 Detailed Description

Plugin class of the plingploing music generator.

4.388.2 Constructor & Destructor Documentation

4.388.2.1 if_t() plingploing::if_t::if_t (

```
algo_comm_t iac,
const char * ,
const char * )
```

4.388.3 Member Function Documentation

4.388.3.1 process() `mha_wave_t * plingploing::if_t::process (mha_wave_t * s)`

4.388.3.2 prepare() `void plingploing::if_t::prepare (mhaconfig_t & cf) [virtual]`

Implements **MHAPlugin::plugin_t< plingploing_t >** (p. 1148).

4.388.3.3 update() `void plingploing::if_t::update () [private]`

4.388.4 Member Data Documentation

4.388.4.1 patchbay `MHAEvents::patchbay_t< if_t> plingploing::if_t::patchbay [private]`

4.388.4.2 level `MHAParser::float_t plingploing::if_t::level [private]`

Output level in dB SPL.

4.388.4.3 pitch `MHAParser::float_t plingploing::if_t::pitch [private]`

Bass pitch in Hz.

4.388.4.4 fun1_key `MHAParser::float_t` `plingploing::if_t::fun1_key` [private]

Key1.

4.388.4.5 fun1_range `MHAParser::float_t` `plingploing::if_t::fun1_range` [private]

Range1.

4.388.4.6 fun2_key `MHAParser::float_t` `plingploing::if_t::fun2_key` [private]

Key 2.

4.388.4.7 fun2_range `MHAParser::float_t` `plingploing::if_t::fun2_range` [private]

Range 2.

4.388.4.8 bpm `MHAParser::float_t` `plingploing::if_t::bpm` [private]

Speed in beats per minute (bpm)

4.388.4.9 minlen `MHAParser::float_t` `plingploing::if_t::minlen` [private]

Minimum note length in beats.

4.388.4.10 maxlen `MHAParser::float_t` `plingploing::if_t::maxlen` [private]

Maximum note length in beats.

4.388.4.11 bassmod `MHAParser::float_t plingploing::if_t::bassmod [private]`

Bass key modulation depth.

4.388.4.12 bassperiod `MHAParser::float_t plingploing::if_t::bassperiod [private]`

Bass key modulation period.

The documentation for this class was generated from the following file:

- `plingploing.cpp`

4.389 plingploing::plingploing_t Class Reference

Run-time configuration of the plingploing music generator.

Public Member Functions

- `plingploing_t (mhaconfig_t, mha_real_t level, mha_real_t pitch, mha_real_t k1, mha_real_t k2, mha_real_t i1, mha_real_t i2, mha_real_t bpm, mha_real_t minlen, mha_real_t maxlen, mha_real_t bassmod, mha_real_t bassperiod)`
- `void process (mha_wave_t *)`

Private Attributes

- `mhaconfig_t cf`
- `mha_real_t pitch_`
- `unsigned int bt`
- `unsigned int t`
- `unsigned int len`
- `mha_real_t dur_`
- `mha_real_t minlen_`
- `mha_real_t maxlen_`
- `mha_real_t bass`
- `mha_real_t freq`
- `mha_real_t fun1_key`
- `mha_real_t fun1_range`
- `mha_real_t fun1`
- `mha_real_t fun2`
- `mha_real_t fun2_key`

- `mha_real_t fun2_range`
- `mha_real_t dist`
- `mha_real_t dist1`
- `mha_real_t alph`
- `mha_real_t rms`
- `mha_real_t bassmod_`
- `mha_real_t bassperiod_`
- `MHAWindow::hanning_t hann1`
- `MHAWindow::hanning_t hann2`
- `mha_real_t level`

4.389.1 Detailed Description

Run-time configuration of the plingploing music generator.

4.389.2 Constructor & Destructor Documentation

4.389.2.1 `plingploing_t()` `plingploing::plingploing_t::plingploing_t (`

```
    mhaconfig_t c,
    mha_real_t level,
    mha_real_t pitch,
    mha_real_t k1,
    mha_real_t k2,
    mha_real_t i1,
    mha_real_t i2,
    mha_real_t bpm,
    mha_real_t minlen,
    mha_real_t maxlen,
    mha_real_t bassmod,
    mha_real_t bassperiod )
```

4.389.3 Member Function Documentation

4.389.3.1 `process()` `void plingploing::plingploing_t::process (`

```
    mha_wave_t * s )
```

4.389.4 Member Data Documentation

4.389.4.1 cf `mhaconfig_t` `plingploing::plingploing_t::cf` [private]

4.389.4.2 pitch_ `mha_real_t` `plingploing::plingploing_t::pitch_` [private]

4.389.4.3 bt `unsigned int` `plingploing::plingploing_t::bt` [private]

4.389.4.4 t `unsigned int` `plingploing::plingploing_t::t` [private]

4.389.4.5 len `unsigned int` `plingploing::plingploing_t::len` [private]

4.389.4.6 dur_ `mha_real_t` `plingploing::plingploing_t::dur_` [private]

4.389.4.7 minlen_ `mha_real_t` `plingploing::plingploing_t::minlen_` [private]

4.389.4.8 maxlen_ `mha_real_t` `plingploing::plingploing_t::maxlen_` [private]

4.389.4.9 bass `mha_real_t` `plingploing::plingploing_t::bass` [private]

4.389.4.10 freq `mha_real_t` `plingploing::plingploing_t::freq` [private]

4.389.4.11 fun1_key `mha_real_t` `plingploing::plingploing_t::fun1_key` [private]

4.389.4.12 fun1_range `mha_real_t` `plingploing::plingploing_t::fun1_range` [private]

4.389.4.13 fun1 `mha_real_t` `plingploing::plingploing_t::fun1` [private]

4.389.4.14 fun2 `mha_real_t` `plingploing::plingploing_t::fun2` [private]

4.389.4.15 fun2_key `mha_real_t` `plingploing::plingploing_t::fun2_key` [private]

4.389.4.16 fun2_range `mha_real_t` `plingploing::plingploing_t::fun2_range` [private]

4.389.4.17 dist `mha_real_t` `plingploing::plingploing_t::dist` [private]

4.389.4.18 dist1 `mha_real_t` `plingploing::plingploing_t::dist1` [private]

4.389.4.19 alph `mha_real_t` `plingploing::plingploing_t::alph` [private]

4.389.4.20 rms `mha_real_t` `plingploing::plingploing_t::rms` [private]

4.389.4.21 bassmod_ `mha_real_t` `plingploing::plingploing_t::bassmod_` [private]

4.389.4.22 bassperiod_ `mha_real_t` `plingploing::plingploing_t::bassperiod_` [private]

4.389.4.23 hann1 `MHAWindow::hanning_t` `plingploing::plingploing_t::hann1` [private]

4.389.4.24 hann2 `MHAWindow::hanning_t` `plingploing::plingploing_t::hann2` [private]

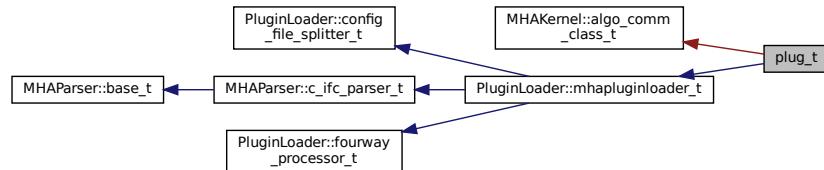
4.389.4.25 level `mha_real_t` `plingploing::plingploing_t::level` [private]

The documentation for this class was generated from the following file:

- `plingploing.cpp`

4.390 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

4.390.1 Constructor & Destructor Documentation

4.390.1.1 plug_t() `plug_t::plug_t (`
`const std::string & libname,`
`const std::string & chain,`
`const std::string & algo)`

4.390.1.2 ~plug_t() `plug_t::~plug_t () throw () [inline]`

4.390.2 Member Function Documentation

4.390.2.1 get_process_wave() `MHAProc_wave2wave_t plug_t::get_process_wave ()`

4.390.2.2 get_process_spec() `MHAProc_wave2spec_t plug_t::get_process_spec ()`

4.390.2.3 get_handle() `void * plug_t::get_handle ()`

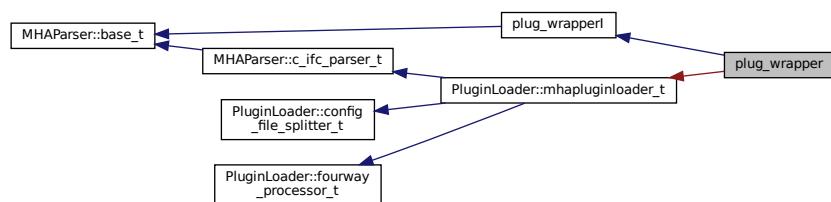
4.390.2.4 get_ac() `algo_comm_t plug_t::get_ac () [inline]`

The documentation for this class was generated from the following file:

- `analysispath.cpp`

4.391 plug_wrapper Class Reference

Inheritance diagram for plug_wrapper:



Public Member Functions

- `plug_wrapper (algo_comm_t iac, const std::string & libname)`
- virtual `~plug_wrapper ()=default`
- virtual `std::vector< std::string > get_categories ()`
- virtual `std::string parse (const std::string &str)`
- virtual `bool has_parser ()`
- virtual `std::string get_documentation ()`
- virtual `bool has_process (mha_domain_t in, mha_domain_t out)`

Additional Inherited Members

4.391.1 Constructor & Destructor Documentation

4.391.1.1 **plug_wrapper()** plug_wrapper::plug_wrapper (

```
    algo_comm_t iac,  
    const std::string & libname ) [inline]
```

4.391.1.2 ~**plug_wrapper()** virtual plug_wrapper::~plug_wrapper () [virtual],

[default]

4.391.2 Member Function Documentation

4.391.2.1 **get_categories()** virtual std::vector<std::string> plug_wrapper::get_categories () [inline], [virtual]

Implements **plug_wrapperI** (p. 1350).

4.391.2.2 **parse()** virtual std::string plug_wrapper::parse (

```
    const std::string & str ) [inline], [virtual]
```

Reimplemented from **PluginLoader::mhapluginloader_t** (p. 1368).

4.391.2.3 **has_parser()** virtual bool plug_wrapper::has_parser () [inline], [virtual]

Implements **plug_wrapperI** (p. 1350).

```
4.391.2.4 get_documentation() virtual std::string plug_wrapper::get_documentation()
( ) [inline], [virtual]
```

Implements **plug_wrapperl** (p. 1350).

```
4.391.2.5 has_process() virtual bool plug_wrapper::has_process (
    mha_domain_t in,
    mha_domain_t out ) [inline], [virtual]
```

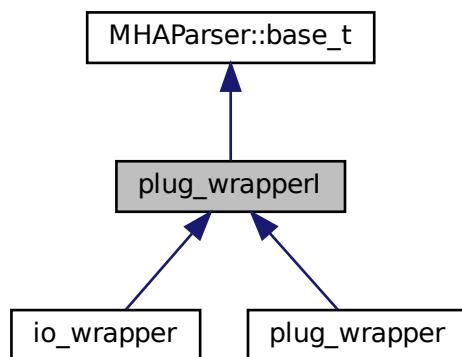
Implements **plug_wrapperl** (p. 1351).

The documentation for this class was generated from the following file:

- **generatemhaplugindoc.cpp**

4.392 plug_wrapperl Class Reference

Inheritance diagram for plug_wrapperl:



Public Member Functions

- **plug_wrapperl ()**
- **virtual ~plug_wrapperl ()=default**
- **virtual std::vector< std::string > get_categories ()=0**
- **virtual std::string parse (const std::string &str)=0**
- **virtual bool has_parser ()=0**
- **virtual std::string get_documentation ()=0**
- **virtual bool has_process (mha_domain_t, mha_domain_t)=0**

Additional Inherited Members

4.392.1 Constructor & Destructor Documentation

4.392.1.1 `plug_wrapperI()` `plug_wrapperI::plug_wrapperI () [inline]`

4.392.1.2 `~plug_wrapperI()` `virtual plug_wrapperI::~plug_wrapperI () [virtual], [default]`

4.392.2 Member Function Documentation

4.392.2.1 `get_categories()` `virtual std::vector<std::string> plug_wrapperI::get_← categories () [pure virtual]`

Implemented in **plug_wrapper** (p. 1348), and **io_wrapper** (p. 642).

4.392.2.2 `parse()` `virtual std::string plug_wrapperI::parse (const std::string & str) [pure virtual]`

Reimplemented from **MHAParser::base_t** (p. 1031).

Implemented in **plug_wrapper** (p. 1348), and **io_wrapper** (p. 642).

4.392.2.3 `has_parser()` `virtual bool plug_wrapperI::has_parser () [pure virtual]`

Implemented in **plug_wrapper** (p. 1348), and **io_wrapper** (p. 642).

```
4.392.2.4 get_documentation() virtual std::string plug_wrapperI::get_documentation()
( ) [pure virtual]
```

Implemented in [plug_wrapper](#) (p. 1348), and [io_wrapper](#) (p. 643).

```
4.392.2.5 has_process() virtual bool plug_wrapperI::has_process (
    mha_domain_t ,
    mha_domain_t ) [pure virtual]
```

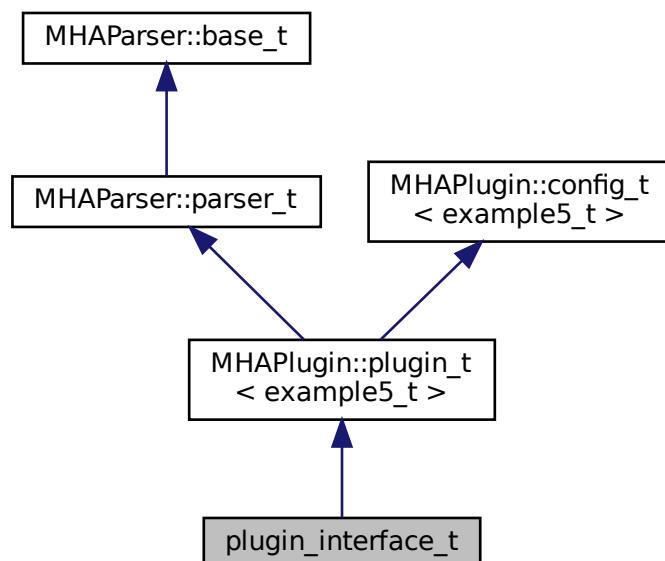
Implemented in [plug_wrapper](#) (p. 1349), and [io_wrapper](#) (p. 643).

The documentation for this class was generated from the following file:

- [generatemhaplugindoc.cpp](#)

4.393 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

4.393.1 Constructor & Destructor Documentation

4.393.1.1 `plugin_interface_t()` `plugin_interface_t::plugin_interface_t (`

```
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.393.2 Member Function Documentation

4.393.2.1 `process()` `mha_spec_t * plugin_interface_t::process (`

```
    mha_spec_t * spec )
```

```
4.393.2.2 prepare() void plugin_interface_t::prepare (
    mhaconfig_t & tfcfg ) [virtual]
```

Implements **MHAPlugIn::plugin_t< example5_t >** (p. 1148).

```
4.393.2.3 update_cfg() void plugin_interface_t::update_cfg ( ) [private]
```

4.393.3 Member Data Documentation

```
4.393.3.1 scale_ch MHAParser::int_t plugin_interface_t::scale_ch [private]
```

```
4.393.3.2 factor MHAParser::float_t plugin_interface_t::factor [private]
```

```
4.393.3.3 patchbay MHAEEvents::patchbay_t< plugin_interface_t > plugin_interface_t::patchbay [private]
```

The documentation for this class was generated from the following file:

- **example5.cpp**

4.394 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**

4.394.1 Constructor & Destructor Documentation

4.394.1.1 pluginbrowser_t() `pluginbrowser_t::pluginbrowser_t ()`

4.394.2 Member Function Documentation

4.394.2.1 get_paths() `void pluginbrowser_t::get_paths ()`

4.394.2.2 scan_plugin() `plugindescription_t pluginbrowser_t::scan_plugin (`
`const std::string & name)`

4.394.2.3 add_plugins() `void pluginbrowser_t::add_plugins ()`

4.394.2.4 clear_plugins() `void pluginbrowser_t::clear_plugins ()`

4.394.2.5 scan_plugins() `void pluginbrowser_t::scan_plugins ()`

4.394.2.6 add_plugin() void pluginbrowser_t::add_plugin (const std::string & name)

4.394.2.7 get_plugins() std::list< **plugindescription_t**> pluginbrowser_t::get_plugins () const [inline]

4.394.3 Member Data Documentation

4.394.3.1 plugin_extension std::string pluginbrowser_t::plugin_extension [private]

4.394.3.2 library_paths std::list<std::string> pluginbrowser_t::library_paths [private]

4.394.3.3 plugins std::list< **plugindescription_t**> pluginbrowser_t::plugins [private]

4.394.3.4 p std::map<std::string, **pluginloader_t***> pluginbrowser_t::p [private]

The documentation for this class was generated from the following files:

- **pluginbrowser.h**
- **pluginbrowser.cpp**

4.395 **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**

4.395.1 Member Data Documentation

4.395.1.1 name std::string plugindescription_t::name

4.395.1.2 fullname std::string plugindescription_t::fullname

4.395.1.3 documentation std::string plugindescription_t::documentation

4.395.1.4 categories std::vector<std::string> plugindescription_t::categories

4.395.1.5 wave2wave bool plugindescription_t::wave2wave

4.395.1.6 wave2spec bool plugindescription_t::wave2spec

4.395.1.7 spec2wave bool plugindescription_t::spec2wave

4.395.1.8 spec2spec bool plugindescription_t::spec2spec

4.395.1.9 query_cmds std::vector<std::string> plugindescription_t::query_cmds

4.395.1.10 queries std::map<std::string, std::string> plugindescription_t::queries

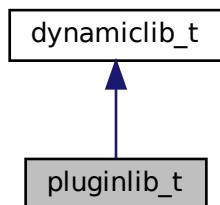
The documentation for this class was generated from the following file:

- **pluginbrowser.h**

4.396 pluginlib_t Class Reference

Specialisation of **dynamiclib_t** (p. 445) for mha plugin libraries.

Inheritance diagram for pluginlib_t:



Public Member Functions

- **pluginlib_t** (const std::string &name_)

C'tor of the wrapper class.
- virtual void * **resolve** (const std::string &name_) override

Resolves the plugin callback specified by name_, e.g.
- virtual ~**pluginlib_t** ()

D'tor.

Additional Inherited Members

4.396.1 Detailed Description

Specialisation of **dynamiclib_t** (p. 445) for mha plugin libraries.

4.396.2 Constructor & Destructor Documentation

4.396.2.1 **pluginlib_t()** pluginlib_t::pluginlib_t (const std::string & name_) [explicit]

C'tor of the wrapper class.

Takes the file name of a shared library w/o the suffix as argument, searches for the library in the system-dependent standard paths for libraries and in MHA_LIBRARY_DIR. Calls load_lib for the actual work.

Parameters

<i>name_</i>	File name of the shared library, without suffix
--------------	---

4.396.2.2 ~**pluginlib_t()** pluginlib_t::~pluginlib_t () [virtual]

D'tor.

4.396.3 Member Function Documentation

4.396.3.1 `resolve()` `void * pluginlib_t::resolve (`
`const std::string & name_) [override], [virtual]`

Resolves the plugin callback specified by `name_`, e.g.

'process', 'prepare', etc... and returns a pointer to it or a `nullptr` if the function was not found. Automatically adds the required prefixes and suffixes for dynamically/statically compiled mha plugins

Parameters

<code>name_</code>	Name of the function to be resolved
--------------------	-------------------------------------

Returns

Pointer to the function

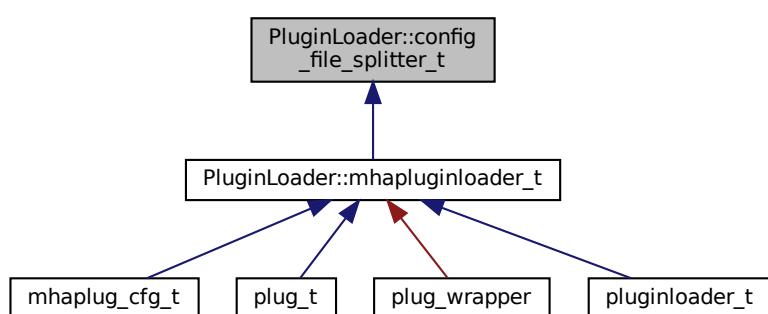
Reimplemented from `dynamiclib_t` (p. 447).

The documentation for this class was generated from the following files:

- `mha_os.h`
- `mha_os.cpp`

4.397 PluginLoader::config_file_splitter_t Class Reference

Inheritance diagram for `PluginLoader::config_file_splitter_t`:



Public Member Functions

- **config_file_splitter_t** (const std::string &name)
- const std::string & **get_configname** () const
- const std::string & **get_libname** () const
- const std::string & **get_origname** () const
- const std::string & **get_configfile** () const

Private Attributes

- std::string **libname**
- std::string **configname**
- std::string **origname**
- std::string **configfile**

4.397.1 Constructor & Destructor Documentation

4.397.1.1 config_file_splitter_t() PluginLoader::config_file_splitter_t::config_file_splitter_t (const std::string & name)

4.397.2 Member Function Documentation

4.397.2.1 get_configname() const std::string& PluginLoader::config_file_splitter_t::get_configname () const [inline]

4.397.2.2 get_libname() const std::string& PluginLoader::config_file_splitter_t::get_libname () const [inline]

4.397.2.3 get_origname() const std::string& PluginLoader::config_file_splitter_t::get_origname () const [inline]

4.397.2.4 get_configfile() const std::string& PluginLoader::config_file_splitter_t::get_configfile () const [inline]

4.397.3 Member Data Documentation

4.397.3.1 libname std::string PluginLoader::config_file_splitter_t::libname [private]

4.397.3.2 configname std::string PluginLoader::config_file_splitter_t::configname [private]

4.397.3.3 origname std::string PluginLoader::config_file_splitter_t::origname [private]

4.397.3.4 configfile std::string PluginLoader::config_file_splitter_t::configfile [private]

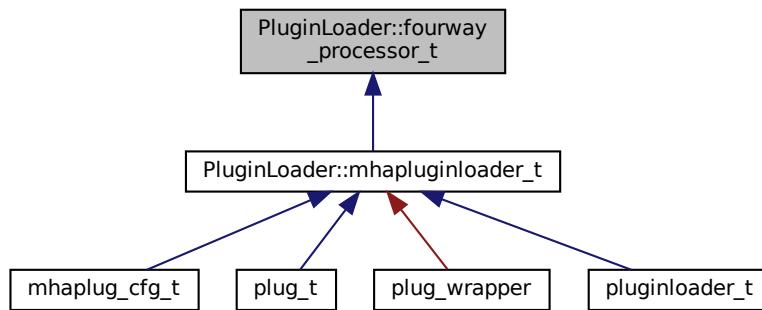
The documentation for this class was generated from the following files:

- **mhapluginloader.h**
- **mhapluginloader.cpp**

4.398 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. 1364) are released here in **fourway_processor_t::release** (p. 1365).*
- virtual std::string **parse** (const std::string &query)=0
Parser interface.
- virtual ~**fourway_processor_t** ()
Classes with virtual methods need virtual destructor.

4.398.1 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.

4.398.2 Constructor & Destructor Documentation

4.398.2.1 ~fourway_processor_t() virtual PluginLoader::fourway_processor_t::~fourway_processor_t () [inline], [virtual]

Classes with virtual methods need virtual destructor.

This destructor is empty.

4.398.3 Member Function Documentation

4.398.3.1 process() [1/4] virtual void PluginLoader::fourway_processor_t::process (
`mha_wave_t * s_in,`
`mha_wave_t ** s_out) [pure virtual]`

Pure waveform processing.

Parameters

<code>s_in</code>	input waveform signal
<code>s_out</code>	output waveform signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 1369).

4.398.3.2 process() [2/4] virtual void PluginLoader::fourway_processor_t::process (
`mha_spec_t * s_in,`
`mha_spec_t ** s_out) [pure virtual]`

Pure spectrum processing.

Parameters

<code>s_in</code>	input spectrum signal
<code>s_out</code>	output spectrum signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 1369).

4.398.3.3 process() [3/4] `virtual void PluginLoader::fourway_processor_t::process (mha_wave_t * s_in, mha_spec_t ** s_out) [pure virtual]`

Signal processing with domain transformation from waveform to spectrum.

Parameters

<i>s_in</i>	input waveform signal
<i>s_out</i>	output spectrum signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 1369).

4.398.3.4 process() [4/4] `virtual void PluginLoader::fourway_processor_t::process (mha_spec_t * s_in, mha_wave_t ** s_out) [pure virtual]`

Signal processing with domain transformation from spectrum to waveform.

Parameters

<i>s_in</i>	input spectrum signal
<i>s_out</i>	output waveform signal

Implemented in **PluginLoader::mhapluginloader_t** (p. 1369).

4.398.3.5 prepare() `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. 1364), settings reflects domain and dimensions of the input signal. When fourway_processor_t::prepare (p. 1364) returns, settings reflects domain and dimensions of the output signal.
-----------------	---

Implemented in **PluginLoader::mhapluginloader_t** (p. 1368).

4.398.3.6 release() `virtual void PluginLoader::fourway_processor_t::release () [pure virtual]`

Resources allocated for signal processing in **fourway_processor_t::prepare** (p. 1364) are released here in **fourway_processor_t::release** (p. 1365).

Implemented in **PluginLoader::mhapluginloader_t** (p. 1369).

4.398.3.7 parse() `virtual std::string PluginLoader::fourway_processor_t::parse (const std::string & query) [pure virtual]`

Parser interface.

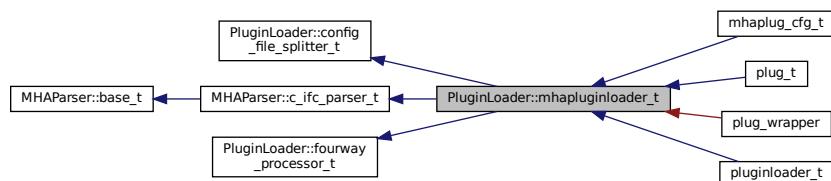
Implemented in **PluginLoader::mhapluginloader_t** (p. 1368), and **plug_wrapper** (p. 1348).

The documentation for this class was generated from the following file:

- **mhapluginloader.h**

4.399 PluginLoader::mhapluginloader_t Class Reference

Inheritance diagram for PluginLoader::mhapluginloader_t:



Public Member Functions

- std::string **parse** (const std::string &str) override
- **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** &) override
- void **release** () override
- void **process** (**mha_wave_t** *, **mha_wave_t** **) override
- void **process** (**mha_spec_t** *, **mha_spec_t** **) override
- void **process** (**mha_wave_t** *, **mha_spec_t** **) override
- void **process** (**mha_spec_t** *, **mha_wave_t** **) override
- std::string **getfullname** () const
- std::string **get_documentation** () const
- std::vector< std::string > **get_categories** () const
- bool **is_prepared** () const

Protected Member Functions

- void **test_error** ()
- void **test_version** ()
- void **mha_test_struct_size** (unsigned int s)
- void **resolve_and_init** ()

Protected Attributes

- int **lib_err**
- **algo_comm_t** ac
- **pluginlib_t** lib_handle
- void * lib_data
- **MHAGetVersion_t** MHAGetVersion_cb
- **MHAInit_t** MHAInit_cb
- **MHADestroy_t** MHADestroy_cb
- **MHAPrepare_t** MHAPrepare_cb
- **MHARelease_t** MHARelease_cb
- **MHAProc_wave2wave_t** MHAProc_wave2wave_cb
- **MHAProc_spec2spec_t** MHAProc_spec2spec_cb
- **MHAProc_wave2spec_t** MHAProc_wave2spec_cb
- **MHAProc_spec2wave_t** MHAProc_spec2wave_cb
- **MHASet_t** MHASet_cb

- **MHACpp_t MHACpp_cb**
- **MHAError_t MHAError_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

4.399.1 Constructor & Destructor Documentation

4.399.1.1 mhapluginloader_t() `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.
----------------------	--

4.399.1.2 ~mhapluginloader_t() `PluginLoader::mhapluginloader_t::~mhapluginloader_t () throw ()`

4.399.2 Member Function Documentation

4.399.2.1 parse() `std::string PluginLoader::mhapluginloader_t::parse (const std::string & str) [override], [virtual]`

Implements **PluginLoader::fourway_processor_t** (p. 1365).

Reimplemented in **plug_wrapper** (p. 1348).

4.399.2.2 has_process() `bool PluginLoader::mhapluginloader_t::has_process (mha_domain_t in, mha_domain_t out) const`

4.399.2.3 has_parser() `bool PluginLoader::mhapluginloader_t::has_parser () const`

4.399.2.4 input_domain() `mha_domain_t PluginLoader::mhapluginloader_t::input_domain () const`

4.399.2.5 output_domain() `mha_domain_t PluginLoader::mhapluginloader_t::output_domain () const`

4.399.2.6 `prepare()` void PluginLoader::mhaplugloader_t::prepare (
 mhaconfig_t & tf) [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1364).

4.399.2.7 `release()` void PluginLoader::mhaplugloader_t::release () [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1365).

4.399.2.8 `process()` [1/4] void PluginLoader::mhaplugloader_t::process (
 mha_wave_t * s_in,
 mha_wave_t ** s_out) [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1363).

4.399.2.9 `process()` [2/4] void PluginLoader::mhaplugloader_t::process (
 mha_spec_t * s_in,
 mha_spec_t ** s_out) [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1363).

4.399.2.10 `process()` [3/4] void PluginLoader::mhaplugloader_t::process (
 mha_wave_t * s_in,
 mha_spec_t ** s_out) [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1364).

4.399.2.11 `process()` [4/4] void PluginLoader::mhaplugloader_t::process (
 mha_spec_t * s_in,
 mha_wave_t ** s_out) [override], [virtual]

Implements **PluginLoader::fourway_processor_t** (p. 1364).

4.399.2.12 getfullname() std::string PluginLoader::mhaplugloader_t::getfullname () const [inline]

4.399.2.13 get_documentation() std::string PluginLoader::mhaplugloader_t::get_documentation () const [inline]

4.399.2.14 get_categories() std::vector<std::string> PluginLoader::mhaplugloader_t::get_categories () const [inline]

4.399.2.15 is_prepared() bool PluginLoader::mhaplugloader_t::is_prepared () const [inline]

4.399.2.16 test_error() void PluginLoader::mhaplugloader_t::test_error () [protected]

4.399.2.17 test_version() void PluginLoader::mhaplugloader_t::test_version () [protected]

4.399.2.18 mha_test_struct_size() void PluginLoader::mhaplugloader_t::mha_test_struct_size (unsigned int s) [protected]

4.399.2.19 resolve_and_init() void PluginLoader::mhaplugloader_t::resolve_and_init () [protected]

4.399.3 Member Data Documentation

4.399.3.1 lib_err int PluginLoader::mhapluginloader_t::lib_err [protected]

4.399.3.2 ac algo_comm_t PluginLoader::mhapluginloader_t::ac [protected]

4.399.3.3 lib_handle pluginlib_t PluginLoader::mhapluginloader_t::lib_handle [protected]

4.399.3.4 lib_data void* PluginLoader::mhapluginloader_t::lib_data [protected]

4.399.3.5 MHAGetVersion_cb MHAGetVersion_t PluginLoader::mhapluginloader_t::MHAGetVersion_cb [protected]

4.399.3.6 MHAInit_cb MHAInit_t PluginLoader::mhapluginloader_t::MHAInit_cb [protected]

4.399.3.7 MHADestroy_cb MHADestroy_t PluginLoader::mhapluginloader_t::MHADestroy_cb [protected]

4.399.3.8 MHAPrepare_cb MHAPrepare_t PluginLoader::mhapluginloader_t::MHAPrepare_cb [protected]

4.399.3.9 MHARelease_cb `MHARelease_t` `PluginLoader::mhapluginloader_t::MHA::Release_cb` [protected]

4.399.3.10 MHAProc_wave2wave_cb `MHAProc_wave2wave_t` `PluginLoader::mhapluginloader_t::MHAProc_wave2wave_cb` [protected]

4.399.3.11 MHAProc_spec2spec_cb `MHAProc_spec2spec_t` `PluginLoader::mhapluginloader_t::MHAProc_spec2spec_cb` [protected]

4.399.3.12 MHAProc_wave2spec_cb `MHAProc_wave2spec_t` `PluginLoader::mhapluginloader_t::MHAProc_wave2spec_cb` [protected]

4.399.3.13 MHAProc_spec2wave_cb `MHAProc_spec2wave_t` `PluginLoader::mhapluginloader_t::MHAProc_spec2wave_cb` [protected]

4.399.3.14 MHASet_cb `MHASet_t` `PluginLoader::mhapluginloader_t::MHASet_cb` [protected]

4.399.3.15 MHASetcpp_cb `MHASetcpp_t` `PluginLoader::mhapluginloader_t::MHA::Setcpp_cb` [protected]

4.399.3.16 MHAStrError_cb `MHAStrError_t` `PluginLoader::mhapluginloader_t::MHA::StrError_cb` [protected]

4.399.3.17 cf_input `mhaconfig_t` `PluginLoader::mhaplugloader_t::cf_input` [protected]

4.399.3.18 cf_output `mhaconfig_t` `PluginLoader::mhaplugloader_t::cf_output` [protected]

4.399.3.19 plugin_documentation `std::string` `PluginLoader::mhaplugloader_t::plugin_documentation` [protected]

4.399.3.20 plugin_categories `std::vector<std::string>` `PluginLoader::mhaplugloader_t::plugin_categories` [protected]

4.399.3.21 b_check_version `bool` `PluginLoader::mhaplugloader_t::b_check_version` [protected]

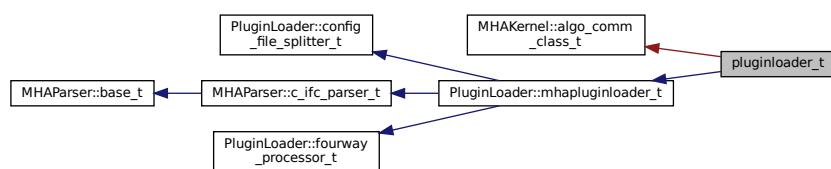
4.399.3.22 b_is_prepared `bool` `PluginLoader::mhaplugloader_t::b_is_prepared` [protected]

The documentation for this class was generated from the following files:

- `mhaplugloader.h`
- `mhaplugloader.cpp`

4.400 pluginloader_t Class Reference

Inheritance diagram for pluginloader_t:



Public Member Functions

- **pluginloader_t** (const std::string &name)
- **~pluginloader_t** () throw ()

Additional Inherited Members

4.400.1 Constructor & Destructor Documentation

4.400.1.1 pluginloader_t() pluginloader_t::pluginloader_t (const std::string & *name*)

4.400.1.2 ~pluginloader_t() pluginloader_t::~pluginloader_t () throw ()

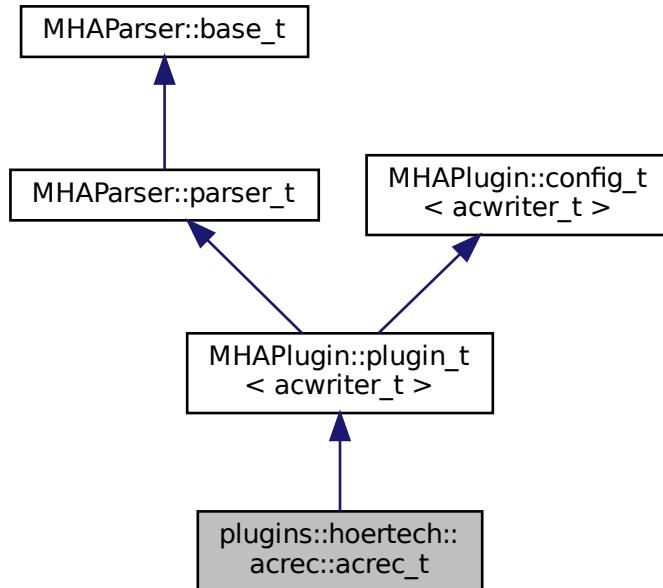
The documentation for this class was generated from the following files:

- **pluginbrowser.h**
- **pluginbrowser.cpp**

4.401 plugins::hoertech::acrec::acrec_t Class Reference

Plugin interface class of plugin acrec.

Inheritance diagram for plugins::hoertech::acrec::acrec_t:



Public Member Functions

- template<class mha_signal_t>
`mha_signal_t * process (mha_signal_t *s)`
Process callback.
- void `prepare (mhaconfig_t &cf)`
Prepare callback.
- void `release ()`
Ensure recorded data is flushed to disk.
- `acrec_t (const algo_comm_t &iac, const std::string &, const std::string &)`
Plugin interface constructor.

Private Member Functions

- void `start_new_session ()`
Configuration callback called whenever configuration variable "record" is written to.

Private Attributes

- `MHAParser::bool_t record`
- `MHAParser::int_t fifolen`
- `MHAParser::int_t minwrite`
- `MHAParser::string_t prefix`
- `MHAParser::string_t varname`
- `MHAParser::bool_t use_date`
- `MHAEvents::patchbay_t< acrec_t > patchbay`
- `comm_var_t cv`
- `algo_comm_t ac`

Additional Inherited Members

4.401.1 Detailed Description

Plugin interface class of plugin acrec.

4.401.2 Constructor & Destructor Documentation

```
4.401.2.1 acrec_t() acrec_t::acrec_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

Plugin interface constructor.

Parameters

<i>iac</i>	Algorithm communication variable space.
------------	---

4.401.3 Member Function Documentation

```
4.401.3.1 process() template<class mha_signal_t >
mha_signal_t * acrec_t::process (
    mha_signal_t * s )
```

Process callback.

Pushes the data from one AC variable into the fifo.

Returns

the unmodified input signal.

Parameters

s	input signal. The audio signal is not used or modified.
----------	---

```
4.401.3.2 prepare() void acrec_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Prepare callback.

acrec does not modify the signal parameters.

Parameters

cf	The signal parameters.
-----------	------------------------

Implements **MHAPlugin::plugin_t< acwriter_t >** (p. 1148).

```
4.401.3.3 release() void acrec_t::release ( ) [virtual]
```

Ensure recorded data is flushed to disk.

Reimplemented from **MHAPlugin::plugin_t< acwriter_t >** (p. 1149).

4.401.3.4 start_new_session() void acrec_t::start_new_session () [private]

Configuration callback called whenever configuration variable "record" is written to.

4.401.4 Member Data Documentation

4.401.4.1 record MHAParser::bool_t plugins::hoertech::acrec::acrec_t::record [private]

4.401.4.2 fifolen MHAParser::int_t plugins::hoertech::acrec::acrec_t::fifolen [private]

4.401.4.3 minwrite MHAParser::int_t plugins::hoertech::acrec::acrec_t::minwrite [private]

4.401.4.4 prefix MHAParser::string_t plugins::hoertech::acrec::acrec_t::prefix [private]

4.401.4.5 varname MHAParser::string_t plugins::hoertech::acrec::acrec_t::varname [private]

4.401.4.6 use_date MHAParser::bool_t plugins::hoertech::acrec::acrec_t::use_date [private]

4.401.4.7 patchbay `MHAEvents::patchbay_t< acrec_t>` `plugins::hoertech::acrec::acrec_t::patchbay` [private]

4.401.4.8 cv `comm_var_t` `plugins::hoertech::acrec::acrec_t::cv` [private]

4.401.4.9 ac `algo_comm_t` `plugins::hoertech::acrec::acrec_t::ac` [private]

The documentation for this class was generated from the following files:

- `acrec.hh`
- `acrec.cpp`

4.402 **plugins::hoertech::acrec::acwriter_t** Class Reference

acwriter_t (p. 1379) decouples signal processing from writing to disk.

Public Types

- `typedef double output_type`
The numeric data type used for outputting the data to disk.

Public Member Functions

- `acwriter_t (bool active, unsigned fifosize, unsigned minwrite, const std::string &prefix, bool use_date, const std::string & varname)`
Constructor allocates fifo and disk output buffer.
- `~acwriter_t ()=default`
Deallocates memory but does not terminate the write_thread.
- `void process (comm_var_t *)`
Place the data present in the algorithm communication variable into the fifo for output to disk.
- `void exit_request ()`
Terminate output thread.
- `const char * get_varname () const`
getter for ac variable name

Private Member Functions

- void **write_thread ()**
Main method of the disk writer thread.
- void **create_datafile** (const std::string &prefix, bool use_date)
Open data file for output.

Private Attributes

- std::atomic< bool > **close_session**
Cross-thread-synchronization.
- const bool **active**
The writer thread and the output file will only be created when active is true.
- std::unique_ptr< **mha_fifo_if_t**< **output_type** >> **fifo**
Fifo for decoupling signal processing thread from disk writer thread.
- const unsigned int **disk_write_threshold_min_num_samples**
Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.
- std::thread **writethread**
The thread that writes to disk.
- std::unique_ptr< **output_type**[]> **diskbuffer**
Intermediate buffer to receive data from fifo and store on disk.
- std::fstream **outfile**
Output file.
- unsigned **num_channels** = 0U
Number of channels of AC variable using stride.
- bool **is_num_channels_known** = false
The number of channels is determined during the first process callback.
- bool **is_complex** = false
If the AC variable is of complex valued type or not.
- const std::string **varname**
The name of the ac variable to publish.

4.402.1 Detailed Description

acwriter_t (p. 1379) decouples signal processing from writing to disk.

Data arriving in numeric AC variables is converted to data type double, placed into a fifo pipeline to transport the data from the signal processing thread to the disk writing thread, and finally written to disk in chunks of at least minwrite numbers. All numbers are written to disk as binary doubles (8 bytes) in host byte order.

4.402.2 Member Typedef Documentation

4.402.2.1 **output_type** `typedef double plugins::hoertech::acrec::acwriter_t::output_type`

The numeric data type used for outputting the data to disk.

4.402.3 Constructor & Destructor Documentation

4.402.3.1 **acwriter_t()** `acwriter_t::acwriter_t (`

```
    bool active,
    unsigned fifosize,
    unsigned minwrite,
    const std::string & prefix,
    bool use_date,
    const std::string & varname )
```

Constructor allocates fifo and disk output buffer.

It spawns a new thread for writing data to disk when `active==true`. In order to terminate the thread, method `exit_request` **must** be called before this object is destroyed.

Parameters

<code>active</code>	Only write data to disk when this is true.
<code>fifosize</code>	Capacity of both the fifo pipeline and of the disk buffer.
<code>minwrite</code>	Wait for a fifo fill count of at least <code>minwrite</code> doubles before flushing the contents of the fifo to disk. Fifo is also flushed before this object is destroyed.
<code>prefix</code>	Path and start of output file name. Will be extended with file name extension ".dat".
<code>use_date</code>	When true, the current date and time will be appended to the output file name before the file name extension.
<code>varname</code>	Name of AC variable to save into file. Can be accessed through getter method <code>get_varname()</code> (p. 1382). Stored here to avoid races between processing thread and configuration thread.

4.402.3.2 ~acwriter_t() `plugins::hoertech::acrec::acwriter_t::~acwriter_t () [default]`

Deallocates memory but does not terminate the write_thread.

write_thread must be terminated before the destructor executes by calling exit_request.

4.402.4 Member Function Documentation**4.402.4.1 process()** `void acwriter_t::process (comm_var_t * s)`

Place the data present in the algorithm communication variable into the fifo for output to disk.

4.402.4.2 exit_request() `void acwriter_t::exit_request ()`

Terminate output thread.

4.402.4.3 get_varname() `const char* plugins::hoertech::acrec::acwriter_t::get_varname () const [inline]`

getter for ac variable name

Returns

name as char* as needed by get_var

4.402.4.4 write_thread() `void acwriter_t::write_thread () [private]`

Main method of the disk writer thread.

Periodically wakes up and checks if data needs to be written to disk.

4.402.4.5 create_datafile() `void acwriter_t::create_datafile (const std::string & prefix, bool use_date) [private]`

Open data file for output.

Combine prefix, date, and file name extension

Parameters

<i>prefix</i>	Path and start of output file name. Will be extended with file name extension ".dat".
<i>use_date</i>	When true, the current date and time will be appended to the output file name before the file name extension.

4.402.5 Member Data Documentation

4.402.5.1 close_session std::atomic<bool> `plugins::hoertech::acrec::acwriter_t::close_session` [private]

cross-thread-synchronization.

write_thread() (p. 1382) terminates after this is set to true by **exit_request()** (p. 1382).

4.402.5.2 active const bool `plugins::hoertech::acrec::acwriter_t::active` [private]

The writer thread and the output file will only be created when active is true.

4.402.5.3 fifo std::unique_ptr< `mha_fifo_lf_t< output_type>` > `plugins::hoertech::acrec::acwriter_t::fifo` [private]

Fifo for decoupling signal processing thread from disk writer thread.

4.402.5.4 disk_write_threshold_min_num_samples const unsigned int `plugins::hoertech::acrec::acwriter_t::disk_write_threshold_min_num_samples` [private]

Minimum number of samples that need to be waiting in the fifo before the disk writer thread writes them to disk.

4.402.5.5 writethread std::thread plugins::hoertech::acrec::acwriter_t::writethread [private]

The thread that writes to disk.

4.402.5.6 diskbuffer std::unique_ptr< **output_type** []> plugins::hoertech::acrec::acwriter_t::diskbuffer [private]

Intermediate buffer to receive data from fifo and store on disk.

4.402.5.7 outfile std::fstream plugins::hoertech::acrec::acwriter_t::outfile [private]

Output file.

4.402.5.8 num_channels unsigned plugins::hoertech::acrec::acwriter_t::num_channels = 0U [private]

Number of channels of AC variable using stride.

If the number of channels changes during processing, an exception is thrown.

4.402.5.9 is_num_channels_known bool plugins::hoertech::acrec::acwriter_t::is_num_channels_known = false [private]

The number of channels is determined during the first process callback.

is_num_channels_known is set to true after the first process callback.

4.402.5.10 is_complex bool plugins::hoertech::acrec::acwriter_t::is_complex = false [private]

If the AC variable is of complex valued type or not.

If this changes during processing, then an exception is thrown.

4.402.5.11 varname const std::string plugins::hoertech::acrec::acwriter_t::varname
[private]

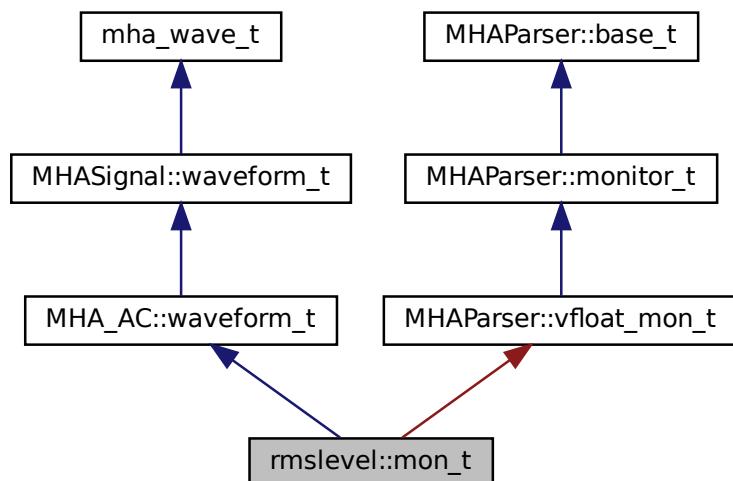
The name of the ac variable to publish.

The documentation for this class was generated from the following files:

- **acrec.hh**
- **acrec.cpp**

4.403 rmslevel::mon_t Class Reference

Inheritance diagram for rmslevel::mon_t:



Public Member Functions

- `mon_t` (unsigned int nch, const std::string & **name**, **algo_comm_t** **ac**, const std::string &**base**, **MHParse::parser_t** &**p**, const std::string & **help**)
- `mon_t` (const `mon_t` &)=delete
- `mon_t` (`mon_t` &&)=delete
- `mon_t` & **operator=** (const `mon_t` &)=delete
- `mon_t` & **operator=** (`mon_t` &&)=delete
- `void store ()`

Additional Inherited Members**4.403.1 Constructor & Destructor Documentation****4.403.1.1 mon_t() [1/3]** rmslevel::mon_t::mon_t (

```
    unsigned int nch,
    const std::string & name,
    algo_comm_t ac,
    const std::string & base,
    MHAParser::parser_t & p,
    const std::string & help )
```

4.403.1.2 mon_t() [2/3] rmslevel::mon_t::mon_t (

```
    const mon_t & ) [delete]
```

4.403.1.3 mon_t() [3/3] rmslevel::mon_t::mon_t (

```
    mon_t && ) [delete]
```

4.403.2 Member Function Documentation**4.403.2.1 operator=() [1/2]** mon_t& rmslevel::mon_t::operator= (

```
    const mon_t & ) [delete]
```

4.403.2.2 operator=() [2/2] mon_t& rmslevel::mon_t::operator= (

```
    mon_t && ) [delete]
```

4.403.2.3 store() void rmslevel::mon_t::store ()

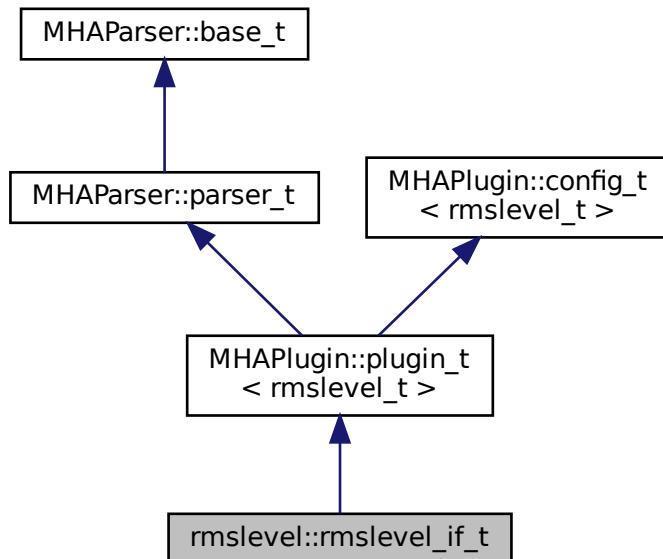
The documentation for this class was generated from the following file:

- [rmslevel.cpp](#)

4.404 rmslevel::rmslevel_if_t Class Reference

Interface class of the rmslevel plugin.

Inheritance diagram for rmslevel::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 Member Functions

- `void update ()`

Private Attributes

- **MHAEvents::patchbay_t< rmslevel_if_t > patchbay**
- std::string **name**
- **MHAParser::kw_t unit**

Additional Inherited Members

4.404.1 Detailed Description

Interface class of the rmslevel plugin.

4.404.2 Constructor & Destructor Documentation

```
4.404.2.1 rmslevel_if_t() rmslevel::rmslevel_if_t::rmslevel_if_t (
    const algo_comm_t & iac,
    const std::string & ith,
    const std::string & ial )
```

4.404.3 Member Function Documentation

```
4.404.3.1 process() [1/2] mha_spec_t * rmslevel::rmslevel_if_t::process (
    mha_spec_t * s )
```

```
4.404.3.2 process() [2/2] mha_wave_t * rmslevel::rmslevel_if_t::process (
    mha_wave_t * s )
```

4.404.3.3 `prepare()` void rmslevel::rmslevel_if_t::prepare (
 mhaconfig_t & tf) [virtual]

Implements **MHAPlugin::plugin_t< rmslevel_t >** (p. 1148).

4.404.3.4 `update()` void rmslevel::rmslevel_if_t::update () [private]

4.404.4 Member Data Documentation

4.404.4.1 `patchbay` MHAEvents::patchbay_t< rmslevel_if_t > rmslevel::rmslevel_if_t::patchbay [private]

4.404.4.2 `name` std::string rmslevel::rmslevel_if_t::name [private]

4.404.4.3 `unit` MHAParser::kw_t rmslevel::rmslevel_if_t::unit [private]

The documentation for this class was generated from the following file:

- **rmslevel.cpp**

4.405 rmslevel::rmslevel_t Class Reference

Run-time configuration class of the rmslevel plugin.

Public Member Functions

- **rmslevel_t** (const **mhaconfig_t** &tf, **algo_comm_t** ac, const std::string &name, **MHAParser::parser_t** &p, **UNIT** unit_)

C'tor of the runtime configuration class.
- **~rmslevel_t** ()
- **mha_spec_t * process** (**mha_spec_t** *)
- **mha_wave_t * process** (**mha_wave_t** *)
- **void fill** (**mha_spec_t** *)
- **void sum_and_fill** (**mha_spec_t** *)
- **void fill** (**mha_wave_t** *)
- **void sum_and_fill** (**mha_wave_t** *)
- **void insert** ()

Private Attributes

- std::unordered_map< std::string, std::unique_ptr< **mon_t** > > **monitors**
- unsigned int **ffflen**
- **UNIT** **unit**
- **mha_domain_t** **domain**
- std::vector< **mha_real_t** > **freq_offsets**

freq_offsets provides the conversion of dB(SPL) to dB(HL) for every frequency bin in the stft used by coloured_intensity.

4.405.1 Detailed Description

Run-time configuration class of the rmslevel plugin.

4.405.2 Constructor & Destructor Documentation

```
4.405.2.1 rmslevel_t() rmslevel::rmslevel_t::rmslevel_t (
    const mhaconfig_t & tf,
    algo_comm_t ac,
    const std::string & name,
    MHAParser::parser_t & p,
    UNIT unit_ )
```

C'tor of the runtime configuration class.

Parameters

<i>tf</i>	Input signal configuration
<i>ac</i>	AC space, needed to publish ac variables
<i>name</i>	Configured name of the plugin within the parser structure
<i>p</i>	Instance of the parent interface class, needed to publish monitor variables
<i>unit_</i>	Configured dB scale
<i>num_← channels_</i>	Number of monitor channels if channel summation shall be used, zero otherwise

4.405.2.2 ~rmslevel_t() rmslevel::rmslevel_t::~rmslevel_t () [inline]

4.405.3 Member Function Documentation

4.405.3.1 process() [1/2] mha_spec_t * rmslevel::rmslevel_t::process (mha_spec_t * *s*)

4.405.3.2 process() [2/2] mha_wave_t * rmslevel::rmslevel_t::process (mha_wave_t * *s*)

4.405.3.3 fill() [1/2] void rmslevel::rmslevel_t::fill (mha_spec_t *)

4.405.3.4 sum_and_fill() [1/2] void rmslevel::rmslevel_t::sum_and_fill (mha_spec_t *)

4.405.3.5 `fill()` [2/2] void rmslevel::rmslevel_t::fill (
 mha_wave_t *)

4.405.3.6 `sum_and_fill()` [2/2] void rmslevel::rmslevel_t::sum_and_fill (
 mha_wave_t *)

4.405.3.7 `insert()` void rmslevel::rmslevel_t::insert ()

4.405.4 Member Data Documentation

4.405.4.1 `monitors` std::unordered_map<std::string, std::unique_ptr< mon_t > > rmslevel↔
 ::rmslevel_t::monitors [private]

4.405.4.2 `fftlen` unsigned int rmslevel::rmslevel_t::fftlen [private]

4.405.4.3 `unit` UNIT rmslevel::rmslevel_t::unit [private]

4.405.4.4 `domain` mha_domain_t rmslevel::rmslevel_t::domain [private]

4.405.4.5 freq_offsets std::vector< mha_real_t> rmslevel::rmslevel_t::freq_offsets
[private]

freq_offsets provides the conversion of dB(SPL) to dB(HL) for every frequency bin in the stft used by coloured_intensity.

Unused when not in spectral domain and unit=hl.

The documentation for this class was generated from the following file:

- **rmslevel.cpp**

4.406 rohBeam::configOptions Struct Reference

Public Attributes

- bool **enable_adaptive_beam**
- int **binaural_type_index**
- float **alpha_postfilter**
- float **alpha_blocking_XkXi**
- float **alpha_blocking_XkY**

4.406.1 Member Data Documentation

4.406.1.1 enable_adaptive_beam bool rohBeam::configOptions::enable_adaptive_beam

4.406.1.2 binaural_type_index int rohBeam::configOptions::binaural_type_index

4.406.1.3 alpha_postfilter float rohBeam::configOptions::alpha_postfilter

4.406.1.4 **alpha_blocking_XkXi** float rohBeam::configOptions::alpha_blocking_XkXi

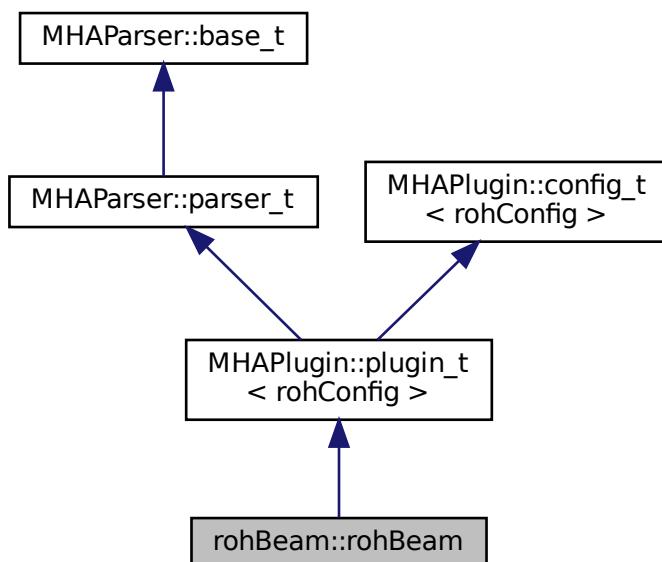
4.406.1.5 **alpha_blocking_XkY** float rohBeam::configOptions::alpha_blocking_XkY

The documentation for this struct was generated from the following file:

- **rohBeam.hh**

4.407 rohBeam::rohBeam Class Reference

Inheritance diagram for rohBeam::rohBeam:



Public Member Functions

- **rohBeam (algo_comm_t & ac, const std::string &chain_name, const std::string &algo_name)**
- **~rohBeam ()**
- **mha_spec_t * process (mha_spec_t *)**
- **void prepare (mhaconfig_t &)**
- **void release (void)**

Private Member Functions

- void **update_cfg** ()
- float **compute_head_model_T** (float)
- float **compute_head_model_alpha** (float)
- Eigen::MatrixXcf * **compute_head_model_mat** (float src_az_degrees)
- **MHASignal::matrix_t** * **compute_delaycomp_vec** (Eigen::MatrixXcf *headModel)
- std::vector< Eigen::MatrixXcf > * **noise_integrate_hrtf** ()
- Eigen::VectorXcf **solve_MVDR** (Eigen::VectorXcf propVec, Eigen::MatrixXcf noiseM)
- const Eigen::MatrixXf **compute_uncorr** (float w)
- const Eigen::MatrixXf **compute_diff2D** (float)
- const Eigen::MatrixXf **compute_diff3D** (float)
- **MHASignal::matrix_t** * **compute_beamW** (Eigen::MatrixXcf *)
- float **compute_wng** (Eigen::VectorXcf freqRes, Eigen::VectorXcf propVec)
- void **export_beam_design** (const **MHASignal::matrix_t** &beamW, const Eigen::MatrixXcf &headModel)
- **noiseFuncPtr** **get_noise_model_func** (void)
- void **on_model_param_valuechanged** ()

Private Attributes

- const typedef Eigen::MatrixXf(rohBeam::* **noiseFuncPtr**)(float)
- **MHAParser::kw_t** **prop_type**
- **MHAParser::string_t** **sampled_hrir_path**
- **MHAParser::float_t** **source_azimuth_degrees**
- **MHAParser::vfloat_t** **mic_azimuth_degrees_vec**
- **MHAParser::float_t** **head_model_sphere_radius_cm**
- **MHAParser::mfloat_t** **intermic_distance_cm**
- **MHAParser::kw_t** **noise_field_model**
- **MHAParser::bool_t** **enable_adaptive_beam**
- **MHAParser::kw_t** **binaural_type**
- **MHAParser::float_t** **diag_loading_mu**
- **MHAParser::bool_t** **enable_export**
- **MHAParser::bool_t** **enable_wng_optimization**
- **MHAParser::float_t** **tau_postfilter_ms**
- **MHAParser::float_t** **tau_blocking_XkXi_ms**
- **MHAParser::float_t** **tau_blocking_XkY_ms**
- **MHAEvents::patchbay_t**< rohBeam > **patchbay**
- bool **prepared**
- **MHA_AC::spectrum_t** * **beamExport**
- **MHA_AC::waveform_t** * **noiseModelExport**
- **MHA_AC::spectrum_t** * **propExport**

Additional Inherited Members

4.407.1 Constructor & Destructor Documentation

4.407.1.1 `rohBeam()` `rohBeam::rohBeam(`
 `algo_comm_t & ac,`
 `const std::string & chain_name,`
 `const std::string & algo_name)`

4.407.1.2 `~rohBeam()` `rohBeam::rohBeam::~rohBeam()`

4.407.2 Member Function Documentation

4.407.2.1 `process()` `mha_spec_t* rohBeam::rohBeam::process(`
 `mha_spec_t *)`

4.407.2.2 `prepare()` `void rohBeam::rohBeam::prepare(`
 `mhaconfig_t &) [virtual]`

Implements **MHAPlugin::plugin_t< rohConfig >** (p. 1148).

4.407.2.3 `release()` `void rohBeam::rohBeam::release(`
 `void) [inline], [virtual]`

Reimplemented from **MHAPlugin::plugin_t< rohConfig >** (p. 1149).

4.407.2.4 update_cfg() void rohBeam::rohBeam::update_cfg () [private]

4.407.2.5 compute_head_model_T() float rohBeam::rohBeam::compute_head_model_T (float) [private]

4.407.2.6 compute_head_model_alpha() float rohBeam::rohBeam::compute_head_model_alpha (float) [private]

4.407.2.7 compute_head_model_mat() Eigen::MatrixXcf* rohBeam::rohBeam::compute_head_model_mat (float src_az_degrees) [private]

4.407.2.8 compute_delaycomp_vec() MHASignal::matrix_t* rohBeam::rohBeam::compute_delaycomp_vec (Eigen::MatrixXcf * headModel) [private]

4.407.2.9 noise_integrate_hrtf() std::vector<Eigen::MatrixXcf>* rohBeam::rohBeam::noise_integrate_hrtf () [private]

4.407.2.10 solve_MVDR() Eigen::VectorXcf rohBeam::rohBeam::solve_MVDR (Eigen::VectorXcf propVec, Eigen::MatrixXcf noiseM) [private]

4.407.2.11 compute_uncorr() const Eigen::MatrixXf rohBeam::rohBeam::compute_←
uncorr (float w) [private]

4.407.2.12 compute_diff2D() const Eigen::MatrixXf rohBeam::rohBeam::compute_diff2D
(float) [private]

4.407.2.13 compute_diff3D() const Eigen::MatrixXf rohBeam::rohBeam::compute_diff3D
(float) [private]

4.407.2.14 compute_beamW() MHASignal::matrix_t* rohBeam::rohBeam::compute_beamW
(Eigen::MatrixXcf *) [private]

4.407.2.15 compute_wng() float rohBeam::rohBeam::compute_wng (Eigen::VectorXcf freqRes,
Eigen::VectorXcf propVec) [private]

4.407.2.16 export_beam_design() void rohBeam::rohBeam::export_beam_design (const MHASignal::matrix_t & beamW,
const Eigen::MatrixXcf & headModel) [private]

4.407.2.17 get_noise_model_func() noiseFuncPtr rohBeam::rohBeam::get_noise_←
model_func (void) [private]

4.407.2.18 on_model_param_valuechanged() void rohBeam::rohBeam::on_model_←
param_valuechanged () [private]

4.407.3 Member Data Documentation

4.407.3.1 noiseFuncPtr const typedef Eigen::MatrixXf(rohBeam::* rohBeam::rohBeam←
::noiseFuncPtr) (float) [private]

4.407.3.2 prop_type MHAParser::kw_t rohBeam::rohBeam::prop_type [private]

4.407.3.3 sampled_hrir_path MHAParser::string_t rohBeam::rohBeam::sampled_hrir_←
path [private]

4.407.3.4 source_azimuth_degrees MHAParser::float_t rohBeam::rohBeam::source_←
azimuth_degrees [private]

4.407.3.5 mic_azimuth_degrees_vec MHAParser::vfloat_t rohBeam::rohBeam::mic_←
azimuth_degrees_vec [private]

4.407.3.6 head_model_sphere_radius_cm MHAParser::float_t rohBeam::rohBeam←
::head_model_sphere_radius_cm [private]

4.407.3.7 intermic_distance_cm `MHAParser::mfloat_t rohBeam::rohBeam::intermic_←`
`distance_cm [private]`

4.407.3.8 noise_field_model `MHAParser::kw_t rohBeam::rohBeam::noise_field_model`
[private]

4.407.3.9 enable_adaptive_beam `MHAParser::bool_t rohBeam::rohBeam::enable_←`
`adaptive_beam [private]`

4.407.3.10 binaural_type `MHAParser::kw_t rohBeam::rohBeam::binaural_type [private]`

4.407.3.11 diag_loading_mu `MHAParser::float_t rohBeam::rohBeam::diag_loading_mu`
[private]

4.407.3.12 enable_export `MHAParser::bool_t rohBeam::rohBeam::enable_export [private]`

4.407.3.13 enable_wng_optimization `MHAParser::bool_t rohBeam::rohBeam::enable_←`
`wng_optimization [private]`

4.407.3.14 tau_postfilter_ms `MHAParser::float_t rohBeam::rohBeam::tau_postfilter←`
`_ms [private]`

4.407.3.15 tau_blocking_XkXi_ms `MHAParser::float_t` `rohBeam::rohBeam::tau_blocking_XkXi_ms` [private]

4.407.3.16 tau_blocking_XkY_ms `MHAParser::float_t` `rohBeam::rohBeam::tau_blocking_XkY_ms` [private]

4.407.3.17 patchbay `MHAEvents::patchbay_t< rohBeam>` `rohBeam::rohBeam::patchbay` [private]

4.407.3.18 prepared `bool` `rohBeam::rohBeam::prepared` [private]

4.407.3.19 beamExport `MHA_AC::spectrum_t*` `rohBeam::rohBeam::beamExport` [private]

4.407.3.20 noiseModelExport `MHA_AC::waveform_t*` `rohBeam::rohBeam::noiseModelExport` [private]

4.407.3.21 propExport `MHA_AC::spectrum_t*` `rohBeam::rohBeam::propExport` [private]

The documentation for this class was generated from the following file:

- **rohBeam.hh**

4.408 rohBeam::rohConfig Class Reference

Public Member Functions

- **rohConfig** (const **mhaconfig_t** **in_cfg**, const **mhaconfig_t** **out_cfg**, std::unique_ptr< Eigen::MatrixXcf > **headModel_**, std::unique_ptr< **MHASignal::matrix_t** > **beamW_**, std::unique_ptr< **MHASignal::matrix_t** > **delayComp_**, const **configOptions** &**options**)
- **rohConfig** (**rohConfig** ***lastConfig**, const **mhaconfig_t** **const mhaconfig_t** **out_cfg**, std::unique_ptr< Eigen::MatrixXcf > **headModel_**, std::unique_ptr< **MHASignal::matrix_t** > **beamW_**, std::unique_ptr< **MHASignal::matrix_t** > **delayComp_**, const **configOptions** &**options**)
- **~rohConfig** ()
- **rohConfig** (const **rohConfig** &)=**delete**
- **rohConfig** & **operator=** (const **rohConfig** &)=**delete**
- **mha_spec_t** * **process** (**mha_spec_t** *)
- void **init_dynamic** ()

Private Member Functions

- void **phasereconstruction** (**MHASignal::spectrum_t** *)
- void **postfilter** (**mha_spec_t** *, **MHASignal::spectrum_t** *)
- void **copyfixedbfoutput** (**MHASignal::spectrum_t** *)

Private Attributes

- int **nfreq**
- int **nchan_block**
- **mhaconfig_t** **in_cfg**
- **mhaconfig_t** **out_cfg**
- bool **enable_adaptive_beam**
- int **binaural_type_index**
- std::unique_ptr< Eigen::MatrixXcf > **headModel**
- std::unique_ptr< **MHASignal::matrix_t** > **beamW**
- std::unique_ptr< **MHASignal::matrix_t** > **delayComp**
- **MHASignal::spectrum_t** * **beam1**
- **MHASignal::spectrum_t** * **beamA**
- **MHASignal::spectrum_t** * **blockSpec**
- **MHASignal::spectrum_t** * **outSpec**
- float **alpha_postfilter**
- float **alpha_blocking_XkXi**
- float **alpha_blocking_XkY**
- std::vector< Eigen::MatrixXcf > **corrXpXp**
- std::vector< Eigen::VectorXcf > **corrXpYf**
- Eigen::VectorXf **corrZZ**
- Eigen::VectorXf **corrLL**

- Eigen::VectorXf **corrRR**
- Eigen::HouseholderQR< Eigen::MatrixXcf > **hhCorrXpXp**
- Eigen::VectorXcf **nextXpYf**
- Eigen::VectorXcf **blockXp**
- Eigen::VectorXcf **freqResp**
- Eigen::ArrayXf **magResp**
- float **minLim**
- float **maxLim**

4.408.1 Constructor & Destructor Documentation

4.408.1.1 rohConfig() [1/3] rohBeam::rohConfig::rohConfig (

```
    const mhaconfig_t in_cfg,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

4.408.1.2 rohConfig() [2/3] rohBeam::rohConfig::rohConfig (

```
    rohConfig * lastConfig,
    const mhaconfig_t,
    const mhaconfig_t out_cfg,
    std::unique_ptr< Eigen::MatrixXcf > headModel_,
    std::unique_ptr< MHASignal::matrix_t > beamW_,
    std::unique_ptr< MHASignal::matrix_t > delayComp_,
    const configOptions & options )
```

4.408.1.3 ~rohConfig() rohBeam::rohConfig::~rohConfig ()

4.408.1.4 rohConfig() [3/3] rohBeam::rohConfig::rohConfig (

```
    const rohConfig & ) [delete]
```

4.408.2 Member Function Documentation

4.408.2.1 operator=() `rohConfig& rohBeam::rohConfig::operator= (const rohConfig &) [delete]`

4.408.2.2 process() `mha_spec_t * rohBeam::rohConfig::process (mha_spec_t * inSpec)`

4.408.2.3 init_dynamic() `void rohBeam::rohConfig::init_dynamic ()`

4.408.2.4 phasereconstruction() `void rohBeam::rohConfig::phasereconstruction (MHASignal::spectrum_t * prevSpecPost) [private]`

4.408.2.5 postfilter() `void rohBeam::rohConfig::postfilter (mha_spec_t * inSpec, MHASignal::spectrum_t * prevSpecPost) [private]`

4.408.2.6 copyfixedbfoutput() `void rohBeam::rohConfig::copyfixedbfoutput (MHASignal::spectrum_t * prevSpecPost) [private]`

4.408.3 Member Data Documentation

4.408.3.1 nfreq int rohBeam::rohConfig::nfreq [private]

4.408.3.2 nchan_block int rohBeam::rohConfig::nchan_block [private]

4.408.3.3 in_cfg mhaconfig_t rohBeam::rohConfig::in_cfg [private]

4.408.3.4 out_cfg mhaconfig_t rohBeam::rohConfig::out_cfg [private]

4.408.3.5 enable_adaptive_beam bool rohBeam::rohConfig::enable_adaptive_beam [private]

4.408.3.6 binaural_type_index int rohBeam::rohConfig::binaural_type_index [private]

4.408.3.7 headModel std::unique_ptr<Eigen::MatrixXcf> rohBeam::rohConfig::headModel [private]

4.408.3.8 beamW std::unique_ptr< MHASignal::matrix_t > rohBeam::rohConfig::beamW [private]

4.408.3.9 delayComp std::unique_ptr< MHASignal::matrix_t > rohBeam::rohConfig::delayComp [private]

4.408.3.10 beam1 `MHASignal::spectrum_t*` `rohBeam::rohConfig::beam1` [private]

4.408.3.11 beamA `MHASignal::spectrum_t*` `rohBeam::rohConfig::beamA` [private]

4.408.3.12 blockSpec `MHASignal::spectrum_t*` `rohBeam::rohConfig::blockSpec` [private]

4.408.3.13 outSpec `MHASignal::spectrum_t*` `rohBeam::rohConfig::outSpec` [private]

4.408.3.14 alpha_postfilter `float` `rohBeam::rohConfig::alpha_postfilter` [private]

4.408.3.15 alpha_blocking_XkXi `float` `rohBeam::rohConfig::alpha_blocking_XkXi` [private]

4.408.3.16 alpha_blocking_XkY `float` `rohBeam::rohConfig::alpha_blocking_XkY` [private]

4.408.3.17 corrXpXp `std::vector<Eigen::MatrixXcf>` `rohBeam::rohConfig::corrXpXp`
[private]

4.408.3.18 corrXpYf `std::vector<Eigen::VectorXcf>` `rohBeam::rohConfig::corrXpYf`
[private]

4.408.3.19 corrZZ Eigen::VectorXf rohBeam::rohConfig::corrZZ [private]

4.408.3.20 corrLL Eigen::VectorXf rohBeam::rohConfig::corrLL [private]

4.408.3.21 corrRR Eigen::VectorXf rohBeam::rohConfig::corrRR [private]

4.408.3.22 hhCorrXpXp Eigen::HouseholderQR<Eigen::MatrixXcf> rohBeam::rohConfig↔
::hhCorrXpXp [private]

4.408.3.23 nextXpYf Eigen::VectorXcf rohBeam::rohConfig::nextXpYf [private]

4.408.3.24 blockXp Eigen::VectorXcf rohBeam::rohConfig::blockXp [private]

4.408.3.25 freqResp Eigen::VectorXcf rohBeam::rohConfig::freqResp [private]

4.408.3.26 magResp Eigen::ArrayXf rohBeam::rohConfig::magResp [private]

4.408.3.27 minLim float rohBeam::rohConfig::minLim [private]

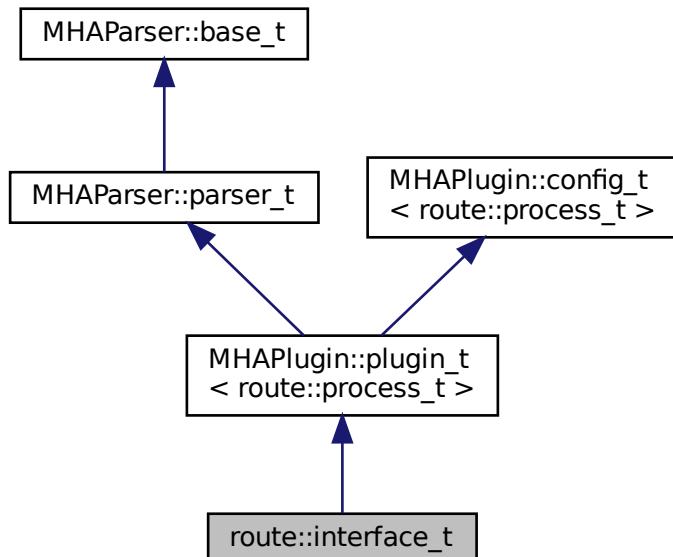
4.408.3.28 **maxLim** float rohBeam::rohConfig::maxLim [private]

The documentation for this class was generated from the following files:

- **rohBeam.hh**
- **rohBeam.cpp**

4.409 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

4.409.1 Constructor & Destructor Documentation

```
4.409.1.1 interface_t() route::interface_t::interface_t (
    algo_comm_t iac,
    const std::string & ,
    const std::string & ialg )
```

4.409.2 Member Function Documentation

```
4.409.2.1 prepare() void route::interface_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements `MHAPlugIn::plugin_t< route::process_t >` (p. [1148](#)).

```
4.409.2.2 release() void route::interface_t::release ( ) [virtual]
```

Reimplemented from `MHAPlugIn::plugin_t< route::process_t >` (p. [1149](#)).

4.409.2.3 process() [1/2] `mha_wave_t * route::interface_t::process (mha_wave_t * s)`

4.409.2.4 process() [2/2] `mha_spec_t * route::interface_t::process (mha_spec_t * s)`

4.409.2.5 update() `void route::interface_t::update () [private]`

4.409.3 Member Data Documentation

4.409.3.1 patchbay `MHAEVENTS::patchbay_t< route::interface_t> route::interface_t::patchbay [private]`

4.409.3.2 route_out `MHAPARSER::vstring_t route::interface_t::route_out [private]`

4.409.3.3 route_ac `MHAPARSER::vstring_t route::interface_t::route_ac [private]`

4.409.3.4 cfin `mhaconfig_t route::interface_t::cfin [private]`

4.409.3.5 cfout `mhaconfig_t route::interface_t::cfout [private]`

4.409.3.6 cfac `mhaconfig_t` `route::interface_t::cfac` [private]

4.409.3.7 prepared `bool` `route::interface_t::prepared` [private]

4.409.3.8 stopped `bool` `route::interface_t::stopped` [private]

4.409.3.9 algo `std::string` `route::interface_t::algo` [private]

The documentation for this class was generated from the following file:

- `route.cpp`

4.410 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`

4.410.1 Constructor & Destructor Documentation

```
4.410.1.1 process_t() 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 )
```

4.410.2 Member Function Documentation

```
4.410.2.1 process() [1/2] mha_wave_t * route::process_t::process (
    mha_wave_t * s )
```

```
4.410.2.2 process() [2/2] mha_spec_t * route::process_t::process (
    mha_spec_t * s )
```

4.410.3 Member Data Documentation

```
4.410.3.1 wout MHAMultiSrc::waveform_t route::process_t::wout [private]
```

```
4.410.3.2 sout MHAMultiSrc::spectrum_t route::process_t::sout [private]
```

```
4.410.3.3 wout_ac MHAMultiSrc::waveform_t route::process_t::wout_ac [private]
```

4.410.3.4 sout_ac `MHAMultiSrc::spectrum_t route::process_t::sout_ac` [private]

The documentation for this class was generated from the following file:

- `route.cpp`

4.411 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 Ufilt`
Input signal cache (second filter)
- `MHASignal::waveform_t Pu`
Power of input signal delayline.
- `MHASignal::waveform_t fu`
Filtered input signal.
- `MHASignal::waveform_t fulft`
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`

4.411.1 Constructor & Destructor Documentation

4.411.1.1 `rt_nlms_t()` `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)`

4.411.1.2 `~rt_nlms_t()` `rt_nlms_t::~rt_nlms_t () [inline]`

4.411.2 Member Function Documentation

4.411.2.1 `process()` `mha_wave_t * rt_nlms_t::process (`
 `mha_wave_t * sUD,`
 `mha_real_t rho,`
 `mha_real_t c,`
 `unsigned int norm_type,`
 `unsigned int estim_type,`
 `mha_real_t lambda_smooth)`

4.411.2.2 `insert()` `void rt_nlms_t::insert ()`

4.411.3 Member Data Documentation

4.411.3.1 ac algo_comm_t rt_nlms_t::ac [private]

4.411.3.2 ntaps unsigned int rt_nlms_t::ntaps [private]

4.411.3.3 frames unsigned int rt_nlms_t::frames [private]

4.411.3.4 channels unsigned int rt_nlms_t::channels [private]

4.411.3.5 F MHA_AC::waveform_t rt_nlms_t::F [private]

4.411.3.6 U MHASignal::waveform_t rt_nlms_t::U [private]

Input signal cache.

4.411.3.7 Uflt MHASignal::waveform_t rt_nlms_t::Uflt [private]

Input signal cache (second filter)

4.411.3.8 Pu MHASignal::waveform_t rt_nlms_t::Pu [private]

Power of input signal delayline.

4.411.3.9 fu `MHASignal::waveform_t rt_nlms_t::fu` [private]

Filtered input signal.

4.411.3.10 fult `MHASignal::waveform_t rt_nlms_t::fult` [private]

Filtered input signal.

4.411.3.11 fu_previous `MHASignal::waveform_t rt_nlms_t::fu_previous` [private]**4.411.3.12 y_previous** `MHASignal::waveform_t rt_nlms_t::y_previous` [private]**4.411.3.13 P_Sum** `MHASignal::waveform_t rt_nlms_t::P_Sum` [private]**4.411.3.14 name_u_** `std::string rt_nlms_t::name_u_` [private]**4.411.3.15 name_d_** `std::string rt_nlms_t::name_d_` [private]**4.411.3.16 name_e_** `std::string rt_nlms_t::name_e_` [private]

4.411.3.17 n_no_update_ int rt_nlms_t::n_no_update_ [private]

4.411.3.18 no_iter int rt_nlms_t::no_iter [private]

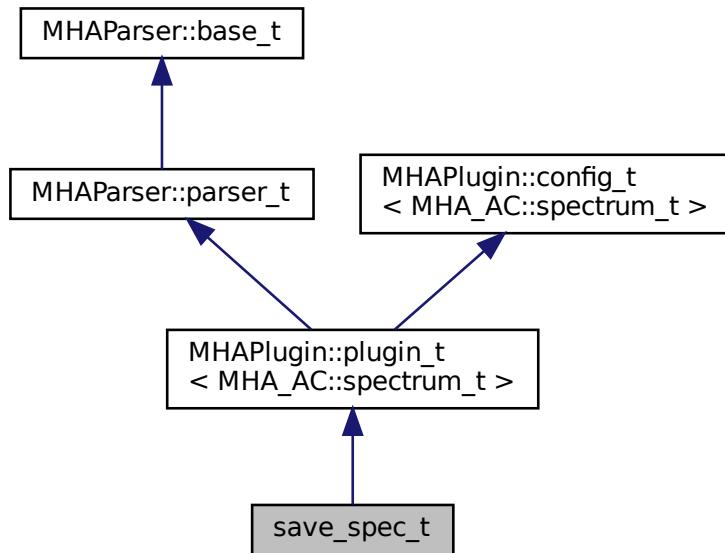
4.411.3.19 s_E mha_wave_t rt_nlms_t::s_E [private]

The documentation for this class was generated from the following file:

- **nlms_wave.cpp**

4.412 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

4.412.1 Constructor & Destructor Documentation

```
4.412.1.1 save_spec_t() save_spec_t::save_spec_t (
    const algo_comm_t & iac,
    const std::string & ith,
    const std::string & ial ) [inline]
```

4.412.2 Member Function Documentation

```
4.412.2.1 process() mha_spec_t* save_spec_t::process (
    mha_spec_t * s ) [inline]
```

```
4.412.2.2 prepare() void save_spec_t::prepare (
    mhaconfig_t & tf ) [inline], [virtual]
```

Implements **MHAPlugin::plugin_t< MHA_AC::spectrum_t >** (p. 1148).

4.412.3 Member Data Documentation

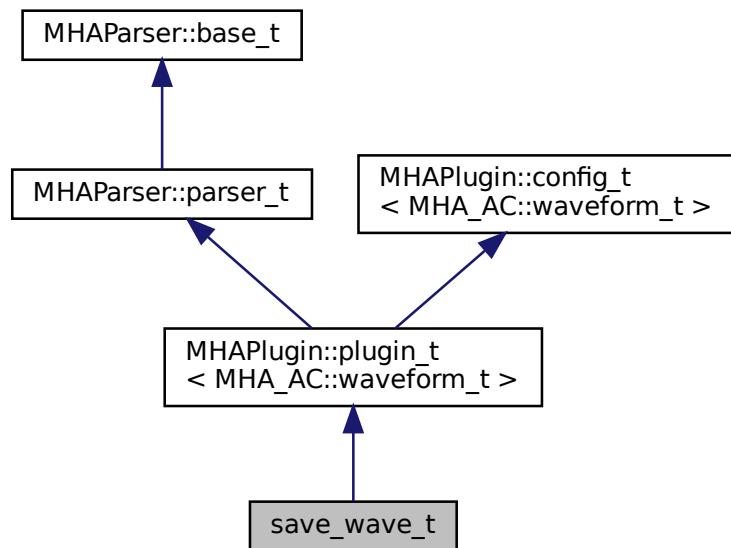
```
4.412.3.1 basename std::string save_spec_t::basename [private]
```

The documentation for this class was generated from the following file:

- **save_spec.cpp**

4.413 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

4.413.1 Constructor & Destructor Documentation

4.413.1.1 `save_wave_t()` `save_wave_t::save_wave_t (`
 `const algo_comm_t & iac,`
 `const std::string & ith,`
 `const std::string & ial) [inline]`

4.413.2 Member Function Documentation

4.413.2.1 `process()` `mha_wave_t* save_wave_t::process (`
 `mha_wave_t * s) [inline]`

4.413.2.2 `prepare()` `void save_wave_t::prepare (`
 `mhaconfig_t & tf) [inline], [virtual]`

Implements `MHAPlugin::plugin_t< MHA_AC::waveform_t >` (p. [1148](#)).

4.413.3 Member Data Documentation

4.413.3.1 `basename` `std::string save_wave_t::basename [private]`

The documentation for this class was generated from the following file:

- `save_wave.cpp`

4.414 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`

4.414.1 Constructor & Destructor Documentation

4.414.1.1 cfg_t() `cfg_t::cfg_t (`
 `int nfft,`
 `int inch,`
 `int outch,`
 `algo_comm_t ac,`
 `std::string name)`

4.414.2 Member Function Documentation

4.414.2.1 process() `mha_spec_t * cfg_t::process (`
 `mha_spec_t * s)`

4.414.3 Member Data Documentation

4.414.3.1 in_spec_copy `MHA_AC::spectrum_t shadowfilter_begin::cfg_t::in_spec_`
`copy [private]`

4.414.3.2 out_spec `MHASignal::spectrum_t shadowfilter_begin::cfg_t::out_spec`
`[private]`

4.414.3.3 nch `MHA_AC::int_t shadowfilter_begin::cfg_t::nch [private]`

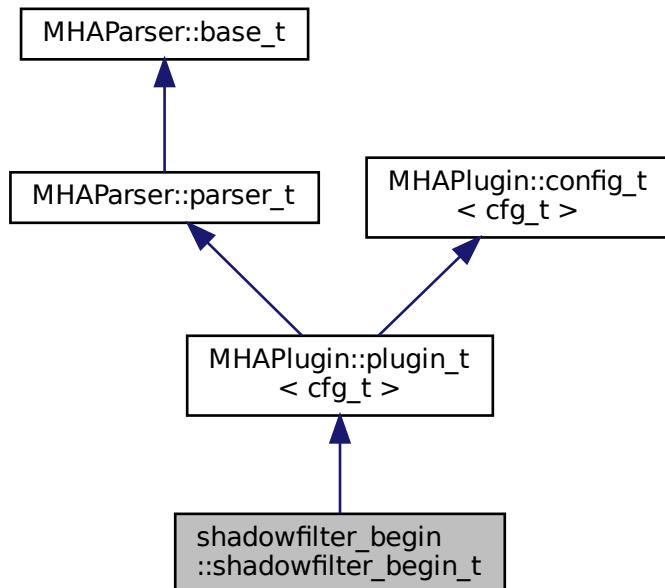
4.414.3.4 ntracks `MHA_AC::int_t shadowfilter_begin::cfg_t::ntracks [private]`

The documentation for this class was generated from the following file:

- `shadowfilter_begin.cpp`

4.415 `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**
- MHParse::int_t **nch**
- MHParse::int_t **ntracks**

Additional Inherited Members

4.415.1 Constructor & Destructor Documentation

```
4.415.1.1 shadowfilter_begin_t() shadowfilter_begin::shadowfilter_begin_t::shadowfilter_begin_t (
    const algo_comm_t & iac,
    const std::string & ith,
    const std::string & ial )
```

4.415.2 Member Function Documentation

```
4.415.2.1 process() mha_spec_t * shadowfilter_begin::shadowfilter_begin_t::process (
(           mha_spec_t * s )
```

```
4.415.2.2 prepare() void shadowfilter_begin::shadowfilter_begin_t::prepare (
(           mhaconfig_t & tf ) [virtual]
```

Implements **MHAPlugin::plugin_t< cfg_t >** (p. 1148).

4.415.3 Member Data Documentation

4.415.3.1 basename std::string shadowfilter_begin::shadowfilter_begin_t::basename
[private]

4.415.3.2 nch MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::nch [private]

4.415.3.3 ntracks MHAParser::int_t shadowfilter_begin::shadowfilter_begin_t::ntracks [private]

The documentation for this class was generated from the following file:

- `shadowfilter_begin.cpp`

4.416 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`

4.416.1 Constructor & Destructor Documentation

```
4.416.1.1 cfg_t() cfg_t::cfg_t (
    int nfft_,
    algo_comm_t ac_,
    std::string name_ )
```

4.416.2 Member Function Documentation

```
4.416.2.1 process() mha_spec_t * cfg_t::process (
    mha_spec_t * s )
```

4.416.3 Member Data Documentation

4.416.3.1 ac algo_comm_t shadowfilter_end::cfg_t::ac [private]

4.416.3.2 name std::string shadowfilter_end::cfg_t::name [private]

4.416.3.3 nfft int shadowfilter_end::cfg_t::nfft [private]

4.416.3.4 ntracks int shadowfilter_end::cfg_t::ntracks [private]

4.416.3.5 nch_out int shadowfilter_end::cfg_t::nch_out [private]

4.416.3.6 in_spec `mha_spec_t shadowfilter_end::cfg_t::in_spec [private]`

4.416.3.7 out_spec `MHASignal::spectrum_t shadowfilter_end::cfg_t::out_spec` [private]

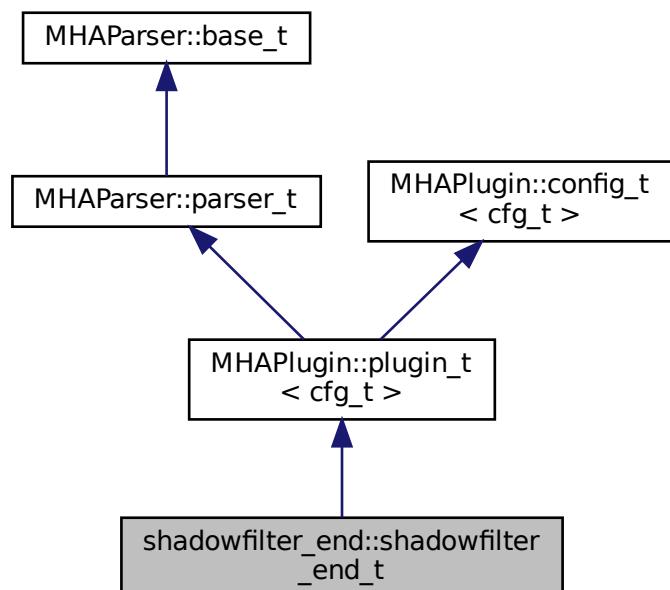
4.416.3.8 gains `MHA_AC::spectrum_t shadowfilter_end::cfg_t::gains` [private]

The documentation for this class was generated from the following file:

- shadowfilter_end.cpp

4.417 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

4.417.1 Constructor & Destructor Documentation

4.417.1.1 shadowfilter_end_t() shadowfilter_end::shadowfilter_end_t::shadowfilter_end_t (const algo_comm_t & iac, const std::string & ith, const std::string & ial)

4.417.2 Member Function Documentation

4.417.2.1 process() mha_spec_t * shadowfilter_end::shadowfilter_end_t::process (mha_spec_t * s)

4.417.2.2 prepare() void shadowfilter_end::shadowfilter_end_t::prepare (mhaconfig_t & tf) [virtual]

Implements `MHAPlugin::plugin_t< cfg_t >` (p. 1148).

4.417.3 Member Data Documentation

4.417.3.1 basename `MHAParser::string_t shadowfilter_end::shadowfilter_end_t::basename` [private]

The documentation for this class was generated from the following file:

- `shadowfilter_end.cpp`

4.418 sine_cfg_t Struct Reference

Runtime configuration of the sine plugin.

Public Member Functions

- `sine_cfg_t` (double sampling_rate, `mha_real_t` frequency, `mha_real_t` newlev, int _mix, const std::vector< int > &_channels)

Constructor computes data members from input parameters.

Public Attributes

- double `phase_increment_div_2pi`
Phase increment per sample, divided by 2 pi for easier phase wrapping.
- double `amplitude`
Amplitude of the sinusoid in Pascal.
- int `mix`
0 for mode replace, 1 for mode mix. Used as factor on input signal.
- const std::vector< int > `channels`
Indices of affected audio channels.

4.418.1 Detailed Description

Runtime configuration of the sine plugin.

4.418.2 Constructor & Destructor Documentation

4.418.2.1 sine_cfg_t() `sine_cfg_t::sine_cfg_t (`
 `double sampling_rate,`
 `mha_real_t frequency,`
 `mha_real_t newlev,`
 `int _mix,`
 `const std::vector< int > & _channels) [inline]`

Constructor computes data members from input parameters.

4.418.3 Member Data Documentation

4.418.3.1 phase_increment_div_2pi `double sine_cfg_t::phase_increment_div_2pi`

Phase increment per sample, divided by 2 pi for easier phase wrapping.

4.418.3.2 amplitude `double sine_cfg_t::amplitude`

Amplitude of the sinusoid in Pascal.

4.418.3.3 mix `int sine_cfg_t::mix`

0 for mode replace, 1 for mode mix. Used as factor on input signal.

4.418.3.4 channels `const std::vector<int> sine_cfg_t::channels`

Indices of affected audio channels.

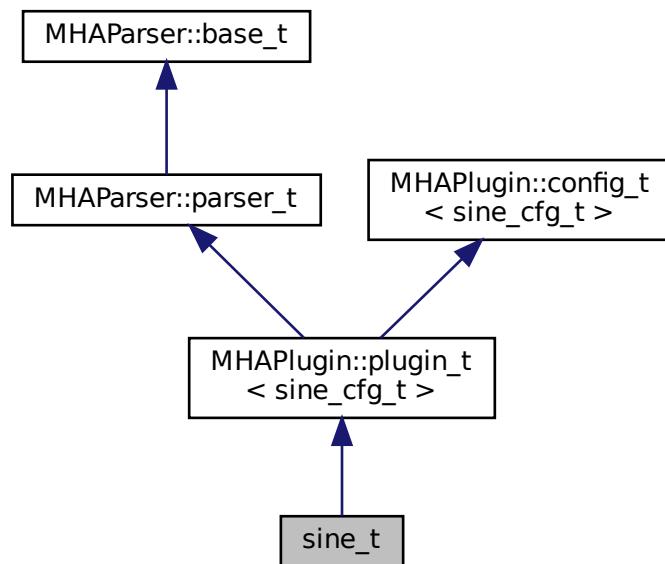
The documentation for this struct was generated from the following file:

- `sine.cpp`

4.419 sine_t Class Reference

Interface class of plugin `sine`, a sinusoid generator plugin.

Inheritance diagram for `sine_t`:



Public Member Functions

- **`sine_t` (const `algo_comm_t` &, const `std::string` &`chain_name`, const `std::string` &`algo_name`)**
Constructor initializes and connects configuration variables.
- **`mha_wave_t * process (mha_wave_t *)`**
Computes sinusoid and mixes/replaces input signal.
- **`void prepare (mhaconfig_t &)`**
Adapts range of channel variable and prepares.

Private Member Functions

- **`void update_cfg ()`**
Computes new runtime configuration.

Private Attributes

- `MHAParser::float_t lev`
- `MHAParser::float_t frequency`
- `MHAParser::kw_t mode`
- `MHAParser::vint_t channels`
- double `phase_div_2pi`
- `MHAEvents::patchbay_t< sine_t > patchbay`

Additional Inherited Members

4.419.1 Detailed Description

Interface class of plugin `sine`, a sinusoid generator plugin.

4.419.2 Constructor & Destructor Documentation

```
4.419.2.1 sine_t() sine_t::sine_t (
    const algo_comm_t & iac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructor initializes and connects configuration variables.

4.419.3 Member Function Documentation

```
4.419.3.1 process() mha_wave_t * sine_t::process (
    mha_wave_t * s )
```

Computes sinusoid and mixes/replaces input signal.

If the amplitude has changed since the last process callback, spread out the amplitude change linearly across all samples of the buffer to avoid clicks.

4.419.3.2 `prepare()` `void sine_t::prepare (mhaconfig_t & tf) [virtual]`

Adapts range of channel variable and prepares.

Implements `MHAPlugIn::plugin_t< sine_cfg_t >` (p. 1148).

4.419.3.3 `update_cfg()` `void sine_t::update_cfg () [private]`

Computes new runtime configuration.

4.419.4 Member Data Documentation

4.419.4.1 `lev` `MHAParser::float_t sine_t::lev [private]`

4.419.4.2 `frequency` `MHAParser::float_t sine_t::frequency [private]`

4.419.4.3 `mode` `MHAParser::kw_t sine_t::mode [private]`

4.419.4.4 `channels` `MHAParser::vint_t sine_t::channels [private]`

4.419.4.5 `phase_div_2pi` `double sine_t::phase_div_2pi [private]`

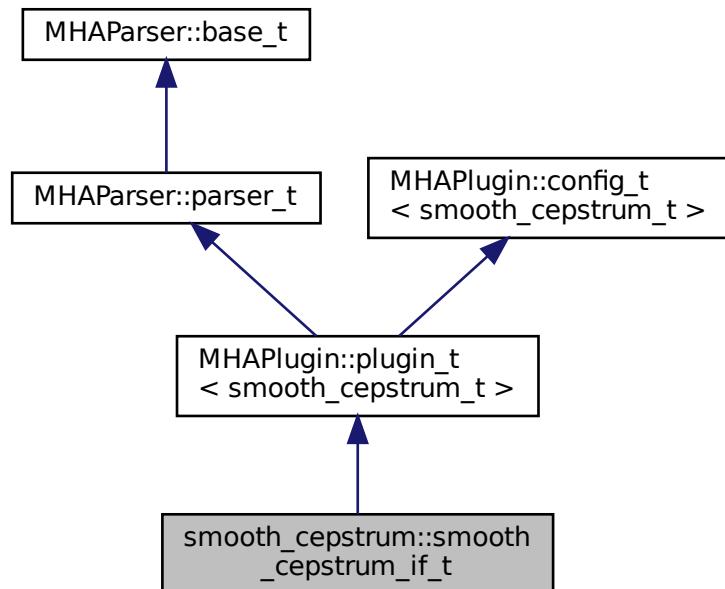
4.419.4.6 patchbay `MHAEEvents::patchbay_t< sine_t> sine_t::patchbay` [private]

The documentation for this class was generated from the following file:

- `sine.cpp`

4.420 smooth_cepstrum::smooth_cepstrum_if_t Class Reference

Inheritance diagram for `smooth_cepstrum::smooth_cepstrum_if_t`:



Public Member Functions

- **smooth_cepstrum_if_t (algo_comm_t & ac, const std::string &chain_name, const std::string &algo_name)**
Constructs the beamforming plugin.
- **mha_spec_t * process (mha_spec_t *)**
This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.
- **void prepare (mhaconfig_t &)**
Plugin preparation.
- **void release (void)**

Private Member Functions

- void `update_cfg ()`
- void `on_model_param_valuechanged ()`

Private Attributes

- `MHAParser::float_t xi_min_db`
- `MHAParser::float_t f0_low`
- `MHAParser::float_t f0_high`
- `MHAParser::float_t delta_pitch`
- `MHAParser::float_t lambda_thresh`
- `MHAParser::float_t alpha_pitch`
- `MHAParser::float_t beta_const`
- `MHAParser::float_t kappa_const`
- `MHAParser::float_t gain_min_db`
- `MHAParser::vfloat_t win_f0`
- `MHAParser::vfloat_t alpha_const_vals`
- `MHAParser::vfloat_t alpha_const_limits_hz`
- `MHAParser::string_t noisePow_name`
- `MHAParser::parser_t spp`
- `MHAParser::float_t prior_q`
- `MHAParser::float_t xi_opt_db`
- `MHAEvents::patchbay_t< smooth_cepstrum_if_t > patchbay`
- bool `prepared`

Additional Inherited Members

4.420.1 Constructor & Destructor Documentation

```
4.420.1.1 smooth_cepstrum_if_t() smooth_cepstrum::smooth_cepstrum_if_t::smooth_cepstrum_if_t (
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs the beamforming plugin.

4.420.2 Member Function Documentation

```
4.420.2.1 process() mha_spec_t * smooth_cepstrum::smooth_cepstrum_if_t::process (
    mha_spec_t * signal )
```

This plugin implements noise reduction using spectral subtraction: by nonnegative subtraction from the output magnitude of the estimated noise magnitude spectrum.

4.420 smooth_cepstrum::smooth_cepstrum_if_t Class Reference

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.

4.420.2.2 prepare() `void smooth_cepstrum::smooth_cepstrum_if_t::prepare (mhaconfig_t & signal_info) [virtual]`

Plugin preparation.

This plugin checks that the input signal has the spectral domain and contains at least one channel

Parameters

<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< smooth_cepstrum_t >` (p. [1148](#)).

4.420.2.3 release() `void smooth_cepstrum::smooth_cepstrum_if_t::release (void) [inline], [virtual]`

Reimplemented from `MHAPlugin::plugin_t< smooth_cepstrum_t >` (p. [1149](#)).

4.420.2.4 update_cfg() `void smooth_cepstrum::smooth_cepstrum_if_t::update_cfg () [private]`

4.420.2.5 on_model_param_valuechanged() `void smooth_cepstrum::smooth_cepstrum_if_t::on_model_param_valuechanged () [private]`

4.420.3 Member Data Documentation

4.420.3.1 xi_min_db `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::xi_min_db` [private]

4.420.3.2 f0_low `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::f0_low` [private]

4.420.3.3 f0_high `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::f0_high` [private]

4.420.3.4 delta_pitch `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::delta_pitch` [private]

4.420.3.5 lambda_thresh `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::lambda_thresh` [private]

4.420.3.6 alpha_pitch `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::alpha_pitch` [private]

4.420.3.7 beta_const `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::beta_const` [private]

4.420.3.8 kappa_const `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::kappa_const` [private]

4.420.3.9 gain_min_db `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::gain_min_db` [private]

4.420.3.10 win_f0 `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::win_f0` [private]

4.420.3.11 alpha_const_vals `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::alpha_const_vals` [private]

4.420.3.12 alpha_const_limits_hz `MHAParser::vfloat_t` `smooth_cepstrum::smooth_cepstrum_if_t::alpha_const_limits_hz` [private]

4.420.3.13 noisePow_name `MHAParser::string_t` `smooth_cepstrum::smooth_cepstrum_if_t::noisePow_name` [private]

4.420.3.14 spp `MHAParser::parser_t` `smooth_cepstrum::smooth_cepstrum_if_t::spp` [private]

4.420.3.15 prior_q `MHAParser::float_t` `smooth_cepstrum::smooth_cepstrum_if_t::prior_q` [private]

4.420.3.16 xi_opt_db `MHAParser::float_t smooth_cepstrum::smooth_cepstrum_if_t::xi_opt_db` [private]

4.420.3.17 patchbay `MHAEvents::patchbay_t< smooth_cepstrum_if_t> smooth_cepstrum::smooth_cepstrum_if_t::patchbay` [private]

4.420.3.18 prepared `bool smooth_cepstrum::smooth_cepstrum_if_t::prepared` [private]

The documentation for this class was generated from the following files:

- **smooth_cepstrum.hh**
- **smooth_cepstrum.cpp**

4.421 smooth_cepstrum::smooth_cepstrum_t Class Reference

Public Member Functions

- **smooth_cepstrum_t (algo_comm_t & ac, smooth_params & params)**
- **smooth_cepstrum_t (const smooth_cepstrum_t &)=delete**
- **smooth_cepstrum_t & operator= (const smooth_cepstrum_t &)=delete**
- **~smooth_cepstrum_t ()**
- **mha_spec_t * process (mha_spec_t *)**

Private Attributes

- **algo_comm_t ac**
- **smooth_params params**
- **unsigned int fftlen**
- **mha_fft_t mha_fft**
- **unsigned int nfreq**
- **unsigned int nchan**
- **float ola_powspec_scale**
- **float q_low**
- **float q_high**
- **MHASignal::waveform_t winF0**
- **float xi_min**
- **float gain_min**
- **MHASignal::waveform_t alpha_const**

- `MHASignal::waveform_t alpha_prev`
- `MHASignal::waveform_t noisePow`
- `MHASignal::waveform_t powSpec`
- `MHASignal::waveform_t gamma_post`
- `MHASignal::waveform_t xi_ml`
- `MHASignal::spectrum_t lambda_ml_full`
- `MHASignal::spectrum_t lambda_ml_ceps`
- `MHASignal::waveform_t lambda_ml_smooth`
- `MHASignal::waveform_t alpha_hat`
- `MHASignal::waveform_t alpha_frame`
- `MHASignal::spectrum_t lambda_ceps`
- `MHASignal::waveform_t lambda_ceps_prev`
- `MHASignal::spectrum_t log_lambda_spec`
- `MHASignal::waveform_t lambda_spec`
- `MHASignal::waveform_t xi_est`
- `MHASignal::waveform_t gain_wiener`
- `MHASignal::spectrum_t spec_out`
- `double * max_val`
- `int * max_q`
- `int * pitch_set_first`
- `int * pitch_set_last`
- `float priorFact`
- `float xiOpt`
- `float logGLRFact`
- `float GLRexp`
- `MHASignal::waveform_t GLR`

4.421.1 Constructor & Destructor Documentation

4.421.1.1 `smooth_cepstrum_t()` [1/2] `smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (`
`algo_comm_t & ac,`
`smooth_params & params)`

4.421.1.2 `smooth_cepstrum_t()` [2/2] `smooth_cepstrum::smooth_cepstrum_t::smooth_cepstrum_t (`
`const smooth_cepstrum_t &) [delete]`

4.421.1.3 ~smooth_cepstrum_t() `smooth_cepstrum::smooth_cepstrum_t::~smooth_cepstrum_t ()`

4.421.2 Member Function Documentation

4.421.2.1 operator=() `smooth_cepstrum_t& smooth_cepstrum::smooth_cepstrum_t::operator= (const smooth_cepstrum_t &) [delete]`

4.421.2.2 process() `mha_spec_t * smooth_cepstrum::smooth_cepstrum_t::process (mha_spec_t * noisyFrame)`

4.421.3 Member Data Documentation

4.421.3.1 ac `algo_comm_t smooth_cepstrum::smooth_cepstrum_t::ac [private]`

4.421.3.2 params `smooth_params smooth_cepstrum::smooth_cepstrum_t::params [private]`

4.421.3.3 fftlen `unsigned int smooth_cepstrum::smooth_cepstrum_t::ffflen [private]`

4.421.3.4 mha_fft `mha_fft_t smooth_cepstrum::smooth_cepstrum_t::mha_fft [private]`

4.421 smooth_cepstrum::smooth_cepstrum_t Class Reference 1441

4.421.3.5 nfreq unsigned int smooth_cepstrum::smooth_cepstrum_t::nfreq [private]

4.421.3.6 nchan unsigned int smooth_cepstrum::smooth_cepstrum_t::nchan [private]

4.421.3.7 ola_powspec_scale float smooth_cepstrum::smooth_cepstrum_t::ola_powspec←
_scale [private]

4.421.3.8 q_low float smooth_cepstrum::smooth_cepstrum_t::q_low [private]

4.421.3.9 q_high float smooth_cepstrum::smooth_cepstrum_t::q_high [private]

4.421.3.10 winF0 MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::winF0
[private]

4.421.3.11 xi_min float smooth_cepstrum::smooth_cepstrum_t::xi_min [private]

4.421.3.12 gain_min float smooth_cepstrum::smooth_cepstrum_t::gain_min [private]

4.421.3.13 alpha_const MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t←
::alpha_const [private]

4.421.3.14 alpha_prev `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::alpha_prev` [private]

4.421.3.15 noisePow `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::noisePow` [private]

4.421.3.16 powSpec `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::powSpec` [private]

4.421.3.17 gamma_post `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::gamma_post` [private]

4.421.3.18 xi_ml `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::xi_ml` [private]

4.421.3.19 lambda_ml_full `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_full` [private]

4.421.3.20 lambda_ml_ceps `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_ceps` [private]

4.421.3.21 lambda_ml_smooth `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ml_smooth` [private]

4.421 smooth_cepstrum::smooth_cepstrum_t Class Reference 1443

4.421.3.22 alpha_hat `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::alpha_hat` [private]

4.421.3.23 alpha_frame `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::alpha_frame` [private]

4.421.3.24 lambda_ceps `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ceps` [private]

4.421.3.25 lambda_ceps_prev `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_ceps_prev` [private]

4.421.3.26 log_lambda_spec `MHASignal::spectrum_t` `smooth_cepstrum::smooth_cepstrum_t::log_lambda_spec` [private]

4.421.3.27 lambda_spec `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::lambda_spec` [private]

4.421.3.28 xi_est `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::xi_est` [private]

4.421.3.29 gain_wiener `MHASignal::waveform_t` `smooth_cepstrum::smooth_cepstrum_t::gain_wiener` [private]

4.421.3.30 spec_out `MHASignal::spectrum_t smooth_cepstrum::smooth_cepstrum_t::spec_out` [private]

4.421.3.31 max_val `double* smooth_cepstrum::smooth_cepstrum_t::max_val` [private]

4.421.3.32 max_q `int* smooth_cepstrum::smooth_cepstrum_t::max_q` [private]

4.421.3.33 pitch_set_first `int* smooth_cepstrum::smooth_cepstrum_t::pitch_set_first` [private]

4.421.3.34 pitch_set_last `int* smooth_cepstrum::smooth_cepstrum_t::pitch_set_last` [private]

4.421.3.35 priorFact `float smooth_cepstrum::smooth_cepstrum_t::priorFact` [private]

4.421.3.36 xiOpt `float smooth_cepstrum::smooth_cepstrum_t::xiOpt` [private]

4.421.3.37 logGLRFact `float smooth_cepstrum::smooth_cepstrum_t::logGLRFact` [private]

4.421.3.38 GLRexp `float smooth_cepstrum::smooth_cepstrum_t::GLRexp` [private]

4.421.3.39 GLR `MHASignal::waveform_t smooth_cepstrum::smooth_cepstrum_t::GLR`
`[private]`

The documentation for this class was generated from the following files:

- `smooth_cepstrum.hh`
- `smooth_cepstrum.cpp`

4.422 smooth_cepstrum::smooth_params Class Reference

Public Member Functions

- `smooth_params (const mhaconfig_t &in_cfg, float xi_min_db, float f0_low, float f0_high, float delta_pitch, float lambda_thresh, float alpha_pitch, float beta_const, float kappa_const, float prior_q, float xi_opt_db, float gain_min_db, std::vector< float > &winF0, std::vector< float > &alpha_const_vals, std::vector< float > &alpha_consts_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_consts_hz`
- std::string `noisePow_name`

4.422.1 Constructor & Destructor Documentation

4.422.1.1 `smooth_params()` `smooth_cepstrum::smooth_params::smooth_params (`
 `const mhaconfig_t & _in_cfg,`
 `float _xi_min_db,`
 `float _f0_low,`
 `float _f0_high,`
 `float _delta_pitch,`
 `float _lambda_thresh,`
 `float _alpha_pitch,`
 `float _beta_const,`
 `float _kappa_const,`
 `float _prior_q,`
 `float _xi_opt_db,`
 `float _gain_min_db,`
 `std::vector< float > & _winF0,`
 `std::vector< float > & _alpha_const_vals,`
 `std::vector< float > & _alpha_const_limits_hz,`
 `std::string & _noisePow_name) [inline]`

4.422.2 Member Data Documentation

4.422.2.1 `in_cfg` `const mhaconfig_t smooth_cepstrum::smooth_params::in_cfg`

4.422.2.2 `xi_min_db` `float smooth_cepstrum::smooth_params::xi_min_db`

4.422.2.3 `f0_low` `float smooth_cepstrum::smooth_params::f0_low`

4.422.2.4 `f0_high` `float smooth_cepstrum::smooth_params::f0_high`

4.422.2.5 `delta_pitch` `float smooth_cepstrum::smooth_params::delta_pitch`

4.422.2.6 lambda_thresh float smooth_cepstrum::smooth_params::lambda_thresh

4.422.2.7 alpha_pitch float smooth_cepstrum::smooth_params::alpha_pitch

4.422.2.8 beta_const float smooth_cepstrum::smooth_params::beta_const

4.422.2.9 kappa_const float smooth_cepstrum::smooth_params::kappa_const

4.422.2.10 prior_q float smooth_cepstrum::smooth_params::prior_q

4.422.2.11 xi_opt_db float smooth_cepstrum::smooth_params::xi_opt_db

4.422.2.12 gain_min_db float smooth_cepstrum::smooth_params::gain_min_db

4.422.2.13 winF0 std::vector<float> smooth_cepstrum::smooth_params::winF0

4.422.2.14 alpha_const_vals std::vector<float> smooth_cepstrum::smooth_params::alpha_const_vals

4.422.2.15 alpha_const_limits_hz std::vector<float> smooth_cepstrum::smooth_<→
params::alpha_const_limits_hz

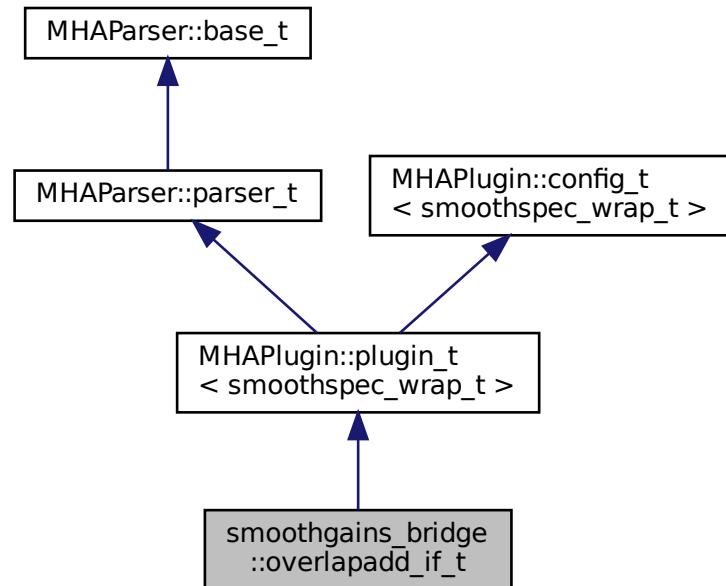
4.422.2.16 noisePow_name std::string smooth_cepstrum::smooth_params::noisePow_<→
name

The documentation for this class was generated from the following file:

- **smooth_cepstrum.hh**

4.423 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

4.423.1 Constructor & Destructor Documentation

4.423.1.1 overlapadd_if_t() smoothgains_bridge::overlapadd_if_t::overlapadd_if_t (

```
const algo_comm_t & iac,
const std::string & ,
const std::string & ialg )
```

4.423.1.2 ~overlapadd_if_t() smoothgains_bridge::overlapadd_if_t::~overlapadd_if_t ()

4.423.2 Member Function Documentation

4.423.2.1 prepare() void smoothgains_bridge::overlapadd_if_t::prepare (

```
mhaconfig_t & t ) [virtual]
```

Implements **MHAPlugin::plugin_t< smoothspec_wrap_t >** (p. 1148).

4.423.2.2 release() void smoothgains_bridge::overlapadd_if_t::release () [virtual]

Reimplemented from **MHAPlugIn::plugin_t<smoothspec_wrap_t>** (p. 1149).

4.423.2.3 process() mha_spec_t * smoothgains_bridge::overlapadd_if_t::process (mha_spec_t * spec)

4.423.2.4 update() void smoothgains_bridge::overlapadd_if_t::update () [private]

4.423.3 Member Data Documentation

4.423.3.1 patchbay MHAEvents::patchbay_t<overlapadd_if_t> smoothgains_bridge<::overlapadd_if_t::patchbay [private]

4.423.3.2 mode MHAParser::kw_t smoothgains_bridge::overlapadd_if_t::mode [private]

4.423.3.3 irswnd MHAParser::window_t smoothgains_bridge::overlapadd_if_t::irswnd [private]

4.423.3.4 epsilon MHAParser::float_t smoothgains_bridge::overlapadd_if_t::epsilon [private]

4.424 smoothgains_bridge::smoothspec_wrap_t Class Reference

4.423.3.5 plugloader `MHAParser::mhapluginloader_t` `smoothgains_bridge::overlapadd_if_t::plugloader` [private]

4.423.3.6 algo `std::string` `smoothgains_bridge::overlapadd_if_t::algo` [private]

4.423.3.7 cf_in `mhaconfig_t` `smoothgains_bridge::overlapadd_if_t::cf_in` [private]

4.423.3.8 cf_out `mhaconfig_t` `smoothgains_bridge::overlapadd_if_t::cf_out` [private]

The documentation for this class was generated from the following file:

- `smoothgains_bridge.cpp`

4.424 smoothgains_bridge::smoothspec_wrap_t Class Reference

Public Member Functions

- `smoothspec_wrap_t (mhaconfig_t spar_in, mhaconfig_t spar_out, const MHAParser::kw_t &mode, const MHAParser::window_t &irswnd, const MHAParser::float_t &epsilon)`
- `mha_spec_t * proc_1 (mha_spec_t *)`
- `mha_spec_t * proc_2 (mha_spec_t *)`

Private Attributes

- **MHASignal::spectrum_t spec_in_copy**
Copy of input spectrum for smoothspec.
- **MHAFilter::smoothspec_t smoothspec**
Smoothspec calculator.
- `bool use_smoothspec`
- `float smoothspec_epsilon`

4.424.1 Constructor & Destructor Documentation

4.424.1.1 smoothspec_wrap_t() `smoothgains_bridge::smoothspec_wrap_t::smoothspec_wrap_t (`

```
mhaconfig_t spar_in,
mhaconfig_t spar_out,
const MHAParser::kw_t & mode,
const MHAParser::window_t & irswnd,
const MHAParser::float_t & epsilon )
```

4.424.2 Member Function Documentation

4.424.2.1 proc_1() `mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_1 (`

```
mha_spec_t * s )
```

4.424.2.2 proc_2() `mha_spec_t * smoothgains_bridge::smoothspec_wrap_t::proc_2 (`

```
mha_spec_t * s )
```

4.424.3 Member Data Documentation

4.424.3.1 spec_in_copy `MHASignal::spectrum_t smoothgains_bridge::smoothspec_wrap_t::spec_in_copy [private]`

Copy of input spectrum for smoothspec.

4.424.3.2 smoothspec `MHAFilter::smoothspec_t smoothgains_bridge::smoothspec_wrap_t::smoothspec [private]`

Smoothspec calculator.

4.424.3.3 use_smoothspec bool smoothgains_bridge::smoothspec_wrap_t::use_smoothspec
[private]

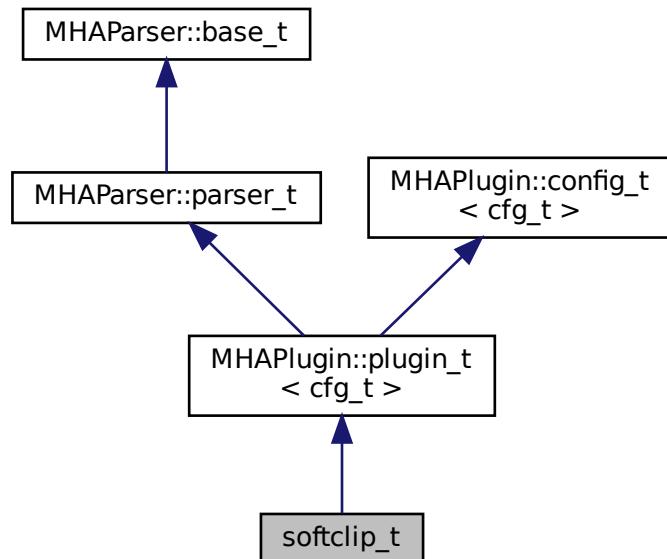
4.424.3.4 smoothspec_epsilon float smoothgains_bridge::smoothspec_wrap_t::smoothspec←
_epsilon [private]

The documentation for this class was generated from the following file:

- **smoothgains_bridge.cpp**

4.425 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

4.425.1 Constructor & Destructor Documentation

4.425.1.1 `softclip_t()` `softclip_t::softclip_t (`
 `const algo_comm_t & iac,`
 `const std::string & chain,`
 `const std::string & name)`

4.425.2 Member Function Documentation

4.425.2.1 `process()` `mha_wave_t * softclip_t::process (`
 `mha_wave_t * s)`

4.425.2.2 `prepare()` `void softclip_t::prepare (`
 `mhaconfig_t & tf) [virtual]`

Implements `MHAPlugin::plugin_t< cfg_t >` (p. 1148).

4.425.2.3 `update()` `void softclip_t::update ()`

4.425.3 Member Data Documentation

4.425.3.1 tftype `mhaconfig_t` `softclip_t::tftype` [private]

4.425.3.2 attack `MHAParser::float_t` `softclip_t::attack` [private]

4.425.3.3 decay `MHAParser::float_t` `softclip_t::decay` [private]

4.425.3.4 start_limit `MHAParser::float_t` `softclip_t::start_limit` [private]

4.425.3.5 slope_db `MHAParser::float_t` `softclip_t::slope_db` [private]

4.425.3.6 patchbay `MHAEvents::patchbay_t< softclip_t>` `softclip_t::patchbay` [private]

The documentation for this class was generated from the following file:

- `softclip.cpp`

4.426 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`

4.426.1 Constructor & Destructor Documentation

4.426.1.1 `softclipper_t()` `softclipper_t::softclipper_t (`
 `const softclipper_variables_t & v,`
 `const mhaconfig_t & tf)`

4.426.2 Member Function Documentation

4.426.2.1 `process()` `mha_real_t softclipper_t::process (`
 `mha_wave_t * s)`

4.426.3 Member Data Documentation

4.426.3.1 `attack` `MHAFilter::o1flt_lowpass_t softclipper_t::attack [private]`

4.426.3.2 `decay` `MHAFilter::o1flt_maxtrack_t softclipper_t::decay [private]`

4.426.3.3 clipmeter `MHAFilter::olflt_lowpass_t` `softclipper_t::cliptmeter` [private]

4.426.3.4 threshold `mha_real_t` `softclipper_t::threshold` [private]

4.426.3.5 hardlimit `mha_real_t` `softclipper_t::hardlimit` [private]

4.426.3.6 slope `mha_real_t` `softclipper_t::slope` [private]

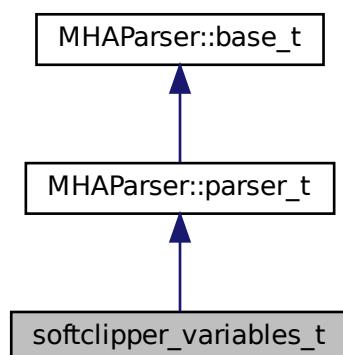
4.426.3.7 linear `bool` `softclipper_t::linear` [private]

The documentation for this class was generated from the following file:

- `transducers.cpp`

4.427 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

4.427.1 Constructor & Destructor Documentation

4.427.1.1 `softclipper_variables_t()` `softclipper_variables_t::softclipper_variables_t()`

4.427.2 Member Data Documentation

4.427.2.1 `tau_attack` `MHAParser::float_t softclipper_variables_t::tau_attack`

4.427.2.2 `tau_decay` `MHAParser::float_t softclipper_variables_t::tau_decay`

4.427.2.3 tau_clip `MHAParser::float_t` `softclipper_variables_t::tau_clip`

4.427.2.4 threshold `MHAParser::float_t` `softclipper_variables_t::threshold`

4.427.2.5 hardlimit `MHAParser::float_t` `softclipper_variables_t::hardlimit`

4.427.2.6 slope `MHAParser::float_t` `softclipper_variables_t::slope`

4.427.2.7 linear `MHAParser::bool_t` `softclipper_variables_t::linear`

4.427.2.8 clipped `MHAParser::float_mon_t` `softclipper_variables_t::clipped`

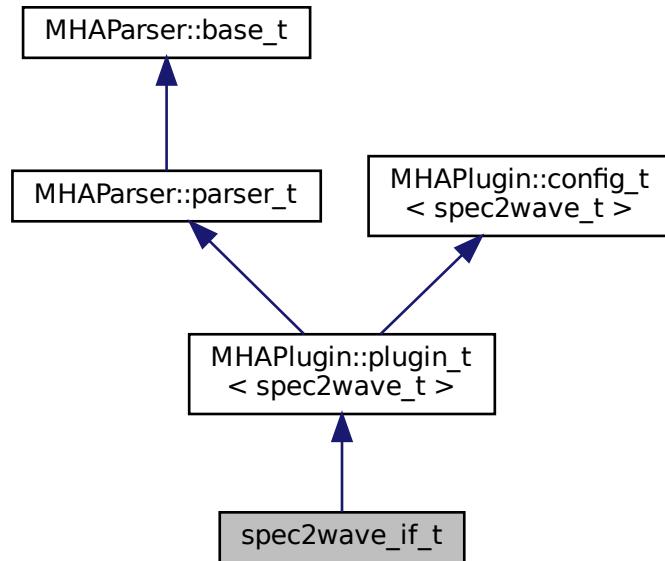
4.427.2.9 max_clipped `MHAParser::float_t` `softclipper_variables_t::max_clipped`

The documentation for this class was generated from the following file:

- `transducers.cpp`

4.428 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 &)`
- `void release ()`
- `mha_wave_t * process (mha_spec_t *)`

Private Member Functions

- `void update ()`
- `void setlock (bool b)`

Private Attributes

- `MHAParser::float_t ramplen`
- `windowselector_t window_config`

Additional Inherited Members

4.428.1 Constructor & Destructor Documentation

```
4.428.1.1 spec2wave_if_t() spec2wave_if_t::spec2wave_if_t (
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & )
```

4.428.2 Member Function Documentation

```
4.428.2.1 prepare() void spec2wave_if_t::prepare (
    mhaconfig_t & t ) [virtual]
```

Implements **MHAPlugIn::plugin_t< spec2wave_t >** (p. [1148](#)).

```
4.428.2.2 release() void spec2wave_if_t::release ( ) [virtual]
```

Reimplemented from **MHAPlugIn::plugin_t< spec2wave_t >** (p. [1149](#)).

```
4.428.2.3 process() mha_wave_t * spec2wave_if_t::process (
    mha_spec_t * spec_in )
```

```
4.428.2.4 update() void spec2wave_if_t::update ( ) [private]
```

4.428.2.5 `setlock()` `void spec2wave_if_t::setlock (`
`bool b)` [private]

4.428.3 Member Data Documentation

4.428.3.1 `ramplen` `MHAParser::float_t spec2wave_if_t::ramplen` [private]

4.428.3.2 `window_config` `windowselector_t spec2wave_if_t::window_config` [private]

The documentation for this class was generated from the following file:

- `spec2wave.cpp`

4.429 `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`

4.429.1 Constructor & Destructor Documentation

4.429.1.1 **spec2wave_t()** 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 )
```

4.429.1.2 ~spec2wave_t() spec2wave_t::~spec2wave_t ()

4.429.2 Member Function Documentation

4.429.2.1 **process()** mha_wave_t * spec2wave_t::process (

```
    mha_spec_t * spec_in )
```

4.429.3 Member Data Documentation

4.429.3.1 **ft** mha_fft_t spec2wave_t::ft [private]

FFT class.

4.429.3.2 **npad1** unsigned int spec2wave_t::npad1 [private]

length of zero padding before window

4.429.3.3 npad2 `unsigned int spec2wave_t::npad2 [private]`

length of zero padding after window

4.429.3.4 ramps `hanning_ramps_t spec2wave_t::ramps [private]`

4.429.3.5 calc_out `MHASignal::waveform_t spec2wave_t::calc_out [private]`

4.429.3.6 out_buf `MHASignal::waveform_t spec2wave_t::out_buf [private]`

4.429.3.7 write_buf `MHASignal::waveform_t spec2wave_t::write_buf [private]`

4.429.3.8 sc `mha_real_t spec2wave_t::sc [private]`

4.429.3.9 nfft `unsigned int spec2wave_t::nfft [private]`

4.429.3.10 nwndshift `unsigned int spec2wave_t::nwndshift [private]`

4.429.3.11 postwindow `MHAWindow::base_t spec2wave_t::postwindow [private]`

The documentation for this class was generated from the following file:

- `spec2wave.cpp`

4.430 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**

4.430.1 Constructor & Destructor Documentation

4.430.1.1 spec_fader_t() `spec_fader_t::spec_fader_t (`
 `unsigned int ch,`
 `mha_real_t fr,`
 `MHAParser::vfloat_t & ng,`
 `MHAParser::float_t & t)`

4.430.1.2 ~spec_fader_t() `spec_fader_t::~spec_fader_t () [inline]`

4.430.2 Member Data Documentation

4.430.2.1 nch `unsigned int spec_fader_t::nch`

4.430.2.2 gains `mha_real_t* spec_fader_t::gains`

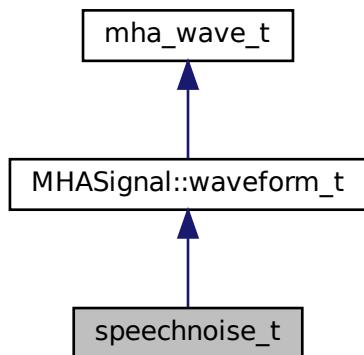
4.430.2.3 **fr** unsigned int spec_fader_t::fr

The documentation for this class was generated from the following file:

- **fader_spec.cpp**

4.431 speechnoise_t Class Reference

Inheritance diagram for speechnoise_t:



Public Types

- enum **noise_type_t** {
 mha, **olnoise**, **LTASS_combined**, **LTASS_female**,
 LTASS_male, **white**, **pink**, **brown**,
 TEN_SPL, **TEN_SPL_250_8k**, **TEN_SPL_50_16k**, **sin125**,
 sin250, **sin500**, **sin1k**, **sin2k**,
 sin4k, **sin8k** }

Public Member Functions

- **speechnoise_t** (float duration, float srat, unsigned int **channels**, **speechnoise_t::noise_type_t** noise_type= **speechnoise_t::mha**)
- **speechnoise_t** (unsigned int length_samples, float srat, 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 srat)

Additional Inherited Members

4.431.1 Member Enumeration Documentation

4.431.1.1 **noise_type_t** 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	

4.431.2 Constructor & Destructor Documentation

4.431.2.1 **speechnoise_t()** [1/2] **speechnoise_t::speechnoise_t** (

float duration,

```
float srate,  
unsigned int channels,  
speechnoise_t::noise_type_t noise_type = speechnoise_t::mha )
```

4.431.2.2 `speechnoise_t()` [2/2] `speechnoise_t::speechnoise_t (`
`unsigned int length_samples,`
`float srate,`
`unsigned int channels,`
`speechnoise_t::noise_type_t noise_type = speechnoise_t::mha)`

4.431.3 Member Function Documentation

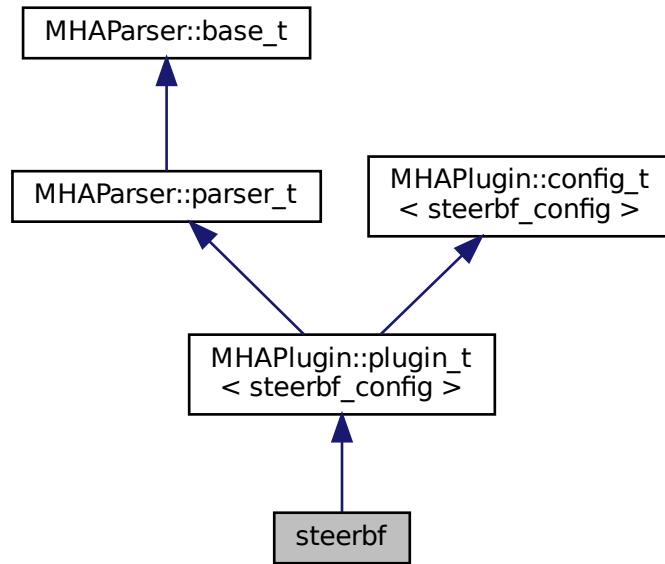
4.431.3.1 `creator()` `void speechnoise_t::creator (`
`speechnoise_t::noise_type_t noise_type,`
`float srate) [private]`

The documentation for this class was generated from the following files:

- **speechnoise.h**
- **speechnoise.cpp**

4.432 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**4.432.1 Constructor & Destructor Documentation**

4.432.1.1 `steerbf()` steerbf::steerbf (

```
    algo_comm_t & ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructs our plugin.

4.432.1.2 `~steerbf()` steerbf::~steerbf ()

4.432.2 Member Function Documentation

4.432.2.1 `process()` mha_spec_t * steerbf::process (

```
    mha_spec_t * signal )
```

Defers to configuration class.

4.432.2.2 `prepare()` void steerbf::prepare (

```
    mhaconfig_t & signal_info ) [virtual]
```

Plugin preparation.

An opportunity to validate configuration parameters before instantiating a configuration.

Parameters

<code>signal_info</code>	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. 1148).

4.432.2.3 `release()` `void steerbf::release (void) [inline], [virtual]`

Reimplemented from `MHAPlugIn::plugin_t<steerbf_config>` (p. 1149).

4.432.2.4 `update_cfg()` `void steerbf::update_cfg () [private]`

4.432.3 Member Data Documentation

4.432.3.1 `bf_src` `MHAParser::string_t steerbf::bf_src`

4.432.3.2 `angle_ind` `parser_int_dyn steerbf::angle_ind`

4.432.3.3 `angle_src` `MHAParser::string_t steerbf::angle_src`

4.432.3.4 `patchbay` `MHAEvents::patchbay_t<steerbf> steerbf::patchbay [private]`

The documentation for this class was generated from the following files:

- `steerbf.h`
- `steerbf.cpp`

4.433 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**

4.433.1 Constructor & Destructor Documentation

4.433.1.1 steerbf_config() steerbf_config::steerbf_config (

```
algo_comm_t & ac,
const mhaconfig_t in_cfg,
steerbf * steerbf )
```

4.433.1.2 ~steerbf_config() steerbf_config::~steerbf_config ()

4.433.2 Member Function Documentation

4.433.2.1 process() mha_spec_t * steerbf_config::process (

```
mha_spec_t * inSpec )
```

4.433.3 Member Data Documentation

4.433.3.1 nchan unsigned int steerbf_config::nchan [private]

4.433.3.2 nfreq unsigned int steerbf_config::nfreq [private]

4.433.3.3 outSpec MHASignal::spectrum_t steerbf_config::outSpec [private]

4.433.3.4 bf_vec mha_spec_t steerbf_config::bf_vec [private]

4.433.3.5 nangle unsigned int steerbf_config::nangle [private]

4.433.3.6 _steerbf steerbf* steerbf_config::_steerbf [private]

4.433.3.7 ac algo_comm_t& steerbf_config::ac [private]

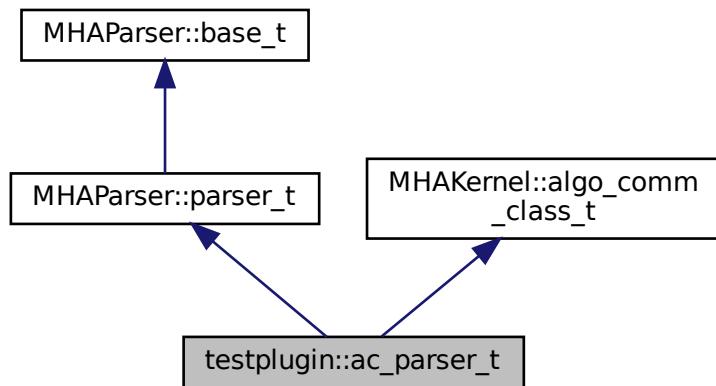
4.433.3.8 bf_src_copy std::string steerbf_config::bf_src_copy [private]

The documentation for this class was generated from the following files:

- **steerbf.h**
- **steerbf.cpp**

4.434 testplugin::ac_parser_t Class Reference

Inheritance diagram for testplugin::ac_parser_t:



Public Types

- enum `data_type_t` {
 `_MHA_AC_CHAR`, `_MHA_AC_INT`, `_MHA_AC_MHAREAL`, `_MHA_AC_FLOAT`,
`_MHA_AC_DOUBLE`, `_MHA_AC_MHACOMPLEX`, `_unknown` }

Public Member Functions

- `ac_parser_t ()`
- `void do_insert_var ()`
Insert variable into AC space.
- `void do_get_var ()`

Public Attributes

- `MHParse::string_t insert_var`
- `MHParse::string_t get_var`
- `MHParse::kw_t data_type`
- `MHParse::int_t num_entries`
- `MHParse::int_t stride`
- `MHParse::string_t char_data`
- `MHParse::vint_t int_data`
- `MHParse::vfloat_t float_data`
- `MHParse::vcomplex_t complex_data`
- `MHAEvents::patchbay_t< ac_parser_t > patchbay`

Additional Inherited Members

4.434.1 Member Enumeration Documentation

4.434.1.1 `data_type_t` `enum testplugin::ac_parser_t::data_type_t`

Enumerator

<code>_MHA_AC_CHAR</code>	
<code>_MHA_AC_INT</code>	
<code>_MHA_AC_MHAREAL</code>	
<code>_MHA_AC_FLOAT</code>	
<code>_MHA_AC_DOUBLE</code>	
<code>_MHA_AC_MHACOMPLEX</code>	
<code>_unknown</code>	

4.434.2 Constructor & Destructor Documentation

4.434.2.1 `ac_parser_t()` `testplugin::ac_parser_t::ac_parser_t () [inline]`

4.434.3 Member Function Documentation

4.434.3.1 `do_insert_var()` `void testplugin::ac_parser_t::do_insert_var () [inline]`

Insert variable into AC space.

This leaks memory by design, as the plugin is for testing only

4.434.3.2 `do_get_var()` `void testplugin::ac_parser_t::do_get_var () [inline]`

4.434.4 Member Data Documentation**4.434.4.1 insert_var** `MHAParser::string_t testplugin::ac_parser_t::insert_var`**4.434.4.2 get_var** `MHAParser::string_t testplugin::ac_parser_t::get_var`**4.434.4.3 data_type** `MHAParser::kw_t testplugin::ac_parser_t::data_type`**4.434.4.4 num_entries** `MHAParser::int_t testplugin::ac_parser_t::num_entries`**4.434.4.5 stride** `MHAParser::int_t testplugin::ac_parser_t::stride`**4.434.4.6 char_data** `MHAParser::string_t testplugin::ac_parser_t::char_data`**4.434.4.7 int_data** `MHAParser::vint_t testplugin::ac_parser_t::int_data`**4.434.4.8 float_data** `MHAParser::vfloat_t testplugin::ac_parser_t::float_data`

4.434.4.9 complex_data `MHAParser::vcomplex_t` `testplugin::ac_parser_t::complex_` ←
data

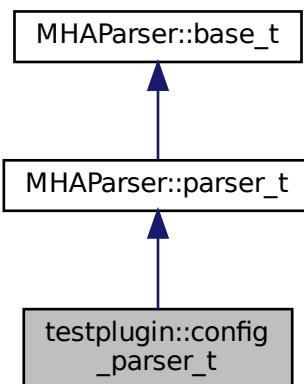
4.434.4.10 patchbay `MHAEvents::patchbay_t< ac_parser_t>` `testplugin::ac_parser_t::patchbay`

The documentation for this class was generated from the following file:

- `testplugin.cpp`

4.435 testplugin::config_parser_t Class Reference

Inheritance diagram for testplugin::config_parser_t:



Public Member Functions

- `void setlock (const bool &b)`
- `config_parser_t ()`
- `mhaconfig_t get () const`
- `void set (mhaconfig_t c)`

Public Attributes

- **MHAParser::int_t channels**
- **MHAParser::kw_t domain**
- **MHAParser::int_t fragsize**
- **MHAParser::int_t wndlen**
- **MHAParser::int_t fftlen**
- **MHAParser::float_t srate**

Additional Inherited Members

4.435.1 Constructor & Destructor Documentation

4.435.1.1 config_parser_t() testplugin::config_parser_t::config_parser_t () [inline]

4.435.2 Member Function Documentation

4.435.2.1 setlock() void testplugin::config_parser_t::setlock (const bool & b) [inline]

4.435.2.2 get() mhaconfig_t testplugin::config_parser_t::get () const [inline]

4.435.2.3 set() void testplugin::config_parser_t::set (mhaconfig_t c) [inline]

4.435.3 Member Data Documentation

4.435.3.1 channels `MHAParser::int_t` `testplugin::config_parser_t::channels`

4.435.3.2 domain `MHAParser::kw_t` `testplugin::config_parser_t::domain`

4.435.3.3 fragsize `MHAParser::int_t` `testplugin::config_parser_t::fragsize`

4.435.3.4 wndlen `MHAParser::int_t` `testplugin::config_parser_t::wndlen`

4.435.3.5 fftlen `MHAParser::int_t` `testplugin::config_parser_t::fftlens`

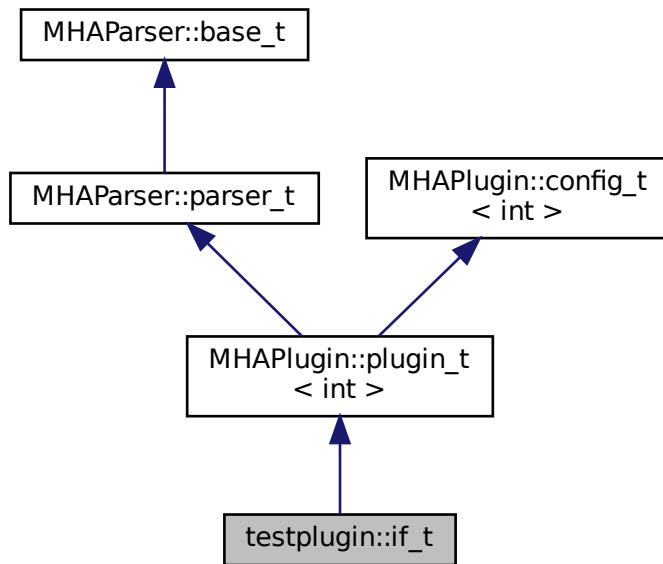
4.435.3.6 srate `MHAParser::float_t` `testplugin::config_parser_t::srate`

The documentation for this class was generated from the following file:

- `testplugin.cpp`

4.436 testplugin::if_t Class Reference

Inheritance diagram for testplugin::if_t:



Public Member Functions

- `if_t (const algo_comm_t &iac, const std::string &ith, const std::string &ial)`
- `mha_spec_t * process (mha_spec_t *s_in)`
- `mha_wave_t * process (mha_wave_t *s_in)`
- `void prepare (mhaconfig_t &)`

Private Member Functions

- `void test_prepare ()`
- `void test_process ()`

Private Attributes

- `config_parser_t config_in`
- `config_parser_t config_out`
- `ac_parser_t ac`
- `signal_parser_t signal`
- `MHAParser::bool_t _prepare`
- `MHAEvents::patchbay_t< if_t > patchbay`
- `MHAParser::mhapluginloader_t plug`

Additional Inherited Members

4.436.1 Constructor & Destructor Documentation

4.436.1.1 if_t() testplugin::if_t::if_t (

```
const algo_comm_t & iac,
const std::string & ith,
const std::string & ial )
```

4.436.2 Member Function Documentation

4.436.2.1 process() [1/2] mha_spec_t* testplugin::if_t::process (

```
mha_spec_t * s_in ) [inline]
```

4.436.2.2 process() [2/2] mha_wave_t* testplugin::if_t::process (

```
mha_wave_t * s_in ) [inline]
```

4.436.2.3 prepare() void testplugin::if_t::prepare (

```
mhaconfig_t & ) [inline], [virtual]
```

Implements **MHAPlugin::plugin_t< int >** (p. 1148).

4.436.2.4 test_prepare() void testplugin::if_t::test_prepare () [private]

4.436.2.5 test_process() void testplugin::if_t::test_process () [private]

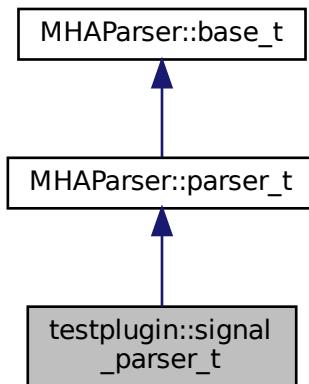
4.436.3 Member Data Documentation**4.436.3.1 config_in** `config_parser_t` `testplugin::if_t::config_in` [private]**4.436.3.2 config_out** `config_parser_t` `testplugin::if_t::config_out` [private]**4.436.3.3 ac** `ac_parser_t` `testplugin::if_t::ac` [private]**4.436.3.4 signal** `signal_parser_t` `testplugin::if_t::signal` [private]**4.436.3.5 _prepare** `MHAParser::bool_t` `testplugin::if_t::_prepare` [private]**4.436.3.6 patchbay** `MHAEEvents::patchbay_t< if_t>` `testplugin::if_t::patchbay` [private]**4.436.3.7 plug** `MHAParser::mhapluginloader_t` `testplugin::if_t::plug` [private]

The documentation for this class was generated from the following file:

- `testplugin.cpp`

4.437 testplugin::signal_parser_t Class Reference

Inheritance diagram for testplugin::signal_parser_t:



Public Member Functions

- **signal_parser_t ()**

Public Attributes

- **MHAParser::mfloat_t input_wave**
- **MHAParser::mcomplex_t input_spec**
- **MHAParser::mfloat_mon_t output_wave**
- **MHAParser::mcomplex_mon_t output_spec**

Additional Inherited Members

4.437.1 Constructor & Destructor Documentation

4.437.1.1 **signal_parser_t()** testplugin::signal_parser_t::signal_parser_t () [inline]

4.437.2 Member Data Documentation

4.437.2.1 `input_wave` `MHAParser::mfloat_t testplugin::signal_parser_t::input_wave`

4.437.2.2 `input_spec` `MHAParser::mcomplex_t testplugin::signal_parser_t::input_spec`

4.437.2.3 `output_wave` `MHAParser::mfloat_mon_t testplugin::signal_parser_t::output_wave`

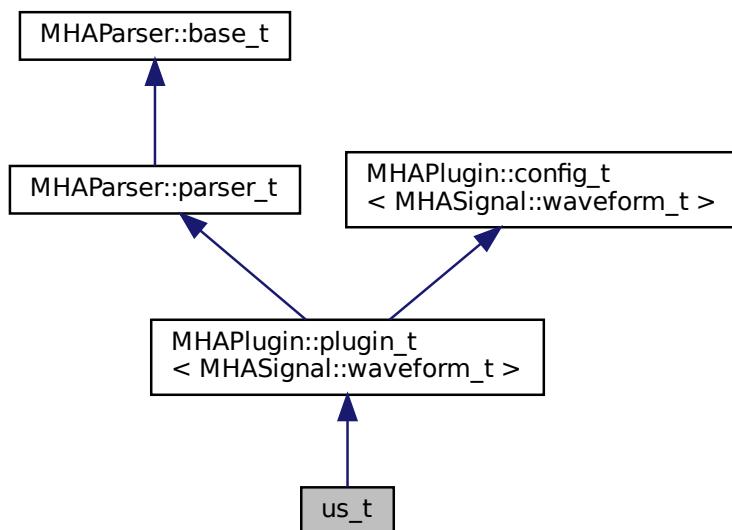
4.437.2.4 `output_spec` `MHAParser::mcomplex_mon_t testplugin::signal_parser_t::output_spec`

The documentation for this class was generated from the following file:

- `testplugin.cpp`

4.438 `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

4.438.1 Constructor & Destructor Documentation

```
4.438.1.1 us_t() us_t::us_t (
    algo_comm_t iac,
    std::string ,
    std::string )
```

4.438.2 Member Function Documentation

```
4.438.2.1 process() mha_wave_t * us_t::process (
    mha_wave_t * s )
```

```
4.438.2.2 prepare() void us_t::prepare (
    mhaconfig_t & cf ) [virtual]
```

Implements **MHAPlugin::plugin_t< MHASignal::waveform_t >** (p. [1148](#)).

4.438.2.3 `release()` `void us_t::release () [virtual]`

Reimplemented from `MHAParser::plugin_t< MHASignal::waveform_t >` (p. 1149).

4.438.3 Member Data Documentation

4.438.3.1 `ratio` `MHAParser::int_t us_t::ratio [private]`

4.438.3.2 `antialias` `MHAFilter::iir_filter_t us_t::antialias [private]`

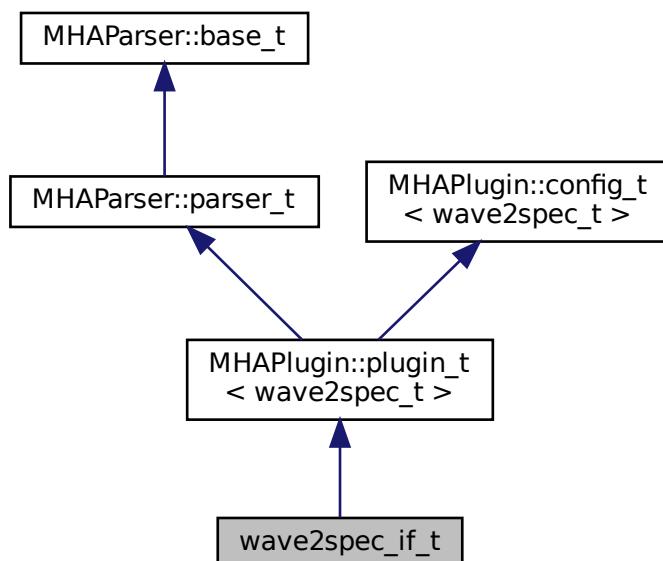
The documentation for this class was generated from the following file:

- `upsample.cpp`

4.439 `wave2spec_if_t` Class Reference

Plugin wave2spec interface class, uses `wave2spec_t` (p. 1491) as runtime configuration.

Inheritance diagram for `wave2spec_if_t`:



Public Member Functions

- **wave2spec_if_t** (const **algo_comm_t** &iac, const std::string &, const std::string &ialg)
Constructor of wave2spec plugin, sets up configuration variables and callbacks.
- **void prepare** (**mhaconfig_t** &t)
prepare for signal processing
- **void release** ()
Unprepare signal processing.
- **void process** (**mha_wave_t** *wave_in, **mha_spec_t** **sout)
processing callback used for domain transformation
- **void process** (**mha_wave_t** *wave_in, **mha_wave_t** **sout)
processing callback used if output of original waveform is requested.

Private Member Functions

- **void update** ()
Create a new runtime configuration from configuration parameters when the plugin is prepared, or when the window position or other window parameters change.
- **void setlock** (bool b)
Lock/Unlock all configuration variables.

Private Attributes

- **MHAParser::int_t nfft**
FFT length selector.
- **MHAParser::int_t nwnd**
Window length selector.
- **MHAParser::float_t wndpos**
Window position selector.
- **windowselector_t window_config**
- **MHAParser::bool_t strict_window_ratio**
Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.
- **MHAParser::bool_t return_wave**
Switch to select return domain.
- **std::string algo**
configured name this plugin, used to name the AC variables
- **MHAParser::vfloat_mon_t zeropadding**

Additional Inherited Members

4.439.1 Detailed Description

Plugin wave2spec interface class, uses **wave2spec_t** (p. 1491) as runtime configuration.

4.439.2 Constructor & Destructor Documentation

4.439.2.1 **wave2spec_if_t()** `wave2spec_if_t::wave2spec_if_t (`

```
    const algo_comm_t & iac,
    const std::string & ,
    const std::string & ialg )
```

Constructor of wave2spec plugin, sets up configuration variables and callbacks.

Parameters

<i>iac</i>	algorithm communication storage accessor
<i>ialg</i>	configured name of this plugin, used to name the AC variables published by wave2spec

4.439.3 Member Function Documentation

4.439.3.1 **prepare()** `void wave2spec_if_t::prepare (`

```
    mhaconfig_t & t ) [virtual]
```

prepare for signal processing

Parameters

<i>in,out</i>	<i>t</i>	signal dimensions, modified by prepare as determined by the STFT configuration
---------------	----------	--

Implements **MHAPlugin::plugin_t< wave2spec_t >** (p. [1148](#)).

4.439.3.2 **release()** `void wave2spec_if_t::release () [virtual]`

Unprepare signal processing.

Reimplemented from **MHAPlugin::plugin_t< wave2spec_t >** (p. [1149](#)).

```
4.439.3.3 process() [1/2] void wave2spec_if_t::process (
    mha_wave_t * wave_in,
    mha_spec_t ** sout )
```

processing callback used for domain transformation

Parameters

<i>wave_in</i>	latest block of audio signal (hop size samples per channel)
<i>sout</i>	output spectrum pointer

```
4.439.3.4 process() [2/2] void wave2spec_if_t::process (
    mha_wave_t * wave_in,
    mha_wave_t ** sout )
```

processing callback used if output of original waveform is requested.

The STFT spectrum is computed and can only be accessed by downstream plugins through the AC variable published by this plugin.

Parameters

<i>wave_in</i>	latest block of audio signal (hop size samples per channel)
<i>sout</i>	output waveform pointer (FFT length samples per channel)

```
4.439.3.5 update() void wave2spec_if_t::update ( ) [private]
```

Create a new runtime configuration from configuration parameters when the plugin is prepared, or when the window position or other window parameters change.

Exceptions

MHA_Error (p. 763)	if the configuration change is not compatible with the current input and FFT length constraints.
--	--

4.439.3.6 setlock() `void wave2spec_if_t::setlock (`
 `bool b) [private]`

Lock/Unlock all configuration variables.

Parameters

<code>b</code>	Desired lock state
----------------	--------------------

4.439.4 Member Data Documentation

4.439.4.1 nfft `MHAParser::int_t wave2spec_if_t::nfft [private]`

FFT length selector.

4.439.4.2 nwnd `MHAParser::int_t wave2spec_if_t::nwnd [private]`

Window length selector.

4.439.4.3 wndpos `MHAParser::float_t wave2spec_if_t::wndpos [private]`

Window position selector.

4.439.4.4 window_config `windowselector_t wave2spec_if_t::window_config [private]`

4.439.4.5 strict_window_ratio `MHAParser::bool_t wave2spec_if_t::strict_window_ratio [private]`

Switch to disallow window sizes that are not a multiple of the fragsize a by power of two.

4.439.4.6 return_wave `MHAParser::bool_t wave2spec_if_t::return_wave [private]`

Switch to select return domain.

4.439.4.7 algo `std::string wave2spec_if_t::algo [private]`

configured name this plugin, used to name the AC variables

4.439.4.8 zeropadding `MHAParser::vfloat_mon_t wave2spec_if_t::zeropadding [private]`

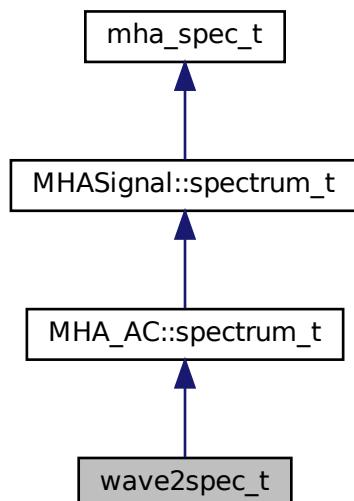
The documentation for this class was generated from the following files:

- `wave2spec.hh`
- `wave2spec.cpp`

4.440 **wave2spec_t Class Reference**

Runtime configuration class for plugin wave2spec.

Inheritance diagram for `wave2spec_t`:



Public Member Functions

- **wave2spec_t** (unsigned int nfft, unsigned int nwnd_, unsigned int nwndshift_, unsigned int nch, **mha_real_t** wndpos, const **MHAWindow::base_t** & **window**, **algo_comm_t ac**, std::string algo)

Constructor computes window and zeropadding, allocates storage and FFT plan.
- void **publish_ac_variables** ()

Insert two AC variables into AC space:
- **mha_spec_t * process** (**mha_wave_t** *wave_in)

Perform signal shift, windowing, zero-padding and FFT.
- ~**wave2spec_t** ()

Destructor removes AC variables from AC space and deallocates memory.
- unsigned **get_zeropadding** (bool after) const

Getter method to read zeropadding computed for the STFT configuration parameters.

Private Member Functions

- void **calc_pre_wnd** (**MHASignal::waveform_t** &dest, const **MHASignal::waveform_t** &src)

Applies analysis window weights to current input signal and writes windowed signal to correct place in FFT buffer.

Private Attributes

- unsigned int **nwnd**

window length
- unsigned int **nwndshift**

window shift or hop size
- **mha_fft_t ft**

FFT instance used for transformation.
- unsigned int **npad1**

length of zero padding before window
- unsigned int **npad2**

length of zero padding after window
- **MHAWindow::base_t window**

Analysis window.
- **MHASignal::waveform_t calc_in**

waveform buffer with FFT length samples per channel
- **MHASignal::waveform_t in_buf**

waveform buffer with window length samples per channel
- **MHASignal::spectrum_t spec_in**

spectrum buffer containing only the positive frequency bins
- std::string **ac_wndshape_name**

name of window shape AC variable

Additional Inherited Members

4.440.1 Detailed Description

Runtime configuration class for plugin wave2spec.

Manages window shift, windowing, zero-padding, and FFT. Inserts current window shape and current STFT spectrum into AC space.

4.440.2 Constructor & Destructor Documentation

```
4.440.2.1 wave2spec_t() wave2spec_t::wave2spec_t (
    unsigned int nfft,
    unsigned int nwnd_,
    unsigned int nwndshift_,
    unsigned int nch,
    mha_real_t wndpos,
    const MHAWindow::base_t & window,
    algo_comm_t ac,
    std::string algo )
```

Constructor computes window and zeropadding, allocates storage and FFT plan.

Parameters

<i>nfft</i>	FFT length
<i>nwnd_</i>	window length in samples
<i>nwndshift_</i>	window shift (hop size) in samples
<i>nch</i>	number of audio channels
<i>wndpos</i>	for cases <i>nfft</i> > <i>nwnd_</i> , where to place the window inside the FFT buffer: 0 = at start, 1 = at end, 0.5 = centered. Position is rounded to full samples and determines zero-padding
<i>window</i>	Analysis window shape
<i>ac</i>	algorithm communication storage accessor
<i>algo</i>	configured name of this plugin, used to name the AC variables published by wave2spec

4.440.2.2 ~wave2spec_t() `wave2spec_t::~wave2spec_t ()`

Destructor removes AC variables from AC space and deallocates memory.

4.440.3 Member Function Documentation

4.440.3.1 publish_ac_variables() `void wave2spec_t::publish_ac_variables ()`

Insert two AC variables into AC space:

- <configured_name>: Contains the current STFT spectrum.
- <configured_name>_wnd: Contains the window shape as individual weights.

4.440.3.2 process() `mha_spec_t * wave2spec_t::process (mha_wave_t * wave_in)`

Perform signal shift, windowing, zero-padding and FFT.

Parameters

<code>wave_in</code>	latest block of audio signal (hop size samples per channel)
----------------------	---

Returns

pointer to current STFT spectrum. Storage is managed by this object. Downstream plug-ins may modify the signal in place.

4.440.3.3 get_zeropadding() `unsigned wave2spec_t::get_zeropadding (bool after) const [inline]`

Getter method to read zeropadding computed for the STFT configuration parameters.

Result is only valid after prepare() has been called.

Returns

Computed zeropadding before or after the analysis window in number of samples.

Parameters

<i>after</i>	When false, return length of zeropadding before the analysis window. When true, return length of zeropadding in samples after the analysis window.
--------------	--

```
4.440.3.4 calc_pre_wnd() void wave2spec_t::calc_pre_wnd (
    MHASignal::waveform_t & dest,
    const MHASignal::waveform_t & src ) [private]
```

Applies analysis window weights to current input signal and writes windowed signal to correct place in FFT buffer.

Ensures zero-padding regions contain only zeros. To be invoked before applying FFT.

Parameters

<i>out</i>	<i>dest</i>	waveform buffer with FFT length audio samples, completely overwritten by this method
<i>in</i>	<i>src</i>	waveform buffer with window length audio samples, these samples are written to dest after window shape has been applied to the individual samples.

4.440.4 Member Data Documentation

```
4.440.4.1 nwnd unsigned int wave2spec_t::nwnd [private]
```

window length

```
4.440.4.2 nwndshift unsigned int wave2spec_t::nwndshift [private]
```

window shift or hop size

4.440.4.3 ft `mha_fft_t` `wave2spec_t::ft` [private]

FFT instance used for transformation.

4.440.4.4 npad1 `unsigned int` `wave2spec_t::npad1` [private]

length of zero padding before window

4.440.4.5 npad2 `unsigned int` `wave2spec_t::npad2` [private]

length of zero padding after window

4.440.4.6 window `MHAWindow::base_t` `wave2spec_t::window` [private]

Analysis window.

4.440.4.7 calc_in `MHASignal::waveform_t` `wave2spec_t::calc_in` [private]

waveform buffer with FFT length samples per channel

4.440.4.8 in_buf `MHASignal::waveform_t` `wave2spec_t::in_buf` [private]

waveform buffer with window length samples per channel

4.440.4.9 spec_in `MHASignal::spectrum_t` `wave2spec_t::spec_in` [private]

spectrum buffer containing only the positive frequency bins

4.440.4.10 ac_wndshape_name std::string wave2spec_t::ac_wndshape_name [private]

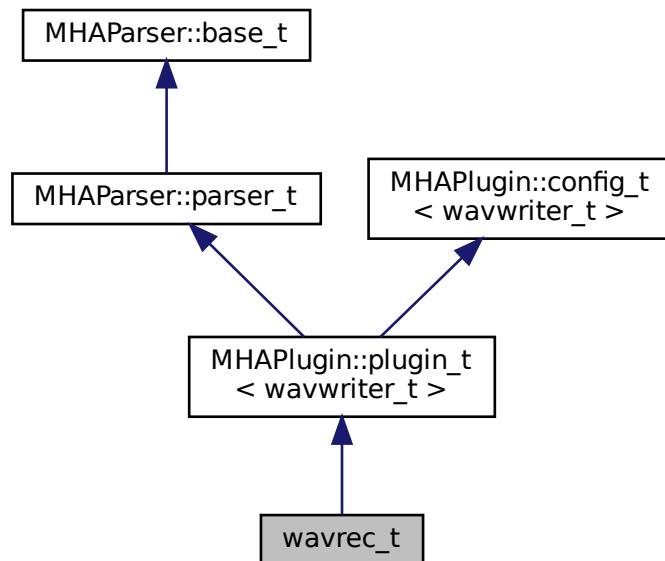
name of window shape AC variable

The documentation for this class was generated from the following files:

- **wave2spec.hh**
- **wave2spec.cpp**

4.441 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`
- `MHAParser::kw_t output_sample_format`
- `MHAEvents::patchbay_t< wavrec_t > patchbay`

Additional Inherited Members

4.441.1 Constructor & Destructor Documentation

4.441.1.1 `wavrec_t()` `wavrec_t::wavrec_t (`
 `const algo_comm_t & iac,`
 `const std::string & ,`
 `const std::string & algo_name)`

4.441.2 Member Function Documentation

4.441.2.1 `process()` `mha_wave_t * wavrec_t::process (`
 `mha_wave_t * s)`

4.441.2.2 `prepare()` `void wavrec_t::prepare (`
 `mhaconfig_t & cf) [virtual]`

Implements `MHAPlugin::plugin_t< wavwriter_t >` (p. 1148).

4.441.2.3 release() void wavrec_t::release () [virtual]

Reimplemented from **MHAPlugIn::plugin_t< wavwriter_t >** (p. 1149).

4.441.2.4 start_new_session() void wavrec_t::start_new_session () [private]

4.441.3 Member Data Documentation

4.441.3.1 record MHAParser::bool_t wavrec_t::record [private]

4.441.3.2 fifolen MHAParser::int_t wavrec_t::fifolen [private]

4.441.3.3 minwrite MHAParser::int_t wavrec_t::minwrite [private]

4.441.3.4 prefix MHAParser::string_t wavrec_t::prefix [private]

4.441.3.5 use_date MHAParser::bool_t wavrec_t::use_date [private]

4.441.3.6 output_sample_format MHAParser::kw_t wavrec_t::output_sample_format [private]

4.441.3.7 **patchbay** `MHAEVENTS::patchbay_t< wavrec_t> wavrec_t::patchbay [private]`

The documentation for this class was generated from the following file:

- `wavrec.cpp`

4.442 **wavwriter_t** Class Reference

Public Member Functions

- `wavwriter_t` (bool active, const `mhaconfig_t` &cf, unsigned int fifosize, unsigned int minwrite, const std::string &prefix, bool use_date, const std::string &format_name_)
- `~wavwriter_t ()`
- void `process` (`mha_wave_t` *)
- void `exit_request ()`

Private Member Functions

- void `write_thread ()`
- void `create_soundfile` (const std::string &prefix, bool use_date)
- void `set_format` (`SF_INFO` &sf_info)

Converts the format_name string to the corresponding int according to libsndfile and writes it into the format field of sf_info throws if no format of this name is available.

Static Private Member Functions

- static void * `write_thread` (void *this_)

Private Attributes

- `std::atomic< bool >` `close_session`
- bool `act_`
- `mhaconfig_t` `cf_`
- `SNDFILE *` `sf`
- `mha_fifo_if_t< mha_real_t >` `fifo`
- unsigned int `minw_`
- `pthread_t` `writethread`
- float * `data`
- `std::string` `format_name`

4.442.1 Constructor & Destructor Documentation

4.442.1.1 `wavwriter_t()` wavwriter_t::wavwriter_t (

```
    bool active,
    const mhaconfig_t & cf,
    unsigned int fifosize,
    unsigned int minwrite,
    const std::string & prefix,
    bool use_date,
    const std::string & format_name_ )
```

4.442.1.2 `~wavwriter_t()` wavwriter_t::~wavwriter_t ()

4.442.2 Member Function Documentation

4.442.2.1 `process()` void wavwriter_t::process (

```
    mha_wave_t * s )
```

4.442.2.2 `exit_request()` void wavwriter_t::exit_request ()

4.442.2.3 `write_thread() [1/2]` static void* wavwriter_t::write_thread (

```
    void * this_ ) [inline], [static], [private]
```

4.442.2.4 `write_thread() [2/2]` void wavwriter_t::write_thread () [private]

4.442.2.5 `create_soundfile()` void wavwriter_t::create_soundfile (const std::string & prefix, bool use_date) [private]

4.442.2.6 `set_format()` void wavwriter_t::set_format (SF_INFO & sf_info) [private]

Converts the format_name string to the corresponding int according to libsndfile and writes it into the format field of sf_info throws if no format of this name is available.

Parameters

<code>sf_info</code>	Destination sf_info struct for the format
----------------------	---

Exceptions

MHA_Error (p. 763)	If no sample format of name format_name is offered by libsndfile
---------------------------	--

4.442.3 Member Data Documentation

4.442.3.1 `close_session` std::atomic<bool> wavwriter_t::close_session [private]

4.442.3.2 `act_` bool wavwriter_t::act_ [private]

4.442.3.3 `cf_` mhaconfig_t wavwriter_t::cf_ [private]

4.442.3.4 `sf` SNDFILE* wavwriter_t::sf [private]

4.442.3.5 fifo `mha_fifo_if_t< mha_real_t> wavwriter_t::fifo` [private]

4.442.3.6 minw_ `unsigned int wavwriter_t::minw_` [private]

4.442.3.7 writethread `pthread_t wavwriter_t::writethread` [private]

4.442.3.8 data `float* wavwriter_t::data` [private]

4.442.3.9 format_name `std::string wavwriter_t::format_name` [private]

The documentation for this class was generated from the following file:

- `wavrec.cpp`

4.443 windnoise::cfg_t Class Reference

Runtime config class for windnoise plugin.

Public Member Functions

- **cfg_t** (const `mhaconfig_t` &signal_info, bool `UseChannel_LF_attenuation`, float tau_lowpass, float LowPassCutOffFrequency, float LowPassFraction_dB, float LowPassWindGain_dB)

constructor translates configuration variables to runtime config
- **mha_spec_t * process** (`mha_spec_t` *signal, std::vector< int > &detected, std::vector< float > &lowpass_quotient)

Detect windnoise.
- **void update_PSD_Lowpass** (const `mha_spec_t` *signal)

Low-pass filters the power spectrum.
- **void threshold_compare** (std::vector< int > &detected, std::vector< float > &lowpass_quotient)

Wind noise detection by comparing low-frequency intensity with broadband intensity.
- **int remapping** (const std::vector< float > &lowpass_quotient)
- **void compensation** (`mha_spec_t` *signal, int best_signal_channel_index)

Public Attributes

- bool **UseChannel_LF_attenuation** = false
FIXME: documentation for UseChannel_LF_attenuation.
- float **alpha_Lowpass** = 0
Filter coefficient for low-pass filtering each bin in the power spectrum with a first-order recursive low-pass filter.
- unsigned **FrequencyBinLowPass** = 0
Only smoothed power spectrum bins < FrequencyBinLowPass are added to the low-pass intensity.
- float **LowPassFraction** = 1
The wind noise detection threshold: We have wind noise if lowFreqIntensity / broadBandIntensity > LowPassFraction.
- float **LowPassWindGain** = 1
FIXME: documentation for LowPassWindGain.
- **MHASignal::waveform_t PSD_Lowpass**
The smoothed-over-time power spectrum.
- **MHASignal::waveform_t powspec**
Temporary storage for the power spectrum of the current input spectrum.

4.443.1 Detailed Description

Runtime config class for windnoise plugin.

Computes power spectra of incoming STFT spectra, smoothes the power spectrum over time by low-pass filtering the intensities of each bin over time, then detects wind noise presence by comparing intensity at low frequency bins to broadband intensity.

4.443.2 Constructor & Destructor Documentation

```
4.443.2.1 cfg_t() cfg_t::cfg_t (
    const mhaconfig_t & signal_info,
    bool UseChannel_LF_attenuation,
    float tau_Lowpass,
    float LowPassCutOffFrequency,
    float LowPassFraction_dB,
    float LowPassWindGain_dB )
```

constructor translates configuration variables to runtime config

4.443.3 Member Function Documentation

```
4.443.3.1 process() mha_spec_t * cfg_t::process (
    mha_spec_t * signal,
    std::vector< int > & detected,
    std::vector< float > & lowpass_quotient )
```

Detect windnoise.

FIXME: cancel it. The process method calls update_PSD_Lowpass and threshold_compare to do its work.

Parameters

in,out	<i>signal</i>	The current STFT spectrum.
out	<i>detected</i>	This Method changes the elements of the vector but not its size. Each element is set to 1 or 0, depending on windnoise being detected in the corresponding audio channel.
out	<i>lowpass_quotient</i>	This Method changes elements of the vector but not its size. Each element is set to the ratio between intensity of the signal, at low frequencies and overall intensity, in the corresponding audio channel.

Exceptions

MHA_Error (p. 763)	if windnoise_indicators.size() != signal.num_channels.
--	--

```
4.443.3.2 update_PSD_Lowpass() void cfg_t::update_PSD_Lowpass (
    const mha_spec_t * signal )
```

Low-pass filters the power spectrum.

```
4.443.3.3 threshold_compare() void cfg_t::threshold_compare (
    std::vector< int > & detected,
    std::vector< float > & lowpass_quotient )
```

Wind noise detection by comparing low-frequency intensity with broadband intensity.

Parameters

<code>out</code>	<code>detected</code>	This Method changes the elements of the vector but not its size. Each element is set to 1 or 0, depending on windnoise being detected in the corresponding audio channel.
<code>out</code>	<code>lowpass_quotient</code>	This Method changes elements of the vector but not its size. Each element is set to the ratio between intensity of the signal, at low frequencies and overall intensity, in the corresponding audio channel.

4.443.3.4 `remapping()` `int cfg_t::remapping (`
`const std::vector< float > & lowpass_quotient)`

4.443.3.5 `compensation()` `void cfg_t::compensation (`
`mha_spec_t * signal,`
`int best_signal_channel_index)`

4.443.4 Member Data Documentation

4.443.4.1 `UseChannel_LF_attenuation` `bool windnoise::cfg_t::UseChannel_LF_attenuation`
`= false`

FIXME: documentation for `UseChannel_LF_attenuation`.

4.443.4.2 `alpha_Lowpass` `float windnoise::cfg_t::alpha_Lowpass = 0`

Filter coefficient for low-pass filtering each bin in the power spectrum with a first-order recursive low-pass filter.

4.443.4.3 FrequencyBinLowPass `unsigned windnoise::cfg_t::FrequencyBinLowPass = 0`

Only smoothed power spectrum bins < FrequencyBinLowPass are added to the low-pass intensity.

4.443.4.4 LowPassFraction `float windnoise::cfg_t::LowPassFraction = 1`

The wind noise detection threshold: We have wind noise if `lowFreqIntensity / broadBandIntensity > LowPassFraction`.

4.443.4.5 LowPassWindGain `float windnoise::cfg_t::LowPassWindGain = 1`

FIXME: documentation for LowPassWindGain.

4.443.4.6 PSD_Lowpass `MHASignal::waveform_t windnoise::cfg_t::PSD_Lowpass`

The smoothed-over-time power spectrum.

4.443.4.7 powspec `MHASignal::waveform_t windnoise::cfg_t::powspec`

Temporary storage for the power spectrum of the current input spectrum.

Only needed to hold the newest squared magnitudes until they are filtered into PSD_Lowpass.

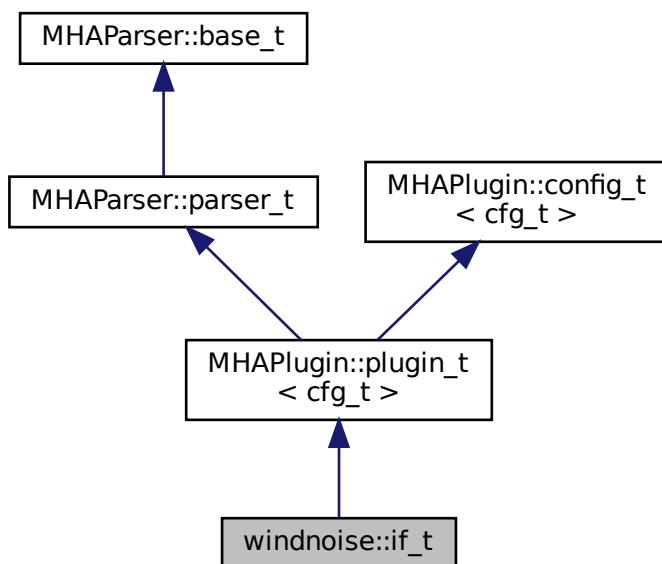
The documentation for this class was generated from the following files:

- `windnoise.hh`
- `windnoise.cpp`

4.444 windnoise::if_t Class Reference

interface class for windnoise plugin

Inheritance diagram for windnoise::if_t:



Public Member Functions

- **if_t (algo_comm_t ac, const std::string &chain_name, const std::string &algo_name)**
Constructor instantiates one windnoise plugin.
- **void prepare (mhaconfig_t &signal_info) override**
Prepare windnoise plugin for signal processing.
- **void release (void) override**
Nothing needs to be deallocated on release.
- **mha_spec_t * process (mha_spec_t *signal)**
signal processing, delegates to cfg_t::process (p. 1505)
- **void update (void)**
update runtime config when configuration parameters have changed
- **void insert ()**
inserts the windnoise detection vector into AC space

Public Attributes

- **MHAParser::bool_t UseChannel_LF_attenuation**
- **MHAParser::float_t tau_Lowpass**
- **MHAParser::float_t LowPassCutOffFrequency**
- **MHAParser::float_t LowPassFraction**
- **MHAParser::float_t LowPassWindGain**
- **MHAParser::kw_t WindNoiseDetector**
- **MHAParser::vint_mon_t detected**
- **MHAParser::vfloat_mon_t lowpass_quotient**
Name of AC variable mirroring the configuration monitor variable "detector".
- **const std::string lowpass_quotient_acname**
Name of AC variable mirroring the configuration monitor variable "lowpass_quotient".

Private Attributes

- **MHAEvents::patchbay_t< if_t > patchbay**
The Event connector.

Additional Inherited Members

4.444.1 Detailed Description

interface class for windnoise plugin

4.444.2 Constructor & Destructor Documentation

```
4.444.2.1 if_t() windnoise::if_t::if_t (
    algo_comm_t ac,
    const std::string & chain_name,
    const std::string & algo_name )
```

Constructor instantiates one windnoise plugin.

4.444.3 Member Function Documentation

```
4.444.3.1 prepare() void windnoise::if_t::prepare (
    mhaconfig_t & signal_info ) [override], [virtual]
```

Prepare windnoise plugin for signal processing.

Parameters

<code>signal_info</code>	signal dimensions, not changed by this plugin
--------------------------	---

Implements `MHAPlugIn::plugin_t< cfg_t >` (p. 1148).

4.444.3.2 `release()` `void windnoise::if_t::release (void) [inline], [override], [virtual]`

Nothing needs to be deallocated on release.

Reimplemented from `MHAPlugIn::plugin_t< cfg_t >` (p. 1149).

4.444.3.3 `process()` `mha_spec_t * windnoise::if_t::process (mha_spec_t * signal)`

signal processing, delegates to `cfg_t::process` (p. 1505)

4.444.3.4 `update()` `void windnoise::if_t::update (void)`

update runtime config when configuration parameters have changed

4.444.3.5 `insert()` `void windnoise::if_t::insert () [inline]`

inserts the windnoise detection vector into AC space

4.444.4 Member Data Documentation

4.444.4.1 patchbay `MHAEVENTS::patchbay_t< if_t> windnoise::if_t::patchbay [private]`

The Event connector.

4.444.4.2 UseChannel_LF_attenuation `MHAPARSER::bool_t windnoise::if_t::Use←
Channel_LF_attenuation`**4.444.4.3 tau_Lowpass** `MHAPARSER::float_t windnoise::if_t::tau_Lowpass`**4.444.4.4 LowPassCutOffFrequency** `MHAPARSER::float_t windnoise::if_t::LowPass←
CutOffFrequency`**4.444.4.5 LowPassFraction** `MHAPARSER::float_t windnoise::if_t::LowPassFraction`**4.444.4.6 LowPassWindGain** `MHAPARSER::float_t windnoise::if_t::LowPassWindGain`**4.444.4.7 WindNoiseDetector** `MHAPARSER::kw_t windnoise::if_t::WindNoiseDetector`**4.444.4.8 detected** `MHAPARSER::vint_mon_t windnoise::if_t::detected`**4.444.4.9 lowpass_quotient** `MHAPARSER::vfloat_mon_t windnoise::if_t::lowpass←
quotient`

4.444.4.10 detected_acname const std::string windnoise::if_t::detected_acname

Name of AC variable mirroring the configuration monitor variable "detector".

Usually "windnoise_detected".

4.444.4.11 lowpass_quotient_acname const std::string windnoise::if_t::lowpass_<quotient_acname>

Name of AC variable mirroring the configuration monitor variable "lowpass_quotient".

Usually "windnoise_lowpass_quotient".

The documentation for this class was generated from the following files:

- **windnoise.hh**
- **windnoise.cpp**

4.445 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 (MHParse::parser_t *p)**
insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.
- **void setlock (bool b_)**
Lock/Unlock variables.

Public Attributes

- **MHAEvents::emitter_t updated**

A collector event that fires when any of the window parameters managed here is written to.

Private Member Functions

- void **invalidate_window_data ()**
invalidates any allocated window samples.
- void **update_parser ()**
invoked when a parser parameter changes.

Private Attributes

- **MHAWindow::base_t * wnd**
Storage for the window data returned by `get_window_data()` (p. [1514](#))
- **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

4.445.1 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. [1287](#)) when needed.

4.445.2 Constructor & Destructor Documentation

4.445.2.1 windowselector_t() `windowselector_t::windowselector_t (const std::string & default_type)`

constructor creates the mha parser variables that describe an overlapadd analysis window.

Parameters

<code>default_type</code>	name of the default analysis window type. Must be one of: "rect", "bartlett", "hanning", "hamming", "blackman"
---------------------------	--

4.445.2.2 ~windowselector_t() `windowselector_t::~windowselector_t ()`

destructor frees window data that were allocated

4.445.3 Member Function Documentation

4.445.3.1 get_window_data() `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. 1287) instance. The referenced instance lives until the window parameters are changed, or this windowselector_t (p. 1512) instance is destroyed.
---------------	--

4.445.3.2 insert_items() `void windowselector_t::insert_items (` `MHAParser::parser_t * p)`

insert the window parameters "wndtype", "wndexp", and "userwnd" as mha configuration parameters into the given mha configuration parser.

Parameters

<i>p</i>	The configuration parser where to insert the window parameters. E.g. the plugin wave2spec's interface class.
----------	--

4.445.3.3 setlock() `void windowselector_t::setlock (` `bool b_)`

Lock/Unlock variables.

Parameters

<i>b</i> ↵	Desired lock state
— ↵	

4.445.3.4 invalidate_window_data() void windowselector_t::invalidate_window_data () [private]

invalidates any allocated window samples.

4.445.3.5 update_parser() void windowselector_t::update_parser () [private]

invoked when a parser parameter changes.

Calls **invalidate_window_data()** (p. 1515) and emits the updated event.

4.445.4 Member Data Documentation

4.445.4.1 updated MHAEvents::emitter_t windowselector_t::updated

A collector event that fires when any of the window parameters managed here is written to.

4.445.4.2 wnd MHAWindow::base_t* windowselector_t::wnd [private]

Storage for the window data returned by **get_window_data()** (p. 1514)

4.445.4.3 wndtype `MHAParser::kw_t windowselector_t::wndtype [private]`

parser variable for window type

4.445.4.4 wndexp `MHAParser::float_t windowselector_t::wndexp [private]`

parser variable for window exponent

4.445.4.5 userwnd `MHAParser::vfloat_t windowselector_t::userwnd [private]`

parser variable for user window samples to use

4.445.4.6 patchbay `MHAEEvents::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`

5 File Documentation

5.1 ac2isl.cpp File Reference

Classes

- struct `ac2isl::type_info`
- class `ac2isl::save_var_base_t`
Interface for ac to Isl bridge variable.
- class `ac2isl::save_var_t< T >`
Implementation for all ac to Isl bridges except complex types.
- class `ac2isl::save_var_t< mha_complex_t >`
Template specialization of the `ac2isl` (p. 78) bridge to take care of complex numbers.
- class `ac2isl::cfg_t`
Runtime configuration class of the `ac2isl` (p. 78) plugin.
- class `ac2isl::ac2isl_t`
Plugin class of `ac2isl` (p. 78).

Namespaces

- **ac2isl**

*All types for the **ac2isl** (p. 78) plugins live in this namespace.*

Variables

- const std::map< int, type_info > **ac2isl::types**

5.2 ac2osc.cpp File Reference

Classes

- class **ac2osc_t**

Plugin class of the ac2osc plugin.

5.3 ac2wave.cpp File Reference

Classes

- class **ac2wave_t**
- class **ac2wave_if_t**

5.4 ac_monitor_type.cpp File Reference

5.5 ac_monitor_type.hh File Reference

Classes

- class **acmon::ac_monitor_t**

A class for converting AC variables to Parser monitors of correct type.

Namespaces

- **acmon**

Namespace for displaying ac variables as parser monitors.

5.6 ac_mul.cpp File Reference

5.7 ac_mul.hh File Reference

Classes

- class **ac_mul_t**
*The class which implements the **ac_mul_t** (p. 191) plugin.*

Enumerations

- enum **arg_type_t** { **ARG_RR**, **ARG_RC**, **ARG_CR**, **ARG_CC** }
Indicates whether the factors of the product are real or complex valued.
- enum **val_type_t** { **VAL_REAL**, **VAL_COMPLEX** }
Indicates whether an AC variable contains real or complex values.

5.7.1 Enumeration Type Documentation

5.7.1.1 **arg_type_t** `enum arg_type_t`

Indicates whether the factors of the product are real or complex valued.

Enumerator

ARG_RR	Both factors are real.
ARG_RC	First factor is real, second is complex.
ARG_CR	First factor is complex, second is real.
ARG_CC	Both factors are complex.

5.7.1.2 **val_type_t** `enum val_type_t`

Indicates whether an AC variable contains real or complex values.

Enumerator

VAL_REAL	AC variable contains real values.
VAL_COMPLEX	AC variable contains complex values.

5.8 ac_proc.cpp File Reference

Classes

- class `ac_proc::interface_t`

Namespaces

- `ac_proc`

5.9 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)`

5.9.1 Macro Definition Documentation

5.9.1.1 PATCH_VAR `#define PATCH_VAR(`

```
var ) patchbay.connect(&var.valuechanged, this, & acConcat_wave::update_cfg)
```

5.9.1.2 INSERT_PATCH `#define INSERT_PATCH(`

```
var ) insert_member(var); PATCH_VAR(var)
```

5.10 acConcat_wave.h File Reference

Classes

- class `acConcat_wave_config`
- class `acConcat_wave`

5.11 acmon.cpp File Reference

Classes

- class `acmon::acmon_t`

Namespaces

- `acmon`
Namespace for displaying ac variables as parser monitors.

5.12 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)`

5.12.1 Macro Definition Documentation

5.12.1.1 `PATCH_VAR` #define PATCH_VAR(

```
var ) patchbay.connect(&var.valuechanged, this, & acPooling_wave::update_cfg)
```

5.12.1.2 `INSERT_PATCH` #define INSERT_PATCH(

```
var ) insert_member(var); PATCH_VAR(var)
```

5.13 acPooling_wave.h File Reference

Classes

- class **acPooling_wave_config**
- class **acPooling_wave**

5.14 acrec.cpp File Reference

5.15 acrec.hh File Reference

Classes

- class **plugins::hoertech::acrec::acwriter_t**
acwriter_t (p. [1379](#)) decouples signal processing from writing to disk.
- class **plugins::hoertech::acrec::acrec_t**
Plugin interface class of plugin acrec.

Namespaces

- **plugins**
- **plugins::hoertech**
- **plugins::hoertech::acrec**

Functions

- std::string **plugins::hoertech::acrec::to_iso8601** (time_t tm)

5.16 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"

5.16.1 Macro Definition Documentation**5.16.1.1 ACSAVE_FMT_TXT** #define ACSAVE_FMT_TXT 0**5.16.1.2 ACSAVE_SFMT_TXT** #define ACSAVE_SFMT_TXT "txt"**5.16.1.3 ACSAVE_FMT_MAT4** #define ACSAVE_FMT_MAT4 1**5.16.1.4 ACSAVE_SFMT_MAT4** #define ACSAVE_SFMT_MAT4 "mat4"**5.16.1.5 ACSAVE_FMT_M** #define ACSAVE_FMT_M 2**5.16.1.6 ACSAVE_SFMT_M** #define ACSAVE_SFMT_M "m"

5.17 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)

5.17.1 Macro Definition Documentation

5.17.1.1 **PATCH_VAR** #define PATCH_VAR(

```
var ) patchbay.connect(&var.valuechanged, this, & acSteer::update_cfg)
```

5.17.1.2 **INSERT_PATCH** #define INSERT_PATCH(

```
var ) insert_member(var); PATCH_VAR(var)
```

5.18 acSteer.h File Reference

Classes

- class **acSteer_config**
- class **acSteer**

5.19 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)

5.19.1 Macro Definition Documentation

5.19.1.1 PATCH_VAR #define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, & **acTransform_wave**::
 ::update_cfg)

5.19.1.2 INSERT_PATCH #define INSERT_PATCH(
 var) insert_member(var); **PATCH_VAR**(var)

5.20 acTransform_wave.h File Reference

Classes

- class **acTransform_wave_config**
- class **acTransform_wave**

5.21 adaptive_feedback_canceller.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, & **adaptive_feedback_canceller**::
 ::update_cfg)
- #define **INSERT_PATCH**(var) insert_member(var); **PATCH_VAR**(var)

Functions

- void **make_friendly_number_by_limiting** (mha_real_t &x)

5.21.1 Macro Definition Documentation

5.21.1.1 PATCH_VAR #define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, & **adaptive_feedback_canceller**::
 ::update_cfg)

5.21.1.2 INSERT_PATCH

```
#define INSERT_PATCH(
```

```
    var )  insert_member(var);  PATCH_VAR(var)
```

5.21.2 Function Documentation**5.21.2.1 make_friendly_number_by_limiting()**

```
void make_friendly_number_by_limiting(
```

```
    mha_real_t & x )  [inline]
```

5.22 adaptive_feedback_canceller.h File Reference**Classes**

- class **adaptive_feedback_canceller_config**
- class **adaptive_feedback_canceller**

5.23 addsndfile.cpp File Reference**Classes**

- class **addsndfile::waveform_proxy_t**
Class helps to specify which instance of MHASignal_waveform_t parent instance is meant in resampled_soundfile_t (p. 257).
- class **addsndfile::resampled_soundfile_t**
Reads sound from file and resamples it if necessary and wanted.
- class **addsndfile::sndfile_t**
- class **addsndfile::level_adapt_t**
- class **addsndfile::addsndfile_if_t**

Namespaces

- **addsndfile**

Macros

- #define **DEBUG(x)** std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

Typedefs

- `typedef MHAPlugin::config_t< level_adapt_t > addsndfile::level_adaptor`
- `typedef MHAPlugin::plugin_t< sndfile_t > addsndfile::wave_reader`

Enumerations

- `enum addsndfile::addsndfile_resampling_mode_t { addsndfile::DONT_RESAMPLE_PERMISSIVE, addsndfile::DONT_RESAMPLE_STRICT, addsndfile::DO_RESAMPLE }`

Specifies the resampling mode in resampled_soundfile_t.

Functions

- `static unsigned addsndfile::resampled_num_frames (unsigned num_source_frames, float source_rate, float target_rate, addsndfile_resampling_mode_t resampling_mode)`

5.23.1 Macro Definition Documentation

```
5.23.1.1 DEBUG #define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl
```

5.24 adm.cpp File Reference

Classes

- class `adm_rtconfig_t`
- class `adm_if_t`

Functions

- `MHASignal::waveform_t * adm_fir_lp (unsigned int fs, unsigned f_pass, unsigned int f_stop, unsigned int order)`
- `MHASignal::waveform_t * adm_fir_decomb (unsigned int fs, float dist_m, unsigned int order)`

5.24.1 Function Documentation

5.24.1.1 **adm_fir_lp()** `MHASignal::waveform_t* adm_fir_lp (`

```
    unsigned int fs,
    unsigned f_pass,
    unsigned int f_stop,
    unsigned int order )
```

5.24.1.2 **adm_fir_decomb()** `MHASignal::waveform_t* adm_fir_decomb (`

```
    unsigned int fs,
    float dist_m,
    unsigned int order )
```

5.25 adm.hh File Reference

Classes

- class **ADM::Linearphase_FIR< F >**
An efficient linear-phase fir filter implementation.
- class **ADM::Delay< F >**
A delay-line class.
- class **ADM::ADM< F >**
Adaptive differential microphone, working for speech frequency range.

Namespaces

- **ADM**

Functions

- static double **ADM::subsampledelay_coeff** (double samples, double f_design, double fs=1.0)
compute IIR coefficient for subsample delay

Variables

- const double **ADM::PI** = 3.14159265358979312
- const double **ADM::C** = 340
- const double **ADM::DELAY_FREQ** = 2000
- const double **ADM::START_BETA** = 0.5

5.26 altconfig.cpp File Reference

5.27 altconfig.hh File Reference

Classes

- class **altconfig_t**
Single class implementing plugin altconfig.

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

5.27.1 Macro Definition Documentation

5.27.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN #define MHAPLUGIN_OVERLOAD_OUTDOM←
AIN

5.28 altplugs.cpp File Reference

Classes

- class **mhaplug_cfg_t**
- class **altplugs_t**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

5.28.1 Macro Definition Documentation

5.28.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN #define MHAPLUGIN_OVERLOAD_OUTDOM↔
AIN

5.29 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 **document_plugin** (MHAKernel::algo_comm_class_t &ac, PluginLoader↔
::mhapluginloader_t &load, int argc, char **argv)
- void **document_io_plugin** (char *lib_name)
- int **main** (int argc, char **argv)

5.29.1 Function Documentation

5.29.1.1 strdom() std::string strdom (mha_domain_t d)

5.29.1.2 print_ac() void print_ac (MHAKernel::algo_comm_class_t & ac,
std::string txt)

5.29.1.3 document_plugin() int document_plugin (MHAKernel::algo_comm_class_t & ac,
PluginLoader::mhapluginloader_t & load,
int argc,
char ** argv)

5.29.1.4 document_io_plugin() void document_io_plugin (char * lib_name)

5.29.1.5 main() int main (int argc, char ** argv)

5.30 analysispath.cpp File Reference

Classes

- class **analysepath_t**
- class **plug_t**
- class **analysispath_if_t**

Functions

- static void * **thread_start** (void *instance)

5.30.1 Function Documentation

5.30.1.1 thread_start() static void* thread_start (void * instance) [static]

5.31 attenuate20.cpp File Reference

Classes

- class **attenuate20_t**

5.32 audiometerbackend.cpp File Reference

Classes

- class **audiometerbackend::Inn3rdoct_t**
- class **audiometerbackend::sine_t**
- class **audiometerbackend::signal_gen_t**
- class **audiometerbackend::level_adapt_t**
- class **audiometerbackend::audiometer_if_t**

Namespaces

- **audiometerbackend**

Macros

- #define **DEBUG(x)** std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x << std::endl

Typedefs

- typedef **MHAPlugin::config_t< level_adapt_t >** **audiometerbackend::level_adaptor**
- typedef **MHAPlugin::plugin_t< signal_gen_t >** **audiometerbackend::generator**

Functions

- static unsigned int **audiometerbackend::gcd** (unsigned int a, unsigned int b)
- **MHASignal::waveform_t audiometerbackend::return_sig** (unsigned int sigtype, unsigned int fs, unsigned int f)

5.32.1 Macro Definition Documentation

```
5.32.1.1 DEBUG #define DEBUG(  
    x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x <<  
std::endl
```

5.33 auditory_profile.cpp File Reference

5.34 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)

5.35 browsemhaplugins.cpp File Reference

Macros

- #define **DEBUG(x)** std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x
 << std::endl

Functions

- int **main** (int argc, char **argv)

5.35.1 Macro Definition Documentation

```
5.35.1.1 DEBUG #define DEBUG( x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x << "=" << x  

<< std::endl
```

5.35.2 Function Documentation

5.35.2.1 main() `int main (`
 `int argc,`
 `char ** argv)`

5.36 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)`

5.37 combinechannels.cpp File Reference

Classes

- class `combc_t`
- class `combc_if_t`

5.38 compiler_id.cpp File Reference

5.39 compiler_id.hh File Reference

Macros

- #define **COMPILER_ID_VENDOR** "gcc"
- #define **COMPILER_ID_MAJOR** __GNUC__
- #define **COMPILER_ID_MINOR** __GNUC_MINOR__
- #define **COMPILER_ID_PATCH** __GNUC_PATCHLEVEL__
- #define **COMPILER_ID_VERSION_HELPER2**(x, y, z) #x "." #y "." #z
- #define **COMPILER_ID_VERSION_HELPER1**(x, y, z) **COMPILER_ID_VERSION_HELPER2**(x,y,z)
- #define **COMPILER_ID_VERSION**
- #define **COMPILER_ID_COMPILER_ID_VENDOR** "-" **COMPILER_ID_VERSION** "-" **COMPILER_ID_STANDARD**

5.39.1 Macro Definition Documentation

5.39.1.1 **COMPILER_ID_VENDOR** #define COMPILER_ID_VENDOR "gcc"

5.39.1.2 **COMPILER_ID_MAJOR** #define COMPILER_ID_MAJOR __GNUC__

5.39.1.3 **COMPILER_ID_MINOR** #define COMPILER_ID_MINOR __GNUC_MINOR__

5.39.1.4 **COMPILER_ID_PATCH** #define COMPILER_ID_PATCH __GNUC_PATCHLEVEL__

```
5.39.1.5 COMPILER_ID_VERSION_HELPER2 #define COMPILER_ID_VERSION_HELPER2(  
    x,  
    y,  
    z ) #x "." #y "." #z
```

```
5.39.1.6 COMPILER_ID_VERSION_HELPER1 #define COMPILER_ID_VERSION_HELPER1(  
    x,  
    y,  
    z ) COMPILER_ID_VERSION_HELPER2(x,y,z)
```

```
5.39.1.7 COMPILER_ID_VERSION #define COMPILER_ID_VERSION
```

```
5.39.1.8 COMPILER_ID #define COMPILER_ID COMPILER_ID_VENDOR "-" COMPILER_ID_←  
VERSION "-" COMPILER_ID_STANDARD
```

5.40 complex_filter.cpp File Reference

5.41 complex_filter.h File Reference

Classes

- class **MHAFilter::complex_bandpass_t**
Complex bandpass filter.
- class **MHAFilter::gamma_flt_t**
Class for gammatone filter.
- class **MHAFilter::thirdoctave_analyzer_t**

Namespaces

- **MHAFilter**
Namespace for IIR and FIR filter classes.

5.42 complex_scale_channel.cpp File Reference

Classes

- class `cfg_t`
- class `complex_scale_channel_t`

5.43 cpupload.cpp File Reference

Classes

- class `cpupload::cpupload_cfg_t`
- class `cpupload::cpupload_if_t`

Namespaces

- `cpupload`

5.44 db.cpp File Reference

Classes

- class `db_t`
- class `db_if_t`

5.45 dbasync.cpp File Reference

Classes

- class `dbasync_native::delay_check_t`
- class `dbasync_native::dbasync_t`
- class `dbasync_native::db_if_t`

Namespaces

- `dbasync_native`

Enumerations

- enum { **dbasync_native::INVALID_THREAD_PRIORITY** = 999999999 }

Functions

- static void * **dbasync_native::thread_start** (void *instance)
- static unsigned **dbasync_native::gcd** (unsigned a, unsigned b)

5.46 dc.cpp File Reference

Macros

- #define **DUPVEC**(x) v.x.data = **MHASignal::dupvec_chk**(v.x.data,s)

Functions

- unsigned int **get_audiochannels** (unsigned int totalchannels, std::string acname, **algo_comm_t** ac)

5.46.1 Macro Definition Documentation

5.46.1.1 DUPVEC #define DUPVEC(

```
x ) v.x.data = MHASignal::dupvec_chk(v.x.data,s)
```

5.46.2 Function Documentation

5.46.2.1 get_audiochannels() unsigned int get_audiochannels (

```
unsigned int totalchannels,
std::string acname,
algo_comm_t ac )
```

5.47 dc.hh File Reference

Classes

- class `dc::dc_vars_t`
- class `dc::dc_vars_validator_t`
- class `dc::dc_t`
- class `dc::dc_if_t`

Namespaces

- `dc`

5.48 dc_afterburn.cpp File Reference

Namespaces

- **DynComp**
dynamic compression related classes and functions

Functions

- float `mylogf` (float x)

5.48.1 Function Documentation

5.48.1.1 mylogf() float mylogf (
 float x)

5.49 dc_afterburn.h File Reference

Classes

- class **DynComp::dc_afterburn_vars_t**
Variables for `dc_afterburn_t` (p. 452) 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

5.50 dc_simple.cpp File Reference

Namespaces

- **dc_simple**

Functions

- void **dc_simple::test_fail** (const std::vector< float > &v, unsigned int s, const std::string &name)
Checks size of vector.
- std::vector< float > **dc_simple::force_resize** (const std::vector< float > &v, unsigned int s, const std::string &name)
Creates a copy of vector v with s elements, provided that v has either s elements or 1 elements.
- **mha_real_t dc_simple::not_zero** (mha_real_t x, const std::string &comment)
Helper function to throw an error if x is 0.

5.51 dc_simple.hh File Reference

Classes

- class **dc_simple::dc_vars_t**
class for dc_simple (p. 86) plugin which registers variables to MHAParser (p. 122).
- class **dc_simple::dc_vars_validator_t**
Helper class to check sizes of configuration variable vectors.
- class **dc_simple::level_smoothen_t**
- class **dc_simple::dc_t**
Runtime config class for dc_simple (p. 86) plugin.
- class **dc_simple::dc_t::line_t**
- class **dc_simple::dc_if_t**
interface class

Namespaces

- **dc_simple**

Typedefs

- `typedef MHAPlugin::plugin_t< dc_t > dc_simple::DC`
- `typedef MHAPlugin::config_t< level_smoothen_t > dc_simple::LEVEL`

Functions

- `void dc_simple::test_fail (const std::vector< float > &v, unsigned int s, const std::string &name)`
Checks size of vector.
- `std::vector< float > dc_simple::force_resize (const std::vector< float > &v, unsigned int s, const std::string &name)`
Creates a copy of vector v with s elements, provided that v has either s elements or 1 elements.
- `mha_real_t dc_simple::not_zero (mha_real_t x, const std::string &comment)`
Helper function to throw an error if x is 0.

5.52 delay.cpp File Reference**Namespaces**

- `delay`

5.53 delay.hh File Reference**Classes**

- class `delay::interface_t`

Namespaces

- `delay`

5.54 delaysum_spec.cpp File Reference**Classes**

- class `delaysum_spec::delaysum_t`
- class `delaysum_spec::delaysum_spec_if_t`

Namespaces

- **delaysum_spec**

5.55 delaysum_wave.cpp File Reference

Classes

- class **delaysum::delaysum_wave_t**
Runtime configuration of the delaysum_wave plugin.
- class **delaysum::delaysum_wave_if_t**
Interface class for the delaysum plugin.

Namespaces

- **delaysum**
This namespace contains the delaysum plugin.

5.56 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)

5.56.1 Macro Definition Documentation

5.56.1.1 PATCH_VAR #define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, & **doasvm_classification::update_cfg**)

5.56.1.2 INSERT_PATCH #define INSERT_PATCH(
 var) **insert_member**(var); **PATCH_VAR**(var)

5.57 doasvm_classification.h File Reference

Classes

- class **doasvm_classification_config**
- class **doasvm_classification**

5.58 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)

5.58.1 Macro Definition Documentation

5.58.1.1 PATCH_VAR #define PATCH_VAR(
 var) patchbay.connect(&var.valuechanged, this, & **doasvm_feature_extraction::update_cfg**)

5.58.1.2 INSERT_PATCH #define INSERT_PATCH(
 var) **insert_member**(var); **PATCH_VAR**(var)

5.59 doasvm_feature_extraction.h File Reference

Classes

- class **doasvm_feature_extraction_config**
- class **doasvm_feature_extraction**

5.60 doc_appendix.h File Reference

5.61 doc_examples.h File Reference

5.62 doc_frameworks.h File Reference

5.63 doc_general.h File Reference

5.64 doc_kernel.h File Reference

5.65 doc_matlab.h File Reference

5.66 doc_mhamain.h File Reference

5.67 doc_parser.h File Reference

5.68 doc_plugins.h File Reference

5.69 doc_system.h File Reference

5.70 doc_toolbox.h File Reference

5.71 double2acvar.cpp File Reference

Classes

- class `double2acvar::double2acvar_t`
Plugin interface class for `double2acvar` (p. 90).

Namespaces

- `double2acvar`

5.72 downsample.cpp File Reference

Classes

- class `ds_t`

5.73 dropgen.cpp File Reference

Classes

- class **dropgen_t**

5.74 droptect.cpp File Reference

Classes

- class **droptect_t**

Detect dropouts in a signal with a constant spectrum.

5.75 equalize.cpp File Reference

Classes

- class **equalize::cfg_t**
- class **equalize::freqgains_t**

Namespaces

- **equalize**

5.76 example1.cpp File Reference

Classes

- class **example1_t**

This C++ class implements the simplest example plugin for the step-by-step tutorial.

5.77 example2.cpp File Reference

Classes

- class **example2_t**

This C++ class implements the second example plugin for the step-by-step tutorial.

5.78 example3.cpp File Reference

Classes

- class **example3_t**
A Plugin class using the openMHA Event mechanism.

5.79 example4.cpp File Reference

Classes

- class **example4_t**
A Plugin class using the spectral signal.

5.80 example5.cpp File Reference

Classes

- class **example5_t**
- class **plugin_interface_t**

Macros

- #define **__declspec(p)**

5.80.1 Macro Definition Documentation

5.80.1.1 **__declspec** #define __declspec(p)

5.81 example6.cpp File Reference

Classes

- class **cfg_t**
- class **example6_t**

Macros

- #define __declspec(p)

5.81.1 Macro Definition Documentation

5.81.1.1 __declspec #define __declspec(p)

5.82 example7.cpp File Reference

5.83 example7.hh File Reference

Classes

- class example7_t

5.84 fader_spec.cpp File Reference

Classes

- class spec_fader_t
- class fader_if_t

5.85 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
```

5.85.1 Macro Definition Documentation

5.85.1.1 DEBUG

```
#define DEBUG(
```



```
    x ) std::cerr << __FILE__ << ":" << __LINE__ << " " #x "=" << x <<
```

```
std::endl
```

5.86 fftfbpow.cpp File Reference

Classes

- class **fftfbpow::fftfbpow_t**
Run time configuration for the fftfbpow plugin.
- class **fftfbpow::fftfbpow_interface_t**
Interface class for fftfbpow plugin.

Namespaces

- **fftfbpow**
Namespace for the fftfbpow plugin.

5.87 fftfilter.cpp File Reference

Classes

- class **fftfilter::fftfilter_t**
- class **fftfilter::interface_t**

Namespaces

- **fftfilter**

Functions

- unsigned int **fftfilter::irs_length** (const **MHAParser::mfloat_t** &irs)
- unsigned int **fftfilter::irs_validator** (const **MHAParser::mfloat_t** &irs, const unsigned int &**channels**, const unsigned int &fragsize, const unsigned int &fftlen)

5.88 fftfilterbank.cpp File Reference

Classes

- class **fftfilterbank::fftfb_plug_t**
- class **fftfilterbank::fftfb_interface_t**

Namespaces

- **fftfilterbank**

5.89 fshift.cpp File Reference

5.90 fshift.hh File Reference

Classes

- class **fshift::fshift_config_t**
fshift runtime config class
- class **fshift::fshift_t**
fshift plugin interface class

Namespaces

- **fshift**

All types for the fshift plugin live in this namespace.

Functions

- int **fshift::fft_find_bin** (**mha_real_t** frequency, unsigned fftlen, **mha_real_t** srate)
Finds bin number of FFT bin nearest to the given frequency.

5.91 fshift_hilbert.cpp File Reference

Classes

- class **fshift_hilbert::hilbert_shifter_t**
- class **fshift_hilbert::frequency_translator_t**

Namespaces

- **fshift_hilbert**
All types for the hilbert frequency shifter live in this namespace.

5.92 gain.cpp File Reference

Classes

- class **gain::scaler_t**
- class **gain::gain_if_t**

Namespaces

- **gain**

5.93 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)

5.93.1 Function Documentation

5.93.1.1 convert_f2logf() `std::vector< mha_real_t> convert_f2logf (`
`const std::vector< mha_real_t > & vF)`

5.93.1.2 isempty() `bool isempty (`
`const std::vector< std::vector< mha_real_t > > & arg)`

5.94 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.

5.95 generatemhaplugindoc.cpp File Reference

Classes

- class **plug_wrapper1**
- class **io_wrapper**
- class **plug_wrapper**
- class **latex_doc_t**

Class to access the information stored in the plugin source code's MHAPLUGIN_DOCUMENTATION macro.

Functions

- std::string **conv2latex** (std::string s, bool iscolored=false)

Escapes various character sequences in texts not intended to be processed by LaTeX for processing by LaTeX.
- static void **print_plugin_references** (const std::set< std::string > &all_categories, std::map< std::string, std::vector< std::string > > main_category_plugins, std::map< std::string, std::vector< std::string > > additional_category_plugins, std::ofstream &ofile, const std::string &category_macro)

Function prints an overview of all categories and their associated plugins into the document.
- std::vector< std::string > **create_latex_doc** (std::map< std::string, std::string > &doc, const std::string &pluginname, const std::string &plugin_macro)

Loads the plugin, creates the latex documentation for the plugin, and adds the latex documentation for this plugin to the plugin's main category entry in doc.
- int **main** (int argc, char **argv)

5.95.1 Function Documentation

5.95.1.1 conv2latex() std::string conv2latex (
 std::string s,
 bool iscolored = false)

Escapes various character sequences in texts not intended to be processed by LaTeX for processing by LaTeX.

Focus is on correct display of symbols contained in these texts. E.g. the help texts of MHA configuration variables can be processed by this function. The contents of the MHAPLUGIN→_DOCUMENTATION is already in LaTeX format and should not be processed by this function.

Returns

A copy of s with various symbols escaped for LaTeX processing

Parameters

s	Text not ready for LaTeX
iscolored	if true, the complete returned text is surrounded with "\\color{monitorcolor}{" and "}"

5.95.1.2 print_plugin_references() static void print_plugin_references (

```

const std::set< std::string > & all_categories,
std::map< std::string, std::vector< std::string > > main_category_<-
plugins,
std::map< std::string, std::vector< std::string > > additional_category_<-
_plugins,
std::ofstream & ofile,
const std::string & category_macro ) [static]

```

Function prints an overview of all categories and their associated plugins into the document.

Parameters

<i>all_categories</i>	A sorted container with all category names
<i>main_category_plugins</i>	map of main categories to plugin names
<i>additional_category_plugins</i>	map of tags to plugin names
<i>ofile</i>	Latex document is produced by writing output to this stream

5.95.1.3 create_latex_doc() std::vector<std::string> create_latex_doc (
 std::map< std::string, std::string > & doc,
 const std::string & plugname,
 const std::string & plugin_macro)

Loads the plugin, creates the latex documentation for the plugin, and adds the latex documentation for this plugin to the plugin's main category entry in doc.

Returns

the vector of all categories.

Parameters

<i>doc</i>	map of main categories to a string containint the documentation of all plugins in that categories. The documentation of the current plugin will be appended to the existing documentation of its main category. Will be created if non-existant.
<i>plugname</i>	Name of the MHA plugin to process
<i>plugin_macro</i>	name of the LaTeX section macro that documents a single plugin (e.g. "section", "subsection", "subsubsection", ...)

5.95.1.4 main() int main (

```
    int argc,
    char ** argv )
```

5.96 gsc_adaptive_stage.cpp File Reference

5.97 gsc_adaptive_stage.hh File Reference

Classes

- class `gsc_adaptive_stage::gsc_adaptive_stage`

Namespaces

- `gsc_adaptive_stage`

Variables

- constexpr `mha_real_t gsc_adaptive_stage::DELT =1e-12`
Small constant to ensure no division by zero occurs.

5.98 gsc_adaptive_stage_if.cpp File Reference

5.99 gsc_adaptive_stage_if.hh File Reference

Classes

- class `gsc_adaptive_stage::gsc_adaptive_stage_if`
Plugin interface class.

Namespaces

- `gsc_adaptive_stage`

5.100 gtfb_analyzer.cpp File Reference

Gammatone Filterbank Analyzer Plugin.

Classes

- struct **gtnb_analyzer::gtnb_analyzer_cfg_t**
Configuration for Gammatone Filterbank Analyzer.
- class **gtnb_analyzer::gtnb_analyzer_t**
Gammatone Filterbank Analyzer Plugin.

Namespaces

- **gtnb_analyzer**

Functions

- static const **mha_complex_t & filter_complex** (const **mha_complex_t &input**, const **mha_complex_t &coeff**, **mha_complex_t *states**, unsigned **orders**)
Filters a complex input sample with the given filter coefficient.
- static const **mha_complex_t & filter_real** (**mha_real_t input**, **mha_complex_t &tmp←_complex**, const **mha_complex_t &coeff**, **mha_complex_t *states**, unsigned **orders**, const **mha_complex_t &normphase**)
Filters a real input sample with the given filter coefficient and applies the given normalization with phase correction.

5.100.1 Detailed Description

Gammatone Filterbank Analyzer Plugin.

5.100.2 Function Documentation

```
5.100.2.1 filter_complex() static const mha_complex_t& filter_complex (  

    const mha_complex_t & input,  

    const mha_complex_t & coeff,  

    mha_complex_t * states,  

    unsigned orders ) [inline], [static]
```

Filters a complex input sample with the given filter coefficient.

No normalization takes place. The implementation is tail-recursive and to exploit compiler optimization.

Parameters

<i>input</i>	The complex input sample
<i>coeff</i>	The complex filter coefficient
<i>states</i>	Pointer to the array of complex filter states.
<i>orders</i>	The filter order

Returns

A const ref to the filtered sample

```
5.100.2.2 filter_real() static const mha_complex_t& filter_real (
    mha_real_t input,
    mha_complex_t & tmp_complex,
    const mha_complex_t & coeff,
    mha_complex_t * states,
    unsigned orders,
    const mha_complex_t & normphase ) [inline], [static]
```

Filters a real input sample with the given filter coefficient and applies the given normalization with phase correction.

Parameters

<i>input</i>	The real input sample
<i>tmp_complex</i>	A reference to a mha_complex_t (p. 744) used for intermediate results. No assumptions should be made about the state of tmp_complex after the return of filter_real. This is an optimization to reduce the number of dtor/ctor calls of mha_complex_t (p. 744)
<i>coeff</i>	The complex filter coefficient
<i>states</i>	Pointer to the array of complex filter states.
<i>orders</i>	The filter order
<i>normphase</i>	Normalization coefficient including the phase correction

Returns

A const ref to the filtered sample

5.101 gtfb_simd.cpp File Reference

Gammatone Filterbank Analyzer Plugin using SIMD.

Classes

- class **gtfb_simd_cfg_t**
- class **gtfb_simd_t**

Macros

- #define **MXCSR_DAZ** (1 << 6) /* Enable denormals are zero mode */
- #define **MXCSR_FTZ** (1 << 15) /* Enable flush to zero mode */
- #define **check_alignment**(ptr, alignment)
Checks alignment of pointer address.
- #define **add4f**(a, b) __builtin_ia32_addps(a,b)
- #define **sub4f**(a, b) __builtin_ia32_subps(a,b)
- #define **mul4f**(a, b) __builtin_ia32_mulps(a,b)

Functions

- void **filter_sisd_complex** (const unsigned bands, const unsigned order, const **mha_complex_t** *inputs, **mha_complex_t** *outputs, const **mha_complex_t** *coefficients, **mha_complex_t** *states)
Filters one sample per band, using SISD operations and the mha_complex operations.
- void **filter_sisd_real** (const unsigned bands, const unsigned order, const **mha_complex_t** *inputs, **mha_complex_t** *outputs, const **mha_complex_t** *coefficients, **mha_complex_t** *states)
Filters one sample per band, using SISD operations and real operations (operating on real and imaginary part as necessary).
- void **filter_simd** (const unsigned bands, const unsigned order, const **mha_real_t** *rinputs, const **mha_real_t** *iinputs, **mha_real_t** *routputs, **mha_real_t** *ioutputs, const **mha_real_t** *rcoefficients, const **mha_real_t** *icoefficients, **mha_real_t** *rstates, **mha_real_t** *istates)
Filters one sample per band, using SIMD operations on float32 To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameters rstates and istates).

5.101.1 Detailed Description

Gammatone Filterbank Analyzer Plugin using SIMD.

A single-instruction-multiple-data (SIMD) implementation of a gammatone filterbank (GTFB)

Not all functions in this file are actually used. Read the functions in this file as a path to convert an algorithm, here complex-valued gammatone filtering as introduced in Hohmann 2002, from a single-instruction-single-data (SISD) implementation that uses complex arithmetic operations to a SIMD implementation of the same, splitting the complex arithmetic operations into their defining real operations, i.e (a+b).real==a.real+b.real, (a+b).imag==a.imag+b.imag, (a*b).real==a.real*b.real-a.imag*b.imag, (a*b).imag==a.real*b.imag+a.imag*b.real.

5.101.2 Macro Definition Documentation

5.101.2.1 MXCSR_DAZ `#define MXCSR_DAZ (1 << 6) /* Enable denormals are zero mode */`

5.101.2.2 MXCSR_FTZ `#define MXCSR_FTZ (1 << 15) /* Enable flush to zero mode */`

5.101.2.3 check_alignment `#define check_alignment(`
`ptr,`
`alignment)`

Checks alignment of pointer address.

Parameters

<i>ptr</i>	pointer to check
<i>alignment</i>	required alignment

Exceptions

MHA_Error (p. [763](#)) if ptr is not aligned as required.

5.101.2.4 add4f `#define add4f(`
`a,`
`b) __builtin_ia32_addps(a,b)`

5.101.2.5 sub4f `#define sub4f(`
`a,`
`b) __builtin_ia32_subps(a,b)`

5.101.2.6 mul4f #define mul4f(
 a,
 b) __builtin_ia32_mulps(a,b)

5.101.3 Function Documentation

5.101.3.1 filter_sisd_complex() void filter_sisd_complex (
 const unsigned bands,
 const unsigned order,
 const mha_complex_t * inputs,
 mha_complex_t * outputs,
 const mha_complex_t * coefficients,
 mha_complex_t * states)

Filters one sample per band, using SISD operations and the mha_complex operations.

To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameter states).

This function is not actually used in this plugin, but can be used for testing. It implements the Hohmann 2002 filtering in the most readable form, and is translated towards a SIMD implementation in the following functions.

Parameters

<i>bands</i>	Number of total bands to compute (i.e. input_channels * num_frequencies)
<i>order</i>	Gammatone filter order
<i>inputs</i>	Pointer to array of complex input samples, only 1 sample per band
<i>outputs</i>	Pointer to array with space for output samples, 1 complex sample per band
<i>coefficients</i>	Pointer to array of recursive filter coefficient, 1 coefficient per band. The same coefficient is reused for all filter orders.
<i>states</i>	Pointer to array of complex filter states. Array size is bands*order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state array from filtering the previous sample again, unmodified. The filter state of band b, order o can be found at index [b+o*bands]

5.101.3.2 filter_sisd_real() void filter_sisd_real (
 const unsigned bands,

```
const unsigned order,
const mha_complex_t * inputs,
mha_complex_t * outputs,
const mha_complex_t * coefficients,
mha_complex_t * states ) [inline]
```

Filters one sample per band, using SISD operations and real operations (operating on real and imaginary part as necessary).

To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameter states).

This function is not actually used in this plugin, but can be used for testing. It reimplements filter_sisd_complex, but expands the complex operations into real arithmetics.

Parameters

<i>bands</i>	Number of total bands to compute (i.e. input_channels * num_frequencies)
<i>order</i>	Gammatone filter order
<i>inputs</i>	Pointer to array of complex input samples, only 1 sample per band
<i>outputs</i>	Pointer to array with space for output samples, 1 complex sample per band
<i>coefficients</i>	Pointer to array of recursive filter coefficient, 1 coefficient per band. The same coefficient is reused for all filter orders.
<i>states</i>	Pointer to array of complex filter states. Array size is bands*order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state array from filtering the previous sample again, unmodified. The filter state of band b, order o can be found at index [b+o*bands]

```
5.101.3.3 filter_simd() void filter_simd (
    const unsigned bands,
    const unsigned order,
    const mha_real_t * rinputs,
    const mha_real_t * iinputs,
    mha_real_t * routputs,
    mha_real_t * ioutputs,
    const mha_real_t * rcoefficients,
    const mha_real_t * icoefficients,
    mha_real_t * rstates,
    mha_real_t * istates ) [inline]
```

Filters one sample per band, using SIMD operations on float32 To process more than one sample, the function must be called repeatedly in correct order (from oldest sample to newest sample), and the calling function must preserve the filter state (parameters rstates and istates).

This reimplements filter_sisd_real, but uses the CPU's vector registers for the arithmetics, and is actually used by this plugin.

Parameters

<i>bands</i>	Number of total bands to compute (i.e. input_channels * num_frequencies) bands is also the size of the arrays pointed to by <i>rinputs</i> , <i>iinputs</i> , <i>routputs</i> , <i>ioutputs</i> , <i>rcoefficients</i> , <i>icoefficients</i> .
<i>order</i>	Gammatone filter order
<i>rinputs</i>	Pointer to array of the real part of input samples
<i>iinputs</i>	Pointer to array of the imaginary part of input samples
<i>routputs</i>	Pointer to array with space for the real parts of the output samples
<i>routputs</i>	Pointer to array with space for the real parts of the output samples
<i>rcoefficients</i>	Pointer to array of the real parts of the recursive filter coefficients
<i>icoefficients</i>	Pointer to array of the imaginary parts of the filter coefficients
<i>rstates</i>	Pointer to array of real parts of filter states. Array size is bands*order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state arrays from filtering the previous sample again, unmodified. The real part of the filter state of band b, order o can be found at index [b+o*bands]
<i>istates</i>	Pointer to array of imaginary parts of filter states. Array size is bands*order. Initialize all elements with zeros before filtering the first sample. Filter state values will be modified by the function. For filtering the next sample, this function needs the filter state arrays from filtering the previous sample again, unmodified. The imaginary part of the filter state of band b, order o can be found at index [b+o*bands]

5.102 gtfb_simple_bridge.cpp File Reference

Classes

- class **gtfb_simple_rt_t**
Runtime configuration class of gtfb_simple_bridge plugin.
- class **gtfb_simple_t**
Interface class of gtfb_simple_bridge plugin.

5.103 hann.cpp File Reference

Macros

- #define **PI** 3.14159265358979323846

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

5.103.1 Macro Definition Documentation

5.103.1.1 PI #define PI 3.14159265358979323846

5.103.2 Function Documentation

5.103.2.1 hannf() float* hannf (const unsigned int N)

5.103.2.2 hann() double* hann (const unsigned int N)

5.104 hann.h File Reference

Functions

- float * **hannf** (const unsigned int N)
- double * **hann** (const unsigned int N)

5.104.1 Function Documentation

5.104.1.1 hannf() float* hannf (const unsigned int N)

5.104.1.2 hann() double* hann (const unsigned int N)

5.105 identity.cpp File Reference

Classes

- class **identity_t**

5.106 ifftshift.cpp File Reference

Functions

- void **ifftshift** (**mha_wave_t** *spec)

5.106.1 Function Documentation

5.106.1.1 ifftshift() void ifftshift (
 mha_wave_t * *spec*)

5.107 ifftshift.h File Reference

Functions

- void **ifftshift** (**mha_wave_t** *spec)

5.107.1 Function Documentation

5.107.1.1 ifftshift() void ifftshift (
 mha_wave_t * *spec*)

5.108 iirfilter.cpp File Reference

Classes

- class **iirfilter_t**

5.109 level_matching.cpp File Reference

Macros

- #define **PATCH_VAR**(var) patchbay.connect(&var.valuechanged, this, & **level_matching**::**level_matching_t**::**update_cfg**)
- #define **INSERT_PATCH**(var) **insert_member**(var); **PATCH_VAR**(var)

5.109.1 Macro Definition Documentation

5.109.1.1 PATCH_VAR #define PATCH_VAR(
 var) patchbay.connect (&var.valuechanged, this, & **level_matching**::**level_matching_t**::**update_cfg**)

5.109.1.2 INSERT_PATCH #define INSERT_PATCH(
 var) **insert_member**(var); **PATCH_VAR**(var)

5.110 level_matching.hh File Reference

Classes

- class **level_matching::channel_pair**
- class **level_matching::level_matching_config_t**
- class **level_matching::level_matching_t**

Namespaces

- **level_matching**

5.111 levelmeter.cpp File Reference

Classes

- class **levelmeter_t**

Macros

- #define **PASCALE** 93.979400086720374929

5.111.1 Macro Definition Documentation

5.111.1.1 **PASCALE** #define PASCALE 93.979400086720374929

5.112 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)

5.112.1 Macro Definition Documentation

5.112.1.1 **PATCH_VAR** #define PATCH_VAR(var) patchbay.connect(&var.valuechanged, this, & **Ipc::update_cfg**)

5.112.1.2 **INSERT_PATCH** #define INSERT_PATCH(var) **insert_member**(var); **PATCH_VAR**(var)

5.112.2 Function Documentation

```
5.112.2.1 Levinson2() void Levinson2 (
    unsigned int P,
    const std::vector< mha_real_t > & R,
    std::vector< mha_real_t > & A )
```

5.113 Ipc.h File Reference

Classes

- class **Ipc_config**
- class **Ipc**

5.114 Ipc_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)

5.114.1 Macro Definition Documentation

```
5.114.1.1 PATCH_VAR #define PATCH_VAR(
    var ) patchbay.connect (&var.valuechanged, this, & lpc_bl_predictor::update_cfg)
```

```
5.114.1.2 INSERT_PATCH #define INSERT_PATCH(
    var ) insert_member(var); PATCH_VAR(var)
```

5.115 **lpc_bl_predictor.h** File Reference

Classes

- class **lpc_bl_predictor_config**
- class **lpc_bl_predictor**

Macros

- #define **EPSILON** 1e-10

5.115.1 Macro Definition Documentation

5.115.1.1 **EPSILON** #define EPSILON 1e-10

5.116 **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)

5.116.1 Macro Definition Documentation

5.116.1.1 **PATCH_VAR** #define PATCH_VAR(var) patchbay.connect (&var.valuechanged, this, & **lpc_burglattice::update_cfg**)

5.116.1.2 **INSERT_PATCH** #define INSERT_PATCH(var) **insert_member**(var); **PATCH_VAR**(var)

5.117 **lpc_burg-lattice.h** File Reference

Classes

- class **lpc_burglattice_config**
- class **lpc_burglattice**

Macros

- #define **EPSILON** 1e-10

5.117.1 Macro Definition Documentation

5.117.1.1 **EPSILON** #define EPSILON 1e-10

5.118 **Isl2ac.cpp** File Reference

5.119 **Isl2ac.hh** File Reference

Classes

- class **Isl2ac::save_var_t**
LSL to AC bridge variable.
- class **Isl2ac::cfg_t**
*Runtime configuration class of the **Isl2ac** (p. 96) plugin.*
- class **Isl2ac::Isl2ac_t**
*Plugin class of **Isl2ac** (p. 96).*

Namespaces

- **Isl2ac**

Enumerations

- enum **Isl2ac::overrun_behavior** { **Isl2ac::overrun_behavior::Discard** =0, **Isl2ac::overrun_behavior::Ignore** }

5.120 matlab_wrapper.cpp File Reference

5.121 matlab_wrapper.hh File Reference

Classes

- struct `matlab_wrapper::types< T >`
- struct `matlab_wrapper::types< MHA_WAVEFORM >`
- struct `matlab_wrapper::types< MHA_SPECTRUM >`
- class `matlab_wrapper::matlab_wrapper_rt_cfg_t`
Thin wrapper around the emxArray containing the user defined configuration variables.
- class `matlab_wrapper::callback`
Utility class connecting a user_config_t instance to its corresponding configuration parser.
- class `matlab_wrapper::matlab_wrapper_t`
Matlab wraper plugin interface class.
- class `matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t`
Wrapper class around the matlab-generated library.

Namespaces

- **matlab_wrapper**
Namespace where all classes of the matlab wrapper plugin live.

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

5.121.1 Macro Definition Documentation

5.121.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN #define MHAPLUGIN_OVERLOAD_OUTDO←
MAIN

5.122 matrixmixer.cpp File Reference

Classes

- class `matrixmixer::cfg_t`
- class `matrixmixer::matmix_t`

Namespaces

- **matrixmixer**

5.123 mconv.cpp File Reference

Classes

- class **mconv::MConv**

Namespaces

- **mconv**

5.124 mha.cpp File Reference

Functions

- int **mhamain** (int argc, char *argv[])
- int **main** (int argc, char *argv[])

5.124.1 Function Documentation

5.124.1.1 mhamain() int mhamain (
 int *argc*,
 char * *argv*[])

5.124.1.2 main() int main (
 int *argc*,
 char * *argv*[])

5.125 mha.hh File Reference

common types for MHA kernel, MHA framework applications and external plugins

Classes

- struct **mha_complex_t**
Type for complex floating point values.
- struct **mha_complex_test_array_t**
*Several places in MHA rely on the fact that you can cast an array of **mha_complex_t** (p. 744) `c[]` to an array of **mha_real_t** `r[]` with `r[0] == c[0].re` `r[1] == c[0].im` `r[2] == c[1].re` ...*
- struct **mha_real_test_array_t**
- struct **mha_direction_t**
Channel source direction structure.
- struct **mha_channel_info_t**
Channel information structure.
- struct **mha_wave_t**
Waveform signal structure.
- struct **mha_spec_t**
- struct **mha_audio_descriptor_t**
*Description of an audio fragment (planned as a replacement of **mhaconfig_t** (p. 850)).*
- struct **mha_audio_t**
*An audio fragment in the openMHA (planned as a replacement of **mha_wave_t** (p. 839) and **mha_spec_t** (p. 793)).*
- 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** 15
Minor version number of MHA.
- #define **MHA_VERSION_RELEASE** 0
Release number of MHA.
- #define **MHA_VERSION_BUILD** 0
Build number of MHA (currently unused)
- #define **MHA_STRUCT_SIZEMATCH** (unsigned int)((sizeof(**mha_real_t**)==4)+2*(sizeof(**mha_complex_t**)==8)+4*(sizeof(**mha_wave_t**)==8+2*sizeof(void*))+8*(sizeof(**mha_spec_t**)==8+2*sizeof(void*))+16*(sizeof(**mhaconfig_t**)==24))

- Test number for structure sizes.*
- #define **MHA_VERSION** (unsigned int)((**MHA_STRUCT_SIZEMATCH** | (**MHA_VERSION_RELEASE** << 8) | (**MHA_VERSION_MINOR** << 16) | (**MHA_VERSION_MAJOR** << 24)))
Full version number of MHA kernel.
 - #define **MHA_VERSION_STRING** **MHA_XSTRF(MHA_VERSION_MAJOR)** "." **MHA_XSTRF(MHA_VERSION_MINOR)**
Version string of MHA kernel (major.minor)
 - #define **MHA_RELEASE_VERSION_STRING** **MHA_XSTRF(MHA_VERSION_MAJOR OR)** "." **MHA_XSTRF(MHA_VERSION_MINOR)** "." **MHA_XSTRF(MHA_VERSION_RELEASE)**
Version string of MHA kernel (major.minor.release)
 - #define **MHA_WAVEFORM** 0
 - #define **MHA_SPECTRUM** 1
 - #define **MHA_DOMAIN_MAX** 2
 - #define **MHA_DOMAIN_UNKNOWN** **MHA_DOMAIN_MAX**
 - #define **MHA_AC_UNKNOWN** 0
 - #define **MHA_AC_CHAR** 1
 - #define **MHA_AC_INT** 2
 - #define **MHA_AC_MHAREAL** 3
 - #define **MHA_AC_FLOAT** 4
 - #define **MHA_AC_DOUBLE** 5
 - #define **MHA_AC_MHACOMPLEX** 6
 - #define **MHA_AC_VEC_FLOAT** 51
 - #define **MHA_AC_USER** 1000

Typedefs

- typedef unsigned int **mha_domain_t**
- typedef float **mha_real_t**
openMHA type for real numbers
- typedef void * **mha_fft_t**
Handle for an FFT object.
- typedef struct **algo_comm_t algo_comm_t**
- typedef unsigned int(* **MHAGetVersion_t**) (void)
- typedef int(* **MHAInit_t**) (**algo_comm_t** algo_comm, const char *chain, const char *algo, void **h)
- typedef int(* **MHAPrepare_t**) (void *h, **mhaconfig_t** *cfg)
- typedef int(* **MHAReset_t**) (void *h)
- typedef void(* **MHADestroy_t**) (void *h)
- typedef int(* **MHASet_t**) (void *h, const char *cmd, char *retval, unsigned int len)
- typedef std::string(* **MHASetcpp_t**) (void *h, const std::string &command)
- typedef 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)`

Variables

- `const typedef char *(* MHAStrError_t)(void *h, int err)`
- `const typedef char *(* MHAPluginDocumentation_t)(void)`
- `const typedef char *(* MHAPluginCategory_t)(void)`

5.125.1 Detailed Description

common types for MHA kernel, MHA framework applications and external plugins

5.125.2 Macro Definition Documentation

5.125.2.1 MHA_CALLBACK_TEST `#define MHA_CALLBACK_TEST(` `x)`

Test macro to compare function type definition and declaration.

5.125.2.2 MHA_CALLBACK_TEST_PREFIX `#define MHA_CALLBACK_TEST_PREFIX(` `prefix,` `x)`

5.125.2.3 MHA_XSTRF `#define MHA_XSTRF(` `x)` `MHA_STRF(` `x)`

5.125.2.4 MHA_STRF #define MHA_STRF(
 x) #x

5.125.2.5 MHA_VERSION_MAJOR #define MHA_VERSION_MAJOR 4

Major version number of MHA.

5.125.2.6 MHA_VERSION_MINOR #define MHA_VERSION_MINOR 15

Minor version number of MHA.

5.125.2.7 MHA_VERSION_RELEASE #define MHA_VERSION_RELEASE 0

Release number of MHA.

5.125.2.8 MHA_VERSION_BUILD #define MHA_VERSION_BUILD 0

Build number of MHA (currently unused)

5.125.2.9 MHA_STRUCT_SIZEMATCH #define MHA_STRUCT_SIZEMATCH ((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.

5.125.2.10 MHA_VERSION #define MHA_VERSION (unsigned int)((MHA_STRUCT_SIZEMA→
TCH | (MHA_VERSION_RELEASE << 8) | (MHA_VERSION_MINOR << 16) | (MHA_VERSION_MAJOR
<< 24)))

Full version number of MHA kernel.

5.125.2.11 MHA_VERSION_STRING #define MHA_VERSION_STRING MHA_XSTRF(MHA_VERSION_MAJOR) ". " MHA_XSTRF(MHA_VERSION_MINOR)

Version string of MHA kernel (major.minor)

5.125.2.12 MHA_RELEASE_VERSION_STRING #define MHA_RELEASE_VERSION_STRING MHA_XSTRF(MHA_VERSION_MAJOR) ". " MHA_XSTRF(MHA_VERSION_MINOR) ". " MHA_XSTRF(MHA_VERSION_RELEASE)

Version string of MHA kernel (major.minor.release)

5.125.2.13 MHA_WAVEFORM #define MHA_WAVEFORM 0

5.125.2.14 MHA_SPECTRUM #define MHA_SPECTRUM 1

5.125.2.15 MHA_DOMAIN_MAX #define MHA_DOMAIN_MAX 2

5.125.2.16 MHA_DOMAIN_UNKNOWN #define MHA_DOMAIN_UNKNOWN MHA_DOMAIN_MAX

5.125.2.17 MHA_AC_UNKNOWN #define MHA_AC_UNKNOWN 0

5.125.2.18 MHA_AC_CHAR #define MHA_AC_CHAR 1

5.125.2.19 MHA_AC_INT #define MHA_AC_INT 2

5.125.2.20 MHA_AC_MHAREAL #define MHA_AC_MHAREAL 3

5.125.2.21 MHA_AC_FLOAT #define MHA_AC_FLOAT 4

5.125.2.22 MHA_AC_DOUBLE #define MHA_AC_DOUBLE 5

5.125.2.23 MHA_AC_MHACOMPLEX #define MHA_AC_MHACOMPLEX 6

5.125.2.24 MHA_AC_VEC_FLOAT #define MHA_AC_VEC_FLOAT 51

5.125.2.25 MHA_AC_USER #define MHA_AC_USER 1000

5.125.3 Typedef Documentation

5.125.3.1 mha_domain_t typedef unsigned int mha_domain_t

5.125.3.2 algo_comm_t typedef struct algo_comm_t algo_comm_t

5.125.3.3 MHAGetVersion_t `typedef unsigned int(* MHAGetVersion_t) (void)`

5.125.3.4 MHAInit_t `typedef int(* MHAInit_t) (algo_comm_t algo_comm, const char *chain, const char *algo, void **h)`

5.125.3.5 MHAPrepare_t `typedef int(* MHAPrepare_t) (void *h, mhaconfig_t *cfg)`

5.125.3.6 MHARelease_t `typedef int(* MHARelease_t) (void *h)`

5.125.3.7 MHADestroy_t `typedef void(* MHADestroy_t) (void *h)`

5.125.3.8 MHASet_t `typedef int(* MHASet_t) (void *h, const char *cmd, char *retval, unsigned int len)`

5.125.3.9 MHASetcpp_t `typedef std::string(* MHASetcpp_t) (void *h, const std::string &command)`

5.125.3.10 MHAProc_wave2wave_t `typedef int(* MHAProc_wave2wave_t) (void *h, mha_wave_t *sIn, mha_wave_t **sOut)`

5.125.3.11 MHAProc_wave2spec_t `typedef int(* MHAProc_wave2spec_t) (void *h, mha_wave_t *sIn, mha_spec_t **sOut)`

5.125.3.12 MHAProc_spec2wave_t `typedef int(* MHAProc_spec2wave_t) (void *h,
mha_spec_t *sIn, mha_wave_t **sOut)`

5.125.3.13 MHAProc_spec2spec_t `typedef int(* MHAProc_spec2spec_t) (void *h,
mha_spec_t *sIn, mha_spec_t **sOut)`

5.125.4 Variable Documentation

5.125.4.1 MHAStrError_t `const typedef char*(* MHAStrError_t) (void *h, int err)`

5.125.4.2 MHAPluginDocumentation_t `const typedef char*(* MHAPluginDocumentation_t) (void)`

5.125.4.3 MHAPluginCategory_t `const typedef char*(* MHAPluginCategory_t) (void)`

5.126 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`

5.126.1 Macro Definition Documentation

5.126.1.1 AC_SUCCESS #define AC_SUCCESS 0

5.126.1.2 AC_INVALID_HANDLE #define AC_INVALID_HANDLE -1

5.126.1.3 AC_INVALID_NAME #define AC_INVALID_NAME -2

5.126.1.4 AC_STRING_TRUNCATED #define AC_STRING_TRUNCATED -3

5.126.1.5 AC_INVALID_OUTPTR #define AC_INVALID_OUTPTR -4

5.126.1.6 AC_TYPE_MISMATCH #define AC_TYPE_MISMATCH -5

5.126.1.7 AC_DIM_MISMATCH #define AC_DIM_MISMATCH -6

5.126.2 Variable Documentation

5.126.2.1 algo_comm_default algo_comm_t algo_comm_default

5.127 mha_algo_comm.h File Reference

Header file for Algorithm Communication.

Classes

- class **MHA_AC::spectrum_t**
*Insert a **MHASignal::spectrum_t** (p. 1244) class into the AC space.*
- class **MHA_AC::waveform_t**
*Insert a **MHASignal::waveform_t** (p. 1259) 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.

5.127.1 Detailed Description

Header file for Algorithm Communication.

5.128 mha_algo_comm.hh File Reference

Classes

- class **MHAKernel::comm_var_map_t**
- class **MHAKernel::algo_comm_class_t**

Namespaces

- **MHAKernel**

Macros

- #define **ALGO_COMM_ID_STR** "MFVK3jL5rmeus1XtggEI971aXCR/GU7RRehKz4k←
Qtrg="

Functions

- algo_comm_class_t * **MHAKernel::algo_comm_safe_cast** (void *)

Variables

- algo_comm_t **algo_comm_default**

5.128.1 Macro Definition Documentation

5.128.1.1 ALGO_COMM_ID_STR #define ALGO_COMM_ID_STR "MFVK3jL5rmeus1XtggE←
I971aXCR/GU7RRehKz4kQtrg="

5.128.2 Variable Documentation

5.128.2.1 algo_comm_default algo_comm_t algo_comm_default

5.129 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

5.129.1 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.

5.129.2 Macro Definition Documentation

5.129.2.1 __MHA_FUN__ #define __MHA_FUN__ __FUNC__

5.129.2.2 CHECK_EXPR #define CHECK_EXPR(
 x) {if(! (x)){throw MHA_Error(__FILE__, __LINE__, "The expression \"\" #x
"\\" is invalid.");}}

5.129.2.3 CHECK_VAR #define CHECK_VAR(
 x) {if(! (x)){throw MHA_Error(__FILE__, __LINE__, "The variable \"\" #x
"\\" is not defined.");}}

5.129.2.4 __declspec #define __declspec(
 p)

5.129.2.5 M_PI #define M_PI 3.14159265358979323846

Define pi if it is not defined yet.

5.129.2.6 MIN #define MIN(
 a,
 b) (((a) < (b)) ? (a) : (b))

Macro for minimum function.

5.129.2.7 MAX #define MAX(
 a,
 b) (((a) > (b)) ? (a) : (b))

Macro for maximum function.

5.129.2.8 MHA_EAR_LEFT #define MHA_EAR_LEFT 0

5.129.2.9 MHA_EAR_RIGHT #define MHA_EAR_RIGHT 1

5.129.2.10 MHA_EAR_MAX #define MHA_EAR_MAX 2

5.130 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**]

5.130.1 Macro Definition Documentation

5.130.1.1 STRLEN #define STRLEN 0x1000

5.130.2 Function Documentation

5.130.2.1 mha_strerror() const char* mha_strerror (int *mhaerrno*)

5.130.2.2 mha_set_user_error() void mha_set_user_error (const char * *str*)

5.130.3 Variable Documentation

5.130.3.1 next_except_str char *next_except_str*[**STRLEN**] = ""

5.130.3.2 cstr_strerror const char* *cstr_strerror*[**MHA_ERR_USER**]

5.131 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*)

5.131.1 Macro Definition Documentation

5.131.1.1 MHA_ERR_SUCCESS #define MHA_ERR_SUCCESS 0

5.131.1.2 MHA_ERR_UNKNOWN #define MHA_ERR_UNKNOWN 1

5.131.1.3 MHA_ERR_INVALID_HANDLE #define MHA_ERR_INVALID_HANDLE 2

5.131.1.4 MHA_ERR_NULL #define MHA_ERR_NULL 3

5.131.1.5 MHA_ERR_VARRANGE #define MHA_ERR_VARRANGE 4

5.131.1.6 MHA_ERR_VARFMT #define MHA_ERR_VARFMT 5

5.131.1.7 MHA_ERR_USER #define MHA_ERR_USER 10000

5.131.2 Function Documentation

5.131.2.1 mha_strerror() const char* mha_strerror (int mhaerrno)

5.131.2.2 mha_set_user_error() void mha_set_user_error (const char * str)

5.132 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,...)`

5.132.1 Detailed Description

Implementation of openMHA error handling.

This file forms a separate library.

5.132.2 Function Documentation

5.132.2.1 mha_debug()

```
void mha_debug (
    const char * fmt,
    ... )
```

5.133 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. 763).*
- `#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. 763) with the values.*

Functions

- `void mha_debug (const char *fmt,...) __attribute__((__format__(printf`
Print an info message (stderr on Linux, OutputDebugString in Windows).
- `unsigned mha_error_helpers::digits (unsigned n)`
Compute number of decimal digits required to represent an unsigned integer.
- `unsigned mha_error_helpers::snprintf_required_length (const char *formatstring,...)`
snprintf_required_length Compute the number of bytes (excluding the terminating nul) required to store the result of an snprintf.

5.133.1 Macro Definition Documentation

5.133.1.1 **Getmsg** `#define Getmsg(` `e) ((e).get_msg())`

5.134 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.

5.135 mha_events.cpp File Reference

5.136 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.

5.137 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::expfilt (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)**
Transform the test frequency into the relative position on the filter flank of the given frequency band.

Variables

- **mha_real_t MHAOvIFilter::barkscale::vfreq [BARKSCALE_ENTRIES]**
- **mha_real_t MHAOvIFilter::barkscale::vbark [BARKSCALE_ENTRIES]**

5.137.1 Macro Definition Documentation

5.137.1.1 BARKSCALE_ENTRIES #define BARKSCALE_ENTRIES 50

5.137.2 Function Documentation

```
5.137.2.1 filtershapefun() mha_real_t filtershapefun (
    mha_real_t f,
    MHAOvlFilter::band_descriptor_t b,
    mha_real_t plateau )
```

Transform the test frequency into the relative position on the filter flank of the given frequency band.

Parameters

<i>f</i>	Test frequency in units corresponding to the chosen frequency scale
<i>b</i>	Descriptor of a single filter bank band: E.g. contains center frequencies of this and the two adjacent bands, and the crossover ("edge") frequencies of this band.
<i>plateau</i>	For non-rectangular filter shapes, specifies what frequency portion of the band around its center frequency should have no attenuation applied.

Precondition

$0 \leq \text{plateau} \leq 1$

Returns

The position of frequency *f* on the filter flank as follows: A returned position of 0 means that *f* is equal to the band's center frequency, or should be treated the same as the center frequency (i.e. is within the band's plateau). A returned position of -1 means that *f* is \leq the lowest frequency of the filter flank (or is an even lower frequency). A returned value of -0.5 means that *f* is equal to the lower edge frequency. Positive returned values have equivalent meanings for the high half of the filter flank.

5.138 mha_fftfb.hh File Reference

Classes

- class **MHAOvlFilter::band_descriptor_t**
- class **MHAOvlFilter::scale_var_t**
- class **MHAOvlFilter::fscale_t**
- class **MHAOvlFilter::fscale_bw_t**
- class **MHAOvlFilter::fftfb_vars_t**
 - Set of configuration variables for FFT-based overlapping filters.*
- class **MHAOvlFilter::fspacing_t**
 - Class for frequency spacing, used by filterbank shape generator class.*
- class **MHAOvlFilter::fftfb_t**
 - FFT based overlapping filter bank.*
- class **MHAOvlFilter::overlap_save_filterbank_t**
 - A time-domain minimal phase filter bank with frequency shapes from **MHAOvlFilter::fftfb_t** (p. 1002).*
- class **MHAOvlFilter::overlap_save_filterbank_t::vars_t**
- class **MHAOvlFilter::overlap_save_filterbank_analytic_t**
- class **MHAOvlFilter::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)`

5.139 mha_fifo.cpp File Reference

5.140 mha_fifo.h File Reference

Classes

- class **mha_fifo_t< T >**
A FIFO class.
- class **mha_fifo_lf_t< T >**
A lock-free FIFO class for transferring data from a producer thread to a consumer thread.
- class **mha_drifter_fifo_t< T >**
A FIFO class for blocksize adaptation without Synchronization.
- class **mha_fifo_thread_platform_t**
Abstract base class for synchronizing multithreaded (producer/consumer) fifo operations.
- class **mha_fifo_posix_threads_t**
- class **mha_fifo_thread_guard_t**
Simple Mutex Guard Class.
- class **mha_fifo_lw_t< T >**
This FIFO uses locks to synchronize access.
- class **mha_dblbuf_t< FIFO >**
The doublebuffer adapts blocksizes between an outer process, which provides input data and takes output data, and an inner process, which processes the input signal and generates output data using a different block size than the outer process.
- class **mha_rt_fifo_element_t< T >**
*Object wrapper for **mha_rt_fifo_t** (p. 789).*
- 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`

5.140.1 Macro Definition Documentation

5.140.1.1 mha_fifo_thread_platform_implementation_t #define mha_fifo_thread_←
platform_implementation_t mha_fifo_posix_threads_t

5.141 mha_filter.cpp File Reference

Functions

- std::vector< **mha_real_t** > **diff_coeffs ()**

5.141.1 Function Documentation

5.141.1.1 diff_coeffs() std::vector< **mha_real_t** > diff_coeffs ()

5.142 mha_filter.hh File Reference

Header file for IIR filter classes.

Classes

- class **MHAFilter::filter_t**
Generic IIR filter class.
- class **MHAFilter::diff_t**
Differentiator class (non-normalized)
- class **MHAFilter::o1_ar_filter_t**
First order attack-release lowpass filter.
- class **MHAFilter::o1flt_lowpass_t**
First order low pass filter.
- class **MHAFilter::o1flt_maxtrack_t**
First order maximum tracker.
- class **MHAFilter::o1flt_mintrack_t**
First order minimum tracker.

- class **MHAFilter::iir_filter_state_t**
- class **MHAFilter::iir_filter_t**
IIR filter class wrapper for integration into parser structure.
- class **MHAFilter::adapt_filter_state_t**
- class **MHAFilter::adapt_filter_param_t**
- class **MHAFilter::adapt_filter_t**
Adaptive filter.
- class **MHAFilter::fftfilter_t**
FFT based FIR filter implementation.
- class **MHAFilter::fftfilterbank_t**
FFT based FIR filterbank implementation.
- struct **MHAFilter::transfer_function_t**
a structure containing a source channel number, a target channel number, and an impulse response.
- struct **MHAFilter::transfer_matrix_t**
A sparse matrix of transfer function partitionss.
- class **MHAFilter::partitioned_convolution_t**
A filter class for partitioned convolution.
- struct **MHAFilter::partitioned_convolution_t::index_t**
Bookkeeping class.
- class **MHAFilter::smoothspec_t**
Smooth spectral gains, create a windowed impulse response.
- class **MHAFilter::resampling_filter_t**
Hann shaped low pass filter for resampling.
- class **MHAFilter::polyphase_resampling_t**
A class that performs polyphase resampling.
- class **MHAFilter::blockprocessing_polyphase_resampling_t**
A class that does polyphase resampling and takes into account block processing.
- class **MHAFilter::iir_ord1_real_t**
First order recursive filter.

Namespaces

- **MHAFilter**
Namespace for IIR and FIR filter classes.

Functions

- template<typename T , typename std::enable_if< std::is_floating_point< T >::value, T >::type * = nullptr>
void **MHAFilter::make_friendly_number** (T &x)
- void **MHAFilter::o1_lp_coeffs** (const **mha_real_t** tau, const **mha_real_t** fs, **mha_real_t** &c1, **mha_real_t** &c2)
Set first order filter coefficients from time constant and sampling rate.

- void **MHAFilter::butter_stop_ord1** (double *A, double *B, double f1, double f2, double fs)
Setup a first order butterworth band stop filter.
- std::vector< float > **MHAFilter::fir_lp** (float f_pass_, float f_stop_, float fs_, unsigned order_)
Setup a nth order fir low pass filter.
- **MHASignal::waveform_t * MHAFilter::spec2fir** (const **mha_spec_t** *spec, const unsigned int fftlen, const **MHAWindow::base_t** &window, const bool minphase)
Create a windowed impulse response/FIR filter coefficients from a spectrum.
- unsigned **MHAFilter::gcd** (unsigned a, unsigned b)
greatest common divisor
- double **MHAFilter::sinc** (double x)
 $\sin(x)/x$ function, coping with $x=0$.
- std::pair< unsigned, unsigned > **MHAFilter::resampling_factors** (float source_← sampling_rate, float target_sampling_rate, float factor=1.0f)
Computes rational resampling factor from two sampling rates.

5.142.1 Detailed Description

Header file for IIR filter classes.

5.143 mha_generic_chain.cpp File Reference

Functions

- void **mhaconfig_compare** (**mhaconfig_t** req, **mhaconfig_t** avail, const char *cpref)

5.143.1 Function Documentation

```
5.143.1.1 mhaconfig_compare() void mhaconfig_compare (
    mhaconfig_t req,
    mhaconfig_t avail,
    const char * cpref )
```

5.144 mha_generic_chain.h File Reference

Classes

- class **mhachain::plugs_t**
- class **mhachain::chain_base_t**

Namespaces

- **mhachain**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**

5.144.1 Macro Definition Documentation

5.144.1.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN #define MHAPLUGIN_OVERLOAD_OUTDO←
MAIN

5.145 mha_git_commit_hash.cpp File Reference

Macros

- #define **GITCOMMITHASH** "independent-plugin-build"

Variables

- const char * **mha_git_commit_hash**

store git commit hash in every binary plugin to support reproducible research

5.145.1 Macro Definition Documentation

5.145.1.1 GITCOMMITHASH #define GITCOMMITHASH "independent-plugin-build"

5.145.2 Variable Documentation

5.145.2.1 mha_git_commit_hash const char* mha_git_commit_hash

store git commit hash in every binary plugin to support reproducible research

5.146 mha_git_commit_hash.hh File Reference

Variables

- const char * **mha_git_commit_hash**

store git commit hash in every binary plugin to support reproducible research

5.146.1 Variable Documentation

5.146.1.1 mha_git_commit_hash const char* mha_git_commit_hash

store git commit hash in every binary plugin to support reproducible research

5.147 mha_io_ifc.h File Reference

Typedefs

- typedef int(* **IOProcessEvent_t**) (void *handle, **mha_wave_t** *sIn, **mha_wave_t** **sOut)
Event handler for signal stream.
- typedef void(* **IOStoppedEvent_t**) (void *handle, int proc_err, int io_err)
Event handler for stop event.
- typedef void(* **IOStartedEvent_t**) (void *handle)
Event handler for start event.
- typedef int(* **IOInit_t**) (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- typedef int(* **IOPrepare_t**) (void *handle, int num_inchannels, int num_outchannels)
- typedef int(* **IOStart_t**) (void *handle)
- typedef int(* **IOStop_t**) (void *handle)
- typedef int(* **IOReset_t**) (void *handle)
- typedef int(* **IOSetVar_t**) (void *handle, const char *cmd, char *retval, unsigned int len)
- typedef void(* **IODestroy_t**) (void *handle)

Variables

- const typedef char *(* **IOStrError_t**)(void *handle, int err)

5.147.1 Typedef Documentation

5.147.1.1 IOProcessEvent_t `typedef int(* IOProcessEvent_t) (void *handle, mha_<→ wave_t *sIn, mha_wave_t **sOut)`

Event handler for signal stream.

This event handler needs to be realtime compatible. All signal path processing will be performed in this callback.

5.147.1.2 IOStoppedEvent_t `typedef void(* IOStoppedEvent_t) (void *handle, int proc_err, int io_err)`

Event handler for stop event.

This event handler needs to be realtime compatible. The function must return immediatly.

5.147.1.3 IOStartedEvent_t `typedef void(* IOStartedEvent_t) (void *handle)`

Event handler for start event.

This event handler needs to be realtime compatible. The function must return immediatly.

5.147.1.4 IOInit_t `typedef int(* IOInit_t) (int fragsize, float samplerate, IO-<→ ProcessEvent_t proc_event, void *proc_handle, IOStartedEvent_t start_event, void *start_handle, IOStoppedEvent_t stop_event, void *stop_handle, void **handle)`

5.147.1.5 IOPrepare_t `typedef int(* IOPrepare_t) (void *handle, int num_inchannels, int num_outchannels)`

5.147.1.6 IOStart_t `typedef int(* IOStart_t) (void *handle)`

5.147.1.7 IOStop_t `typedef int(* IOStop_t) (void *handle)`

5.147.1.8 IOReset_t `typedef int(* IOReset_t) (void *handle)`

5.147.1.9 IOSetVar_t `typedef int(* IOSetVar_t) (void *handle, const char *cmd, char *retval, unsigned int len)`

5.147.1.10 IODestroy_t `typedef void(* IODestroy_t) (void *handle)`

5.147.2 Variable Documentation

5.147.2.1 IOStrError_t `const typedef char*(* IOStrError_t) (void *handle, int err)`

5.148 mha_io_utils.cpp File Reference

5.149 mha_io_utils.hh File Reference

Namespaces

- **mhaiutils**

Functions

- `template<typename T >`
`T mhaiutils::to_int_clamped (float val)`

5.150 mha_multisrc.cpp File Reference

Namespaces

- **MHAMultiSrc**

Collection of classes for selecting audio chunks from multiple sources.

5.151 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.

5.152 mha_os.cpp File Reference

Functions

- bool **mha_hasenv** (const std::string &envvar)
Checks if environment variable exists.
- std::string **mha_getenv** (const std::string &envvar)
Get value of environment variable.
- void **mha_delenv** (const std::string &envvar)
Deletes environment variable from process environment if it exists.
- int **mha_setenv** (const std::string &envvar, const std::string & **value**)
Set value of environment variable.
- std::list< std::string > **mha_library_paths** ()
- std::list< std::string > **list_dir** (const std::string &path, const std::string &pattern)

5.152.1 Function Documentation

5.152.1.1 **mha_hasenv()** bool mha_hasenv (

```
const std::string & envvar )
```

Checks if environment variable exists.

Parameters

<code>envvar</code>	Name of environment variable to check
---------------------	---------------------------------------

Returns

true if the environment has a variable of this name

5.152.1.2 mha_getenv() `std::string mha_getenv (`
`const std::string & envvar)`

Get value of environment variable.

Parameters

<code>envvar</code>	Name of environment variable to retrieve
---------------------	--

Returns

content of environment variable if it exists, empty string if the environment variable does not exist

5.152.1.3 mha_delenv() `void mha_delenv (`
`const std::string & envvar)`

Deletes environment variable from process environment if it exists.

Parameters

<code>envvar</code>	Name of environment variable to delete
---------------------	--

5.152.1.4 mha_setenv() `int mha_setenv (`
`const std::string & envvar,`
`const std::string & value)`

Set value of environment variable.

Parameters

<code>envvar</code>	Name of environment variable to set
<code>value</code>	New content for environment variable

Returns

error code: 0 on success, OS dependent error code on failure

5.152.1.5 `mha_library_paths()` `std::list<std::string> mha_library_paths ()`

5.152.1.6 `list_dir()` `std::list<std::string> list_dir (` `const std::string & path,` `const std::string & pattern)`

5.153 mha_os.h File Reference

Classes

- class **`mha_stash_environment_variable_t`**
This class changes the value of an environment variable when constructed and restores the original state of the environment variable when destroyed.
- class **`dynamiclib_t`**
Wrapper class around a shared library.
- class **`pluginlib_t`**
Specialisation of `dynamiclib_t` (p. 445) for mha plugin libraries.

Macros

- `#define mha_loadlib(x) dlopen(x,RTLD_NOW)`
- `#define mha_freelib(x) dlclose(x)`
- `#define mha_freelib_success(x) (x == 0)`
- `#define mha_getlibfun(h, x) x ## _cb = (x ## _t)dlsym(h,#x)`
- `#define mha_getlibfun_checked(h, x) x ## _cb = (x ## _t)dlsym(h,#x);if(! x ## _cb) throw MHA_Error(__FILE__,__LINE__,"Function "#x " is undefined.")`
- `#define mha_loadlib_error(x) dlerror()`
- `#define mha_lib_extension ".so"`
- `#define mha_msleep(milliseconds) usleep((milliseconds)*1000)`
- `#define FMTsz "%zu"`
printf modifier to print integers of type size_t
- `#define MHA_RESOLVE(h, t) t ## _cb = (t ## _t)(h->resolve(#t))`
- `#define MHA_RESOLVE_CHECKED(h, t) t ## _cb = (t ## _t)(h->resolve_checked(#t))`

Typedefs

- `typedef void * mha_libhandle_t`

Functions

- `std::string mha_getenv (const std::string &envvar)`
Get value of environment variable.
- `bool mha_hasenv (const std::string &envvar)`
Checks if environment variable exists.
- `int mha_setenv (const std::string &envvar, const std::string & value)`
Set value of environment variable.
- `void mha_delenv (const std::string &envvar)`
Deletes environment variable from process environment if it exists.
- `std::list< std::string > mha_library_paths ()`
- `std::list< std::string > list_dir (const std::string &path, const std::string &pattern)`
- `void mha_hton (float *data, unsigned int len)`
- `void mha_ntoh (float *data, unsigned int len)`
- `void mha_hton (uint32_t *data, unsigned int len)`
- `void mha_ntoh (uint32_t *data, unsigned int len)`
- `void mha_hton (int32_t *data, unsigned int len)`
- `void mha_ntoh (int32_t *data, unsigned int len)`

5.153.1 Macro Definition Documentation

5.153.1.1 mha_loadlib `#define mha_loadlib(`
 `x) dlopen(x, RTLD_NOW)`

5.153.1.2 mha_freelib `#define mha_freelib(`
 `x) dlclose(x)`

5.153.1.3 mha_freelib_success `#define mha_freelib_success(`
 `x) (x == 0)`

5.153.1.4 mha_getlibfun #define mha_getlibfun(
 h,
 x) *x* ## _cb = (*x* ## _t)dlsym(*h*,#*x*)

5.153.1.5 mha_getlibfun_checked #define mha_getlibfun_checked(
 h,
 x) *x* ## _cb = (*x* ## _t)dlsym(*h*,#*x*);if(! *x* ## _cb) throw MHA_Error(_←
_FILE__, __LINE__, "Function " "#*x* " is undefined.")

5.153.1.6 mha_loadlib_error #define mha_loadlib_error(
 x) dlerror()

5.153.1.7 mha_lib_extension #define mha_lib_extension ".so"

5.153.1.8 mha_msleep #define mha_msleep(
 milliseconds) usleep(*milliseconds*)*1000)

5.153.1.9 FMTsz #define FMTsz "%zu"

printf modifier to print integers of type size_t

5.153.1.10 MHA_RESOLVE #define MHA_RESOLVE(
 h,
 t) *t* ## _cb = (*t* ## _t)(*h*->resolve(#*t*))

```
5.153.1.11 MHA_RESOLVE_CHECKED #define MHA_RESOLVE_CHECKED ( h, t ) t ## _cb = (t ## _t) (h->resolve_checked(#t))
```

5.153.2 Typedef Documentation

5.153.2.1 mha_libhandle_t `typedef void* mha_libhandle_t`

5.153.3 Function Documentation

5.153.3.1 mha_getenv() `std::string mha_getenv (const std::string & envvar)`

Get value of environment variable.

Parameters

<code>envvar</code>	Name of environment variable to retrieve
---------------------	--

Returns

content of environment variable if it exists, empty string if the environment variable does not exist

5.153.3.2 mha_hasenv() `bool mha_hasenv (const std::string & envvar)`

Checks if environment variable exists.

Parameters

<code>envvar</code>	Name of environment variable to check
---------------------	---------------------------------------

Returns

true if the environment has a variable of this name

5.153.3.3 mha_setenv() `int mha_setenv (`
 `const std::string & envvar,`
 `const std::string & value)`

Set value of environment variable.

Parameters

<code>envvar</code>	Name of environment variable to set
<code>value</code>	New content for environment variable

Returns

error code: 0 on success, OS dependent error code on failure

5.153.3.4 mha_delenv() `void mha_delenv (`
 `const std::string & envvar)`

Deletes environment variable from process environment if it exists.

Parameters

<code>envvar</code>	Name of environment variable to delete
---------------------	--

5.153.3.5 mha_library_paths() `std::list<std::string> mha_library_paths ()`

5.153.3.6 list_dir() `std::list<std::string> list_dir (`
 `const std::string & path,`
 `const std::string & pattern)`

5.153.3.7 mha_hton() [1/3] void mha_hton (float * *data*, unsigned int *len*) [inline]

5.153.3.8 mha_ntoh() [1/3] void mha_ntoh (float * *data*, unsigned int *len*) [inline]

5.153.3.9 mha_hton() [2/3] void mha_hton (uint32_t * *data*, unsigned int *len*) [inline]

5.153.3.10 mha_ntoh() [2/3] void mha_ntoh (uint32_t * *data*, unsigned int *len*) [inline]

5.153.3.11 mha_hton() [3/3] void mha_hton (int32_t * *data*, unsigned int *len*) [inline]

5.153.3.12 mha_ntoh() [3/3] void mha_ntoh (int32_t * *data*, unsigned int *len*) [inline]

5.154 mha_parser.cpp File Reference

Namespaces

- **MHAParser**
Name space for the openMHA-Parser configuration language.
- **MHAParser::StrCnv**
String converter namespace.

Macros

- #define **MHAPLATFORM** "undefined-linux"

Functions

- int **MHAParser::get_precision ()**
- int **MHAParser::StrCnv::num_brackets** (const std::string &s)
count number of brackets
- int **MHAParser::StrCnv::bracket_balance** (const std::string &s)
- static std::ostream & **write_float** (std::ostream &o, const float &f)
- static std::string **parse_1_float** (const std::string &s, **mha_real_t** &v)
This internal function parses a floating point number from the beginning of a string.
- static void **check_parenthesis_complex** (const std::string &str)
This function checks for unbalanced parenthesis in the string containing complex number.
- static int **check_sign_complex** (const std::string &str)
This function checks for valid sign (b/w real & img).
- static std::string **parse_1_complex** (const std::string &s, **mha_complex_t** &v)
This internal function parses a complex number from the beginning of a string.

5.154.1 Macro Definition Documentation

5.154.1.1 **MHAPLATFORM** #define MHAPLATFORM "undefined-linux"

5.154.2 Function Documentation

5.154.2.1 **write_float()** static std::ostream& **write_float** (std::ostream & o, const float & f) [inline], [static]

5.154.2.2 **parse_1_float()** static std::string **parse_1_float** (const std::string & s, **mha_real_t** & v) [static]

This internal function parses a floating point number from the beginning of a string.

Parameters

s	The string to parse
---	---------------------

Precondition

s.size() > 0

Parameters

v	The float variable to fill with a value
---	---

Returns

The rest of the string.

5.154.2.3 check_parenthesis_complex() static void check_parenthesis_complex (const std::string & str) [static]

This function checks for unbalanced parenthesis in the string containing complex number.

Parameters

str	The string to check. This function returns normally only when the string starts with an opening bracket and ends with a closing bracket.
-----	--

Precondition

str.size() > 0

Exceptions

MHA_ErrorMsg	This function raises an error with an appropriate error message if the string does not start with an opening bracket '(' or if it does not end with a closing bracket ')'.
--------------	--

5.154.2.4 check_sign_complex() static int check_sign_complex (

```
const std::string & str ) [static]
```

This function checks for valid sign (b/w real & img.

parts of complex number). It also checks if real part is missing in complex number.

Parameters

<i>str</i>	String containing sign and onward imaginary part.
------------	---

Precondition

`str.size() > 0`

Exceptions

<i>This</i>	function raises an error if wrong sign (other than '+' or '-') is found. It also raises error, if it finds 'i' instead of sign. This can happen when real part is missing in complex number and 'i' (of imaginary part) is found where sign was expected.
-------------	---

Returns

+1 for '+' sign and -1 for '-' sign

5.154.2.5 `parse_1_complex()` static std::string parse_1_complex (

```
const std::string & s,
mha_complex_t & v ) [static]
```

This internal function parses a complex number from the beginning of a string.

Parameters

<i>s</i>	The string to parse
<i>v</i>	The complex variable to fill with a value

Exceptions

<i>MHA_ErrorMsg</i>	This function raises an error if s does not have characters except spaces, tabs etc
---------------------	---

Returns

The rest of the string.

5.155 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**

- **MHAParser::vcomplex_t**
Vector variable with float value.
- class **MHAParser::mint_t**
Matrix variable with int value.
- class **MHAParser::mfloat_t**
Matrix variable with float value.
- class **MHAParser::mcomplex_t**
Matrix variable with complex value.
- class **MHAParser::int_mon_t**
Monitor variable with int value.
- class **MHAParser::bool_mon_t**
Monitor with string value.
- class **MHAParser::string_mon_t**
Monitor with string value.
- class **MHAParser::vstring_mon_t**
Vector of monitors with string value.
- class **MHAParser::vint_mon_t**
Vector of ints monitor.
- class **MHAParser::mint_mon_t**
Matrix of ints monitor.
- class **MHAParser::vfloat_mon_t**
Vector of floats monitor.
- class **MHAParser::mfloat_mon_t**
Matrix of floats monitor.
- class **MHAParser::float_mon_t**
Monitor with float value.
- class **MHAParser::complex_mon_t**
Monitor with complex value.
- class **MHAParser::vcomplex_mon_t**
Monitor with vector of complex values.
- class **MHAParser::mcomplex_mon_t**
Matrix of complex numbers monitor.
- class **MHAParser::commit_t< receiver_t >**
Parser variable with event-emission functionality.
- class **MHAParser::mhacconfig_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)

Functions

- std::string **MHAParser::commentate** (const std::string &s)
- void **MHAParser::trim** (std::string &s)
- std::string **MHAParser::cfg_dump** (base_t *, const std::string &)
- std::string **MHAParser::cfg_dump_short** (base_t *, const std::string &)
- std::string **MHAParser::all_dump** (base_t *, const std::string &)
- std::string **MHAParser::mon_dump** (base_t *, const std::string &)
- std::string **MHAParser::all_ids** (base_t *, const std::string &, const std::string &= "")
- void **MHAParser::strreplace** (std::string &, const std::string &, const std::string &)

string replace function
- void **MHAParser::envreplace** (std::string &s)
- void **MHAParser::StrCnv::str2val** (const std::string &, bool &)

Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, float &)

Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, **mha_complex_t** &)

Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, int &)

Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, keyword_list_t &)

Convert from string.
- void **MHAParser::StrCnv::str2val** (const std::string &, std::string &)

Convert from string.
- template<class arg_t >
 void **MHAParser::StrCnv::str2val** (const std::string &s, std::vector< arg_t > &val)

Converter for vector types.
- template<> void **MHAParser::StrCnv::str2val< mha_real_t >** (const std::string &s, std::vector< **mha_real_t** > &v)

- Converter for vector< mha_real_t > with Matlab-style expansion.*
- template<class arg_t >
void **MHAParser::StrCnv::str2val** (const std::string &s, std::vector< std::vector< arg_t > > &val)
Converter for matrix types.
 - std::string **MHAParser::StrCnv::val2str** (const bool &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const float &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const mha_complex_t &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const int &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const keyword_list_t &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::string &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< float > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< mha_complex_t > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< int > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< int > > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< std::string > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< float > > &)
Convert to string.
 - std::string **MHAParser::StrCnv::val2str** (const std::vector< std::vector< std::vector< mha_complex_t > > &)
Convert to string.
 - int **MHAParser::StrCnv::num_brackets** (const std::string &s)
count number of brackets

Variables

- const typedef char *(* **MHAParser::c_parse_err_t**)(void *, int)

5.155.1 Detailed Description

Header file for the MHA-Parser script language.

5.155.2 Macro Definition Documentation

5.155.2.1 DEFAULT_RET_SIZE #define DEFAULT_RET_SIZE 0x100000

5.155.2.2 insert_member #define insert_member(x) insert_item(#x,&x)

Macro to insert a member variable into a parser.

Parameters

x	Member variable to be inserted. Name of member variable will be used as configuration name.
---	---

See also **MHAParser::parser_t::insert_item()** (p. [1099](#)).

5.156 mha_plugin.cpp File Reference

5.157 mha_plugin.hh File Reference

Header file for MHA C++ plugin class templates.

Classes

- class **MHAPlugin::cfg_node_t< runtime_cfg_t >**
A node class for storing MHA plugin runtime configurations as a singly linked list, where the nodes store besides the usual "next" and "data" pointers also a flag that indicates whether this node can be deleted.
- class **MHAPlugin::config_t< runtime_cfg_t >**
Template class for thread safe configuration.
- class **MHAPlugin::plugin_t< runtime_cfg_t >**
The template class for C++ openMHA plugins.

Namespaces

- **MHAPlugin**
Namespace for openMHA plugin class templates and thread-safe runtime configurations.

Macros

- #define **__declspec**(p)
- #define **WINAPI**
- #define **HINSTANCE** int
- #define **MHAPLUGIN_PROC_CALLBACK_PREFIX**(prefix, classname, indom, outdom)
- #define **MHAPLUGIN_SETCPP_CALLBACK_PREFIX**(prefix, classname)
- #define **MHAPLUGIN_INIT_CALLBACKS_PREFIX**(prefix, classname)
- #define **MHAPLUGIN_CALLBACKS_PREFIX**(prefix, classname, indom, outdom)

C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION_PREFIX**(prefix, cat, doc)
- #define **MHAPLUGIN_PROC_CALLBACK**(plugname, classname, indom, outdom) **MHAPLUGIN_PROC_CALLBACK_PREFIX**(MHA_STATIC_ ## plugname ## _classname,indom,outdom)
- #define **MHAPLUGIN_INIT_CALLBACKS**(plugname, classname) **MHAPLUGIN_INIT_CALLBACKS_PREFIX**(MHA_STATIC_ ## plugname ## _,classname)
- #define **MHAPLUGIN_CALLBACKS**(plugname, classname, indom, outdom) **MHAPLUGIN_CALLBACKS_PREFIX**(MHA_STATIC_ ## plugname ## _,classname,indom,outdom)

C++ wrapper macro for the plugin interface.
- #define **MHAPLUGIN_DOCUMENTATION**(plugname, cat, doc) **MHAPLUGIN_DOCUMENTATION_PREFIX**(MHA_STATIC_ ## plugname ## _,cat,doc)

Wrapper macro for the plugin documentation interface.

5.157.1 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`).

5.157.2 Macro Definition Documentation

5.157.2.1 **__declspec** #define **__declspec**(p)

5.157.2.2 WINAPI #define WINAPI**5.157.2.3 HINSTANCE** #define HINSTANCE int

```
5.157.2.4 MHAPLUGIN_PROC_CALLBACK_PREFIX #define MHAPLUGIN_PROC_CALLBACKPREFIX(
    prefix,
    classname,
    indom,
    outdom )
```

```
5.157.2.5 MHAPLUGIN_SETCPP_CALLBACK_PREFIX #define MHAPLUGIN_SETCPP_CALLBACKPREFIX(
    prefix,
    classname )
```

```
5.157.2.6 MHAPLUGIN_INIT_CALLBACKS_PREFIX #define MHAPLUGIN_INIT_CALLBACKSPREFIX(
    prefix,
    classname )
```

```
5.157.2.7 MHAPLUGIN_CALLBACKS_PREFIX #define MHAPLUGIN_CALLBACKSPREFIX(
    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 (wave or spec)
<i>outdom</i>	Output domain (wave or spec)

This macro defines all required openMHA Plugin interface functions and passes calls of these functions to the corresponding member functions of the class ‘classname’. The parameters ‘indom’ and ‘outdom’ specify the input and output domain of the processing method. The MHAInit() and MHADestroy() functions will create or destroy an instance of the class. The appropriate member functions have to be defined in the class. It is suggested to make usage of the **MHAPLUGIN::plugin_t** (p. 1146) template class. Exceptions of type **MHA_Error** (p. 763) are caught and transformed into appropriate error codes with their corresponding error messages.

5.157.2.8 MHAPLUGIN_DOCUMENTATION_PREFIX `#define MHAPLUGIN_DOCUMENTATIONPREFIX(`

```
prefix,
cat,
doc )
```

5.157.2.9 MHAPLUGIN_PROC_CALLBACK `#define MHAPLUGIN_PROC_CALLBACK(`

```
plugname,
classname,
indom,
outdom ) MHAPLUGIN_PROC_CALLBACK_PREFIX(MHA_STATIC_ ## plugname ## _classname, indom, outdom)
```

5.157.2.10 MHAPLUGIN_INIT_CALLBACKS `#define MHAPLUGIN_INIT_CALLBACKS(`

```
plugname,
classname ) MHAPLUGIN_INIT_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _classname)
```

5.157.2.11 MHAPLUGIN_CALLBACKS `#define MHAPLUGIN_CALLBACKS(`

```
plugname,
classname,
indom,
outdom ) MHAPLUGIN_CALLBACKS_PREFIX(MHA_STATIC_ ## plugname ## _classname, indom, outdom)
```

C++ wrapper macro for the plugin interface.

Parameters

<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 (wave or spec)
<i>outdom</i>	Output domain (wave or spec)

This macro defines all required openMHA Plugin interface functions and passes calls of these functions to the corresponding member functions of the class ‘classname’. The parameters ‘indom’ and ‘outdom’ specify the input and output domain of the processing method. The MHAInit() and MHADestroy() functions will create or destroy an instance of the class. The appropriate member functions have to be defined in the class. It is suggested to make usage of the **MHAPLUGIN::plugin_t** (p. 1146) template class. Exceptions of type **MHA_Error** (p. 763) are caught and transformed into appropriate error codes with their corresponding error messages.

```
5.157.2.12 MHAPLUGIN_DOCUMENTATION #define MHAPLUGIN_DOCUMENTATION(
```

 `plugname,
 cat,
 doc) MHAPLUGIN_DOCUMENTATION_PREFIX(MHA_STATIC_ ## plugname ## _←
,cat,doc)`

Wrapper macro for the plugin documentation interface.

Parameters

<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 seperated list of category names. An empty string will categorize the plugin in the category 'other'.

The documentation should contain a description of the plugin including a description of the underlying models, and a paragraph containing hints for usage. The text should be LaTeX compatible (e.g., avoid or quote underscores in the text part); equations should be formatted as LaTeX.

5.158 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)

5.158.1 Function Documentation

5.158.1.1 mha_tic() void mha_tic (
 mha_tictoc_t * t)

5.158.1.2 mha_platform_tic() void mha_platform_tic (
 mha_platform_tictoc_t * t)

5.158.1.3 mha_toc() float mha_toc (
 mha_tictoc_t * t)

5.158.1.4 mha_platform_toc() float mha_platform_toc (
 mha_platform_tictoc_t * t)

5.159 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)**

5.159.1 TypeDef Documentation

5.159.1.1 mha_platform_tictoc_t `typedef mha_tictoc_t mha_platform_tictoc_t`**5.159.2 Function Documentation****5.159.2.1 mha_platform_tic()** `void mha_platform_tic (`
 `mha_platform_tictoc_t * t)`**5.159.2.2 mha_platform_toc()** `float mha_platform_toc (`
 `mha_platform_tictoc_t * t)`**5.160 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 ()`

5.160.1 Typedef Documentation**5.160.1.1 rb_f_t** `typedef VALUE(* rb_f_t) (...)`**5.160.2 Function Documentation**

5.160.2.1 `mha_free()` static void mha_free (void * *mha*) [static]

5.160.2.2 `mha_alloc()` static VALUE mha_alloc (VALUE *klass*) [static]

5.160.2.3 `mha_exit_request()` static VALUE mha_exit_request (VALUE *self*) [static]

5.160.2.4 `mha_parse()` static VALUE mha_parse (VALUE *self*, VALUE *request*) [static]

5.160.2.5 `Init_mha_ruby()` void Init_mha_ruby ()

5.161 `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)

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)

Transform 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)
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.
- std::vector< float > **std_vector_float** (const **mha_wave_t** &w)
*Converts a **mha_wave_t** (p. 839) 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. 839) 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. 793) 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.

5.161.1 Macro Definition Documentation

5.161.1.1 MHA_ID_UINT_VECTOR #define MHA_ID_UINT_VECTOR "MHASignal::uint_<vector_t"

5.161.1.2 MHA_ID_MATRIX #define MHA_ID_MATRIX "MHASignal::matrix_t"

5.161.1.3 ASSERT_EQUAL_DIM #define ASSERT_EQUAL_DIM(
 a,
 b)

5.161.1.4 ASSERT_EQUAL_DIM_PTR #define ASSERT_EQUAL_DIM_PTR(
 a,
 b)

5.161.2 Function Documentation

5.161.2.1 set_minabs() void set_minabs (
 mha_spec_t & self,
 const mha_real_t & m)

5.161.2.2 safe_div() mha_spec_t& safe_div (
 mha_spec_t & self,
 const mha_spec_t & v,
 mha_real_t eps)

In-Place division with lower limit on divisor.

```
5.161.2.3 intensity() static mha_real_t intensity (
    const mha_spec_t & s,
    unsigned int channel,
    unsigned int fftlen,
    mha_real_t * sqfreq_response = 0 ) [static]
```

5.162 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. 793))*
- class **MHASignal::waveform_t**
*signal processing class for waveform data (based on **mha_wave_t** (p. 839))*
- 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. 839).*
- **mha_real_t** **MHASignal::lin2db** (**mha_real_t** x, **mha_real_t** eps)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t** **MHASignal::lin2db** (**mha_real_t** x)
Conversion from linear scale to dB (no SPL reference)
- **mha_real_t** **MHASignal::db2lin** (**mha_real_t** x)
Conversion from dB scale to linear (no SPL reference)
- **mha_real_t** **MHASignal::sq2db** (**mha_real_t** x, **mha_real_t** eps=0.0f)
conversion from squared values to dB (no SPL reference)
- **mha_real_t** **MHASignal::db2sq** (**mha_real_t** x)
conversion from dB to squared values (no SPL reference)
- **mha_real_t** **MHASignal::pa2dbspl** (**mha_real_t** x, **mha_real_t** eps)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t** **MHASignal::pa2dbspl** (**mha_real_t** x)
Conversion from linear Pascal scale to dB SPL.
- **mha_real_t** **MHASignal::dbspl2pa** (**mha_real_t** x)
Conversion from dB SPL to linear Pascal scale.
- **mha_real_t** **MHASignal::pa22dbspl** (**mha_real_t** x, **mha_real_t** eps=0.0f)
Conversion from squared Pascal scale to dB SPL.
- **mha_real_t** **MHASignal::dbspl2pa2** (**mha_real_t** x)
conversion from dB SPL to squared Pascal scale
- **mha_real_t** **MHASignal::smp2sec** (**mha_real_t** n, **mha_real_t** srate)
conversion from samples to seconds
- **mha_real_t** **MHASignal::sec2smp** (**mha_real_t** sec, **mha_real_t** srate)
conversion from seconds to samples
- **mha_real_t** **MHASignal::bin2freq** (**mha_real_t** bin, unsigned fftlen, **mha_real_t** sratet)
conversion from fft bin index to frequency
- **mha_real_t** **MHASignal::freq2bin** (**mha_real_t** freq, unsigned fftlen, **mha_real_t** sratet)
conversion from frequency to fft bin index

- **mha_real_t MHASignal::smp2rad** (**mha_real_t** samples, unsigned bin, unsigned fftlen)
 - conversion from frequency to fft bin index*
- **mha_real_t MHASignal::rad2smp** (**mha_real_t** phase_shift, unsigned bin, unsigned fftlen)
 - conversion from delay in samples to phase shift*
- 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)
 - Return size of a waveform 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.*
- **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)

Access to an element of a spectrum.
- **mha_complex_t** & **value** (**mha_spec_t** *s, unsigned int k)

Access to an element of a spectrum.
- **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. 839) 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. 839) 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. 793) 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_wave_t & operator/= (mha_wave_t &, const mha_wave_t &)**
Element-wise division operator.
- **mha_spec_t & operator+= (mha_spec_t &, const mha_spec_t &)**
Addition operator.
- **mha_spec_t & operator+= (mha_spec_t &, const mha_real_t &)**
Addition operator.
- **void set_minabs (mha_spec_t &, const mha_real_t &)**
- **mha_spec_t & safe_div (mha_spec_t &self, const mha_spec_t &v, mha_real_t eps)**
In-Place division with lower limit on divisor.
- **mha_wave_t & operator^= (mha_wave_t &self, const mha_real_t &arg)**
Exponent operator.
- **void MHASignal::copy_channel (mha_spec_t &self, const mha_spec_t &src, unsigned sch, unsigned dch)**
Copy one channel of a source signal.
- **void MHASignal::copy_channel (mha_wave_t &self, const mha_wave_t &src, unsigned src_channel, unsigned dest_channel)**
Copy one channel of a source signal.
- **mha_real_t MHASignal::rmslevel (const mha_spec_t &s, unsigned int channel, unsigned int fftlen)**
Return RMS level of a spectrum channel.
- **mha_real_t MHASignal::colored_intensity (const mha_spec_t &s, unsigned int channel, unsigned int fftlen, mha_real_t *sqfreq_response=nullptr)**
Colored spectrum intensity.
- **mha_real_t MHASignal::maxabs (const mha_spec_t &s, unsigned int channel)**
Find maximal absolute value.
- **mha_real_t MHASignal::rmslevel (const mha_wave_t &s, unsigned int channel)**
Return RMS level of a waveform channel.
- **mha_real_t MHASignal::maxabs (const mha_wave_t &s, unsigned int channel)**
Find maximal absolute value.

- **mha_real_t MHASignal::maxabs (const mha_wave_t &s)**
Find maximal absolute value.
- **mha_real_t MHASignal::max (const mha_wave_t &s)**
Find maximal value.
- **mha_real_t MHASignal::min (const mha_wave_t &s)**
Find minimal value.
- **mha_real_t MHASignal::sumsqr_channel (const mha_wave_t &s, unsigned int channel)**
Calculate sum of squared values in one channel.
- **mha_real_t MHASignal::sumsqr_frame (const mha_wave_t &s, unsigned int frame)**
Calculate sum over all channels of squared values.
- **void MHASignal::scale (mha_spec_t *dest, const mha_wave_t *src)**
- **void MHASignal::limit (mha_wave_t &s, const mha_real_t & min, const mha_real_t & max)**
Limit the singal in the waveform buffer to the range [min, max].
- **mha_complex_t & set (mha_complex_t &self, mha_real_t real, mha_real_t imag=0)**
*Assign real and imaginary parts to a **mha_complex_t** (p. 744) variable.*
- **mha_complex_t mha_complex (mha_real_t real, mha_real_t imag=0)**
*Create a new **mha_complex_t** (p. 744) 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. 744) 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. 744).*
- **mha_complex_t & expi (mha_complex_t &self, mha_real_t angle)**
*replaces the value of the given **mha_complex_t** (p. 744) 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)**

- **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. 744) 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)**
- **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. 744) with its conjugate.*
- **void conjugate (mha_spec_t &self)**

*Replace (!) the value of this **mha_spec_t** (p. 793) 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)**

- **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).
- **bool almost (const mha_complex_t &self, const mha_complex_t &other, mha_<real_t times_epsilon=1e2)**

Divide a complex by its absolute value, thereby normalizing it (projecting onto the unit circle), with a safety margin.
- **bool operator< (const mha_complex_t &x, const mha_complex_t &y)**

Compare two complex numbers for equality except for a small relative error.
- **std::ostream & operator<< (std::ostream &o, const mha_complex_t &c)**

ostream operator for mha_complex_t (p. 744)
- **std::istream & operator>> (std::istream &i, mha_complex_t &c)**

preliminary istream operator for mha_complex_t (p. 744) 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)**

Transform waveform segment into spectrum.
- **void mha_fft_wave2spec (mha_fft_t h, const mha_wave_t *in, mha_spec_t *out, bool swaps)**

Transform waveform segment into spectrum.
- **void mha_fft_spec2wave (mha_fft_t h, const mha_spec_t *in, mha_wave_t *out)**

Transform spectrum into waveform segment.
- **void mha_fft_spec2wave (mha_fft_t h, const mha_spec_t *in, mha_wave_t *out, unsigned int offset)**

Transform 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.
- **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.

5.162.1 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.

5.162.2 Macro Definition Documentation

5.162.2.1 **M_PI** #define M_PI 3.14159265358979323846

5.162.2.2 mha_round #define mha_round(
x) (int)((float)x+0.5)

5.162.3 Function Documentation

5.162.3.1 mha_min_1() unsigned int mha_min_1 (
unsigned int a) [inline]

5.162.3.2 value() [1/2] mha_real_t& value (
mha_wave_t * s,
unsigned int k) [inline]

5.162.3.3 value() [2/2] mha_complex_t& value (
mha_spec_t * s,
unsigned int k) [inline]

5.162.3.4 set_minabs() void set_minabs (
mha_spec_t & ,
const mha_real_t &)

5.162.3.5 safe_div() mha_spec_t& safe_div (
mha_spec_t & self,
const mha_spec_t & v,
mha_real_t eps)

In-Place division with lower limit on divisor.

5.162.3.6 operator<<() std::ostream& operator<< (std::ostream & o, const mha_complex_t & c) [inline]

ostream operator for **mha_complex_t** (p. 744)

5.162.3.7 operator>>() std::istream& operator>> (std::istream & i, mha_complex_t & c) [inline]

preliminary istream operator for **mha_complex_t** (p. 744) without error checking

5.163 mha_signal_fft.h File Reference

Classes

- class **MHASignal::fft_t**

Namespaces

- **MHASignal**
Namespace for audio signal handling and processing classes.

5.164 mha_tablelookup.cpp File Reference

5.165 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.

5.165.1 Detailed Description

Header file for table lookup classes.

5.166 mha_tcp.cpp File Reference

Classes

- class **MHA_TCP::sock_init_t**

Namespaces

- **MHA_TCP**

A Namespace for TCP helper classes.

Macros

- #define **INVALID_SOCKET** (-1)
- #define **SOCKET_ERROR** (-1)
- #define **closesocket(fd)** (close((fd)))
- #define **ASYNC_CONNECT_STARTED** EINPROGRESS

Typedefs

- typedef int **SOCKET**

Functions

- std::string **MHA_TCP::STRERROR** (int err)

Portable conversion from error number to error string.
- std::string **MHA_TCP::HSTRERROR** (int err)

Portable conversion from hostname error number to error string.
- int **MHA_TCP::N_ERRNO** ()

Portable access to last network error number.
- int **MHA_TCP::H_ERRNO** ()

Portable access to last hostname error number.
- int **MHA_TCP::G_ERRNO** ()

Portable access to last non-network error number.
- static sockaddr_in **host_port_to_sock_addr** (const std::string &host, unsigned short port)
- static SOCKET **tcp_connect_to** (const std::string &host, unsigned short port)
- static SOCKET **tcp_connect_to_with_timeout** (const std::string &host, unsigned short port, **Timeout_Watcher** &timeout_watcher)
- static void * **thread_start_func** (void *thread)

Variables

- class **MHA_TCP::sock_init_t MHA_TCP::sock_initializer**

5.166.1 Macro Definition Documentation

5.166.1.1 INVALID_SOCKET #define INVALID_SOCKET (-1)

5.166.1.2 SOCKET_ERROR #define SOCKET_ERROR (-1)

5.166.1.3 closesocket #define closesocket(fd) (close((fd)))

5.166.1.4 ASYNC_CONNECT_STARTED #define ASYNC_CONNECT_STARTED EINPROGRESS

5.166.2 Typedef Documentation

5.166.2.1 SOCKET `typedef int SOCKET`

5.166.3 Function Documentation

5.166.3.1 host_port_to_sock_addr() `static sockaddr_in host_port_to_sock_addr (`
 `const std::string & host,`
 `unsigned short port) [static]`

5.166.3.2 tcp_connect_to() `static SOCKET tcp_connect_to (`
 `const std::string & host,`
 `unsigned short port) [static]`

5.166.3.3 tcp_connect_to_with_timeout() `static SOCKET tcp_connect_to_with_timeout (`
 `const std::string & host,`
 `unsigned short port,`
 `Timeout_Watcher & timeout_watcher) [static]`

5.166.3.4 thread_start_func() `static void* thread_start_func (`
 `void * thread) [static]`

5.167 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. 802) 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.

5.167.1 Macro Definition Documentation

5.167.1.1 Sleep #define Sleep(x) usleep((x)*1000);

5.168 mha_tcp_server.cpp File Reference

Namespaces

- **mha_tcp**
namespace for network communication classes of MHA

5.169 mha_tcp_server.hh File Reference

Classes

- class **mha_tcp::buffered_socket_t**
An asio TCP socket with an associated streambuf buffer for receiving lines of text, as well as string buffers for sending responses.
- class **mha_tcp::server_t**
Class for accepting TCP connections from clients.

Namespaces

- **mha_tcp**
namespace for network communication classes of MHA

5.170 mha_toolbox.h File Reference**5.171 mha_utils.cpp File Reference****5.172 mha_utils.hh File Reference****Namespaces**

- **MHAUtils**

Functions

- bool **MHAUtils::is_multiple_of** (const unsigned big, const unsigned small)
- bool **MHAUtils::is_power_of_two** (const unsigned n)
- bool **MHAUtils::is_multiple_of_by_power_of_two** (const unsigned big, const unsigned small)
- std::string **MHAUtils::strip** (const std::string &line)
- std::string **MHAUtils::remove** (const std::string &str_, char c)
- bool **MHAUtils::is_denormal** (**mha_real_t** x)
Get the normal-ness of a mha_real_t.
- bool **MHAUtils::is_denormal** (const **mha_complex_t** &x)
Get the normal-ness of a complex number.
- bool **MHAUtils::is_denormal** (const std::complex< **mha_real_t** > &x)
Get the normal-ness of a complex number.
- **mha_real_t MHAUtils::spl2hl** (**mha_real_t** f)
Get the offset of between dB(SPL) and dB(HL) for a given frequency according to ISO 389-7-2005 (freefield); e.g.

5.173 mha_windowparser.cpp File Reference**Variables**

- float(*) **wnd_funcs** [])(float)

5.173.1 Variable Documentation

5.173.1.1 **wnd_funcs** float(* wnd_funcs[]) (float)

5.174 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.

5.175 mhachain.cpp File Reference

Classes

- class **mhachain::mhachain_t**

Namespaces

- **mhachain**

5.176 mhafw_lib.cpp File Reference

5.177 mhafw_lib.h File Reference

Classes

- class **io_lib_t**
Class for loading MHA sound IO module.
- class **fw_vars_t**
- class **fw_t**

5.178 MHAIoalsa.cpp File Reference

Classes

- class **alsa_base_t**
- class **alsa_dev_par_parser_t**
Parser variables corresponding to one alsa device.
- class **alsa_t< T >**
Our representation of one alsa device.
- class **io_alsa_t**
MHA IO interface class for ALSA IO.

Macros

- #define **DBG**(x) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)
- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOalsa_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOalsa_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOalsa_IOStart
- #define **IOStop** MHA_STATIC_MHAIOalsa_IOStop
- #define **IOResume** MHA_STATIC_MHAIOalsa_IOResume
- #define **IORelease** MHA_STATIC_MHAIOalsa_IOResume
- #define **IOSetVar** MHA_STATIC_MHAIOalsa_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOalsa_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOalsa_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOalsa_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
IO library initialization function, called by framework after loading this IO library into the MHA process.
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
IO library prepare function, called after the MHA prepared the processing plugins.
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOResume** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

5.178.1 Macro Definition Documentation

5.178.1.1 `DBG` `#define DBG(`
`x) fprintf(stderr,"%s:%d\n",__FILE__,__LINE__)`

5.178.1.2 `ERR_SUCCESS` `#define ERR_SUCCESS 0`

5.178.1.3 `ERR_IHANDLE` `#define ERR_IHANDLE -1`

5.178.1.4 `ERR_USER` `#define ERR_USER -1000`

5.178.1.5 `MAX_USER_ERR` `#define MAX_USER_ERR 0x500`

5.178.1.6 `IOInit` `#define IOInit MHA_STATIC_MHAIoalsa_IOInit`

5.178.1.7 `IOPrepare` `#define IOPrepare MHA_STATIC_MHAIoalsa_IOPrepare`

5.178.1.8 `IOStart` `#define IOStart MHA_STATIC_MHAIoalsa_IOStart`

5.178.1.9 `IOStop` `#define IOStop MHA_STATIC_MHAIoalsa_IOStop`

5.178.1.10 IORelease #define IORelease MHA_STATIC_MHAIOalsa_IORelease

5.178.1.11 IOSetVar #define IOSetVar MHA_STATIC_MHAIOalsa_IOSetVar

5.178.1.12 IOStrError #define IOStrError MHA_STATIC_MHAIOalsa_IOStrError

5.178.1.13 IODestroy #define IODestroy MHA_STATIC_MHAIOalsa_IODestroy

5.178.1.14 dummy_interface_test void dummy_interface_test MHA_STATIC_MHAIOalsa_<→
dummy_interface_test

5.178.2 Function Documentation

5.178.2.1 IOInit() int IOInit (

```
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    void ** handle )
```

IO library initialization function, called by framework after loading this IO library into the MHA process.

Gives plugin callback functions and callback handles to interact with the MHA framework.

Parameters

<i>handle</i>	output parameter. IO library returns pointer to void to the caller via this parameter. All other function calls from the MHA framework will use this handle.
---------------	---

5.178.2.2 IOPrepare() `int IOPrepare (`

```
    void * handle,  
    int nch_in,  
    int nch_out )
```

IO library prepare function, called after the MHA prepared the processing plugins.

5.178.2.3 IOStart() `int IOStart (`

```
    void * handle )
```

5.178.2.4 IOStop() `int IOStop (`

```
    void * handle )
```

5.178.2.5 IOReset() `int IOReset (`

```
    void * handle )
```

5.178.2.6 IOSetVar() `int IOSetVar (`

```
    void * handle,  
    const char * command,  
    char * retval,  
    unsigned int maxretlen )
```

```
5.178.2.7 IOStrError() const char* IOStrError (
    void * handle,
    int err )
```

```
5.178.2.8 IODestroy() void IODestroy (
    void * handle )
```

5.178.3 Variable Documentation

```
5.178.3.1 user_err_msg char user_err_msg[ MAX_USER_ERR] [static]
```

5.179 MHAIOAsterisk.cpp File Reference

Classes

- class **io_asterisk_parser_t**
The parser interface of the IOAsterisk library.
- class **io_asterisk_sound_t**
Sound data handling of io tcp library.
- class **io_asterisk_fwcb_t**
TCP sound-io library's interface to the framework callbacks.
- class **io_asterisk_t**
The tcp sound io library.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x2000
- #define **MHA_ErrorMsg2**(x, y) **MHA_Error**(__FILE__, __LINE__, (x), (y))
- #define **MHA_ErrorMsg3**(x, y, z) **MHA_Error**(__FILE__, __LINE__, (x), (y), (z))
- #define **MIN_TCP_PORT** 0
- #define **MIN_TCP_PORT_STR** "0"
- #define **MAX_TCP_PORT** 65535
- #define **MAX_TCP_PORT_STR** "65535"

- #define **IOInit** MHA_STATIC_MHAIOTCP_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOTCP_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOTCP_IOStart
- #define **IOStop** MHA_STATIC_MHAIOTCP_IOStop
- #define **IOResume** MHA_STATIC_MHAIOTCP_IOResume
- #define **IOSetVar** MHA_STATIC_MHAIOTCP_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOTCP_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOTCP_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOTCP_dummy_interface_test

Functions

- static int **copy_error** (**MHA_Error** &e)
- static void * **thread_startup_function** (void *parameter)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int num_inchannels, int num_outchannels)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOResume** (void *handle)
- int **IOSetVar** (void *handle, const char *cmd, char *retval, unsigned int len)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

5.179.1 Macro Definition Documentation

5.179.1.1 **ERR_SUCCESS** #define **ERR_SUCCESS** 0

5.179.1.2 **ERR_IHANDLE** #define **ERR_IHANDLE** -1

5.179.1.3 ERR_USER #define ERR_USER -1000

5.179.1.4 MAX_USER_ERR #define MAX_USER_ERR 0x2000

5.179.1.5 MHA_ErrorMsg2 #define MHA_ErrorMsg2(

```
    x,  
    y )  MHA_Error(__FILE__, __LINE__, (x), (y))
```

5.179.1.6 MHA_ErrorMsg3 #define MHA_ErrorMsg3(

```
    x,  
    y,  
    z )  MHA_Error(__FILE__, __LINE__, (x), (y), (z))
```

5.179.1.7 MIN_TCP_PORT #define MIN_TCP_PORT 0

5.179.1.8 MIN_TCP_PORT_STR #define MIN_TCP_PORT_STR "0"

5.179.1.9 MAX_TCP_PORT #define MAX_TCP_PORT 65535

5.179.1.10 MAX_TCP_PORT_STR #define MAX_TCP_PORT_STR "65535"

5.179.1.11 IOInit #define IOInit MHA_STATIC_MHAIOTCP_IOInit

5.179.1.12 IOPrepare #define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare

5.179.1.13 IOStart #define IOStart MHA_STATIC_MHAIOTCP_IOStart

5.179.1.14 IOStop #define IOStop MHA_STATIC_MHAIOTCP_IOStop

5.179.1.15 IOReset #define IOReset MHA_STATIC_MHAIOTCP_IOReset

5.179.1.16 IOSetVar #define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar

5.179.1.17 IOStrError #define IOStrError MHA_STATIC_MHAIOTCP_IOStrError

5.179.1.18 IODestroy #define IODestroy MHA_STATIC_MHAIOTCP_IODestroy

5.179.1.19 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOTC←
P_dummy_interface_test

5.179.2 Function Documentation

5.179.2.1 copy_error() static int copy_error (
 MHA_Error & e) [static]

5.179.2.2 thread_startup_function() static void* thread_startup_function (
 void * parameter) [static]

5.179.2.3 IOInit() int IOInit (
 int fragsize,
 float samplerate,
 IOProcessEvent_t proc_event,
 void * proc_handle,
 IOStartedEvent_t start_event,
 void * start_handle,
 IOSoppedEvent_t stop_event,
 void * stop_handle,
 void ** handle)

5.179.2.4 IOPrepare() int IOPrepare (
 void * handle,
 int num_inchannels,
 int num_outchannels)

5.179.2.5 IOStart() int IOStart (
 void * handle)

5.179.2.6 IOStop() int IOStop (
 void * handle)

5.179.2.7 `IOResume()` `int IOResume (`
`void * handle)`

5.179.2.8 `IOSetVar()` `int IOSetVar (`
`void * handle,`
`const char * cmd,`
`char * retval,`
`unsigned int len)`

5.179.2.9 `IOStrError()` `const char* IOStrError (`
`void * handle,`
`int err)`

5.179.2.10 `IODestroy()` `void IODestroy (`
`void * handle)`

5.179.3 Variable Documentation

5.179.3.1 `user_err_msg` `char user_err_msg[MAX_USER_ERR] [static]`

5.180 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 **IOResume** MHA_STATIC_MHAIOFile_IOResume
- #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 **IOResume** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

5.180.1 Macro Definition Documentation

```
5.180.1.1 DEBUG #define DEBUG(
    x ) std::cerr << __FILE__ << ":" << __LINE__ << " " << #x " = " <<
x << std::endl
```

5.180.1.2 ERR_SUCCESS #define ERR_SUCCESS 0

5.180.1.3 ERR_IHANDLE #define ERR_IHANDLE -1

5.180.1.4 ERR_USER #define ERR_USER -1000

5.180.1.5 MAX_USER_ERR #define MAX_USER_ERR 0x500

5.180.1.6 IOInit #define IOInit MHA_STATIC_MHAIOFile_IOInit

5.180.1.7 IOPrepare #define IOPrepare MHA_STATIC_MHAIOFile_IOPrepare

5.180.1.8 IOStart #define IOStart MHA_STATIC_MHAIOFile_IOStart

5.180.1.9 IOStop #define IOStop MHA_STATIC_MHAIOFile_IOStop

5.180.1.10 IOReset #define IOReset MHA_STATIC_MHAIOFile_IOReset

5.180.1.11 IOSetVar #define IOSetVar MHA_STATIC_MHAIOFile_IOSetVar

5.180.1.12 IOStrError #define IOStrError MHA_STATIC_MHAIOFile_IOStrError

5.180.1.13 IODestroy #define IODestroy MHA_STATIC_MHAIOFile_IODestroy

5.180.1.14 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIO←
File_dummy_interface_test

5.180.2 Function Documentation

5.180.2.1 IOInit() int IOInit (

```
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    void ** handle )
```

5.180.2.2 IOPrepare() int IOPrepare (

```
    void * handle,
    int nch_in,
    int nch_out )
```

5.180.2.3 IOStart() int IOStart (
 void * *handle*)

5.180.2.4 IOStop() int IOStop (
 void * *handle*)

5.180.2.5 IOReset() int IOReset (
 void * *handle*)

5.180.2.6 IOSetVar() int IOSetVar (
 void * *handle*,
 const char * *command*,
 char * *retval*,
 unsigned int *maxretlen*)

5.180.2.7 IOStrError() const char* IOStrError (
 void * *handle*,
 int *err*)

5.180.2.8 IODestroy() void IODestroy (
 void * *handle*)

5.180.3 Variable Documentation

5.180.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] [static]

5.181 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 **IOResume** MHA_STATIC_MHAIOJack_IOResume
- #define **IOSetVar** MHA_STATIC_MHAIOJack_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOJack_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOJack_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOJack_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOResume** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

5.181.1 Macro Definition Documentation

5.181.1.1 ERR_SUCCESS #define ERR_SUCCESS 0

5.181.1.2 ERR_IHANDLE #define ERR_IHANDLE -1

5.181.1.3 ERR_USER #define ERR_USER -1000

5.181.1.4 MAX_USER_ERR #define MAX_USER_ERR 0x500

5.181.1.5 IOInit #define IOInit MHA_STATIC_MHAIOJack_IOInit

5.181.1.6 IOPrepare #define IOPrepare MHA_STATIC_MHAIOJack_IOPrepare

5.181.1.7 IOStart #define IOStart MHA_STATIC_MHAIOJack_IOStart

5.181.1.8 IOStop #define IOStop MHA_STATIC_MHAIOJack_IOStop

5.181.1.9 IOReset #define IOReset MHA_STATIC_MHAIOJack_IOReset

5.181.1.10 IOSetVar #define IOSetVar MHA_STATIC_MHAIOJack_IOSetVar

5.181.1.11 IOStrError #define IOStrError MHA_STATIC_MHAIOJack_IOStrError

5.181.1.12 IODestroy #define IODestroy MHA_STATIC_MHAIOJack_IODestroy

5.181.1.13 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOJack_dummy_interface_test

5.181.2 Function Documentation

5.181.2.1 IOInit() int IOInit (

```
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    void ** handle )
```

5.181.2.2 IOPrepare() int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

5.181.2.3 IOStart() int IOStart (

```
void * handle )
```

5.181.2.4 IOStop() int IOStop (

```
void * handle )
```

5.181.2.5 IOReset() int IOReset (

```
void * handle )
```

5.181.2.6 IOSetVar() int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

5.181.2.7 IOStrError() const char* IOStrError (

```
void * handle,
int err )
```

5.181.2.8 IODestroy() void IODestroy (

```
void * handle )
```

5.181.3 Variable Documentation

5.181.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] = "" [static]

5.182 MHAIOJackdb.cpp File Reference

Classes

- class **MHAIOJackdb::io_jack_t**
Main class for JACK IO.

Namespaces

- **MHAIOJackdb**

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x500
- #define **IOInit** MHA_STATIC_MHAIOJackdb_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOJackdb_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOJackdb_IOStart
- #define **IOStop** MHA_STATIC_MHAIOJackdb_IOStop
- #define **IOReset** MHA_STATIC_MHAIOJackdb_IOReset
- #define **IOSetVar** MHA_STATIC_MHAIOJackdb_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOJackdb_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOJackdb_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOJackdb_dummy_interface_test

Functions

- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOReset** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""

5.182.1 Macro Definition Documentation

5.182.1.1 ERR_SUCCESS #define ERR_SUCCESS 0

5.182.1.2 ERR_IHANDLE #define ERR_IHANDLE -1

5.182.1.3 ERR_USER #define ERR_USER -1000

5.182.1.4 MAX_USER_ERR #define MAX_USER_ERR 0x500

5.182.1.5 IOInit #define IOInit MHA_STATIC_MHAIOJackdb_IOInit

5.182.1.6 IOPrepare #define IOPrepare MHA_STATIC_MHAIOJackdb_IOPrepare

5.182.1.7 IOStart #define IOStart MHA_STATIC_MHAIOJackdb_IOStart

5.182.1.8 IOStop #define IOStop MHA_STATIC_MHAIOJackdb_IOStop

5.182.1.9 IOReset #define IOReset MHA_STATIC_MHAIOJackdb_IOReset

5.182.1.10 IOSetVar #define IOSetVar MHA_STATIC_MHAIOJackdb_IOSetVar

5.182.1.11 IOStrError #define IOStrError MHA_STATIC_MHAIOJackdb_IOStrError

5.182.1.12 IODestroy #define IODestroy MHA_STATIC_MHAIOJackdb_IODestroy

5.182.1.13 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOJackdb_dummy_interface_test

5.182.2 Function Documentation

5.182.2.1 IOInit() int IOInit (

```
    int fragsize,
    float samplerate,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    void ** handle )
```

5.182.2.2 IOPrepare() int IOPrepare (
 void * *handle*,
 int *nch_in*,
 int *nch_out*)

5.182.2.3 IOStart() int IOStart (
 void * *handle*)

5.182.2.4 IOStop() int IOStop (
 void * *handle*)

5.182.2.5 IOReset() int IOReset (
 void * *handle*)

5.182.2.6 IOSetVar() int IOSetVar (
 void * *handle*,
 const char * *command*,
 char * *retval*,
 unsigned int *maxretlen*)

5.182.2.7 IOStrError() const char* IOStrError (
 void * *handle*,
 int *err*)

5.182.2.8 IODestroy() void IODestroy (
 void * *handle*)

5.182.3 Variable Documentation

5.182.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] = "" [static]

5.183 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 **IOResume** MHA_STATIC_MHAIOParser_IOResume
- #define **IORelease** MHA_STATIC_MHAIOParser_IOResume
- #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, **IStartedEvent_t** start_event, void *start_handle, **IStoppedEvent_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 **IOResume** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

5.183.1 Macro Definition Documentation

5.183.1.1 `ERR_SUCCESS` `#define ERR_SUCCESS 0`

5.183.1.2 `ERR_IHANDLE` `#define ERR_IHANDLE -1`

5.183.1.3 `ERR_USER` `#define ERR_USER -1000`

5.183.1.4 `MAX_USER_ERR` `#define MAX_USER_ERR 0x500`

5.183.1.5 `IOInit` `#define IOInit MHA_STATIC_MHAIOParser_IOInit`

5.183.1.6 `IOPrepare` `#define IOPrepare MHA_STATIC_MHAIOParser_IOPrepare`

5.183.1.7 `IOStart` `#define IOStart MHA_STATIC_MHAIOParser_IOStart`

5.183.1.8 IOStop #define IOStop MHA_STATIC_MHAIOParser_IOStop

5.183.1.9 IOReset #define IOReset MHA_STATIC_MHAIOParser_IOReset

5.183.1.10 IOSetVar #define IOSetVar MHA_STATIC_MHAIOParser_IOSetVar

5.183.1.11 IOStrError #define IOStrError MHA_STATIC_MHAIOParser_IOStrError

5.183.1.12 IODestroy #define IODestroy MHA_STATIC_MHAIOParser_IODestroy

5.183.1.13 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOParser_dummy_interface_test

5.183.2 Function Documentation

5.183.2.1 IOInit() int IOInit (

```
    int fragsize,
    float ,
    IOProcessEvent_t proc_event,
    void * proc_handle,
    IOStartedEvent_t start_event,
    void * start_handle,
    IOStoppedEvent_t stop_event,
    void * stop_handle,
    void ** handle )
```

5.183.2.2 IOPrepare() int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

5.183.2.3 IOStart() int IOStart (

```
void * handle )
```

5.183.2.4 IOStop() int IOStop (

```
void * handle )
```

5.183.2.5 IOResearch() int IOResearch (

```
void * handle )
```

5.183.2.6 IOSetVar() int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

5.183.2.7 IOStrError() const char* IOStrError (

```
void * handle,
int err )
```

5.183.2.8 IODestroy() void IODestroy (

```
void * handle )
```

5.183.3 Variable Documentation

5.183.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] [static]

5.184 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 **IOResume** MHA_STATIC_MHAIOPortAudio_IOResume
- #define **IOSetVar** MHA_STATIC_MHAIOPortAudio_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOPortAudio_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOPortAudio_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOPortAudio_dummy_interface_←
test

Functions

- static std::string **MHAIOPortAudio::parserFriendlyName** (const std::string &in)
- int **portaudio_callback** (const void *input, void *output, unsigned long frameCount, const PaStreamCallbackTimeInfo *timeInfo, PaStreamCallbackFlags statusFlags, void *userData)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int nch_in, int nch_out)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOResume** (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)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**] = ""
- PaStreamCallback **portaudio_callback**

5.184.1 Macro Definition Documentation

5.184.1.1 **ERR_SUCCESS** #define ERR_SUCCESS 0

5.184.1.2 **ERR_IHANDLE** #define ERR_IHANDLE -1

5.184.1.3 **ERR_USER** #define ERR_USER -1000

5.184.1.4 **MAX_USER_ERR** #define MAX_USER_ERR 0x500

5.184.1.5 IOInit #define IOInit MHA_STATIC_MHAIOPortAudio_IOInit

5.184.1.6 IOPrepare #define IOPrepare MHA_STATIC_MHAIOPortAudio_IOPrepare

5.184.1.7 IOStart #define IOStart MHA_STATIC_MHAIOPortAudio_IOStart

5.184.1.8 IOStop #define IOStop MHA_STATIC_MHAIOPortAudio_IOStop

5.184.1.9 IOReset #define IOReset MHA_STATIC_MHAIOPortAudio_IOReset

5.184.1.10 IOSetVar #define IOSetVar MHA_STATIC_MHAIOPortAudio_IOSetVar

5.184.1.11 IOStrError #define IOStrError MHA_STATIC_MHAIOPortAudio_IOStrError

5.184.1.12 IODestroy #define IODestroy MHA_STATIC_MHAIOPortAudio_IODestroy

5.184.1.13 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOPortAudio_dummy_interface_test

5.184.2 Function Documentation

5.184.2.1 portaudio_callback() int portaudio_callback (

```
const void * input,
void * output,
unsigned long frameCount,
const PaStreamCallbackTimeInfo * timeInfo,
PaStreamCallbackFlags statusFlags,
void * userData )
```

5.184.2.2 IOInit() int IOInit (

```
int fragsize,
float samplerate,
IOProcessEvent_t proc_event,
void * proc_handle,
IOStartedEvent_t start_event,
void * start_handle,
IOSoppedEvent_t stop_event,
void * stop_handle,
void ** handle )
```

5.184.2.3 IOPrepare() int IOPrepare (

```
void * handle,
int nch_in,
int nch_out )
```

5.184.2.4 IOStart() int IOStart (

```
void * handle )
```

5.184.2.5 IOStop() int IOStop (

```
void * handle )
```

5.184.2.6 IORelease() int IORelease (

```
void * handle )
```

5.184.2.7 IOSetVar() int IOSetVar (

```
void * handle,
const char * command,
char * retval,
unsigned int maxretlen )
```

5.184.2.8 IOStrError() const char* IOStrError (

```
void * handle,
int err )
```

5.184.2.9 IODestroy() void IODestroy (

```
void * handle )
```

5.184.3 Variable Documentation

5.184.3.1 user_err_msg char user_err_msg[**MAX_USER_ERR**] = "" [static]

5.184.3.2 portaudio_callback PaStreamCallback portaudio_callback

5.185 MHAIOTCP.cpp File Reference

Classes

- class **io_tcp_parser_t**
The parser interface of the IOTCP library.
- class **io_tcp_sound_t**
Sound data handling of io tcp library.
- union **io_tcp_sound_t::float_union**
This union helps in conversion of floats from host byte order to network byte order and back again.
- class **io_tcp_fwcb_t**
TCP sound-io library's interface to the framework callbacks.
- class **io_tcp_t**
The tcp sound io library.

Macros

- #define **ERR_SUCCESS** 0
- #define **ERR_IHANDLE** -1
- #define **ERR_USER** -1000
- #define **MAX_USER_ERR** 0x2000
- #define **MHA_ErrorMsg2**(x, y) **MHA_Error**(__FILE__, __LINE__, (x), (y))
- #define **MHA_ErrorMsg3**(x, y, z) **MHA_Error**(__FILE__, __LINE__, (x), (y), (z))
- #define **MIN_TCP_PORT** 0
- #define **MIN_TCP_PORT_STR** "0"
- #define **MAX_TCP_PORT** 65535
- #define **MAX_TCP_PORT_STR** "65535"
- #define **IOInit** MHA_STATIC_MHAIOTCP_IOInit
- #define **IOPrepare** MHA_STATIC_MHAIOTCP_IOPrepare
- #define **IOStart** MHA_STATIC_MHAIOTCP_IOStart
- #define **IOStop** MHA_STATIC_MHAIOTCP_IOStop
- #define **IOResource** MHA_STATIC_MHAIOTCP_IOResource
- #define **IOSetVar** MHA_STATIC_MHAIOTCP_IOSetVar
- #define **IOStrError** MHA_STATIC_MHAIOTCP_IOStrError
- #define **IODestroy** MHA_STATIC_MHAIOTCP_IODestroy
- #define **dummy_interface_test** MHA_STATIC_MHAIOTCP_dummy_interface_test

Functions

- static int **copy_error** (**MHA_Error** &e)
- static void * **thread_startup_function** (void *parameter)
- int **IOInit** (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void *proc_← handle, **IOStartedEvent_t** start_event, void *start_handle, **IOStoppedEvent_t** stop_← event, void *stop_handle, void **handle)
- int **IOPrepare** (void *handle, int num_inchannels, int num_outchannels)
- int **IOStart** (void *handle)
- int **IOStop** (void *handle)
- int **IOResume** (void *handle)
- int **IORelease** (void *handle)
- int **IOSetVar** (void *handle, const char *cmd, char *retval, unsigned int len)
- const char * **IOStrError** (void *handle, int err)
- void **IODestroy** (void *handle)

Variables

- static char **user_err_msg** [**MAX_USER_ERR**]

5.185.1 Macro Definition Documentation

5.185.1.1 **ERR_SUCCESS** #define ERR_SUCCESS 0

5.185.1.2 **ERR_IHANDLE** #define ERR_IHANDLE -1

5.185.1.3 **ERR_USER** #define ERR_USER -1000

5.185.1.4 **MAX_USER_ERR** #define MAX_USER_ERR 0x2000

5.185.1.5 MHA_ErrorMsg2 #define MHA_ErrorMsg2(

```
  x,  
  y )  MHA_Error(__FILE__, __LINE__, (x), (y))
```

5.185.1.6 MHA_ErrorMsg3 #define MHA_ErrorMsg3(

```
  x,  
  y,  
  z )  MHA_Error(__FILE__, __LINE__, (x), (y), (z))
```

5.185.1.7 MIN_TCP_PORT #define MIN_TCP_PORT 0

5.185.1.8 MIN_TCP_PORT_STR #define MIN_TCP_PORT_STR "0"

5.185.1.9 MAX_TCP_PORT #define MAX_TCP_PORT 65535

5.185.1.10 MAX_TCP_PORT_STR #define MAX_TCP_PORT_STR "65535"

5.185.1.11 IOInit #define IOInit MHA_STATIC_MHAIOTCP_IOInit

5.185.1.12 IOPrepare #define IOPrepare MHA_STATIC_MHAIOTCP_IOPrepare

5.185.1.13 IOStart #define IOStart MHA_STATIC_MHAIOTCP_IOStart

5.185.1.14 IOStop #define IOStop MHA_STATIC_MHAIOTCP_IOStop

5.185.1.15 IOReset #define IOReset MHA_STATIC_MHAIOTCP_IOReset

5.185.1.16 IOSetVar #define IOSetVar MHA_STATIC_MHAIOTCP_IOSetVar

5.185.1.17 IOStrError #define IOStrError MHA_STATIC_MHAIOTCP_IOStrError

5.185.1.18 IODestroy #define IODestroy MHA_STATIC_MHAIOTCP_IODestroy

5.185.1.19 dummy_interface_test #define dummy_interface_test MHA_STATIC_MHAIOTCP_dummy_interface_test

5.185.2 Function Documentation

5.185.2.1 copy_error() static int copy_error (
 MHA_Error & e) [static]

5.185.2.2 `thread_startup_function()` static void* thread_startup_function (void * parameter) [static]

5.185.2.3 `IOInit()` int IOInit (int fragsize, float samplerate, **IOProcessEvent_t** proc_event, void * proc_handle, **IOStartedEvent_t** start_event, void * start_handle, **IOSoppedEvent_t** stop_event, void * stop_handle, void ** handle)

5.185.2.4 `IOPrepare()` int IOPrepare (void * handle, int num_inchannels, int num_outchannels)

5.185.2.5 `IOStart()` int IOStart (void * handle)

5.185.2.6 `IOStop()` int IOStop (void * handle)

5.185.2.7 `IOResale()` int IOResale (void * handle)

```
5.185.2.8 IOSetVar() int IOSetVar (
    void * handle,
    const char * cmd,
    char * retval,
    unsigned int len )
```

```
5.185.2.9 IOStrError() const char* IOStrError (
    void * handle,
    int err )
```

```
5.185.2.10 IODestroy() void IODestroy (
    void * handle )
```

5.185.3 Variable Documentation

```
5.185.3.1 user_err_msg char user_err_msg[ MAX_USER_ERR] [static]
```

5.186 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

5.186.1 Function Documentation

5.186.1.1 `jack_error_handler()` static void jack_error_handler (const char * msg) [static]

5.186.1.2 `dummy_jack_proc_cb()` static int dummy_jack_proc_cb (jack_nframes_t , void *) [static]

5.186.1.3 `make_friendly_number()` void make_friendly_number (jack_default_audio_sample_t & x) [inline]

5.186.2 Variable Documentation

5.186.2.1 `last_jack_err_msg` char last_jack_err_msg[**MAX_USER_ERR**] = ""

5.186.2.2 `last_jack_err` int last_jack_err = 0

5.187 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**]

5.187.1 Macro Definition Documentation

5.187.1.1 MHAJACK_FW_STARTED #define MHAJACK_FW_STARTED 1

5.187.1.2 MHAJACK_STOPPED #define MHAJACK_STOPPED 2

5.187.1.3 MHAJACK_STARTING #define MHAJACK_STARTING 8

5.187.1.4 IO_ERROR_JACK #define IO_ERROR_JACK 11

5.187.1.5 IO_ERROR_MHAJACKLIB #define IO_ERROR_MHAJACKLIB 12

5.187.1.6 MAX_USER_ERR #define MAX_USER_ERR 0x500

5.187.2 Variable Documentation

5.187.2.1 last_jack_err_msg char last_jack_err_msg[MAX_USER_ERR]

5.188 mhamain.cpp File Reference

Classes

- class **mhaserver_t**
MHA Framework listening on TCP port for commands.
- class **mhaserver_t::tcp_server_t**

Macros

- #define **HELP_TEXT**
- #define **NORELEASE_WARNING**
- #define **VERSION_EXTENSION** "+"
- #define **GREETING_TEXT**

Functions

- void **create_lock** (unsigned int p, const std::string &s)
- void **remove_lock** (unsigned int p)
- int **mhamain** (int argc, char *argv[])

5.188.1 Macro Definition Documentation

5.188.1.1 **HELP_TEXT** #define HELP_TEXT

5.188.1.2 **NORELEASE_WARNING** #define NORELEASE_WARNING

5.188.1.3 **VERSION_EXTENSION** #define VERSION_EXTENSION "+"

5.188.1.4 **GREETING_TEXT** #define GREETING_TEXT

5.188.2 Function Documentation

5.188.2.1 `create_lock()` `void create_lock (`
`unsigned int p,`
`const std::string & s)`

5.188.2.2 `remove_lock()` `void remove_lock (`
`unsigned int p)`

5.188.2.3 `mhamain()` `int mhamain (`
`int argc,`
`char * argv[])`

5.189 `mhapluginloader.cpp` File Reference

5.190 `mhapluginloader.h` File Reference

Classes

- class `PluginLoader::config_file_splitter_t`
- class `PluginLoader::fourway_processor_t`

This abstract class defines the interface for classes that implement all types of signal domain processing supported by the MHA: wave2wave, spec2spec, wave2spec, and spec2wave.

- class `PluginLoader::mhapluginloader_t`
- class `MHAParser::mhapluginloader_t`

Class to create a plugin loader in a parser, including the load logic.

Namespaces

- `PluginLoader`
- `MHAParser`

Name space for the openMHA-Parser configuration language.

Functions

- const char * `PluginLoader::mhastrdomain (mha_domain_t)`
- void `PluginLoader::mhaconfig_compare (const mhaconfig_t &req, const mhaconfig_t &avail, const std::string &pref="")`

Compare two `mhaconfig_t` (p. 850) structures, and report differences as an error.

5.191 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)

5.191.1 Function Documentation

5.191.1.1 write_wave() void write_wave (

```
    const mha_wave_t & sig,
    const char * fname,
    const float & srate,
    const int & format )
```

5.191.1.2 validator_channels() unsigned int validator_channels (

```
    std::vector< int > channel_map,
    unsigned int channels )
```

5.191.1.3 validator_length() unsigned int validator_length (

```
    unsigned int maxlen,
    unsigned int frames,
    unsigned int startpos )
```

5.192 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=SFFORMAT_WAV|SF_FORMAT_FLOAT|SF_ENDIAN_FILE)

5.192.1 Function Documentation

```
5.192.1.1 write_wave() void write_wave (
    const mha_wave_t & sig,
    const char * fname,
    const float & srate = 44100,
    const int & format = SF_FORMAT_WAV|SF_FORMAT_FLOAT|SF_ENDIAN_FILE )
```

5.193 multibandcompressor.cpp File Reference

Classes

- class **multibandcompressor::plugin_signals_t**
- class **multibandcompressor::fftfb_plug_t**
- class **multibandcompressor::interface_t**

Namespaces

- **multibandcompressor**

5.194 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)

5.194.1 Macro Definition Documentation

5.194.1.1 **NORMALIZATION_TYPES** #define NORMALIZATION_TYPES "[none default sum]"

5.194.1.2 **NORM_NONE** #define NORM_NONE 0

5.194.1.3 **NORM_DEFAULT** #define NORM_DEFAULT 1

5.194.1.4 **NORM_SUM** #define NORM_SUM 2

5.194.1.5 **ESTIMATION_TYPES** #define ESTIMATION_TYPES "[previous current]"

5.194.1.6 ESTIM_PREV #define ESTIM_PREV 0

5.194.1.7 ESTIM_CUR #define ESTIM_CUR 1

5.194.2 Function Documentation

5.194.2.1 make_friendly_number_by_limiting() void make_friendly_number_by_limiting
(
 mha_real_t & x) [inline]

5.195 noise.cpp File Reference

Classes

- class **cfg_t**
- class **noise_t**

5.196 noise_psd_estimator.cpp File Reference

Classes

- class **noise_psd_estimator::noise_psd_estimator_t**
- class **noise_psd_estimator::noise_psd_estimator_if_t**

Namespaces

- **noise_psd_estimator**

Macros

- #define **POWSPEC_FACTOR** 0.0025

5.196.1 Macro Definition Documentation

5.196.1.1 POWSPEC_FACTOR `#define POWSPEC_FACTOR 0.0025`

5.197 osc2ac.cpp File Reference

Classes

- class **osc_variable_t**
Class for converting messages received at a single osc address to a single AC variable.
- class **osc_server_t**
OSC receiver implemented using liblo.
- class **osc2ac_t**

5.198 overlapadd.cpp File Reference

Namespaces

- **overlapadd**

5.199 overlapadd.hh File Reference

Classes

- class **overlapadd::overlapadd_t**
- class **overlapadd::overlapadd_if_t**

Namespaces

- **overlapadd**

5.200 plingploing.cpp File Reference

Classes

- class **plingploing::plingploing_t**
Run-time configuration of the plingploing music generator.
- class **plingploing::if_t**
Plugin class of the plingploing music generator.

Namespaces

- **plingploing**

All classes for the plingploing music generator live in this namespace.

Functions

- double **plingploing::drand** (double a, double b)

5.201 pluginbrowser.cpp File Reference

5.202 pluginbrowser.h File Reference

Classes

- class **plugindescription_t**
- class **pluginloader_t**
- class **pluginbrowser_t**

5.203 resampling.cpp File Reference

Classes

- class **MHAPlugIn_Resampling::resampling_t**
- class **MHAPlugIn_Resampling::resampling_if_t**

Namespaces

- **MHAPlugIn_Resampling**

5.204 rmslevel.cpp File Reference

Classes

- class **rmslevel::mon_t**
- class **rmslevel::rmslevel_t**

Run-time configuration class of the rmslevel plugin.
- class **rmslevel::rmslevel_if_t**

Interface class of the rmslevel plugin.

Namespaces

- **rmslevel**

Enumerations

- enum **rmslevel::UNIT** { **rmslevel::UNIT::SPL** =0, **rmslevel::UNIT::HL** =1 }

5.205 rohBeam.cpp File Reference

Namespaces

- **rohBeam**

Variables

- auto **rohBeam::scalarify** =[](auto t){return t(0);}

5.206 rohBeam.hh File Reference

Classes

- struct **rohBeam::configOptions**
- class **rohBeam::rohConfig**
- class **rohBeam::rohBeam**

Namespaces

- **rohBeam**

Macros

- #define **NDEBUG**

Functions

- double **rohBeam::j0** (double x)
Cylindrical bessel function of the first kind of order 0.

Variables

- constexpr float `rohBeam::CONST_C` = 343.0115f
- constexpr int `rohBeam::refL` = 0
- constexpr int `rohBeam::refR` = 3

5.206.1 Macro Definition Documentation

5.206.1.1 `NDEBUG` #define NDEBUG

5.207 route.cpp File Reference

Classes

- class `route::process_t`
- class `route::interface_t`

Namespaces

- `route`

5.208 save_spec.cpp File Reference

Classes

- class `save_spec_t`

5.209 save_wave.cpp File Reference

Classes

- class `save_wave_t`

5.210 shadowfilter_begin.cpp File Reference

Classes

- class `shadowfilter_begin::cfg_t`
- class `shadowfilter_begin::shadowfilter_begin_t`

Namespaces

- `shadowfilter_begin`

5.211 shadowfilter_end.cpp File Reference

Classes

- class `shadowfilter_end::cfg_t`
- class `shadowfilter_end::shadowfilter_end_t`

Namespaces

- `shadowfilter_end`

5.212 sine.cpp File Reference

Classes

- struct `sine_cfg_t`
Runtime configuration of the sine plugin.
- class `sine_t`
Interface class of plugin sine, a sinusoid generator plugin.

5.213 smooth_cepstrum.cpp File Reference

Macros

- #define `INSERT_VAR(var)` `insert_item(#var, &var)`
- #define `PATCH_VAR(var)`
- #define `INSERT_PATCH(var)` `INSERT_VAR(var); PATCH_VAR(var)`

5.213.1 Macro Definition Documentation

5.213.1.1 `INSERT_VAR` `#define INSERT_VAR(`
`var) insert_item(#var, &var)`

5.213.1.2 `PATCH_VAR` `#define PATCH_VAR(`
`var)`

5.213.1.3 `INSERT_PATCH` `#define INSERT_PATCH(`
`var) INSERT_VAR(var); PATCH_VAR(var)`

5.214 `smooth_cepstrum.hh` File Reference

Classes

- class `smooth_cepstrum::smooth_params`
- class `smooth_cepstrum::smooth_cepstrum_t`
- class `smooth_cepstrum::smooth_cepstrum_if_t`

Namespaces

- `smooth_cepstrum`

5.215 `smoothgains_bridge.cpp` File Reference

Classes

- class `smoothgains_bridge::smoothspec_wrap_t`
- class `smoothgains_bridge::overlapadd_if_t`

Namespaces

- `smoothgains_bridge`

5.216 softclip.cpp File Reference

Classes

- class `cfg_t`
- class `softclip_t`

5.217 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)

5.217.1 Function Documentation

5.217.1.1 `max()` `unsigned int max (`
 `unsigned int a,`
 `unsigned int b) [inline]`

5.217.1.2 `min()` `unsigned int min (`
 `unsigned int a,`
 `unsigned int b) [inline]`

5.218 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}
- 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}

5.218.1 Macro Definition Documentation

5.218.1.1 NUM_ENTR_MHAORIG #define NUM_ENTR_MHAORIG 76

5.218.1.2 NUM_ENTR_LTASS #define NUM_ENTR_LTASS 25

5.218.1.3 NUM_ENTR_OLNOISE #define NUM_ENTR_OLNOISE 49

5.218.2 Function Documentation

5.218.2.1 fhz2bandno() float fhz2bandno (float x)

5.218.2.2 erb_hz_f_hz() float erb_hz_f_hz (float f_hz)

5.218.2.3 hz2hz() 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

5.218.2.4 bandw_correction() float bandw_correction (float *f*, float *ldb*)

5.218.3 Variable Documentation

5.218.3.1 vMHAOrigSpec float vMHAOrigSpec[**NUM_ENTR_MHAORIG**] = { -1.473, 0, -4.←
 939, -10.14, -13.94, -14.83, -14.27, -15.66, -16.16, -18.22, -20.5, -21.23, -22.13,
 -22.58, -23.98, -26.58, -26.4, -25.15, -23.89, -25.54, -27, -30.15, -31.68, -30.←
 14, -27.55, -25.79, -25.89, -26.11, -27.48, -30.37, -33.13, -36.23, -36.64, -36.←
 35, -35.03, -35.48, -36.35, -37.95, -40.53, -42.37, -41.29, -38.49, -36.32, -34.85,
 -34.05, -33.81, -33.48, -34.1, -35.19, -36.29, -36.94, -37.53, -38.71, -38.7, -38.←
 92, -40.36, -41.26, -42.19, -43.65, -44.37, -43.95, -43.15, -42.57, -41.57, -41.←
 86, -42.34, -42.87, -42.35, -42.71, -42.85, -43.47, -47.43, -67.54, -76.3, -77.43,
 -77.43 }

5.218.3.2 vMHAOrigFreq float vMHAOrigFreq[**NUM_ENTR_MHAORIG**] = { 172.266, 344.←
 532, 516.797, 689.063, 861.329, 1033.59, 1205.86, 1378.13, 1550.39, 1722.66, 1894.92, 2067.←
 19, 2239.46, 2411.72, 2583.99, 2756.25, 2928.52, 3100.78, 3273.05, 3445.32, 3617.58, 3789.←
 85, 3962.11, 4134.38, 4306.64, 4478.91, 4651.18, 4823.44, 4995.71, 5167.97, 5340.24, 5512.←
 51, 5684.77, 5857.04, 6029.3, 6201.57, 6373.83, 6546.1, 6718.37, 6890.63, 7062.9, 7235.16, 7407.←
 43, 7579.69, 7751.96, 7924.23, 8096.49, 8268.76, 8441.02, 8613.29, 8785.56, 8957.82, 9130.←
 09, 9302.35, 9474.62, 9646.88, 9819.15, 9991.42, 10163.7, 10335.9, 10508.2, 10680.5, 10852.←
 7, 11025, 11197.3, 11369.5, 11541.8, 11714.1, 11886.3, 12058.6, 12230.9, 12403.1, 12575.←
 4, 12747.7, 12919.9, 13092.2 }

5.218.3.3 vLTASS_freq float vLTASS_freq[**NUM_ENTR_LTASS**] = { 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000 }

5.218.3.4 vLTASS_combined_lev float vLTASS_combined_lev[**NUM_ENTR_LTASS**] = { 38. ← 6, 43.5, 54.4, 57.7, 56.8, 60.2, 60.3, 59.0, 62.1, 62.1, 60.5, 56.8, 53.7, 53.0, 52.0, 48.7, 48.1, 46.8, 45.6, 44.5, 44.3, 43.7, 43.4, 41.3, 40.7 }

5.218.3.5 vLTASS_female_lev float vLTASS_female_lev[**NUM_ENTR_LTASS**] = { 37. ← 0, 36.0, 37.5, 40.1, 53.4, 62.2, 60.9, 58.1, 61.7, 61.7, 60.4, 58, 54.3, 52.3, 51.7, 48.8, 47. ← 3, 46.7, 45.3, 44.6, 45.2, 44.9, 45.0, 42.8, 41.1 }

5.218.3.6 vLTASS_male_lev float vLTASS_male_lev[**NUM_ENTR_LTASS**] = { 38.6, 43. ← 5, 54.4, 57.7, 56.8, 58.2, 59.7, 60.0, 62.4, 62.6, 60.6, 55.7, 53.1, 53.7, 52.3, 48.7, 48.9, 47. ← 0, 46.0, 44.4, 43.3, 42.4, 41.9, 39.8, 40.4 }

5.218.3.7 vOlnoiseFreq float vOlnoiseFreq[**NUM_ENTR_OLNOISE**] = { 62.5, 70.1539, 78. ← 7451, 88.3884, 99.2126, 111.362, 125, 140.308, 157.49, 176.777, 198.425, 222.725, 250, 280. ← 616, 314.98, 353.553, 396.85, 445.449, 500, 561.231, 629.961, 707.107, 793.701, 890.899, 1000, 1122. ← 46, 1259.92, 1414.21, 1587.4, 1781.8, 2000, 2244.92, 2519.84, 2828.43, 3174.8, 3563.59, 4000, 4489. ← 85, 5039.68, 5656.85, 6349.6, 7127.19, 8000, 8979.7, 10079.4, 11313.7, 12699.2, 14254.4, 16000 }

5.218.3.8 vOlnoiseLev float vOlnoiseLev[**NUM_ENTR_OLNOISE**] = { 45.9042, 38.044, 48. ← 9444, 61.3697, 67.6953, 69.7451, 71.6201, 71.2431, 65.2754, 63.2547, 70.2264, 72.1434, 73. ← 4433, 73.2659, 69.8424, 71.0132, 70.9577, 70.3492, 68.691, 64.8436, 64.0435, 64.2879, 60. ← 5889, 60.6596, 60.3727, 61.2003, 61.8477, 61.1478, 61.2312, 58.6584, 57.2892, 56.8299, 56. ← 0191, 53.3018, 56.0525, 54.3592, 50.8823, 55.992, 54.6768, 47.2616, 46.9914, 45.209, 50. ← 413, 47.5848, 43.3215, 43.754, 38.5773, -0.39427, 5.74224 }

5.219 speechnoise.h File Reference

Classes

- class **speechnoise_t**

5.220 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. 1178) instance manages the plugin that performs processing on the reduced set of channels.*
- class **MHAPlugin_Split::split_t**
Implements split plugin.

Namespaces

- **MHAPlugin_Split**

Macros

- #define **MHAPLUGIN_OVERLOAD_OUTDOMAIN**
This define modifies the definition of MHAPLUGIN_CALLBACKS and friends.
- #define **posixthreads** 1
- #define **native_thread_platform_type** posix_threads_t

Enumerations

- enum { **MHAPlugin_Split::INVALID_THREAD_PRIORITY** = 999999999 }
Invalid thread priority.

5.220.1 Detailed Description

Source code for the split plugin. The split plugin splits the audio signal by channel. The splitted paths execute in parallel.

5.220.2 Macro Definition Documentation

5.220.2.1 MHAPLUGIN_OVERLOAD_OUTDOMAIN #define MHAPLUGIN_OVERLOAD_OUTDO←
MAIN

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.

5.220.2.2 posixthreads #define posixthreads 1

5.220.2.3 native_thread_platform_type #define native_thread_platform_type posix_←
threads_t

5.221 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)

5.221.1 Macro Definition Documentation

5.221.1.1 PATCH_VAR #define PATCH_VAR(
var) patchbay.connect(&var.valuechanged, this, & **steerbf**::**update_cfg**)

5.221.1.2 INSERT_PATCH #define INSERT_PATCH(
var) **insert_member**(var); **PATCH_VAR**(var)

5.222 steerbf.h File Reference

Classes

- class `parser_int_dyn`
- class `steerbf_config`
- class `steerbf`

5.223 testalsadvice.c File Reference

Functions

- int `main` (int argc, char **argv)

5.223.1 Function Documentation

```
5.223.1.1 main() int main (
    int argc,
    char ** argv )
```

5.224 testplugin.cpp File Reference

Classes

- class `testplugin::config_parser_t`
- class `testplugin::ac_parser_t`
- class `testplugin::signal_parser_t`
- class `testplugin::if_t`

Namespaces

- `testplugin`

5.225 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)

5.225.1 Typedef Documentation

5.225.1.1 rmslevelmeter `typedef MHAPlugin::config_t< MHASignal::async_rmslevel_t > rmslevelmeter`

5.225.1.2 rtcalibrator `typedef MHAPlugin::plugin_t< calibrator_runtime_layer_t > rtcalibrator`

5.225.2 Function Documentation

5.225.2.1 `kw_index2type()` `speechnoise_t::noise_type_t kw_index2type (`
`unsigned int idx)`

5.225.2.2 `vint_0123n1()` `std::vector<int> vint_0123n1 (`
`unsigned int n)`

5.226 upsample.cpp File Reference

Classes

- class `us_t`

5.227 wave2spec.cpp File Reference

5.228 wave2spec.hh File Reference

Classes

- class `wave2spec_t`
Runtime configuration class for plugin wave2spec.
- class `wave2spec_if_t`
Plugin wave2spec interface class, uses `wave2spec_t` (p. 1491) as runtime configuration.

Macros

- `#define MHAPLUGIN_OVERLOAD_OUTDOMAIN`

5.228.1 Macro Definition Documentation

5.228.1.1 `MHAPLUGIN_OVERLOAD_OUTDOMAIN` `#define MHAPLUGIN_OVERLOAD_OUTDO→`
`MAIN`

5.229 wavrec.cpp File Reference

Classes

- class **wavwriter_t**
- class **wavrec_t**

5.230 windnoise.cpp File Reference

Namespaces

- **windnoise**

namespace for plugin windnoise which detects and cancels wind noise

Macros

- #define **register_configuration_variable(v)**

5.230.1 Macro Definition Documentation

5.230.1.1 **register_configuration_variable** #define register_configuration_variable(v)

5.231 windnoise.hh File Reference

Classes

- class **windnoise::cfg_t**
Runtime config class for windnoise plugin.
- class **windnoise::if_t**
interface class for windnoise plugin

Namespaces

- **windnoise**

namespace for plugin windnoise which detects and cancels wind noise

5.232 windowselector.cpp File Reference

5.233 windowselector.h File Reference

Classes

- class **windowselector_t**
A combination of mha parser variables to describe an overlapped analysis window.

Index

_MHA_AC_CHAR
 testplugin::ac_parser_t, 1475

_MHA_AC_DOUBLE
 testplugin::ac_parser_t, 1475

_MHA_AC_FLOAT
 testplugin::ac_parser_t, 1475

_MHA_AC_INT
 testplugin::ac_parser_t, 1475

_MHA_AC_MHACOMPLEX
 testplugin::ac_parser_t, 1475

_MHA_AC_MHAREAL
 testplugin::ac_parser_t, 1475

_MHA_FUN_
 mha_defs.h, 1582

_declspec
 example5.cpp, 1545
 example6.cpp, 1546
 mha_defs.h, 1583
 mha_plugin.hh, 1616

_ac
 gtfb_simple_rt_t, 568

_cf
 DynComp::dc_afterburn_t, 454

_channels
 DynComp::dc_afterburn_t, 454

_conjugate
 Complex arithmetics in the openMHA, 67

_fmax
 audiometerbackend::lnn3rdoct_t, 322

_fmin
 audiometerbackend::lnn3rdoct_t, 322

_linphase_asym
 MHAFilter::smoothspec_t, 932

_order
 gtfb_simple_rt_t, 566

_pre_stages
 gtfb_simple_rt_t, 567

_prepare
 testplugin::if_t, 1482

_reciprocal
 Complex arithmetics in the openMHA, 67

_srates
 DynComp::dc_afterburn_t, 454

_steerbf
 steerbf_config, 1473

_unknown
 testplugin::ac_parser_t, 1475

~Async_Notify
 MHA_TCP::Async_Notify, 798

~Connection
 MHA_TCP::Connection, 804

~Delay
 ADM::Delay< F >, 267

~Event_Watcher
 MHA_TCP::Event_Watcher, 812

~Linearphase_FIR
 ADM::Linearphase_FIR< F >, 269

~MHA_Error
 MHA_Error, 764

~Server
 MHA_TCP::Server, 815

~Thread
 MHA_TCP::Thread, 830

~Timeout_Watcher
 MHA_TCP::Timeout_Watcher, 834

~Wakeups_Event
 MHA_TCP::Wakeups_Event, 836

~acConcat_wave
 acConcat_wave, 200

~acConcat_wave_config
 acConcat_wave_config, 202

~acPooling_wave
 acPooling_wave, 212

~acPooling_wave_config
 acPooling_wave_config, 216

~acSteer
 acSteer, 230

~acSteer_config
 acSteer_config, 232

~actransform_wave
 actransform_wave, 235

~actransform_wave_config
 actransform_wave_config, 238

~acmon_t
 acmon::acmon_t, 208

~acspace2matrix_t
 MHA_AC::acspace2matrix_t, 726

~acwriter_t
 plugins::hoertech::acrec::acwriter_t, 1381

~adaptive_feedback_canceller
 adaptive_feedback_canceller, 241

~adaptive_feedback_canceller_config
 adaptive_feedback_canceller_config, 245

~adm_rtconfig_t
 adm_rtconfig_t, 277

~algo_comm_class_t

MHAKernel::algo_comm_class_t, 986
 ~alsa_base_t
 alsa_base_t, 287
 ~alsa_t
 alsa_t< T >, 292
 ~analysepath_t
 analysepath_t, 307
 ~analysispath_if_t
 analysispath_if_t, 311
 ~bark2hz_t
 MHAOvIFilter::barkscale::bark2hz_t, 999
 ~base_t
 MHAParser::base_t, 1030
 ~bbcalib_interface_t
 bbcalib_interface_t, 335
 ~blockprocessing_polyphase_resampling_t
 MHAFilter::blockprocessing_polyphase_resamp
 869
 ~c_ifc_parser_t
 MHAParser::c_ifc_parser_t, 1046
 ~cfg_node_t
 MHAPlugin::cfg_node_t< runtime_cfg_t >, 1139
 ~cfg_t
 acsave::cfg_t, 223
 equalize::cfg_t, 462
 ~config_t
 MHAPlugin::config_t< runtime_cfg_t >, 1143
 ~connector_base_t
 MHAEvents::connector_base_t, 853
 ~connector_t
 MHAEvents::connector_t< receiver_t >, 856
 ~db_if_t
 db_if_t, 371
 dbasync_native::db_if_t, 375
 ~dbasync_t
 dbasync_native::dbasync_t, 378
 ~delay_spec_t
 MHASignal::delay_spec_t, 1198
 ~delay_t
 MHASignal::delay_t, 1200
 ~delay_wave_t
 MHASignal::delay_wave_t, 1202
 ~doasvm_classification
 doasvm_classification, 421
 ~doasvm_classification_config
 doasvm_classification_config, 424
 ~doasvm_feature_extraction
 doasvm_feature_extraction, 426
 ~doasvm_feature_extraction_config
 doasvm_feature_extraction_config, 430
 ~domain_handler_t
 MHAPlugin_Split::domain_handler_t, 1160
 ~double2acvar_t
 double2acvar::double2acvar_t, 434
 ~double_t
 MHA_AC::double_t, 729
 ~doublebuffer_t
 MHASignal::doublebuffer_t, 1205
 ~dynamiclib_t
 dynamiclib_t, 447
 ~emitter_t
 MHAEvents::emitter_t, 859
 ~fft_t
 MHAEvents::fft_t, 1208
 ~fftfb_t
 MHAOvIFilter::fftfb_t, 1004
 ~fftfilter_t
 MHAFilter::fftfilter_t, 877
 ~fftfilterbank_t
 MHAFilter::fftfilterbank_t, 883
 ~filter_t
 MHAFilter::filter_t, 888
 ~float_t
 MHA_AC::float_t, 731
 ~fourway_processor_t
 PluginLoader::fourway_processor_t, 1363
 ~fshift_config_t
 fshift::fshift_config_t, 509
 ~fshift_t
 fshift::fshift_t, 512
 ~fw_t
 fw_t, 524
 ~gaintable_t
 DynComp::gaintable_t, 458
 ~gamma_flt_t
 MHAFilter::gamma_flt_t, 892
 ~gsc_adaptive_stage
 gsc_adaptive_stage::gsc_adaptive_stage, 538
 ~gsc_adaptive_stage_if
 gsc_adaptive_stage::gsc_adaptive_stage_if, 545
 ~gtfb_analyzer_cfg_t
 gtfb_analyzer::gtfb_analyzer_cfg_t, 550
 ~gtfb_simd_cfg_t
 gtfb_simd_cfg_t, 557
 ~hanning_ramps_t
 hanning_ramps_t, 574

~hilbert_fftw_t
 MHASignal::hilbert_fftw_t, 1212

~hilbert_shifter_t
 fshift_hilbert::hilbert_shifter_t, 519

~hilbert_t
 MHASignal::hilbert_t, 1215

~hz2bark_t
 MHAOvIFilter::barkscale::hz2bark_t, 1000

~int_t
 MHA_AC::int_t, 733

~io_asterisk_fwcb_t
 io_asterisk_fwcb_t, 585

~io_asterisk_parser_t
 io_asterisk_parser_t, 590

~io_asterisk_sound_t
 io_asterisk_sound_t, 597

~io_asterisk_t
 io_asterisk_t, 601

~io_file_t
 io_file_t, 606

~io_lib_t
 io_lib_t, 612

~io_parser_t
 io_parser_t, 616

~io_portaudio_t
 MHAIOPortAudio::io_portaudio_t, 958

~io_tcp_fwcb_t
 io_tcp_fwcb_t, 621

~io_tcp_parser_t
 io_tcp_parser_t, 626

~io_tcp_sound_t
 io_tcp_sound_t, 633

~io_tcp_t
 io_tcp_t, 638

~io_wrapper
 io_wrapper, 642

~level_matching_config_t
 level_matching::level_matching_config_t,
 651

~level_matching_t
 level_matching::level_matching_t, 654

~linear_table_t
 MHATableLookup::linear_table_t, 1276

~lpc
 lpc, 661

~lpc_bl_predictor
 lpc_bl_predictor, 664

~lpc_bl_predictor_config
 lpc_bl_predictor_config, 667

~lpc_burglattice
 lpc_burglattice, 670

~lpc_burglattice_config
 lpc_burglattice_config, 673

~lpc_config
 lpc_config, 676

~matlab_wrapper_rt_cfg_t
 matlab_wrapper::matlab_wrapper_rt_cfg_t,
 695

~matrix_t
 MHASignal::matrix_t, 1226

~mha_dblbuf_t
 mha_dblbuf_t< FIFO >, 749

~mha_fifo_lw_t
 mha_fifo_lw_t< T >, 770

~mha_fifo_posix_threads_t
 mha_fifo_posix_threads_t, 773

~mha_fifo_t
 mha_fifo_t< T >, 778

~mha_fifo_thread_guard_t
 mha_fifo_thread_guard_t, 783

~mha_fifo_thread_platform_t
 mha_fifo_thread_platform_t, 784

~mha_rt_fifo_element_t
 mha_rt_fifo_element_t< T >, 788

~mha_rt_fifo_t
 mha_rt_fifo_t< T >, 790

~mha_stash_environment_variable_t
 mha_stash_environment_variable_t, 796

~mhaplug_cfg_t
 mhaplug_cfg_t, 1138

~mhaplugloader_t
 MHAParser::mhaplugloader_t, 1087
 PluginLoader::mhaplugloader_t, 1368

~mhaserver_t
 mhaserver_t, 1191

~osc_server_t
 osc_server_t, 1321

~overlapadd_if_t
 overlapadd::overlapadd_if_t, 1329
 smoothgains_bridge::overlapadd_if_t,
 1449

~overlapadd_t
 overlapadd::overlapadd_t, 1332

~parser_t
 MHAParser::parser_t, 1099

~partitioned_convolution_t
 MHAFilter::partitioned_convolution_t, 916

~patchbay_t
 MHAEvents::patchbay_t< receiver_t >,
 861

~plug_t
 plug_t, 1346

~plug_wrapper
 plug_wrapper, 1348
 ~plug_wrapperl
 plug_wrapperl, 1350
 ~plugin_t
 MHAPlugin::plugin_t< runtime_cfg_t >, 1148
 ~pluginlib_t
 pluginlib_t, 1358
 ~pluginloader_t
 pluginloader_t, 1374
 ~plugs_t
 mhachain::plugs_t, 847
 ~port_t
 MHAJack::port_t, 982
 ~posix_threads_t
 MHAPlugin_Split::posix_threads_t, 1168
 ~rmslevel_t
 rmslevel::rmslevel_t, 1391
 ~rohBeam
 rohBeam::rohBeam, 1396
 ~rohConfig
 rohBeam::rohConfig, 1403
 ~rt_nlms_t
 rt_nlms_t, 1414
 ~save_var_base_t
 ac2lsl::save_var_base_t, 170
 ~save_var_t
 ac2lsl::save_var_t< mha_complex_t >, 177
 ac2lsl::save_var_t< T >, 173
 acsave::save_var_t, 227
 lsl2ac::save_var_t, 687
 ~server_t
 mha_tcp::server_t, 820
 ~sf_t
 MHASndFile::sf_t, 1272
 ~smooth_cepstrum_t
 smooth_cepstrum::smooth_cepstrum_t, 1439
 ~smoothspec_t
 MHAFilter::smoothspec_t, 930
 ~spec2wave_t
 spec2wave_t, 1463
 ~spec_fader_t
 spec_fader_t, 1465
 ~spectrum_t
 MHA_AC::spectrum_t, 735
 MHASignal::spectrum_t, 1246
 ~split_t
 MHAPlugin_Split::split_t, 1173
 ~splitted_part_t
 MHAPlugin_Split::splitted_part_t, 1180
 ~steerbf
 steerbf, 1470
 ~steerbf_config
 steerbf_config, 1472
 ~table_t
 MHATableLookup::table_t, 1280
 ~thread_platform_t
 MHAPlugin_Split::thread_platform_t, 1186
 ~uint_vector_t
 MHASignal::uint_vector_t, 1256
 ~uni_processor_t
 MHAPlugin_Split::uni_processor_t, 1188
 ~wave2spec_t
 wave2spec_t, 1493
 ~waveform_t
 MHA_AC::waveform_t, 739
 MHASignal::waveform_t, 1262
 ~wavewriter_t
 wavewriter_t, 1501
 ~windowselector_t
 windowselector_t, 1514
 ~wrapped_plugin_t
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 704

A

- ipc_config, 677
- MHAFilter::filter_t, 890
- MHAFilter::gamma_filt_t, 894
- MHAFilter::iir_filter_t, 899

a

- MHAParser::base_t::replace_t, 1040

A_

- MHAFilter::complex_bandpass_t, 874
- MHAFilter::iir_ord1_real_t, 903

abandonned

- mha_rt_fifo_element_t< T >, 789

abs

- Complex arithmetics in the openMHA, 65

abs2

- Complex arithmetics in the openMHA, 65

ac

- ac2lsl::cfg_t, 169
- ac2wave_t, 190
- ac_mul_t, 194
- acConcat_wave_config, 203
- acmon::acmon_t, 209
- acPooling_wave_config, 217
- acsave::cfg_t, 224

acsave::save_var_t, 228
acTransform_wave_config, 238
adaptive_feedback_canceller_config, 246
doasvm_classification_config, 424
fw_t, 529
gsc_adaptive_stage::gsc_adaptive_stage, 539
latex_doc_t, 646
lpc_bl_predictor_config, 668
lpc_burglattice_config, 673
lsl2ac::save_var_t, 690
MHA_AC::ac2matrix_helper_t, 721
MHA_AC::double_t, 730
MHA_AC::float_t, 731
MHA_AC::int_t, 733
MHA_AC::spectrum_t, 736
MHA_AC::waveform_t, 740
mhachain::plugs_t, 848
MHAKernel::algo_comm_class_t, 989
MHAMultiSrc::base_t, 992
MHAPlugin::plugin_t< runtime_cfg_t >, 1151
PluginLoader::mhaplugloader_t, 1371
plugins::hoertech::acrec::acrec_t, 1379
rt_nlms_t, 1414
shadowfilter_end::cfg_t, 1425
smooth_cepstrum::smooth_cepstrum_t, 1440
steerbf_config, 1473
testplugin::if_t, 1482
AC variable, 4
ac2lsl, 78
 types, 78
ac2lsl.cpp, 1516
ac2lsl::ac2lsl_t, 162
 ac2lsl_t, 163
 activate, 165
 get_all_names_from_ac_space, 164
 is_first_run, 166
 patchbay, 165
 prepare, 163
 process, 164
 release, 164
 rt_strict, 165
 skip, 165
 source_id, 165
 update, 165
 vars, 165
ac2lsl::cfg_t, 166
 ac, 169
 cfg_t, 167
 check_vars, 167
 create_or_replace_var, 167
 process, 168
 skip, 168
 skipcnt, 168
 source_id, 168
 srate, 168
 update_varlist, 168
 varlist, 168
ac2lsl::save_var_base_t, 169
 ~save_var_base_t, 170
 data_type, 171
 get_buf_address, 170
 info, 170
 num_entries, 171
 send_frame, 170
 set_buf_address, 170
ac2lsl::save_var_t< mha_complex_t >, 175
 ~save_var_t, 177
 buf, 178
 data_type, 178
 get_buf_address, 177
 info, 177
 num_entries, 177
 save_var_t, 176
 send_frame, 178
 set_buf_address, 177
 stream, 178
ac2lsl::save_var_t< T >, 171
 ~save_var_t, 173
 buf, 175
 data_type, 174
 data_type_, 175
 get_buf_address, 173
 info, 174
 num_entries, 174
 save_var_t, 172
 send_frame, 174
 set_buf_address, 173
 stream, 175
ac2lsl::type_info, 179
 format, 179
 name, 179
ac2lsl_t
 ac2lsl::ac2lsl_t, 163
ac2matrix_helper_t
 MHA_AC::ac2matrix_helper_t, 721
ac2matrix_t
 MHA_AC::ac2matrix_t, 723
ac2osc.cpp, 1517
ac2osc_t, 180

ac2osc_t, 181
 acspace, 184
 b_record, 184
 framerate, 184
 host, 183
 is_first_run, 185
 lo_addr, 185
 mode, 183
 patchbay, 184
 port, 183
 prepare, 182
 process, 182
 release, 182
 rt_strict, 184
 rtmem, 184
 send_osc_float, 182
 skip, 184
 skipcnt, 184
 ttl, 183
 update_mode, 183
 vars, 183
 ac2wave.cpp, 1517
 ac2wave_if_t, 185
 ac2wave_if_t, 186
 delay_ac, 188
 delay_in, 188
 gain_ac, 187
 gain_in, 187
 name, 187
 patchbay, 188
 prepare, 187
 prepared, 188
 process, 186, 187
 release, 187
 update, 187
 zeros, 188
 ac2wave_t, 188
 ac, 190
 ac2wave_t, 189
 channels, 189
 delay_ac, 190
 delay_in, 190
 frames, 189
 gain_ac, 190
 gain_in, 190
 name, 190
 process, 189
 w, 190
 ac_
 combc_t, 359
 MHAParser::mhapluginloader_t, 1089
 ac_data
 osc_variable_t, 1326
 AC_DIM_MISMATCH
 mha_algo_comm.cpp, 1579
 ac_double
 double2acvar::double2acvar_t, 435
 ac_fifo
 analysepath_t, 308
 ac_insert
 osc_server_t, 1322
 osc_variable_t, 1325
 AC_INVALID_HANDLE
 mha_algo_comm.cpp, 1579
 AC_INVALID_NAME
 mha_algo_comm.cpp, 1579
 AC_INVALID_OUTPTR
 mha_algo_comm.cpp, 1579
 ac_monitor_t
 acmon::ac_monitor_t, 205
 ac_monitor_type.cpp, 1517
 ac_monitor_type.hh, 1517
 ac_mul.cpp, 1518
 ac_mul.hh, 1518
 ARG_CC, 1518
 ARG_CR, 1518
 ARG_RC, 1518
 ARG_RR, 1518
 arg_type_t, 1518
 VAL_COMPLEX, 1519
 VAL_REAL, 1519
 val_type_t, 1518
 ac_mul_t, 191
 ac, 194
 ac_mul_t, 192
 algo, 194
 argt, 195
 get_arg_type_and_dimension, 193, 194
 num_channels, 195
 num_frames, 195
 prepare_, 193
 process, 193, 194
 process_cc, 194
 process_cr, 194
 process_rc, 194
 process_rr, 194
 release_, 193
 res_c, 195
 res_r, 195
 scan_syntax, 193
 str_a, 195
 str_b, 195

ac_parser_t
 testplugin::ac_parser_t, 1475

ac_proc, 78

ac_proc.cpp, 1519

ac_proc::interface_t, 196
 algo, 198
 b_permute, 198
 input, 198
 interface_t, 197
 permute, 198
 plug, 198
 prepare, 197
 process, 197, 198
 release, 197
 s_in, 199
 s_in_perm, 198
 s_out, 198

ac_resynthesis_gain
 gtfb_simple_rt_t, 568

AC_STRING_TRUNCATED
 mha_algo_comm.cpp, 1579

AC_SUCCESS
 mha_algo_comm.cpp, 1579

AC_TYPE_MISMATCH
 mha_algo_comm.cpp, 1579

ac_wndshape_name
 wave2spec_t, 1496

acb
 gtfb_simple_rt_t, 567

accept
 MHA_TCP::Server, 816

accept_event
 MHA_TCP::Server, 817

accept_loop
 io_asterisk_t, 602
 io_tcp_t, 639

acceptor
 mha_tcp::server_t, 823

acceptor_started
 mhaserver_t, 1191

accf
 gtfb_simple_rt_t, 567

acConcat_wave, 199
 ~acConcat_wave, 200
 acConcat_wave, 200
 name_con_AC, 201
 num_AC, 201
 numchannels, 201
 patchbay, 202
 prefix_names_AC, 201
 prepare, 200

process, 200
release, 201
samples_AC, 201
update_cfg, 201

acConcat_wave.cpp, 1519
 INSERT_PATCH, 1519
 PATCH_VAR, 1519

acConcat_wave.h, 1520

acConcat_wave_config, 202
 ~acConcat_wave_config, 202

ac, 203

acConcat_wave_config, 202

numSamples_AC, 203

process, 203

strNames_AC, 203

vGCC, 203

vGCC_con, 203

ack_fail
 mhaserver_t, 1192

ack_ok
 mhaserver_t, 1192

acmon, 79

acmon.cpp, 1520

acmon::ac_monitor_t, 204
 ac_monitor_t, 205
 dimstr, 205
 getvar, 205
 mon, 206
 mon_complex, 206
 mon_mat, 206
 mon_mat_complex, 206
 name, 205
 p_parser, 206
 use_mat, 206

acmon::acmon_t, 207
 ~acmon_t, 208
 ac, 209

acmon_t, 208
 algo, 210
 b_cont, 210
 b_snapshot, 210
 chain, 210
 dimensions, 210
 dispmode, 210
 patchbay, 210
 prepare, 208
 process, 209
 recmode, 210
 release, 209
 save_vars, 209
 update_recmode, 209

varlist, 209
 vars, 210
 acmon_t
 acmon::acmon_t, 208
 acname
 osc_variable_t, 1326
 acPooling_wave, 211
 ~acPooling_wave, 212
 acPooling_wave, 212
 alpha, 214
 like_ratio_name, 215
 lower_threshold, 214
 max_pool_ind_name, 215
 neighbourhood, 214
 numsamples, 213
 p_biased_name, 214
 p_name, 214
 patchbay, 215
 pool_name, 214
 pooling_type, 214
 pooling_wndlen, 214
 prepare, 213
 prob_bias, 215
 process, 213
 release, 213
 update_cfg, 213
 upper_threshold, 214
 acPooling_wave.cpp, 1520
 INSERT_PATCH, 1520
 PATCH_VAR, 1520
 acPooling_wave.h, 1521
 acPooling_wave_config, 215
 ~acPooling_wave_config, 216
 ac, 217
 acPooling_wave_config, 216
 alpha, 218
 c, 217
 insert, 216
 like_ratio, 217
 low_thresh, 218
 neigh, 218
 p, 217
 p_biased, 217
 p_max, 217
 pool, 218
 pooling_ind, 217
 pooling_option, 217
 pooling_size, 218
 prob_bias_func, 218
 process, 216
 raw_p_name, 217
 up_thresh, 218
 acrec.cpp, 1521
 acrec.hh, 1521
 acrec_t
 plugins::hoertech::acrec::acrec_t, 1376
 acsave, 79
 acsave.cpp, 1521
 ACSAVE_FMT_M, 1522
 ACSAVE_FMT_MAT4, 1522
 ACSAVE_FMT_TXT, 1522
 ACSAVE_SFMT_M, 1522
 ACSAVE_SFMT_MAT4, 1522
 ACSAVE_SFMT_TXT, 1522
 acsave::acsave_t, 219
 acsave_t, 220
 algo, 222
 b_flushed, 222
 b_prepared, 222
 bflush, 221
 chain, 222
 event_start_recording, 221
 event_stop_and_flush, 221
 fileformat, 222
 fname, 222
 patchbay, 223
 prepare, 220
 process, 221
 reclen, 222
 release, 221
 variables, 222
 varlist, 222
 varlist_t, 220
 acsave::cfg_t, 223
 ~cfg_t, 223
 ac, 224
 cfg_t, 223
 flush_data, 224
 max_frames, 225
 nvars, 224
 rec_frames, 224
 store_frame, 224
 varlist, 224
 acsave::mat4head_t, 225
 cols, 225
 imag, 225
 namelen, 226
 rows, 225
 t, 225
 acsave::save_var_t, 226
 ~save_var_t, 227
 ac, 228

b_complex, 228
data, 227
framecnt, 228
maxframe, 228
name, 228
ndim, 228
nframes, 228
save_m, 227
save_mat4, 227
save_txt, 227
save_var_t, 226
store_frame, 227
ACSAVE_FMT_M
 acsave.cpp, 1522
ACSAVE_FMT_MAT4
 acsave.cpp, 1522
ACSAVE_FMT_TXT
 acsave.cpp, 1522
ACSAVE_SFMT_M
 acsave.cpp, 1522
ACSAVE_SFMT_MAT4
 acsave.cpp, 1522
ACSAVE_SFMT_TXT
 acsave.cpp, 1522
acsave_t
 acsave::acsave_t, 220
acspace
 ac2osc_t, 184
acspace2matrix_t
 MHA_AC::acspace2matrix_t, 725
acspace_template
 analysispath_if_t, 313
acSteer, 229
 ~acSteer, 230
 acSteer, 230
 acSteerName1, 231
 acSteerName2, 231
 nrefmic, 231
 nsteerchan, 231
 patchbay, 232
 prepare, 230
 process, 230
 release, 231
 steerFile, 231
 update_cfg, 231
acSteer.cpp, 1523
 INSERT_PATCH, 1523
 PATCH_VAR, 1523
acSteer.h, 1523
acSteer_config, 232
 ~acSteer_config, 232
acSteer_config, 232
 acSteer_name, 232
 insert, 233
 nangle, 233
 nchan, 233
 nfreq, 233
 nrefmic, 233
 nsteerchan, 233
 specSteer1, 233
 specSteer2, 233
acSteerName1
 acSteer, 231
acSteerName2
 acSteer, 231
act_
 wavwriter_t, 1502
actgains
 fader_if_t, 488
activate
 ac2lsl::ac2lsl_t, 165
 lsl2ac::lsl2ac_t, 683
activate_query
 MHAParser::base_t, 1036
active
 addsndfile::addsndfile_if_t, 254
 plugins::hoertech::acrec::acwriter_t, 1383
acTransform_wave, 234
 ~acTransform_wave, 235
 acTransform_wave, 235
 ang_name, 236
 numsamples, 237
 patchbay, 237
 prepare, 236
 process, 235
 raw_p_max_name, 237
 raw_p_name, 236
 release, 236
 rotated_p_max_name, 237
 rotated_p_name, 237
 to_from, 237
 update_cfg, 236
acTransform_wave.cpp, 1523
 INSERT_PATCH, 1524
 PATCH_VAR, 1523
acTransform_wave.h, 1524
acTransform_wave_config, 237
 ~acTransform_wave_config, 238
 ac, 238
 acTransform_wave_config, 238
 ang_name, 239
 offset, 239
 process, 238

raw_p_max_name, 239
 raw_p_name, 239
 resolution, 239
 rotated_i, 239
 rotated_p, 239
 to_from, 239
 acvar
 MHA_AC::ac2matrix_helper_t, 722
 acwriter_t
 plugins::hoertech::acrec::acwriter_t, 1381
 adapt_filter_param_t
 MHAFilter::adapt_filter_param_t, 863
 adapt_filter_state_t
 MHAFilter::adapt_filter_state_t, 864
 adapt_filter_t
 MHAFilter::adapt_filter_t, 866
 adaptation_ratio
 adm_if_t, 275
 adm_rtconfig_t, 279
 adaptive_feedback_canceller, 240
 ~adaptive_feedback_canceller, 241
 adaptive_feedback_canceller, 241
 afc_delay, 243
 c, 242
 delay_d, 243
 delay_w, 243
 gains, 243
 lpc_order, 243
 n_no_update, 244
 name_e, 243
 name_f, 243
 name_lpc, 243
 ntaps, 243
 patchbay, 244
 prepare, 242
 process, 241
 release, 242
 rho, 242
 update_cfg, 242
 adaptive_feedback_canceller.cpp, 1524
 INSERT_PATCH, 1524
 make_friendly_number_by_limiting, 1525
 PATCH_VAR, 1524
 adaptive_feedback_canceller.h, 1525
 adaptive_feedback_canceller_config, 244
 ~adaptive_feedback_canceller_config,
 245
 ac, 246
 adaptive_feedback_canceller_config, 245
 channels, 246
 EPrew, 249
 F, 246
 F_Uflt, 248
 frames, 246
 insert, 246
 iter, 247
 n_no_update_, 247
 name_d_, 247
 name_lpc_, 247
 no_iter, 247
 ntaps, 246
 process, 245
 PSD_val, 247
 Pu, 246
 s_E, 246
 s_E_afc_delay, 247
 s_LPC, 249
 s_U, 247
 s_U_delay, 248
 s_U_delayflt, 248
 s_Usmpl, 249
 s_W, 248
 s_Wflt, 248
 s_Y_delay, 248
 s_Y_delayflt, 248
 smpl, 249
 UbufferPrew, 248
 UPrew, 249
 UPrewW, 249
 v_G, 247
 YPrew, 249
 add
 MHASignal::loop_wavefragment_t, 1218
 add4f
 gtfb_simd.cpp, 1557
 add_connection
 mha_tcp::server_t, 823
 add_entry
 MHAParser::keyword_list_t, 1069
 MHATableLookup::linear_table_t, 1277
 MHATableLookup::xy_table_t, 1283
 add_fun
 MHAOvlFilter::scale_var_t, 1025
 add_parent_on_insert
 MHAParser::base_t, 1036
 add_plug
 altpugs_t, 304
 add_plugin
 pluginbrowser_t, 1354
 add_plugins
 pluginbrowser_t, 1354
 add_replace_pair

MHAParser::base_t, 1037
added_via_plugs
 altplugs_t, 305
addsndfile, 79
 addsndfile_resampling_mode_t, 80
 DO_RESAMPLE, 80
 DONT_RESAMPLE_PERMISSIVE, 80
 DONT_RESAMPLE_STRICT, 80
 level_adaptor, 80
 resampled_num_frames, 81
 wave_reader, 80
addsndfile.cpp, 1525
 DEBUG, 1526
addsndfile::addsndfile_if_t, 250
 active, 254
 addsndfile_if_t, 251
 change_mode, 252
 channels, 253
 filename, 252
 level, 253
 levelmode, 253
 loop, 252
 mapping, 253
 mhachannels, 254
 mode, 253
 numchannels, 253
 patchbay, 254
 path, 252
 prepare, 251
 process, 251
 ramplen, 253
 release, 252
 resamplingmode, 253
 scan_dir, 252
 search_pattern, 254
 search_result, 254
 set_level, 252
 startpos, 253
 uint_mode, 254
 update, 252
addsndfile::level_adapt_t, 255
 can_update, 256
 get_level, 256
 ilen, 256
 l_new, 256
 l_old, 257
 level_adapt_t, 255
 pos, 256
 update_frame, 256
 wnd, 256
addsndfile::resampled_soundfile_t, 257
 resampled_soundfile_t, 258
addsndfile::sndfile_t, 259
 sndfile_t, 259
addsndfile::waveform_proxy_t, 260
 waveform_proxy_t, 261
addsndfile_if_t
 addsndfile::addsndfile_if_t, 251
addsndfile_resampling_mode_t
 addsndfile, 80
ADM, 81
 ADM::ADM< F >, 262
 C, 82
 DELAY_FREQ, 82
 PI, 82
 START_BETA, 82
 subsampledelay_coeff, 81
adm
 adm_rtconfig_t, 278
adm.cpp, 1526
 adm_fir_decomb, 1527
 adm_fir_lp, 1527
adm.hh, 1527
ADM::ADM< F >, 261
 ADM, 262
 beta, 263
 m_beta, 264
 m_decomb, 264
 m_delay_back, 264
 m_delay_front, 264
 m_lp_bf, 264
 m_lp_result, 264
 m_mu_beta, 264
 m_powerfilter_coeff, 265
 m_powerfilter_norm, 265
 m_powerfilter_state, 265
 process, 263
ADM::Delay< F >, 265
 ~Delay, 267
 Delay, 266
 m_coeff, 267
 m_fullsamples, 267
 m_norm, 268
 m_now_in, 268
 m_state, 268
 process, 267
ADM::Linearphase_FIR< F >, 268
 ~Linearphase_FIR, 269
 Linearphase_FIR, 269
 m_alphas, 270
 m_now, 270
 m_order, 270

m_output, 270
 process, 270
 adm_fir_decomb
 adm.cpp, 1527
 adm_fir_lp
 adm.cpp, 1527
 adm_if_t, 271
 adaptation_ratio, 275
 adm_if_t, 272
 beta, 274
 bypass, 274
 coeff_decomb, 274
 coeff_lp, 274
 decomb_order, 274
 distances, 274
 framecnt, 275
 front_channels, 273
 input_channels, 275
 is_prepared, 273
 lp_order, 274
 mu_beta, 274
 out, 273
 patchbay, 275
 prepare, 273
 process, 272
 rear_channels, 273
 release, 273
 srate, 275
 tau_beta, 274
 update, 273
 adm_rtconfig_t, 275
 ~adm_rtconfig_t, 277
 adaptation_ratio, 279
 adm, 278
 adm_rtconfig_t, 277
 adm_t, 276
 adms, 279
 check_index, 278
 decomb_coeffs, 279
 front_channel, 278
 front_channels, 279
 get_adaptation_ratio, 278
 lp_coeffs, 279
 num_adms, 278
 rear_channel, 278
 rear_channels, 279
 adm_t
 adm_rtconfig_t, 276
 adms
 adm_rtconfig_t, 279
 afc_delay
 adaptive_feedback_canceller, 243
 algo
 ac_mul_t, 194
 ac_proc::interface_t, 198
 acmon::acmon_t, 210
 acsave::acsave_t, 222
 analysispath_if_t, 313
 coherence::cohflt_if_t, 350
 db_if_t, 372
 dbasync_native::db_if_t, 377
 dc::dc_if_t, 383
 fftfilterbank::fftfb_interface_t, 505
 MHAPlugin_Resampling::resampling_if_t,
 1155
 multibandcompressor::interface_t, 1302
 nlms_t, 1308
 overlapadd::overlapadd_if_t, 1331
 route::interface_t, 1411
 smoothgains_bridge::overlapadd_if_t,
 1451
 wave2spec_if_t, 1491
 algo_comm_class_t
 MHAKernel::algo_comm_class_t, 985
 algo_comm_default
 mha_algo_comm.cpp, 1579
 mha_algo_comm.hh, 1582
 ALGO_COMM_ID_STR
 mha_algo_comm.hh, 1581
 algo_comm_id_string
 MHAKernel::algo_comm_class_t, 989
 algo_comm_id_string_len
 MHAKernel::algo_comm_class_t, 989
 algo_comm_safe_cast
 MHAKernel, 115
 algo_comm_t, 280
 get_entries, 285
 get_error, 286
 get_var, 284
 get_var_double, 285
 get_var_float, 285
 get_var_int, 284
 handle, 281
 insert_var, 281
 insert_var_double, 282
 insert_var_float, 282
 insert_var_int, 281
 is_var, 283
 mha.hh, 1576
 remove_ref, 283
 remove_var, 282
 algo_name

lpc, 662
algos
 mhachain::chain_base_t, 843
 mhachain::plugs_t, 848
 MHAParser_Split::split_t, 1176
all_dump
 MHAParser, 127
all_ids
 MHAParser, 127
alloc_plugs
 mhachain::plugs_t, 847
almost
 Complex arithmetics in the openMHA, 68
alp
 gsc_adaptive_stage::gsc_adaptive_stage,
 541
 gsc_adaptive_stage::gsc_adaptive_stage_ifalphaPSD
 548
alph
 plingploing::plingploing_t, 1345
alpha
 acPooling_wave, 214
 acPooling_wave_config, 218
 cfg_t, 347
 coherence::cohfilt_t, 352
 coherence::vars_t, 355
alpha_blocking_XkXi
 rohBeam::configOptions, 1393
 rohBeam::rohConfig, 1406
alpha_blocking_XkY
 rohBeam::configOptions, 1394
 rohBeam::rohConfig, 1406
alpha_const
 smooth_cepstrum::smooth_cepstrum_t,
 1441
alpha_const_limits_hz
 smooth_cepstrum::smooth_cepstrum_if_t,
 1437
 smooth_cepstrum::smooth_params, 1447
alpha_const_vals
 smooth_cepstrum::smooth_cepstrum_if_t,
 1437
 smooth_cepstrum::smooth_params, 1447
alpha_frame
 smooth_cepstrum::smooth_cepstrum_t,
 1443
alpha_hat
 smooth_cepstrum::smooth_cepstrum_t,
 1442
alpha_Lowpass
 windnoise::cfg_t, 1506
alpha_pitch
 smooth_cepstrum::smooth_cepstrum_if_t,
 1436
 smooth_cepstrum::smooth_params, 1447
alpha_postfilter
 rohBeam::configOptions, 1393
 rohBeam::rohConfig, 1406
alpha_prev
 smooth_cepstrum::smooth_cepstrum_t,
 1441
alphaPH1mean
 noise_psd_estimator::noise_psd_estimator_if_t,
 1310
alphaPH1mean_
 noise_psd_estimator::noise_psd_estimator_t,
 1314
noise_psd_estimator::noise_psd_estimator_if_t,
 1311
alphaPSD_
 noise_psd_estimator::noise_psd_estimator_t,
 1314
alsa_base_t, 286
 ~alsa_base_t, 287
 alsa_base_t, 287
 pcm, 288
 read, 288
 start, 287
 stop, 287
 write, 288
alsa_dev_par_parser_t, 289
 alsa_dev_par_parser_t, 290
 device, 290
 nperiods, 290
 stream_dir, 290
alsa_start_counter
 io_alsa_t, 583
alsa_t
 alsa_t< T >, 292
alsa_t< T >, 291
 ~alsa_t, 292
 alsa_t, 292
 buffer, 294
 channels, 294
 fragsize, 294
 frame_data, 294
 gain, 294
 invgain, 294
 pcm_format, 294
 read, 293
 start, 293

stop, 293
 wave, 294
 write, 293
 altconfig.cpp, 1528
 altconfig.hh, 1528
 MHAPLUGIN_OVERLOAD_OUTDOMAIN, amplitude
 1528
 altconfig_t, 295
 altconfig_t, 296
 configs, 299
 event_select_all, 298
 on_set_algos, 297
 on_set_select, 297
 parser_algos, 299
 patchbay, 299
 prepare, 297
 release, 297
 restore_state, 298
 save_state, 298
 select_plug, 299
 selectall, 299
 altplugs.cpp, 1528
 MHAPLUGIN_OVERLOAD_OUTDOMAIN,
 1529
 altplugs_t, 300
 add_plug, 304
 added_via_plugs, 305
 altplugs_t, 301
 cfin, 305
 cfout, 305
 current, 304
 delete_plug, 304
 event_add_plug, 303
 event_delete_plug, 303
 event_select_plug, 303
 event_set_plugs, 302
 fallback_spec, 305
 fallback_wave, 305
 nondefault_labels, 304
 parse, 302
 parser_plugs, 303
 patchbay, 304
 plugs, 304
 prepare, 301
 prepared, 305
 proc_ramp, 303
 process, 302
 ramp_counter, 305
 ramp_len, 305
 ramplen, 304
 release, 301
 select_plug, 304
 selected_plug, 304
 update_ramplen, 303
 update_selector_list, 303
 use_own_ac, 303
 analysepath_t, 306
 ~analysepath_t, 307
 ac_fifo, 308
 analysepath_t, 306
 attr, 309
 cond_to_process, 309
 flag_terminate_inner_thread, 309
 has_inner_error, 309
 inner_ac_copy, 308
 inner_error, 308
 inner_input, 308
 inner_out_domain, 308
 inner_process_wave2spec, 307
 inner_process_wave2wave, 307
 input_to_process, 309
 libdata, 308
 outer_ac, 308
 outer_ac_copy, 308
 priority, 309
 ProcessMutex, 309
 rt_process, 307
 scheduler, 309
 svc, 307
 thread, 309
 wave_fifo, 308
 analysispath.cpp, 1530
 thread_start, 1530
 analysispath_if_t, 310
 ~analysispath_if_t, 311
 acspace_template, 313
 algo, 313
 analysispath_if_t, 311
 chain, 313
 fifolen, 312
 fragsize, 312
 libname, 312
 loadlib, 312
 patchbay, 312

plug, 313
prepare, 311
priority, 312
process, 311
release, 312
vars, 313
analytic
 fshift_hilbert::hilbert_shifter_t, 519
ang_name
 acTransform_wave, 236
 acTransform_wave_config, 239
angle
 Complex arithmetics in the openMHA, 62
angle_ind
 steerbf, 1471
angle_src
 steerbf, 1471
angles
 doasvm_classification, 422
announce_port
 mhaserver_t, 1193
antialias
 ds_t, 444
 us_t, 1486
apply_gains
 MHAoVlFilter::fftfb_t, 1004
 multibandcompressor::plugin_signals_t,
 1303
aquire_mutex
 mha_fifo_posix_threads_t, 774
 mha_fifo_thread_platform_t, 785
arg
 MHA_TCP::Thread, 830
ARG_CC
 ac_mul.hh, 1518
ARG_CR
 ac_mul.hh, 1518
ARG_RC
 ac_mul.hh, 1518
ARG_RR
 ac_mul.hh, 1518
arg_type_t
 ac_mul.hh, 1518
argt
 ac_mul_t, 195
array
 matlab_wrapper::types< MHA_WAVEFORM
 >, 711
array_type
 matlab_wrapper::types< MHA_SPECTRUM
 >, 710
ASSERT_EQUAL_DIM
 mha_signal.cpp, 1625
ASSERT_EQUAL_DIM_PTR
 mha_signal.cpp, 1625
assign
 MHASignal::waveform_t, 1266, 1267
 Vector and matrix processing toolbox, 44,
 45
assign_channel
 MHASignal::waveform_t, 1267
assign_frame
 MHASignal::waveform_t, 1267
async_accept_has_been_triggered
 mha_tcp::server_t, 823
ASYNC_CONNECT_STARTED
 mha_tcp.cpp, 1638
Async_Notify
 MHA_TCP::Async_Notify, 797
async_poll_msg
 fw_t, 527
async_read
 fw_t, 527
async_rmslevel_t
 MHASignal::async_rmslevel_t, 1196
atomic_read_ptr
 mha_fifo_if_t< T >, 769
atomic_write_ptr
 mha_fifo_if_t< T >, 769
attack
 cfg_t, 347
 dc::dc_t, 386
 dc_simple::level_smoothen_t, 407
 softclip_t, 1455
 softclipper_t, 1456
attenuate20.cpp, 1530
attenuate20_t, 314
 attenuate20_t, 314
 prepare, 315
 process, 315
 release, 315
attr
 analysepath_t, 309
 dbasync_native::dbasync_t, 379
 MHAPlugin_Split::posix_threads_t, 1170
audiometer_if_t
 audiometerbackend::audiometer_if_t, 316
 audiometerbackend, 82
 gcd, 83
 generator, 83
 level_adaptor, 83
 return_sig, 83

audiometerbackend.cpp, 1531
 DEBUG, 1531
 audiometerbackend::audiometer_if_t, 315
 audiometer_if_t, 316
 change_mode, 317
 freq, 317
 level, 317
 mode, 317
 outchannel, 318
 patchbay, 318
 pmode, 318
 prepare, 317
 process, 316
 ramplen, 318
 set_level, 317
 sigtype, 317
 update, 317
 audiometerbackend::level_adapt_t, 318
 can_update, 319
 get_level, 319
 ilen, 320
 l_new, 320
 l_old, 320
 level_adapt_t, 319
 pos, 320
 update_frame, 319
 wnd, 320
 audiometerbackend::Inn3rdoct_t, 321
 _fmax, 322
 _fmin, 322
 bandpass, 322
 dev, 322
 iterate_Inn, 322
 Inn3rdoct_t, 321
 random, 322
 rng, 322
 audiometerbackend::signal_gen_t, 323
 signal_gen_t, 323
 audiometerbackend::sine_t, 324
 sine_t, 324
 auditory_profile.cpp, 1532
 auditory_profile.h, 1532
 AuditoryProfile, 84
 AuditoryProfile::fmap_t, 325
 get_frequencies, 325
 get_values, 325
 isempty, 326
 AuditoryProfile::parser_t, 326
 get_current_profile, 327
 L, 327
 parser_t, 327
 R, 327
 AuditoryProfile::parser_t::ear_t, 328
 ear_t, 328
 get_ear, 329
 HTL, 329
 UCL, 329
 AuditoryProfile::parser_t::fmap_t, 329
 f, 331
 fmap_t, 330
 get_fmap, 330
 name_, 331
 patchbay, 331
 validate, 330
 value, 331
 AuditoryProfile::profile_t, 331
 get_ear, 332
 L, 332
 R, 332
 AuditoryProfile::profile_t::ear_t, 333
 convert_empty2normal, 333
 HTL, 333
 UCL, 334
 available_streams
 Isl2ac::Isl2ac_t, 685
 average
 coherence::vars_t, 355
 avg_ipd
 coherence::cohflt_t, 352
 azimuth
 mha_direction_t, 754
B
 MHAFilter::filter_t, 890
 MHAFilter::iir_filter_t, 899
b
 doasvm_classification, 422
 MHAParser::base_t::replace_t, 1040
B_
 MHAFilter::complex_bandpass_t, 875
 MHAFilter::iir_ord1_real_t, 903
b_check_version
 PluginLoader::mhaplugloader_t, 1373
b_complex
 acsave::save_var_t, 228
b_cont
 acmon::acmon_t, 210
b_est
 lpc_bl_predictor_config, 668
b_exit_request
 fw_t, 530
b_flushed
 acsave::acsave_t, 222

b_fw_started
 io_parser_t, 619

b_interactive
 mhaserver_t, 1193

b_is_input
 calibrator_runtime_layer_t, 338
 calibrator_t, 341

b_is_prepared
 PluginLoader::mhaplugloader_t, 1373

b_loop
 MHASignal::loop_wavefragment_t, 1221

b_ltg
 coherence::cohfilt_t, 353

b_permute
 ac_proc::interface_t, 198

b_prepared
 acsave::acsave_t, 222
 io_file_t, 609
 io_parser_t, 619
 mhachain::chain_base_t, 844
 mhachain::plugs_t, 848
 MHAJack::client_t, 979

b_process
 io_alsa_t, 581

b_ready
 MHAJack::client_avg_t, 967

b_record
 ac2osc_t, 184

b_snapshot
 acmon::acmon_t, 210

b_starting
 io_parser_t, 619

b_stopped
 io_parser_t, 619
 MHAJack::client_avg_t, 966
 MHAJack::client_noncont_t, 969

b_use_clipping
 calibrator_runtime_layer_t, 338

b_use_fir
 calibrator_runtime_layer_t, 338

b_use_profiling
 mhachain::plugs_t, 850

backward
 lpc_bl_predictor_config, 668
 lpc_burglattice_config, 674
 MHASignal::fft_t, 1209

backward_scale
 MHASignal::fft_t, 1210

band_weights
 dc::dc_vars_t, 391

bandpass

audiometerbackend::Inn3rdoct_t, 322

bands
 gtfb_analyzer::gtfb_analyzer_cfg_t, 550
 gtfb_simd_cfg_t, 558
 MHAOvlFilter::fspacing_t, 1017

bandsXchannels
 gtfb_simd_cfg_t, 558

bandw_correction
 speechnoise.cpp, 1700

bark2hz_t
 MHAOvlFilter::barkscale::bark2hz_t, 999

BARKSCALE_ENTRIES
 mha_fftfb.cpp, 1590

bartlett
 MHAWindow, 154

bartlett_t
 MHAWindow::bartlett_t, 1287

base_t
 MHAMultiSrc::base_t, 992
 MHAParser::base_t, 1030
 MHAWindow::base_t, 1288

basename
 save_spec_t, 1418
 save_wave_t, 1420
 shadowfilter_begin::shadowfilter_begin_t,
 1423
 shadowfilter_end::shadowfilter_end_t,
 1428

bass
 plingploing::plingploing_t, 1343

bassmod
 plingploing::if_t, 1340

bassmod_
 plingploing::plingploing_t, 1345

bassperiod
 plingploing::if_t, 1341

bassperiod_
 plingploing::plingploing_t, 1345

bbcilib_interface_t, 334
 ~bbcilib_interface_t, 335

bbcilib_interface_t, 335
 calib_in, 336
 calib_out, 336
 plugloader, 336
 prepare, 335
 process, 335
 release, 335

beam1
 rohBeam::rohConfig, 1405

beamA
 rohBeam::rohConfig, 1406

beamExport
 rohBeam::rohBeam, 1401

beamW
 rohBeam::rohConfig, 1405

beta
 ADM::ADM< F >, 263
 adm_if_t, 274

beta_const
 smooth_cepstrum::smooth_cepstrum_if_t, 1436
 smooth_cepstrum::smooth_params, 1447

bf_src
 steerbf, 1471

bf_src_copy
 steerbf_config, 1473

bf_vec
 steerbf_config, 1473

bflush
 acsave::acsave_t, 221

bin1
 MHAOvlFilter::fftfb_t, 1004

bin2
 MHAOvlFilter::fftfb_t, 1005

bin2freq
 Vector and matrix processing toolbox, 39

binaural_type
 rohBeam::rohBeam, 1400

binaural_type_index
 rohBeam::configOptions, 1393
 rohBeam::rohConfig, 1405

blinvert
 coherence::cohflt_t, 353

blackman
 MHAWindow, 155

blackman_t
 MHAWindow::blackman_t, 1290

blockprocessing_polyphase_resampling_t
 MHAFilter::blockprocessing_polyphase_resampling_t, 868

blocks
 droptect_t, 441

blockSpec
 rohBeam::rohConfig, 1406

blockXp
 rohBeam::rohConfig, 1407

bookkeeping
 MHAFilter::partitioned_convolution_t, 918
 MHAParser::mhapluginloader_t, 1089

bool_mon_t
 MHAParser::bool_mon_t, 1041

bool_t
 MHAParser::bool_t, 1043

bpm
 plingploing::if_t, 1340

bprofiling
 mhachain::chain_base_t, 843

bracket_balance
 MHAParser::StrCnv, 130

brown
 speechnoise_t, 1467

browsemhaplugins.cpp, 1532
 DEBUG, 1532
 main, 1533

bt
 plingploing::plingploing_t, 1343

buf
 ac2lsl::save_var_t< mha_complex_t >, 178
 ac2lsl::save_var_t< T >, 175
 lsl2ac::save_var_t, 689
 mha_fifo_t< T >, 781
 mha_spec_t, 794
 mha_wave_t, 840

buf_c_in
 MHASignal::hilbert_fftw_t, 1213

buf_c_out
 MHASignal::hilbert_fftw_t, 1213

buf_in
 MHASignal::fft_t, 1211

buf_out
 MHASignal::fft_t, 1211

buf_r_in
 MHASignal::hilbert_fftw_t, 1213

buf_r_out
 MHASignal::hilbert_fftw_t, 1213

buffer
 alsa_t< T >, 294
 MHASignal::delay_spec_t, 1199
 MHASignal::delay_t, 1201
 MHASignal::delay_wave_t, 1203

buffered_incoming_bytes
 MHA_TCP::Connection, 809

buffered_outgoing_bytes
 MHA_TCP::Connection, 809

buffersize
 lsl2ac::lsl2ac_t, 684

bufSize
 gsc_adaptive_stage::gsc_adaptive_stage, 539

bufsize
 lsl2ac::save_var_t, 692

burn

DynComp::dc_afterburn_rt_t, 450
DynComp::dc_afterburn_t, 453
multibandcompressor::interface_t, 1302
butter_stop_ord1
 MHAFilter, 107
bw
 MHAOvIFilter::fscale_bw_t, 1012
bw_
 MHAFilter::gamma_flt_t, 895
bw_generator
 MHAFilter::thirdoctave_analyzer_t, 934
bw_hz
 MHAOvIFilter::fscale_bw_t, 1012
bw_name
 dc::dc_vars_t, 390
bwv
 MHAOvIFilter::fftfb_ac_info_t, 1002
 multibandcompressor::fftfb_plug_t, 1299
bypass
 adm_if_t, 274
 db_if_t, 372
 dc::dc_t, 387
 dc::dc_vars_t, 390
 dc_simple::dc_vars_t, 404
 DynComp::dc_afterburn_vars_t, 457

C

ADM, 82

c

 acPooling_wave_config, 217
 adaptive_feedback_canceller, 242
 doasvm_classification_config, 425
 io_tcp_sound_t::float_union, 637
 mha_complex_test_array_t, 746
 nlms_t, 1307

c1_a
 MHAFilter::o1_ar_filter_t, 907

c1_r
 MHAFilter::o1_ar_filter_t, 907

c2_a
 MHAFilter::o1_ar_filter_t, 907

c2_r
 MHAFilter::o1_ar_filter_t, 907

c_ifc_parser_t
 MHAParser::c_ifc_parser_t, 1046

c_min
 coherence::cohflt_t, 352

c_parse_cmd
 MHAParser::c_ifc_parser_t, 1047

c_parse_cmd_t
 MHAParser, 126

c_parse_err
 MHAParser::c_ifc_parser_t, 1047
c_parse_err_t
 MHAParser, 128

c_scale
 coherence::cohflt_t, 352

calc_in
 wave2spec_t, 1496

calc_out
 overlapadd::overlapadd_t, 1334
 spec2wave_t, 1464

calc_pre_wnd
 wave2spec_t, 1495

calc_sine
 cpupload::cpupload_cfg_t, 366

calib_in
 bbcalib_interface_t, 336

calib_out
 bbcalib_interface_t, 336

calibrator_runtime_layer_t, 336
 b_is_input, 338
 b_use_clipping, 338
 b_use_fir, 338
 calibrator_runtime_layer_t, 337
 fir, 338
 firfir2ffflen, 337
 firflen, 337
 gain, 338
 pmode, 338
 process, 337
 quant, 338
 softclip, 338
 speechnoise, 338

calibrator_t, 339
 b_is_input, 341
 calibrator_t, 340
 patchbay, 341
 prepare, 340
 prepared, 341
 process, 340
 read_levels, 341
 release, 340
 update, 340
 update_tau_level, 341
 vars, 341

calibrator_variables_t, 341
 calibrator_variables_t, 342
 config_parser, 344
 do_clipping, 344
 fir, 342
 fragsize, 343
 nbits, 342

num_channels, 343
 peaklevel, 342
 rmslevel, 343
 softclip, 344
 spnoise_channels, 343
 spnoise_level, 343
 spnoise_mode, 343
 spnoise_parser, 343
 srate, 343
 tau_level, 343
callback
 matlab_wrapper::callback, 693
 matlab_wrapper::matlab_wrapper_t, 701
callbacks
 matlab_wrapper::matlab_wrapper_t, 701
can_read
 MHAFilter::blockprocessing_polyphase_resampling_t,
 870
can_read_bytes
 MHA_TCP::Connection, 807
can_read_line
 MHA_TCP::Connection, 806
can_sysread
 MHA_TCP::Connection, 805
can_syswrite
 MHA_TCP::Connection, 805
can_update
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 319
 fader_wave::level_adapt_t, 492
catch_condition
 MHAParser::mhaplugloader_t, 1089
catch_thread
 MHAParser::mhaplugloader_t, 1372
 MHAPlugin_Split::dummy_threads_t,
 1166
 MHAParser::mhaplugloader_t, 1168
 MHAPlugin_Split::thread_platform_t,
 1186
categories
 plugindescription_t, 1356
cb_patchbay
 matlab_wrapper::matlab_wrapper_t, 701
cdata
 mha_audio_t, 743
 MHASignal::matrix_t, 1232
center_frequencies
 dc::dc_vars_t, 391
 dc_simple::dc_if_t, 396
cf
 mha_audio_descriptor_t, 741
 MHAFilter::thirdoctave_analyzer_t, 934
 MHAOvlFilter::band_descriptor_t, 998
 MHAOvlFilter::ffftb_vars_t, 1009
 plingploing::plingploing_t, 1343
cf2bands
 MHAOvlFilter::fspacing_t, 1016
cf_
 MHAFilter::gamma_flt_t, 895
 wavwriter_t, 1502
cf_generator
 MHAFilter::thirdoctave_analyzer_t, 934
cf_h
 MHAOvlFilter::band_descriptor_t, 998
cf_in
 overlapadd::overlapadd_if_t, 1331
 smoothgains_bridge::overlapadd_if_t,
 1451
cf_m
 MHAOvlFilter::band_descriptor_t, 998
cf_name
 dc::dc_vars_t, 390
cf_out
 overlapadd::overlapadd_if_t, 1331
 smoothgains_bridge::overlapadd_if_t,
 1451
cf_out_
 MHAParser::mhaplugloader_t, 1089
cf_output
 PluginLoader::mhaplugloader_t, 1373
cfac
 route::interface_t, 1410
cfg
 MHAParser::config_t< runtime_cfg_t >,
 1145
cfg_
 MHAFilter::thirdoctave_analyzer_t, 934
cfg_dump
 MHAParser, 126
cfg_dump_short
 MHAParser, 126
cfg_node_current
 MHAParser::config_t< runtime_cfg_t >,
 1146
cfg_node_t
 MHAParser::cfg_node_t< runtime_cfg_t >,
 1139
cfg_root
 MHAParser::config_t< runtime_cfg_t >,

1146
cfg_t, 344
 ac2lsl::cfg_t, 167
 acsave::cfg_t, 223
 alpha, 347
 attack, 347
 cfg_t, 345
 channel, 346
 decay, 347
 equalize::cfg_t, 462
 frozen_noise_, 347
 gain_spec_, 346
 gain_wave_, 346
 lsl2ac::cfg_t, 679
 matrixmixer::cfg_t, 712
 pos, 347
 process, 346
 rand_dist, 347
 replace_, 346
 rng, 347
 scale, 346
 shadowfilter_begin::cfg_t, 1421
 shadowfilter_end::cfg_t, 1424
 start_lin, 347
 use_frozen_, 346
 windnoise::cfg_t, 1504
cfin
 altplugs_t, 305
 fw_t, 529
 mhachain::chain_base_t, 844
 route::interface_t, 1410
cfout
 altplugs_t, 305
 fw_t, 530
 mhachain::chain_base_t, 844
 route::interface_t, 1410
cfv
 MHAOvlFilter::fftfb_ac_info_t, 1001
 multibandcompressor::fftfb_plug_t, 1299
cg
 coherence::cohflt_t, 352
ch
 MHASignal::doublebuffer_t, 1207
chain
 acmon::acmon_t, 210
 acsave::acsave_t, 222
 analysispath_if_t, 313
 db_if_t, 372
 dbasync_native::db_if_t, 377
 mhachain::chain_base_t, 844
 mhachain::plugs_t, 848
 MHAPlugin_Resampling::resampling_if_t, 1155
 chain_base_t
 mhachain::chain_base_t, 842
chains
 MHAPlugin_Split::split_t, 1177
chance
 dropgen_t, 438
change_mode
 addsndfile::addsndfile_if_t, 252
 audiometerbackend::audiometer_if_t, 317
channel
 cfg_t, 346
 example5_t, 481
 MHAMultiSrc::channel_t, 993
channel_gain_name
 combc_if_t, 358
channel_gains_
 combc_t, 360
channel_info
 mha_spec_t, 794
 mha_wave_t, 840
channel_no
 example6_t, 483
channel_pair
 level_matching::channel_pair, 647, 648
channelconfig_out_
 MHAOvlFilter::overlap_save_filterbank_t, 1021
channels
 ac2wave_t, 189
 adaptive_feedback_canceller_config, 246
 addsndfile::addsndfile_if_t, 253
 alsa_t< T >, 294
 fftfilter::fftfilter_t, 499
 gtfb_analyzer::gtfb_analyzer_cfg_t, 550
 gtfb_simd_cfg_t, 558
 level_matching::level_matching_t, 656
 mhaconfig_t, 851
 MHAFilter::fftfilter_t, 880
 MHAFilter::filter_t, 891
 MHAParser::mhaconfig_mon_t, 1084
 MHAPlugin_Split::split_t, 1176
 MHASignal::delay_t, 1201
 rt_nlms_t, 1415
 sine_cfg_t, 1429
 sine_t, 1432
 testplugin::config_parser_t, 1478
 Vector and matrix processing toolbox, 38
channels_t
 MHAMultiSrc::channels_t, 993

char_data
 testplugin::ac_parser_t, 1476

chdir
 mha_audio_descriptor_t, 742

check_alignment
 gtfb_simd.cpp, 1557

CHECK_EXPR
 mha_defs.h, 1583

check_index
 adm_rtconfig_t, 278

check_low
 MHParse::range_var_t, 1107

check_parenthesis_complex
 mha_parser.cpp, 1609

check_range
 MHParse::range_var_t, 1107

check_sign_complex
 mha_parser.cpp, 1609

check_sound_data_type
 io_tcp_sound_t, 633

check_up
 MHParse::range_var_t, 1107

CHECK_VAR
 mha_defs.h, 1583

check_vars
 ac2isl::cfg_t, 167

chnname
 dc::dc_vars_t, 390

chunkbytes_in
 io_asterisk_sound_t, 597
 io_tcp_sound_t, 634

chunksize
 isl2ac::isl2ac_t, 684
 isl2ac::save_var_t, 691

ci
 matrixmixer::matmix_t, 715

class_signal_type
 matlab_wrapper::types< MHA_SPECTRUMcmd_release
 >, 711

matlab_wrapper::types< MHA_WAVEFORMcmd_start
 >, 711

cleanup_plugins
 mhachain::plugins_t, 848

cleanup_unused_cfg
 MHParse::config_t< runtime_cfg_t >, 1145

clear
 mha_fifo_t< T >, 780
 MHATableLookup::linear_table_t, 1278
 MHATableLookup::table_t, 1280
 MHATableLookup::xy_table_t, 1284

Vector and matrix processing toolbox, 44

clear_chains
 MHAParse::split_t, 1174

clear_plugins
 pluginbrowser_t, 1354

Client
 MHA_TCP::Client, 801

client_avg_t
 MHAJack::client_avg_t, 964

client_noncont_t
 MHAJack::client_noncont_t, 968

client_t
 MHAJack::client_t, 972

clientid
 dc::dc_vars_t, 391
 dc_simple::dc_if_t, 396

clientname
 MHAIOJack::io_jack_t, 943
 MHAIOJackdb::io_jack_t, 951

clipmeter
 softclipper_t, 1456

clipped
 softclipper_variables_t, 1459

close_session
 plugins::hoertech::acrec::acwriter_t, 1383
 wavwriter_t, 1502

closed
 MHA_TCP::Connection, 810

closesocket
 mha_tcp.cpp, 1638

cLTASS
 gtfb_simple_rt_t, 568
 gtfb_simple_t, 573
 MHAOvlFilter::fftfb_ac_info_t, 1002
 MHAOvlFilter::fftfb_vars_t, 1010

cmd_prepare
 MHAIOPortAudio::io_portaudio_t, 959

matlab_wrapper::types< MHA_WAVEFORMcmd_start
 >, 711

MHAIOPortAudio::io_portaudio_t, 960

MHAIOPortAudio::io_portaudio_t, 959

cmd_stop
 MHAIOPortAudio::io_portaudio_t, 959

co
 matrixmixer::matmix_t, 715

coeff
 gtfb_analyzer::gtfb_analyzer_cfg_t, 551
 gtfb_analyzer::gtfb_analyzer_t, 555
 gtfb_simd_t, 563

coeff_decomb
 adm_if_t, 274

coeff_lp
 adm_if_t, 274

coh_c
 coherence::cohflt_t, 353

coh_rlp
 coherence::cohflt_t, 353

coherence, 84
 getcipd, 84

coherence.cpp, 1533

coherence::cohflt_if_t, 348
 algo, 350
 cohflt_if_t, 349
 patchbay, 349
 prepare, 349
 process, 349
 release, 349
 update, 349
 vars, 350

coherence::cohflt_t, 350
 alpha, 352
 avg_ipd, 352
 b_ltg, 353
 blinvert, 353
 c_min, 352
 c_scale, 352
 cg, 352
 coh_c, 353
 coh_rlp, 353
 cohflt_t, 351
 g, 352
 gain, 353
 gain_delay, 353
 insert, 351
 limit, 352
 lp1i, 353
 lp1ltg, 353
 lp1r, 352
 nbands, 352
 process, 351
 s_out, 353
 staticgain, 354

coherence::vars_t, 354
 alpha, 355
 average, 355
 delay, 356
 invert, 355
 limit, 355
 ltgcomp, 356
 ltgtau, 356
 mapping, 355
 staticgain, 356

tau, 355
tau_unit, 355
vars_t, 355

cohflt_if_t
 coherence::cohflt_if_t, 349

cohflt_t
 coherence::cohflt_t, 351

collect_result
 MHAPlugin_Split::split_t, 1175
 MHAPlugin_Split::splitted_part_t, 1182

colored_intensity
 Vector and matrix processing toolbox, 54

cols
 acsave::mat4head_t, 225

combc_if_t, 356
 channel_gain_name, 358
 combc_if_t, 357
 element_gain_name, 358
 interleaved, 358
 outchannels, 358
 prepare, 357
 process, 357

combc_t, 358
 ac_, 359
 channel_gains_, 360
 combc_t, 359
 element_gain_name_, 360
 interleaved_, 360
 nbands, 360
 process, 359
 s_out, 360
 w_out, 360

combinechannels.cpp, 1533

comm_var_t, 360
 data, 362
 data_type, 361
 num_entries, 361
 stride, 362

commentate
 MHAParser, 126

commit
 DynComp::dc_afterburn_vars_t, 457

commit_pending
 DynComp::dc_afterburn_t, 454

commit_t
 MHAParser::commit_t< receiver_t >, 1049

Communication between algorithms, 23
 get_var_float, 26
 get_var_int, 26
 get_var_spectrum, 25

get_var_vfloat, 27
 get_var_waveform, 25
 comp_each_iter
 lpc, 662
 lpc_config, 677
 comp_iter
 lpc_config, 677
 compensation
 windnoise::cfg_t, 1506
 COMPILER_ID
 compiler_id.hh, 1535
 compiler_id.cpp, 1534
 compiler_id.hh, 1534
 COMPILER_ID, 1535
 COMPILER_ID_MAJOR, 1534
 COMPILER_ID_MINOR, 1534
 COMPILER_ID_PATCH, 1534
 COMPILER_ID_VENDOR, 1534
 COMPILER_ID_VERSION, 1535
 COMPILER_ID_VERSION_HELPER1, 1535
 COMPILER_ID_VERSION_HELPER2, 1534
 COMPILER_ID_MAJOR
 compiler_id.hh, 1534
 COMPILER_ID_MINOR
 compiler_id.hh, 1534
 COMPILER_ID_PATCH
 compiler_id.hh, 1534
 COMPILER_ID_VENDOR
 compiler_id.hh, 1534
 COMPILER_ID_VERSION
 compiler_id.hh, 1535
 COMPILER_ID_VERSION_HELPER1
 compiler_id.hh, 1535
 COMPILER_ID_VERSION_HELPER2
 compiler_id.hh, 1534
 Complex arithmetics in the openMHA, 58
 _conjugate, 67
 _reciprocal, 67
 abs, 65
 abs2, 65
 almost, 68
 angle, 62
 conjugate, 67
 expi, 61, 64
 mha_complex, 60
 normalize, 68
 operator!=, 67
 operator<, 69
 operator*, 64, 65
 operator*=, 64
 operator+, 62, 63
 operator+=, 62
 operator-, 63, 66
 operator-=, 63
 operator/, 65, 66
 operator/=, 65, 66
 operator==, 66
 reciprocal, 67
 safe_div, 66
 set, 60, 61
 stdcomplex, 61
 complex_bandpass_t
 MHAFilter::complex_bandpass_t, 872
 complex_data
 testplugin::ac_parser_t, 1476
 complex_filter.cpp, 1535
 complex_filter.h, 1535
 complex_mon_t
 MHAParser::complex_mon_t, 1051
 complex_ofs
 MHASignal::matrix_t, 1232
 complex_scale_channel.cpp, 1536
 complex_scale_channel_t, 362
 complex_scale_channel_t, 363
 factor, 364
 patchbay, 364
 prepare, 363
 process, 363
 scale_ch, 364
 update_cfg, 364
 complex_t
 MHAParser::complex_t, 1053
 compression
 dc_simple::dc_t, 399
 compute_beamW
 rohBeam::rohBeam, 1398
 compute_delaycomp_vec
 rohBeam::rohBeam, 1397
 compute_diff2D
 rohBeam::rohBeam, 1398
 compute_diff3D
 rohBeam::rohBeam, 1398
 compute_head_model_alpha
 rohBeam::rohBeam, 1397
 compute_head_model_mat
 rohBeam::rohBeam, 1397
 compute_head_model_T
 rohBeam::rohBeam, 1397
 compute_uncorr
 rohBeam::rohBeam, 1397

compute_wng
 rohBeam::rohBeam, 1398

Concept of Variables and Data Exchange in the openMHA, 4

cond_to_process
 analysepath_t, 309

config_file_splitter_t
 PluginLoader::config_file_splitter_t, 1360

config_in
 testplugin::if_t, 1482

config_out
 testplugin::if_t, 1482

config_parser
 calibrator_variables_t, 344

config_parser_t
 testplugin::config_parser_t, 1478

config_t
 MHAPlugin::config_t< runtime_cfg_t >, 1143

configfile
 PluginLoader::config_file_splitter_t, 1361

configname
 PluginLoader::config_file_splitter_t, 1361

configs
 altconfig_t, 299

configuration, 4

configuration variable, 4

conflux
 DynComp::dc_afterburn_rt_t, 451
 DynComp::dc_afterburn_vars_t, 456

conjugate
 Complex arithmetics in the openMHA, 67
 Vector and matrix processing toolbox, 57

connect
 MHAEvents::emitter_t, 859
 MHAEvents::patchbay_t< receiver_t >, 861, 862

connect_input
 MHAJack::client_t, 974

connect_output
 MHAJack::client_t, 974

connect_to
 MHAJack::port_t, 983

connected
 io_asterisk_parser_t, 594
 io_tcp_parser_t, 630

Connection
 MHA_TCP::Connection, 804

connection_loop
 io_asterisk_t, 602
 io_tcp_t, 639

connections
 mha_tcp::server_t, 823
 MHAEvents::emitter_t, 860

connections_in
 MHAIOJack::io_jack_t, 943
 MHAIOJackdb::io_jack_t, 951

connections_out
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 951

connector
 MHAFilter::adapt_filter_t, 867
 MHAFilter::iir_filter_t, 900
 MHAParser::mhapluginloader_t, 1089

connector_base_t
 MHAEvents::connector_base_t, 853

connector_t
 MHAEvents::connector_t< receiver_t >, 856

cons
 MHAEvents::patchbay_t< receiver_t >, 862

consecutive_dropouts
 droptect_t, 441

CONST_C
 rohBeam, 160

contained_frames
 MHASignal::ringbuffer_t, 1238

conv2lateX
 generatemhaplugindoc.cpp, 1551

convert_empty2normal
 AuditoryProfile::profile_t::ear_t, 333

convert_f2logf
 gaintable.cpp, 1549

copy
 MHASignal::spectrum_t, 1247
 MHASignal::waveform_t, 1267, 1268

copy_channel
 MHASignal::spectrum_t, 1248
 MHASignal::waveform_t, 1268
 Vector and matrix processing toolbox, 52, 53

copy_error
 MHAIOAsterisk.cpp, 1653
 MHAIOTCP.cpp, 1679

copy_from_at
 MHASignal::waveform_t, 1268

copy_output_spec
 MHAPlugin_Split::split_t, 1175

copy_output_wave
 MHAPlugin_Split::split_t, 1174

copy_permuted

MHASignal, 149
 copyfixedbfoutput
 rohBeam::rohConfig, 1404
 corr_out
 lpc_config, 678
 corrLL
 rohBeam::rohConfig, 1407
 corrRR
 rohBeam::rohConfig, 1407
 corrXpXp
 rohBeam::rohConfig, 1406
 corrXpYf
 rohBeam::rohConfig, 1406
 corrZZ
 rohBeam::rohConfig, 1406
 cpupload, 85
 cpupload.cpp, 1536
 cpupload::cpupload_cfg_t, 364
 calc_sine, 366
 cpupload_cfg_t, 365
 factor, 366
 phase, 366
 process, 365
 result, 366
 table, 366
 use_sine, 366
 write_to_table, 366
 cpupload::cpupload_if_t, 367
 cpupload_if_t, 368
 factor, 369
 patchbay, 369
 prepare, 368
 process, 368
 table_size, 369
 update, 368
 use_sine, 369
 cpupload_cfg_t
 cpupload::cpupload_cfg_t, 365
 cpupload_if_t
 cpupload::cpupload_if_t, 368
 create_datafile
 plugins::hoertech::acrec::acwriter_t, 1382
 create_latex_doc
 generatemhaplugindoc.cpp, 1552
 create_lock
 mhamain.cpp, 1685
 create_or_replace_var
 ac2lsl::cfg_t, 167
 create_soundfile
 wavwriter_t, 1501
 creator
 speechnoise_t, 1468
 creator_A
 MHAFilter::complex_bandpass_t, 872
 creator_B
 MHAFilter::complex_bandpass_t, 873
 cstr_strerror
 mha_errno.c, 1585
 current
 altpugs_t, 304
 mha_rt_fifo_t< T >, 792
 current_input_signal_buffer_half_index
 MHAFilter::partitioned_convolution_t, 917
 current_message
 mha_tcp::buffered_socket_t, 800
 current_output_partition_index
 MHAFilter::partitioned_convolution_t, 918
 current_powspec
 droptect_t, 442
 current_thread_priority
 MHAPlugin_Split::posix_threads_t, 1169
 current_thread_scheduler
 MHAPlugin_Split::posix_threads_t, 1169
 cv
 lsl2ac::save_var_t, 690
 plugins::hoertech::acrec::acrec_t, 1379
 cvalue
 gtfb_analyzer::gtfb_analyzer_cfg_t, 551

 d
 gsc_adaptive_stage::gsc_adaptive_stage,
 542
 data
 acsave::save_var_t, 227
 comm_var_t, 362
 DynComp::gaintable_t, 461
 MHA_AC::acspace2matrix_t, 728
 MHA_AC::double_t, 729
 MHA_AC::float_t, 731
 MHA_AC::int_t, 733
 mha_rt_fifo_element_t< T >, 789
 MHAParser::bool_mon_t, 1042
 MHAParser::bool_t, 1044
 MHAParser::complex_mon_t, 1051
 MHAParser::complex_t, 1054
 MHAParser::float_mon_t, 1058
 MHAParser::float_t, 1061
 MHAParser::int_mon_t, 1063
 MHAParser::int_t, 1066
 MHAParser::kw_t, 1073
 MHAParser::mcomplex_mon_t, 1075
 MHAParser::mcomplex_t, 1078
 MHAParser::mfloat_mon_t, 1080

MHAParser::mfloat_t, 1082
MHAParser::mint_mon_t, 1092
MHAParser::mint_t, 1094
MHAParser::string_mon_t, 1110
MHAParser::string_t, 1112
MHAParser::vcomplex_mon_t, 1117
MHAParser::vcomplex_t, 1119
MHAParser::vfloat_mon_t, 1121
MHAParser::vfloat_t, 1124
MHAParser::vint_mon_t, 1126
MHAParser::vint_t, 1129
MHAParser::vstring_mon_t, 1131
MHAParser::vstring_t, 1133
MHAParser::cfg_node_t< runtime_cfg_t >, 1140
MHASignal::uint_vector_t, 1258
wavwriter_t, 1503
data_is_initialized
 MHAParser::base_t, 1038
data_type
 ac2lsl::save_var_base_t, 171
 ac2lsl::save_var_t< mha_complex_t >, 178
 ac2lsl::save_var_t< T >, 174
 comm_var_t, 361
 testplugin::ac_parser_t, 1476
data_type_
 ac2lsl::save_var_t< T >, 175
data_type_t
 testplugin::ac_parser_t, 1475
db.cpp, 1536
db2lin
 MHASignal, 140
db2sq
 MHASignal, 141
db_if_t, 370
 ~db_if_t, 371
 algo, 372
 bypass, 372
 chain, 372
 db_if_t, 371
 dbasync_native::db_if_t, 375
 fragsize, 372
 patchbay, 371
 plugloader, 372
 prepare, 371
 process, 371
 release, 371
db_t, 372
 db_t, 373
 inner_process, 373
 plugloader, 373
 dbasync.cpp, 1536
 dbasync_native, 85
 gcd, 86
 INVALID_THREAD_PRIORITY, 86
 thread_start, 86
 dbasync_native::db_if_t, 374
 ~db_if_t, 375
 algo, 377
 chain, 377
 db_if_t, 375
 delay, 376
 fragsize, 376
 framework_thread_priority, 377
 framework_thread_scheduler, 377
 plugloader, 376
 prepare, 375
 process, 375
 release, 376
 sub_ac, 376
 worker_thread_priority, 376
 worker_thread_scheduler, 376
 dbasync_native::dbasync_t, 377
 ~dbasync_t, 378
 attr, 379
 dbasync_t, 378
 inner_input, 379
 outer_output, 379
 outer_process, 378
 plugloader, 379
 priority, 379
 scheduler, 379
 svc, 379
 thread, 379
 dbasync_native::delay_check_t, 380
 delay_check_t, 380
 dbasync_t
 dbasync_native::dbasync_t, 378
DBG
 MHAIOalsa.cpp, 1645
dbspl2pa
 MHASignal, 142
dbspl2pa2
 MHASignal, 143
DC
 dc_simple, 87
dc, 86
dc.cpp, 1537
 DUPVEC, 1537
 get_audiochannels, 1537
dc.hh, 1538

dc::dc_if_t, 381
 algo, 383
 dc_if_t, 382
 patchbay, 383
 prepare, 382
 process, 382
 update, 382
 update_monitors, 382
 dc::dc_t, 383
 attack, 386
 bypass, 387
 dc_t, 384
 decay, 387
 explicit_insert, 385
 ffflen, 387
 get_attack_filter_state, 386
 get_decay_filter_state, 386
 get_level_in_db, 385
 get_level_in_db_adjusted, 386
 get_nbands, 385
 get_nch, 385
 get_rmslevel_filter_state, 386
 gt, 386
 level_in_db, 387
 level_in_db_adjusted, 387
 log_interp, 387
 naudiochannels, 387
 nbands, 387
 nch, 387
 offset, 386
 process, 385
 rmslevel, 386
 dc::dc_vars_t, 388
 band_weights, 391
 bw_name, 390
 bypass, 390
 center_frequencies, 391
 cf_name, 390
 chname, 390
 clientid, 391
 dc_vars_t, 389
 edge_frequencies, 391
 ef_name, 390
 filterbank, 390
 filtered_level, 391
 gainrule, 391
 gtdata, 389
 gtmin, 389
 gtstep, 389
 input_level, 391
 log_interp, 390
 modified, 391
 offset, 390
 preset, 391
 tauattack, 389
 taudecay, 390
 taurmslevel, 389
 dc::dc_vars_validator_t, 392
 dc_vars_validator_t, 392
 dc_afterburn.cpp, 1538
 mylogf, 1538
 dc_afterburn.h, 1538
 dc_afterburn_rt_t
 DynComp::dc_afterburn_rt_t, 450
 dc_afterburn_t
 DynComp::dc_afterburn_t, 453
 dc_afterburn_vars_t
 DynComp::dc_afterburn_vars_t, 456
 dc_if_t
 dc::dc_if_t, 382
 dc_simple::dc_if_t, 394
 dc_simple, 86
 DC, 87
 force_resize, 88
 LEVEL, 87
 not_zero, 88
 test_fail, 87
 dc_simple.cpp, 1539
 dc_simple.hh, 1539
 dc_simple::dc_if_t, 393
 center_frequencies, 396
 clientid, 396
 dc_if_t, 394
 edge_frequencies, 396
 filterbank, 396
 gainrule, 396
 has Been_modified, 395
 modified, 396
 mon_g, 396
 mon_l, 396
 patchbay, 397
 prepare, 394
 prepared, 397
 preset, 396
 process, 395
 read_modified, 395
 release, 394
 update_dc, 395
 update_gain_mon, 395
 update_level, 395
 update_level_mon, 395
 dc_simple::dc_t, 397

compression, 399
dc_t, 398
expansion, 399
expansion_threshold, 399
limiter, 399
limiter_threshold, 399
maxgain, 400
mon_g, 400
mon_l, 400
nbands, 400
process, 399
dc_simple::dc_t::line_t, 400
line_t, 400, 401
m, 401
operator(), 401
y0, 401
dc_simple::dc_vars_t, 402
bypass, 404
dc_vars_t, 402
expansion_slope, 403
expansion_threshold, 403
g50, 403
g80, 403
limiter_threshold, 403
maxgain, 403
tauattack, 403
taudecay, 403
dc_simple::dc_vars_validator_t, 404
dc_vars_validator_t, 405
dc_simple::level_smoothen_t, 406
attack, 407
decay, 408
ffflen, 408
level_smoothen_t, 407
level_spec, 408
level_wave, 408
nbands, 408
process, 407
dc_t
dc::dc_t, 384
dc_simple::dc_t, 398
dc_vars_t
dc::dc_vars_t, 389
dc_simple::dc_vars_t, 402
dc_vars_validator_t
dc::dc_vars_validator_t, 392
dc_simple::dc_vars_validator_t, 405
deallocate_domains
MHAPlugin_Split::domain_handler_t,
1161
DEBUG
addsndfile.cpp, 1526
audiometerbackend.cpp, 1531
browsemhapugins.cpp, 1532
fader_wave.cpp, 1547
MHAIOFile.cpp, 1655
debug
io_asterisk_parser_t, 594
io_tcp_parser_t, 630
debug_file
io_asterisk_parser_t, 595
io_tcp_parser_t, 631
debug_filename
io_asterisk_parser_t, 595
io_tcp_parser_t, 631
decay
cfg_t, 347
dc::dc_t, 387
dc_simple::level_smoothen_t, 408
softclip_t, 1455
softclipper_t, 1456
decomb_coeffs
adm_rtconfig_t, 279
decomb_order
adm_if_t, 274
decrease_condition
mha_fifo_posix_threads_t, 775
decrement
mha_fifo_posix_threads_t, 774
mha_fifo_thread_platform_t, 786
DEFAULT_RETSIZE
mha_parser.hh, 1615
defaultHighInputLatency
MHAIOPortAudio::device_info_t, 956
defaultHighOutputLatency
MHAIOPortAudio::device_info_t, 956
defaultLowInputLatency
MHAIOPortAudio::device_info_t, 956
defaultLowOutputLatency
MHAIOPortAudio::device_info_t, 956
defaultSampleRate
MHAIOPortAudio::device_info_t, 956
Delay
ADM::Delay< F >, 266
delay, 89
coherence::vars_t, 356
dbasync_native::db_if_t, 376
delaysum::delaysum_wave_if_t, 413
mha_dbbuf_t< FIFO >, 752
MHAFilter::gamma_flt_t, 894
MHAFilter::partitioned_convolution_t::index_t,
920

MHAPlugin_Split::split_t, 1177
 MHASignal::delay_spec_t, 1199
 MHASignal::delay_wave_t, 1203
 delay.cpp, 1540
 delay.hh, 1540
 delay::interface_t, 409
 delays, 410
 interface_t, 410
 patchbay, 410
 prepare, 410
 process, 410
 update, 410
 delay_ac
 ac2wave_if_t, 188
 ac2wave_t, 190
 delay_check_t
 dbasync_native::delay_check_t, 380
 delay_d
 adaptive_feedback_canceller, 243
 DELAY_FREQ
 ADM, 82
 delay_in
 ac2wave_if_t, 188
 ac2wave_t, 190
 delay_spec_t
 MHASignal::delay_spec_t, 1198
 delay_t
 MHASignal::delay_t, 1200
 delay_w
 adaptive_feedback_canceller, 243
 delay_wave_t
 MHASignal::delay_wave_t, 1202
 delayComp
 rohBeam::rohConfig, 1405
 delays
 delay::interface_t, 410
 MHASignal::delay_t, 1201
 delays_in
 MHAIOJack::io_jack_t, 943
 delays_out
 MHAIOJack::io_jack_t, 944
 delaysum, 89
 delaysum::delaysum_wave_if_t, 411
 delay, 413
 delaysum_wave_if_t, 412
 patchbay, 413
 prepare, 412
 process, 412
 release, 413
 update_cfg, 413
 weights, 413
 delaysum::delaysum_wave_t, 414
 delaysum_wave_t, 415
 out, 415
 process, 415
 weights, 415
 delaysum_spec, 89
 delaysum_spec.cpp, 1540
 delaysum_spec::delaysum_spec_if_t, 416
 delaysum_spec_if_t, 417
 gain, 418
 groupdelay, 418
 patchbay, 418
 prepare, 417
 process, 417
 update_cfg, 417
 delaysum_spec::delaysum_t, 418
 delaysum_t, 418
 output, 419
 process, 419
 scale, 419
 delaysum_spec_if_t
 delaysum_spec::delaysum_spec_if_t, 417
 delaysum_t
 delaysum_spec::delaysum_t, 418
 delaysum_wave.cpp, 1541
 delaysum_wave_if_t
 delaysum::delaysum_wave_if_t, 412
 delaysum_wave_t
 delaysum::delaysum_wave_t, 415
 delete_plug
 altpugs_t, 304
 DELT
 gsc_adaptive_stage, 96
 delta_phi
 fshift::fshift_config_t, 510
 fshift_hilbert::hilbert_shifter_t, 521
 delta_phi_total
 fshift::fshift_config_t, 510
 fshift_hilbert::hilbert_shifter_t, 521
 delta_pitch
 smooth_cepstrum::smooth_cepstrum_if_t, 1436
 smooth_cepstrum::smooth_params, 1446
 descriptor
 mha_audio_t, 742
 desired_chan
 gsc_adaptive_stage::gsc_adaptive_stage, 540
 desired_delay
 gtfb_simple_t, 572
 desired_fill_count

mha_drifter_fifo_t< T >, 761
detected
 windnoise::if_t, 1511
detected_acname
 windnoise::if_t, 1511
dev
 audiometerbackend::Inn3rdoct_t, 322
dev_in
 io_alsa_t, 582
dev_out
 io_alsa_t, 582
device
 alsa_dev_par_parser_t, 290
device_index_in
 MHAIOPortAudio::io_portaudio_t, 962
device_index_in_updated
 MHAIOPortAudio::io_portaudio_t, 959
device_index_out
 MHAIOPortAudio::io_portaudio_t, 962
device_index_out_updated
 MHAIOPortAudio::io_portaudio_t, 959
device_info
 MHAIOPortAudio::io_portaudio_t, 960
device_info_t
 MHAIOPortAudio::device_info_t, 954
device_name_in
 MHAIOPortAudio::io_portaudio_t, 962
device_name_in_updated
 MHAIOPortAudio::io_portaudio_t, 959
device_name_out
 MHAIOPortAudio::io_portaudio_t, 962
device_name_out_updated
 MHAIOPortAudio::io_portaudio_t, 959
df
 fshift::fshift_config_t, 510
 fshift::fshift_t, 513
 fshift_hilbert::frequency_translator_t, 516
 fshift_hilbert::hilbert_shifter_t, 520
diag_loading_mu
 rohBeam::rohBeam, 1400
diff_coeffs
 mha_filter.cpp, 1593
diff_t
 MHAFilter::diff_t, 876
digits
 mha_error_helpers, 99
dimension
 MHASignal::matrix_t, 1226
dimensions
 acmon::acmon_t, 210
dimstr
 acmon::ac_monitor_t, 205
dir
 mha_channel_info_t, 744
dir_t
 MHAJack::port_t, 981
dir_type
 MHAJack::port_t, 983
dis
 dropgen_t, 438
Discard
 lsl2ac, 97
discard
 MHASignal::ringbuffer_t, 1239
disconnect
 MHAEvents::emitter_t, 859
disk_write_threshold_min_num_samples
 plugins::hoertech::acrec::acwriter_t, 1383
diskbuffer
 plugins::hoertech::acrec::acwriter_t, 1384
dispmode
 acmon::acmon_t, 210
dist
 plingploing::plingploing_t, 1344
dist1
 plingploing::plingploing_t, 1344
distance
 mha_direction_t, 755
distances
 adm_if_t, 274
dm
 lpc_burglattice_config, 674
do_clipping
 calibrator_variables_t, 344
do_get_var
 testplugin::ac_parser_t, 1475
do_insert_var
 testplugin::ac_parser_t, 1475
DO_RESAMPLE
 addsndfile, 80
doagcc
 doasvm_feature_extraction_config, 430
doasvm
 doasvm_classification_config, 424
doasvm_classification, 420
 ~doasvm_classification, 421
 angles, 422
 b, 422
 doasvm_classification, 421
 max_p_ind_name, 423
 p_name, 423
 patchbay, 423

prepare, 421
 process, 421
 release, 422
 update_cfg, 422
 vGCC_name, 423
 w, 422
 x, 422
 y, 422
doasvm_classification.cpp, 1541
 INSERT_PATCH, 1541
 PATCH_VAR, 1541
doasvm_classification.h, 1542
doasvm_classification_config, 423
 ~doasvm_classification_config, 424
 ac, 424
 c, 425
 doasvm, 424
 doasvm_classification_config, 424
 p, 424
 p_max, 424
 process, 424
doasvm_feature_extraction, 425
 ~doasvm_feature_extraction, 426
 doasvm_feature_extraction, 426
 fftlen, 428
 max_lag, 428
 nupsample, 428
 patchbay, 428
 prepare, 427
 process, 426
 release, 428
 update_cfg, 428
 vGCC_name, 428
doasvm_feature_extraction.cpp, 1542
 INSERT_PATCH, 1542
 PATCH_VAR, 1542
doasvm_feature_extraction.h, 1542
doasvm_feature_extraction_config, 429
 ~doasvm_feature_extraction_config, 430
 doagcc, 430
 doasvm_feature_extraction_config, 429
 fft, 431
 fftlen, 430
 G, 432
 G_length, 430
 GCC_end, 431
 GCC_start, 430
 hifftwin, 431
 hifftwin_sum, 431
 hwin, 431
 ifft, 431
 in_spec, 432
 proc_wave, 431
 process, 430
 vGCC, 431
 vGCC_ac, 431
 wndlen, 430
doc_appendix.h, 1543
doc_examples.h, 1543
doc_frameworks.h, 1543
doc_general.h, 1543
doc_kernel.h, 1543
doc_matlab.h, 1543
doc_mhamain.h, 1543
doc_parser.h, 1543
doc_plugins.h, 1543
doc_system.h, 1543
doc_toolbox.h, 1543
doCircularComp
 gsc_adaptive_stage::gsc_adaptive_stage, 540
 gsc_adaptive_stage::gsc_adaptive_stage_if, 547
document_io_plugin
 analysemhaplugin.cpp, 1529
document_plugin
 analysemhaplugin.cpp, 1529
documentation
 plugindescription_t, 1356
domain
 mhaconfig_t, 851
 MHAParser::mhaconfig_mon_t, 1084
 MHAParser::mhaconfig_mon_t, 1084
 MHAPlugin_Split::splitted_part_t, 1183
 rmslevel::rmslevel_t, 1392
 testplugin::config_parser_t, 1479
domain_handler_t
 MHAPlugin_Split::domain_handler_t, 1159
DONT_RESAMPLE_PERMISSIVE
 addsndfile, 80
DONT_RESAMPLE_STRICT
 addsndfile, 80
 double2acvar, 90
 double2acvar.cpp, 1543
 double2acvar::double2acvar_t, 432
 ~double2acvar_t, 434
 ac_double, 435
 double2acvar_t, 433
 is_prepared, 435
 on_configuration_update, 435
 patchbay, 435
 poll_latest_value_and_reinsert, 434

prepare_, 434
process, 434
release_, 435
double2acvar_t
 double2acvar::double2acvar_t, 433
double_t
 MHA_AC::double_t, 729
doublebuffer_t
 MHASignal::doublebuffer_t, 1204
down
 MHASignal::schroeder_t, 1242
downsample.cpp, 1543
downsampling_factor
 MHAFilter::polyphase_resampling_t, 925
downscale
 MHASignal::quantizer_t, 1236
drain
 DynComp::dc_afterburn_vars_t, 456
drain_inv
 DynComp::dc_afterburn_rt_t, 451
drand
 plingploing, 156
dropgen.cpp, 1544
dropgen_t, 436
 chance, 438
 dis, 438
 dropgen_t, 437
 max_sleep_time, 438
 min_sleep_time, 438
 patchbay, 438
 prepare, 437
 process, 437
 r, 438
 random_engine, 438
 release, 437
dropouts
 droptect_t, 441
droptect.cpp, 1544
droptect_t, 439
 blocks, 441
 consecutive_dropouts, 441
 current_powspec, 442
 dropouts, 441
 droptect_t, 440
 filter_activated, 442
 filtered_powspec, 442
 filtered_powspec_mon, 442
 level_mon, 442
 period, 442
 prepare, 440
 process, 441
 release, 441
 reset, 441
 tau, 442
 threshold, 442
ds_t, 443
 antialias, 444
 ds_t, 444
 prepare, 444
 process, 444
 ratio, 444
 release, 444
dt
 mha_audio_descriptor_t, 741
dtime
 MHA_TCP, 103
dummy_interface_test
 MHAIOalsa.cpp, 1647
 MHAIOAsterisk.cpp, 1652
 MHAIOFile.cpp, 1657
 MHAIOJack.cpp, 1661
 MHAIOJackdb.cpp, 1665
 MHAIOParser.cpp, 1669
 MHAIOPortAudio.cpp, 1673
 MHAIOTCP.cpp, 1679
dummy_jack_proc_cb
 mhajack.cpp, 1682
dummy_threads_t
 MHAPlugin_Split::dummy_threads_t,
 1165
dump_mha
 fw_t, 529
dup
 MHAFilter::thirdoctave_analyzer_t, 934
duplicate_vector
 gtfb_simple_rt_t, 566
DUPVEC
 dc.cpp, 1537
dupvec
 Vector and matrix processing toolbox, 41
dupvec_chk
 Vector and matrix processing toolbox, 41
dur_
 plingploing::plingploing_t, 1343
dynamiclib_t, 445
 ~dynamiclib_t, 447
 dynamiclib_t, 446, 447
 fullname, 449
 getmodulename, 448
 getname, 448
 h, 449
 load_lib, 448

modulename, 449
 resolve, 447
 resolve_checked, 447
DynComp, 90
 interp1, 90
 interp2, 91
DynComp::dc_afterburn_rt_t, 449
 burn, 450
 conflux, 451
 dc_afterburn_rt_t, 450
 drain_inv, 451
 lp, 451
 maxgain, 451
 mpo_inv, 451
DynComp::dc_afterburn_t, 452
 _cf, 454
 _channels, 454
 _srates, 454
 burn, 453
 commit_pending, 454
 dc_afterburn_t, 453
 fb_pars_configured, 454
 patchbay, 454
 set_fb_pars, 453
 unset_fb_pars, 453
 update, 453
 update_burner, 453
DynComp::dc_afterburn_vars_t, 455
 bypass, 457
 commit, 457
 conflux, 456
 dc_afterburn_vars_t, 456
 drain, 456
 f, 456
 maxgain, 456
 mpo, 456
 taugain, 456
DynComp::gaintable_t, 457
 ~gaintable_t, 458
 data, 461
 gaintable_t, 458
 get_gain, 459, 460
 get_iofun, 460
 get_vF, 460
 get_vL, 460
 nbands, 460
 nchannels, 460
 num_channels, 461
 num_F, 461
 num_L, 461
 update, 459
 vF, 461
 vFlog, 461
 vL, 461

E
gsc_adaptive_stage::gsc_adaptive_stage, 542
e
gsc_adaptive_stage::gsc_adaptive_stage, 542
E2
gsc_adaptive_stage::gsc_adaptive_stage, 542
e_out
gsc_adaptive_stage::gsc_adaptive_stage, 543
ear_t
 AuditoryProfile::parser_t::ear_t, 328
edge_frequencies
 dc::dc_vars_t, 391
 dc_simple::dc_if_t, 396
ef
 MHAOvIFilter::fftfb_vars_t, 1010
ef2bands
 MHAOvIFilter::fspacing_t, 1016
ef_h
 MHAOvIFilter::band_descriptor_t, 998
ef_l
 MHAOvIFilter::band_descriptor_t, 998
ef_name
 dc::dc_vars_t, 390
efv
 MHAOvIFilter::fftfb_ac_info_t, 1002
 multibandcompressor::fftfb_plug_t, 1299
element_gain_name
 combc_if_t, 358
 gtfb_simple_t, 572
element_gain_name_
 combc_t, 360
 gtfb_simple_rt_t, 568
elevation
 mha_direction_t, 755
emit_event
 MHAEvents::connector_base_t, 853, 854
 MHAEvents::connector_t< receiver_t >, 856, 857
emitter
 MHAEvents::connector_t< receiver_t >, 857
emitter_die
 MHAEvents::connector_base_t, 854
emitter_is_alive

MHAEvents::connector_base_t, 854
empty_string
 MHParse::keyword_list_t, 1070
enable_adaptive_beam
 rohBeam::configOptions, 1393
 rohBeam::rohBeam, 1400
 rohBeam::rohConfig, 1405
enable_export
 rohBeam::rohBeam, 1400
enable_wng_optimization
 rohBeam::rohBeam, 1400
end_time
 MHA_TCP::Timeout_Event, 833
entries
 MHParse::keyword_list_t, 1070
 MHParse::parser_t, 1103
entry
 MHParse::entry_t, 1055
entry_map_t
 MHParse, 126
entry_t
 MHParse::entry_t, 1054
envelope_delay
 MHAFilter::gamma_flt_t, 895
envreplace
 MHParse, 127
eof
 MHA_TCP::Connection, 806
EPrew
 adaptive_feedback_canceller_config, 249
EPSILON
 lpc_bl_predictor.h, 1567
 lpc_burg-lattice.h, 1568
epsilon
 smoothgains_bridge::overlapadd_if_t,
 1450
equal_dim
 Vector and matrix processing toolbox, 42
equalize, 91
equalize.cpp, 1544
equalize::cfg_t, 462
 ~cfg_t, 462
 cfg_t, 462
 fftgains, 463
 nchannels, 463
 num_bins, 463
 operator=, 462
equalize::freqgains_t, 463
 fftgains, 465
 freqgains_t, 464
 id, 465
patchbay, 465
prepare, 464
process, 464
update_gains, 465
update_id, 465
equidist2bands
 MHAOvIFilter::fspacing_t, 1016
erb_hz_f_hz
 speechnoise.cpp, 1699
ERR_IHANDLE
 MHAIOalsa.cpp, 1646
 MHAIOAsterisk.cpp, 1650
 MHAIOFile.cpp, 1656
 MHAIOJack.cpp, 1660
 MHAIOJackdb.cpp, 1664
 MHAIOParser.cpp, 1668
 MHAIOPortAudio.cpp, 1672
 MHAIOTCP.cpp, 1677
err_in
 MHAFilter::adapt_filter_param_t, 863
 MHAFilter::adapt_filter_t, 867
ERR_SUCCESS
 MHAIOalsa.cpp, 1646
 MHAIOAsterisk.cpp, 1650
 MHAIOFile.cpp, 1655
 MHAIOJack.cpp, 1660
 MHAIOJackdb.cpp, 1664
 MHAIOParser.cpp, 1668
 MHAIOPortAudio.cpp, 1672
 MHAIOTCP.cpp, 1677
ERR_USER
 MHAIOalsa.cpp, 1646
 MHAIOAsterisk.cpp, 1650
 MHAIOFile.cpp, 1656
 MHAIOJack.cpp, 1660
 MHAIOJackdb.cpp, 1664
 MHAIOParser.cpp, 1668
 MHAIOPortAudio.cpp, 1672
 MHAIOTCP.cpp, 1677
error
 mha_fifo_lw_t< T >, 772
 MHA_TCP::Thread, 831
Error handling in the openMHA, 28
 MHA_assert, 29
 MHA_assert_equal, 29
 mha_debug, 29
 MHA_ErrorMsg, 28
error_h
 osc_server_t, 1322
errorlog
 fw_t, 528

ESTIM_CUR
 nlms_wave.cpp, 1690
ESTIM_PREV
 nlms_wave.cpp, 1689
estimateDebug
 noise_psd_estimator::noise_psd_estimator_t, 1314
ESTIMATION_TYPES
 nlms_wave.cpp, 1689
estimtype
 nlms_t, 1307
event_add_plug
 altplugs_t, 303
event_delete_plug
 altplugs_t, 303
event_select_all
 altconfig_t, 298
event_select_plug
 altplugs_t, 303
event_set_plugs
 altplugs_t, 302
event_start_recording
 acsave::acsave_t, 221
event_stop_and_flush
 acsave::acsave_t, 221
eventhandler
 MHAEvents::connector_t< receiver_t >, 857
eventhandler_s
 MHAEvents::connector_t< receiver_t >, 857
eventhandler_suu
 MHAEvents::connector_t< receiver_t >, 858
Events
 MHA_TCP::Event_Watcher, 811
events
 MHA_TCP::Event_Watcher, 812
example1.cpp, 1544
example1_t, 466
 example1_t, 467
 prepare, 467
 process, 468
 release, 467
example2.cpp, 1544
example2_t, 468
 example2_t, 470
 factor, 471
 prepare, 470
 process, 471
 release, 470
 scale_ch, 471
example3.cpp, 1545
example3_t, 472
 example3_t, 473
 factor, 475
 on_prereadaccess, 474
 on_scale_ch_readaccess, 474
 on_scale_ch_valuechanged, 474
 on_scale_ch_writeaccess, 473
 patchbay, 475
 prepare, 474
 prepared, 475
 process, 474
 release, 474
 scale_ch, 475
example4.cpp, 1545
example4_t, 476
 example4_t, 477
 factor, 479
 on_prereadaccess, 478
 on_scale_ch_readaccess, 478
 on_scale_ch_valuechanged, 478
 on_scale_ch_writeaccess, 478
 patchbay, 480
 prepare, 478
 prepared, 479
 process, 479
 release, 478
 scale_ch, 479
example5.cpp, 1545
 __declspec, 1545
example5_t, 480
 channel, 481
example5_t, 480
 process, 480
 scale, 481
example6.cpp, 1545
 __declspec, 1546
example6_t, 481
 channel_no, 483
example6_t, 482
 patchbay, 483
 prepare, 482
 process, 482
 rmsdb, 483
 update_cfg, 483
example7.cpp, 1546
example7.hh, 1546
example7_t, 484
 example7_t, 484
 prepare, 485

process, 485
release, 485

exec_fw_command
 fw_t, 526

existed_before
 mha_stash_environment_variable_t, 796

exit_on_stop
 fw_t, 528

exit_request
 fw_t, 525
 plugins::hoertech::acrec::acwriter_t, 1382
 wavwriter_t, 1501

expansion
 dc_simple::dc_t, 399

expansion_slope
 dc_simple::dc_vars_t, 403

expansion_threshold
 dc_simple::dc_t, 399
 dc_simple::dc_vars_t, 403

expfilt
 MHAOvlFilter::ShapeFun, 122

expi
 Complex arithmetics in the openMHA, 61, 64

explicit_insert
 dc::dc_t, 385

export_beam_design
 rohBeam::rohBeam, 1398

export_to
 MHASignal::spectrum_t, 1248
 MHASignal::waveform_t, 1269

expression_t, 485
 MHAParser::expression_t, 1055, 1056

extern_connector
 MHAParser::commit_t< receiver_t >, 1049

F

 adaptive_feedback_canceller_config, 246
 rt_nlms_t, 1415

f

 AuditoryProfile::parser_t::fmap_t, 331
 DynComp::dc_afterburn_vars_t, 456
 io_tcp_sound_t::float_union, 637
 MHAOvlFilter::fftfb_vars_t, 1009
 MHAOvlFilter::fscale_t, 1014

f0_high
 smooth_cepstrum::smooth_cepstrum_if_t, 1436
 smooth_cepstrum::smooth_params, 1446

f0_low

 smooth_cepstrum::smooth_cepstrum_if_t, 1436
 smooth_cepstrum::smooth_params, 1446

f_est
 lpc_bl_predictor_config, 668

f_hz
 MHAOvlFilter::fscale_t, 1014

F_Uflt
 adaptive_feedback_canceller_config, 248

factor
 complex_scale_channel_t, 364
 cpupload::cpupload_cfg_t, 366
 cpupload::cpupload_if_t, 369
 example2_t, 471
 example3_t, 475
 example4_t, 479
 plugin_interface_t, 1353

fader_if_t, 486
 actgains, 488
 fader_if_t, 487
 newgains, 488
 patchbay, 487
 prepare, 487
 process, 487
 tau, 487
 update_cfg, 487

fader_spec.cpp, 1546

fader_wave, 92
 level_adaptor, 92

fader_wave.cpp, 1546
 DEBUG, 1547

fader_wave::fader_wave_if_t, 488
 fader_wave_if_t, 489
 gain, 490
 patchbay, 490
 prepare, 489
 prepared, 490
 process, 489
 ramplen, 490
 release, 489
 set_level, 490

fader_wave::level_adapt_t, 491
 can_update, 492
 get_level, 492
 ilen, 492
 l_new, 492
 l_old, 493
 level_adapt_t, 491
 pos, 492
 update_frame, 492
 wnd, 492

fader_wave_if_t
 fader_wave::fader_wave_if_t, 489

fail_on_async_jackerr
 MHAIOJackdb::io_jack_t, 951

fail_on_async_jackerror
 MHAIOJackdb::io_jack_t, 948
 MHAJack::client_t, 980

fail_on_nonmonotonic
 MHAOvIFilter::ffftfb_vars_t, 1009

fail_on_nonmonotonic_cf
 MHAOvIFilter::fspacing_t, 1016

fail_on_unique_bins
 MHAOvIFilter::ffftfb_vars_t, 1009

fail_on_unique_fftbins
 MHAOvIFilter::fspacing_t, 1016

fallback_spec
 altplugs_t, 305

fallback_wave
 altplugs_t, 305

Fast Fourier Transform functions, 70
 mha_fft_backward, 75
 mha_fft_backward_scale, 76
 mha_fft_forward, 75
 mha_fft_forward_scale, 76
 mha_fft_free, 72
 mha_fft_new, 71
 mha_fft_spec2wave, 73, 74
 mha_fft_spec2wave_scale, 77
 mha_fft_t, 71
 mha_fft_wave2spec, 72, 73
 mha_fft_wave2spec_scale, 76

fatallog
 fw_t, 529

fb
 MHAFilter::thirdoctave_analyzer_t, 934

fb_acinfo
 fftfilterbank::ffftfb_plug_t, 508

fb_pars_configured
 DynComp::dc_afterburn_t, 454

fbpow
 fftfbpow::fftfbpow_t, 497

fcn_init
 matlab_wrapper::matlab_wrapper_t::wrapped_pfftpow_interface_t, 494
 707

fcn_prepare
 matlab_wrapper::matlab_wrapper_t::wrapped_pfftpow_interface_t, 494
 708

fcn_process_ss
 matlab_wrapper::matlab_wrapper_t::wrapped_pfftpow_interface_t, 496
 708

fcn_process_sw
 matlab_wrapper::matlab_wrapper_t::wrapped_pfftpow_interface_t, 496
 fbpow, 497
 fftfbpow_t, 497

matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 708

fcn_process_ws
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 708

fcn_process_ww
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 707

fcn_release
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 708

fcn_terminate
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 707

fd
 MHA_TCP::Connection, 810
 MHA_TCP::OS_EVENT_TYPE, 814

fft
 doasvm_feature_extraction_config, 431
 MHAFilter::fftfilter_t, 881
 MHAFilter::fftfilterbank_t, 886
 MHAFilter::partitioned_convolution_t, 918
 MHAFilter::smoothspec_t, 932
 overlapadd::overlapadd_t, 1333

fft_find_bin
 fshift, 94

fft_t
 MHASignal::fft_t, 1208

ffftfb_ac_info_t
 MHAOvIFilter::ffftfb_ac_info_t, 1001

ffftfb_interface_t
 fftfilterbank::ffftfb_interface_t, 503

ffftfb_plug_t
 fftfilterbank::ffftfb_plug_t, 507
 multibandcompressor::ffftfb_plug_t, 1298

ffftfb_t
 MHAOvIFilter::ffftfb_t, 1003

ffftfb_vars_t
 MHAOvIFilter::ffftfb_vars_t, 1008

fftfbpow, 92

fftfbpow.cpp, 1547

fftfbpow::fftfbpow_interface_t, 493

fftfbpow_interface_t, 494
 name, 495
 patchbay, 496

process, 494
 process, 495
 update_cfg, 495

fftfbpow_t, 496
 fbpow, 497
 fftfbpow_t, 497

fftfbpow_interface_t
 fftfbpow::fftfbpow_interface_t, 494

fftfbpow_t
 fftfbpow::fftfbpow_t, 497

fftfilt
 fftfilter::fftfilter_t, 500

fftfilter, 92
 irs_length, 93
 irs_validator, 93

fftfilter.cpp, 1547

fftfilter::fftfilter_t, 498
 channels, 499
 fftfilt, 500
 fftfilter_t, 498
 fftlen, 499
 fragsize, 499
 irslen, 499
 process, 499

fftfilter::interface_t, 500
 fftlen, 502
 fftlen_final, 502
 interface_t, 501
 irs, 502
 patchbay, 502
 prepare, 501
 process, 501
 update, 502

fftfilter_t
 fftfilter::fftfilter_t, 498
 MHAFilter::fftfilter_t, 877

fftfilterbank, 94

fftfilterbank.cpp, 1548

fftfilterbank::fftfb_interface_t, 503
 algo, 505
 fftfb_interface_t, 503
 nbands, 505
 nchannels, 505
 patchbay, 505
 prepare, 504
 prepared, 505
 process, 504, 505
 release, 504
 return_imag, 505
 update_cfg, 505

fftfilterbank::fftfb_plug_t, 506
 fb_acinfo, 508
 fftfb_plug_t, 507
 imag, 508
 insert, 507
 process, 507
 return_imag_, 508

 s_out, 508

fftfilterbank_t
 MHAFilter::fftfilterbank_t, 882

fftgains
 equalize::cfg_t, 463
 equalize::freqgains_t, 465

fftlen
 dc::dc_t, 387
 dc_simple::level_smoothen_t, 408
 doasvm_feature_extraction, 428
 doasvm_feature_extraction_config, 430
 fftfilter::fftfilter_t, 499
 fftfilter::interface_t, 502
 level_matching::level_matching_config_t, 652
 mhaconfig_t, 852
 MHAFilter::fftfilter_t, 880
 MHAFilter::fftfilterbank_t, 885
 MHAFilter::smoothspec_t, 931
 MHAOvIFilter::fftfb_t, 1006
 MHAOvIFilter::overlap_save_filterbank_t::vars_t, 1023
 MHAParser::mhaconfig_mon_t, 1085
 rmslevel::rmslevel_t, 1392
 smooth_cepstrum::smooth_cepstrum_t, 1440
 testplugin::config_parser_t, 1479

fftlen_final
 fftfilter::interface_t, 502

fftw_plan_fft
 MHASignal::fft_t, 1211

fftw_plan_ifft
 MHASignal::fft_t, 1211

fftw_plan_spec2wave
 MHASignal::fft_t, 1211

fftw_plan_wave2spec
 MHASignal::fft_t, 1211

fhz2bandno
 speechnoise.cpp, 1699

fifo
 plugins::hoertech::acrec::acwriter_t, 1383
 wavwriter_t, 1502

fifo_size
 mha_dblbuf_t< FIFO >, 753

fifolen
 analysispath_if_t, 312
 plugins::hoertech::acrec::acrec_t, 1378
 wavrec_t, 1499

fileformat
 acsave::acsave_t, 222

filename

addsndfile::addsndfile_if_t, 252
 filename_input
 io_file_t, 608
 filename_output
 io_file_t, 608
 fill
 rmslevel::rmslevel_t, 1391
 fill_info
 MHAIOPortAudio::device_info_t, 955
 filled
 MHASignal::async_rmslevel_t, 1198
 filter
 MHAFilter::adapt_filter_state_t, 864
 MHAFilter::adapt_filter_t, 866
 MHAFilter::complex_bandpass_t, 873, 874
 MHAFilter::fftfilter_t, 878, 879
 MHAFilter::fftfilterbank_t, 883, 884
 MHAFilter::filter_t, 889
 MHAFilter::iir_filter_t, 898
 filter_activated
 droptect_t, 442
 filter_analytic
 MHAOvIFilter::overlap_save_filterbank_analytic_MHAParser::float_mon_t, 1057
 1019
 filter_complex
 gtfb_analyzer.cpp, 1554
 filter_partitions
 MHAFilter::partitioned_convolution_t, 917
 filter_real
 gtfb_analyzer.cpp, 1555
 filter_simd
 gtfb_simd.cpp, 1559
 filter_sisd_complex
 gtfb_simd.cpp, 1558
 filter_sisd_real
 gtfb_simd.cpp, 1558
 filter_t
 MHAFilter::filter_t, 888
 filterbank
 dc::dc_vars_t, 390
 dc_simple::dc_if_t, 396
 filtered_level
 dc::dc_vars_t, 391
 filtered_powspec
 droptect_t, 442
 filtered_powspec_mon
 droptect_t, 442
 filtershapefun
 mha_fftfb.cpp, 1590
 FINISHED
 MHA_TCP::Thread, 829
 fir
 calibrator_runtime_layer_t, 338
 calibrator_variables_t, 342
 fir_lp
 MHAFilter, 107
 firchannels
 MHAFilter::fftfilterbank_t, 885
 firfir2ffflen
 calibrator_runtime_layer_t, 337
 firfirlen
 calibrator_runtime_layer_t, 337
 flag_allow_empty_bands
 MHAOvIFilter::fftfb_vars_t, 1009
 flag_terminate_inner_thread
 analysepath_t, 309
 flags
 MHAJack::client_t, 979
 flatten
 MHASignal::waveform_t, 1262
 float_data
 testplugin::ac_parser_t, 1476
 float_mon_t
 MHAParser::float_mon_t, 1057
 float_t
 MHA_AC::float_t, 731
 MHAParser::float_t, 1060
 flush_data
 acsave::cfg_t, 224
 fmap_t
 AuditoryProfile::parser_t::fmap_t, 330
 fmax
 fshift::fshift_t, 513
 fshift_hilbert::frequency_translator_t, 516
 fmin
 fshift::fshift_t, 513
 fshift_hilbert::frequency_translator_t, 516
 FMTsz
 mha_os.h, 1604
 fname
 acsave::acsave_t, 222
 for_each
 MHASignal, 139
 force_remove_item
 MHAParser::parser_t, 1100
 force_resize
 dc_simple, 88
 format
 ac2lsl::type_info, 179
 io_alsa_t, 583
 format_name

wavwriter_t, 1503
forward
 lpc_bl_predictor_config, 668
 lpc_burglattice_config, 674
 MHASignal::fft_t, 1209
forward_scale
 MHASignal::fft_t, 1209
fr
 spec_fader_t, 1465
frac_old
 gsc_adaptive_stage::gsc_adaptive_stage, 539
frag_out
 MHAJack::client_avg_t, 966
 MHAJack::client_noncont_t, 970
fragsize
 alsa_t< T >, 294
 analysispath_if_t, 312
 calibrator_variables_t, 343
 db_if_t, 372
 dbasync_native::db_if_t, 376
 fftfilter::fftfilter_t, 499
 io_asterisk_sound_t, 598
 io_file_t, 607
 io_parser_t, 618
 io_tcp_sound_t, 635
 mconv::MConv, 719
 mhaconfig_t, 851
 MHAFilter::fftfilter_t, 879
 MHAFilter::fftfilterbank_t, 885
 MHAFilter::partitioned_convolution_t, 916
 MHAFilter::resampling_filter_t, 928
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 977
 MHAParser::mhaconfig_mon_t, 1084
 MHAPlugin_Resampling::resampling_if_t, 1154
 testplugin::config_parser_t, 1479
fragsize_in
 MHAFilter::blockprocessing_polyphase_resampling_t, 870
fragsize_out
 MHAFilter::blockprocessing_polyphase_resampling_t, 870
fragsize_ratio
 MHAIOJackdb::io_jack_t, 950
fragsize_validator
 MHAFilter::resampling_filter_t, 927
frame
 MHA_AC::acspace2matrix_t, 727
frame_data
 alsa_t< T >, 294
framecnt
 acsave::save_var_t, 228
 adm_if_t, 275
frameno
 MHA_AC::acspace2matrix_t, 728
 noise_psd_estimator::noise_psd_estimator_t, 1315
framerate
 ac2osc_t, 184
frames
 ac2wave_t, 189
 adaptive_feedback_canceller_config, 246
 gtfb_analyzer::gtfb_analyzer_cfg_t, 550
 rt_nlms_t, 1415
frameshift
 fshift_hilbert::hilbert_shifter_t, 521
framework_thread_priority
 dbasync_native::db_if_t, 377
 MHAPlugin_Split::split_t, 1177
framework_thread_scheduler
 dbasync_native::db_if_t, 377
 MHAPlugin_Split::split_t, 1177
freq
 audiometerbackend::audiometer_if_t, 317
 plingploing::plingploing_t, 1344
freq2bin
 Vector and matrix processing toolbox, 39
freq_offsets
 rmslevel::rmslevel_t, 1392
freqgains_t
 equalize::freqgains_t, 464
freqResp
 rohBeam::rohConfig, 1407
frequency
 sine_t, 1432
frequency_response
 MHAFilter::partitioned_convolution_t, 918
frequency_translator_t
 fshift_hilbert::frequency_translator_t, 515
FrequencyBinLowPass
 windnoise::cfg_t, 1506
fronthop_channel
 adm_rtconfig_t, 278
front_channels
 adm_if_t, 273
 adm_rtconfig_t, 279
frozen_noise_cfg_t, 347
frozennoise_length
 noise_t, 1317

fs
 MHAFilter::o1_ar_filter_t, 907
fs_
 MHAOvIFilter::fspacing_t, 1017
fscale
 gtfb_simple_t, 572
 MHAOvIFilter::ffftfb_vars_t, 1008
fscale_bw_t
 MHAOvIFilter::fscale_bw_t, 1011
fscale_t
 MHAOvIFilter::fscale_t, 1013
fshift, 94
 fft_find_bin, 94
fshift.cpp, 1548
fshift.hh, 1548
fshift::fshift_config_t, 508
 ~fshift_config_t, 509
 delta_phi, 510
 delta_phi_total, 510
 df, 510
 fshift_config_t, 509
 kmax, 510
 kmin, 510
 process, 509
fshift::fshift_t, 511
 ~fshift_t, 512
 df, 513
 fmax, 513
 fmin, 513
fshift_t, 512
 m_df, 514
 m_fmax, 514
 m_fmin, 513
 patchbay, 513
 prepare, 512
 process, 512
 release, 513
 update_cfg, 513
fshift_config_t
 fshift::fshift_config_t, 509
fshift_hilbert, 95
fshift_hilbert.cpp, 1549
fshift_hilbert::frequency_translator_t, 514
 df, 516
 fmax, 516
 fmin, 516
 frequency_translator_t, 515
 irslen, 517
 patchbay, 516
 phasemode, 517
 prepare, 516
 process, 515
 release, 516
 update, 516
fshift_hilbert::hilbert_shifter_t, 517
 ~hilbert_shifter_t, 519
 analytic, 519
 delta_phi, 521
 delta_phi_total, 521
 df, 520
 frameshift, 521
 fullspec, 519
 hilbert_shifter_t, 518
 kmax, 521
 kmin, 520
 mhafft, 520
 mixw_ref, 520
 mixw_shift, 520
 plan_spec2analytic, 520
 process, 519
 shifted, 519
fshift_t
 fshift::fshift_t, 512
fspacing_t
 MHAOvIFilter::fspacing_t, 1015
ft
 spec2wave_t, 1463
 wave2spec_t, 1495
ftype
 MHAOvIFilter::ffftfb_vars_t, 1008
fu
 rt_nlms_t, 1415
fu_previous
 rt_nlms_t, 1416
fuflt
 rt_nlms_t, 1416
fullname
 dynamiclib_t, 449
 MHAParser::base_t, 1036
 plugindescription_t, 1356
fullspec
 fshift_hilbert::hilbert_shifter_t, 519
fun1
 plingploing::plingploing_t, 1344
fun1_key
 plingploing::if_t, 1339
 plingploing::plingploing_t, 1344
fun1_range
 plingploing::if_t, 1340
 plingploing::plingploing_t, 1344
fun2
 plingploing::plingploing_t, 1344

fun2_key
 plingploing::if_t, 1340
 plingploing::plingploing_t, 1344

fun2_range
 plingploing::if_t, 1340
 plingploing::plingploing_t, 1344

fun_t
 MHAWindow::fun_t, 1291

fun_s
 MHAOvlFilter::scale_var_t, 1026

fw_cmd
 fw_t, 528

fw_exiting
 fw_t, 524

fw_fragsize
 io_alsa_t, 581
 MHAIOJack::io_jack_t, 943

fw_running
 fw_t, 524

fw_samplerate
 io_alsa_t, 581
 MHAIOJack::io_jack_t, 943

fw_sleep
 fw_t, 528

fw_sleep_cmd
 fw_t, 527

fw_starting
 fw_t, 524

fw_stopped
 fw_t, 524

fw_stopping
 fw_t, 524

fw_t, 522
 ~fw_t, 524
 ac, 529
 async_poll_msg, 527
 async_read, 527
 b_exit_request, 530
 cfin, 529
 cfout, 530
 dump_mha, 529
 errorlog, 528
 exec_fw_command, 526
 exit_on_stop, 528
 exit_request, 525
 fatallog, 529
 fw_cmd, 528
 fw_exiting, 524
 fw_running, 524
 fw_sleep, 528
 fw_sleep_cmd, 527

 fw_starting, 524
 fw_stopped, 524
 fw_stopping, 524
 fw_t, 524
 fw_unprepared, 524
 fw_until, 528
 fw_until_cmd, 527
 get_input_signal_dimension, 527
 get_parserstate, 527
 inst_name, 529
 io_error, 530
 io_lib, 529
 io_name, 528
 load_io_lib, 527
 load_proc_lib, 526
 nchannels_out, 528
 parserstate, 528
 patchbay, 530
 plugin_paths, 529
 plugins, 529
 prepare, 525
 prepare_vars, 527
 proc_error, 530
 proc_error_string, 530
 proc_lib, 529
 proc_name, 528
 process, 526
 quit, 525
 release, 525
 start, 525
 started, 526
 state, 530
 state_t, 524
 stop, 525
 stopped, 525, 526

fw_unprepared
 fw_t, 524

fw_until
 fw_t, 528

fw_until_cmd
 fw_t, 527

fw_vars_t, 531
 fw_vars_t, 531
 lock_channels, 531
 lock_srate_fragsize, 531
 pfragmentsize, 532
 pinchannels, 532
 psrate, 532
 unlock_channels, 531
 unlock_srate_fragsize, 531

fwcb

io_asterisk_t, 603
 io_tcp_t, 640

G

- doasvm_feature_extraction_config, 432
- g
 - coherence::cohflt_t, 352
- g50
 - dc_simple::dc_vars_t, 403
- g80
 - dc_simple::dc_vars_t, 403
- G_ERRNO
 - MHA_TCP, 102
- G_length
 - doasvm_feature_extraction_config, 430
- gain, 95
 - alsa_t< T >, 294
 - calibrator_runtime_layer_t, 338
 - coherence::cohflt_t, 353
 - delaysum_spec::delaysum_spec_if_t, 418
 - fader_wave::fader_wave_if_t, 490
 - multibandcompressor::plugin_signals_t, 1304
- gain.cpp, 1549
- gain::gain_if_t, 532
 - gain_if_t, 533
 - gains, 534
 - patchbay, 534
 - prepare, 534
 - process, 533
 - release, 534
 - update_gain, 534
 - update_minmax, 534
 - vmax, 535
 - vmin, 534
- gain::scaler_t, 535
 - scaler_t, 535
- gain_ac
 - ac2wave_if_t, 187
 - ac2wave_t, 190
- gain_delay
 - coherence::cohflt_t, 353
- gain_if_t
 - gain::gain_if_t, 533
- gain_in
 - ac2wave_if_t, 187
 - ac2wave_t, 190
- gain_min
 - smooth_cepstrum::smooth_cepstrum_t, 1441
- gain_min_db
- smooth_cepstrum::smooth_cepstrum_if_t, 1437
- smooth_cepstrum::smooth_params, 1447
- gain_spec_
 - cfg_t, 346
- gain_wave_
 - cfg_t, 346
- gain_wiener
 - smooth_cepstrum::smooth_cepstrum_t, 1443
- gainrule
 - dc::dc_vars_t, 391
 - dc_simple::dc_if_t, 396
- gains
 - adaptive_feedback_canceller, 243
 - gain::gain_if_t, 534
 - shadowfilter_end::cfg_t, 1426
 - spec_fader_t, 1465
- gaintable.cpp, 1549
 - convert_f2logf, 1549
 - isempty, 1550
- gaintable.h, 1550
- gaintable_t
 - DynComp::gaintable_t, 458
- gamma_flt_t
 - MHAFilter::gamma_flt_t, 892
- gamma_post
 - smooth_cepstrum::smooth_cepstrum_t, 1442
- gauss
 - MHAOvlFilter::ShapeFun, 122
- GCC_end
 - doasvm_feature_extraction_config, 431
- GCC_start
 - doasvm_feature_extraction_config, 430
- gcd
 - audiometerbackend, 83
 - dbasync_native, 86
 - MHAFilter, 108
- generatemhaplugindoc.cpp, 1550
 - conv2latex, 1551
 - create_latex_doc, 1552
 - main, 1552
 - print_plugin_references, 1551
- generator
 - audiometerbackend, 83
- get
 - testplugin::config_parser_t, 1478
- get_A
 - MHAFilter::gamma_flt_t, 894
- get_a

MHAParser::base_t::replace_t, 1040
get_ac
 latex_doc_t, 645
 plug_t, 1347
get_accept_event
 MHA_TCP::Server, 816
get_adaptation_ratio
 adm_rtconfig_t, 278
get_address
 mha_tcp::server_t, 820
get_all_input_ports
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949
get_all_names_from_ac_space
 ac2lsl::ac2lsl_t, 164
get_all_output_ports
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949
get_all_stream_names
 lsl2ac::lsl2ac_t, 683
get_arg_type_and_dimension
 ac_mul_t, 193, 194
get_attack_filter_state
 dc::dc_t, 386
get_audiochannels
 dc.cpp, 1537
get_available_space
 mha_drifter_fifo_t< T >, 759
 mha_fifo_lf_t< T >, 768
 mha_fifo_t< T >, 779
get_b
 MHAParser::base_t::replace_t, 1040
get_bands
 gtfb_simd_cfg_t, 557
get_buf_address
 ac2lsl::save_var_base_t, 170
 ac2lsl::save_var_t< mha_complex_t >, 177
 ac2lsl::save_var_t< T >, 173
get_buffer
 mha_tcp::buffered_socket_t, 799
get_bw_hz
 MHAOvlFilter::fscale_bw_t, 1011
get_c1
 MHAFilter::o1flt_lowpass_t, 910
get_c_handle
 MHAKernel::algo_comm_class_t, 986
get_categories
 io_wrapper, 642
 latex_doc_t, 645
 plug_wrapper, 1348
 plug_wrapperl, 1350
 PluginLoader::mhaplugloader_t, 1370
get_cdatal
 MHASignal::matrix_t, 1232
get_cf_fftbin
 MHAOvlFilter::fspacing_t, 1015
get_cf_hz
 MHAFilter::thirdoctave_analyzer_t, 933
 MHAOvlFilter::fspacing_t, 1016
get_cfin
 MHAParser::mhaplugloader_t, 1088
get_cfout
 MHAParser::mhaplugloader_t, 1088
get_channelconfig
 MHAOvlFilter::overlap_save_filterbank_t, 1021
get_channels
 gtfb_simd_cfg_t, 557
get_comm_var
 MHASignal::matrix_t, 1226
get_configfile
 PluginLoader::config_file_splitter_t, 1361
get_configname
 PluginLoader::config_file_splitter_t, 1360
get_connected
 io_asterisk_parser_t, 592
 io_tcp_parser_t, 628
get_context
 mha_tcp::server_t, 822
get_cpu_load
 MHAJack::client_t, 976
get_current_profile
 AuditoryProfile::parser_t, 327
get_decay_filter_state
 dc::dc_t, 386
get_delay
 mha_dbdbuf_t< FIFO >, 749
get_delays_in
 MHAIOJack::io_jack_t, 942
get_delays_out
 MHAIOJack::io_jack_t, 942
get_des_fill_count
 mha_drifter_fifo_t< T >, 759
get_documentation
 io_wrapper, 643
 plug_wrapper, 1348
 plug_wrapperl, 1350
 PluginLoader::mhaplugloader_t, 1370
get_ear
 AuditoryProfile::parser_t::ear_t, 329
 AuditoryProfile::profile_t, 332

get_ef_hz
 MHAOvlFilter::fspacing_t, 1016

get_endpoint
 mha_tcp::server_t, 820

get_entries
 algo_comm_t, 285
 MHAKernel::algo_comm_class_t, 988
 MHAParser::keyword_list_t, 1069

get_error
 algo_comm_t, 286
 MHAKernel::algo_comm_class_t, 988

get_f_hz
 MHAOvlFilter::fscale_t, 1013

get_fbpower
 MHAOvlFilter::fftfb_t, 1004

get_fbpower_db
 MHAOvlFilter::fftfb_t, 1004

get_fd
 MHA_TCP::Connection, 806

get_ffflen
 MHAOvlFilter::fftfb_t, 1005

get_fifo_size
 mha_dblbuf_t< FIFO >, 749

get_fill_count
 mha_drifter_fifo_t< T >, 759
 mha_fifo_lf_t< T >, 768
 mha_fifo_t< T >, 779, 781

get_fmap
 AuditoryProfile::parser_t::fmap_t, 330

get_fragsize
 MHAJack::client_t, 974

get_frames
 gtfb_simd_cfg_t, 557

get_frequencies
 AuditoryProfile::fmap_t, 325

get_fun
 MHAOvlFilter::scale_var_t, 1025

get_gain
 DynComp::gaintable_t, 459, 460

get_gf
 gtfb_simple_rt_t, 566

get_handle
 plug_t, 1347

get_idx
 level_matching::channel_pair, 649

get_index
 MHAParser::keyword_list_t, 1069
 MHASignal::matrix_t, 1231

get_inner_error
 mha_dblbuf_t< FIFO >, 750

get_inner_size

 mha_dblbuf_t< FIFO >, 749

get_input_channels
 mha_dblbuf_t< FIFO >, 749

get_input_fifo_fill_count
 mha_dblbuf_t< FIFO >, 750

get_input_fifo_space
 mha_dblbuf_t< FIFO >, 750

get_input_signal_dimension
 fw_t, 527

get_interface
 MHA_TCP::Server, 816

get_iofun
 DynComp::gaintable_t, 460

get_irs
 MHAFilter::fftfilterbank_t, 884

get_last_name
 MHAParser::mhaplugloader_t, 1088

get_last_output
 MHAFilter::o1flt_lowpass_t, 910

get_latex_doc
 latex_doc_t, 645

get_len_A
 MHAFilter::filter_t, 890

get_len_B
 MHAFilter::filter_t, 890

get_length
 MHASignal::uint_vector_t, 1257

get_level
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 319
 fader_wave::level_adapt_t, 492

get_level_in_db
 dc::dc_t, 385

get_level_in_db_adjusted
 dc::dc_t, 386

get_libname
 PluginLoader::config_file_splitter_t, 1360

get_local_address
 io_asterisk_parser_t, 590
 io_tcp_parser_t, 626

get_local_port
 io_asterisk_parser_t, 590
 io_tcp_parser_t, 626

get_longmsg
 MHA_Error, 765

get_ltass_gain_db
 MHAOvlFilter::fftfb_t, 1004

get_main_category
 latex_doc_t, 645

get_mapping
 MHASignal::loop_wavefragment_t, 1219

get_max_fill_count
 mha_fifo_t< T >, 780

get_min_fill_count
 mha_drifter_fifo_t< T >, 760

get_mismatch
 level_matching::channel_pair, 649

get_msg
 MHA_Error, 765

get_my_input_ports
 MHAJack::client_t, 975

get_my_output_ports
 MHAJack::client_t, 975

get_name
 MHAOvlFilter::scale_var_t, 1025

get_nbands
 dc::dc_t, 385

get_nch
 dc::dc_t, 385

get_nelements
 MHASignal::matrix_t, 1227

get_noise_model_func
 rohBeam::rohBeam, 1398

get_nreals
 MHASignal::matrix_t, 1231

get_num_accepted_connections
 mha_tcp::server_t, 821

get_origname
 PluginLoader::config_file_splitter_t, 1360

get_os_event
 MHA_TCP::Timeout_Event, 832
 MHA_TCP::Wakeup_Event, 837

get_outer_size
 mha_dblbuf_t< FIFO >, 749

get_output_channels
 mha_dblbuf_t< FIFO >, 750

get_output_fifo_fill_count
 mha_dblbuf_t< FIFO >, 750

get_output_fifo_space
 mha_dblbuf_t< FIFO >, 750

get_parser_tab
 latex_doc_t, 646

get_parser_var
 latex_doc_t, 646

get_parserstate
 fw_t, 527

get_paths
 pluginbrowser_t, 1354

get_peer_address
 MHA_TCP::Connection, 806

get_peer_port
 MHA_TCP::Connection, 806

get_physical_input_ports
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949

get_physical_output_ports
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949

get_plugins
 pluginbrowser_t, 1355

get_port
 MHA_TCP::Server, 816
 mha_tcp::server_t, 820

get_port_capture_latency
 MHAJack, 112

get_port_capture_latency_int
 MHAJack, 112

get_port_playback_latency
 MHAJack, 114

get_port_playback_latency_int
 MHAJack, 114

get_ports
 MHAJack::client_t, 975

get_precision
 MHAParser, 126

get_process_spec
 plug_t, 1347

get_process_wave
 plug_t, 1346

get_rdata
 MHASignal::matrix_t, 1231

get_read_event
 MHA_TCP::Connection, 806

get_read_ptr
 mha_fifo_t< T >, 781

get_resynthesis_gain
 MHAFilter::gamma_flt_t, 894

get_rmslevel_filter_state
 dc::dc_t, 386

get_server_port_open
 io_asterisk_parser_t, 591
 io_tcp_parser_t, 627

get_short_name
 MHAJack::port_t, 983

get_signal
 MHAPlugin_Split::domain_handler_t,
 1162

get_size
 MHASignal::waveform_t, 1271

get_srate
 MHAJack::client_t, 974

get_time_correction
 lsl2ac::save_var_t, 689

get_type
 MHAParser::window_t, 1136

get_value
 MHAParser::keyword_list_t, 1068

get_values
 AuditoryProfile::fmap_t, 325

get_var
 algo_comm_t, 284
 MHAKernel::algo_comm_class_t, 987
 testplugin::ac_parser_t, 1476

get_var_double
 algo_comm_t, 285
 MHAKernel::algo_comm_class_t, 988

get_var_float
 algo_comm_t, 285
 Communication between algorithms, 26
 MHAKernel::algo_comm_class_t, 988

get_var_int
 algo_comm_t, 284
 Communication between algorithms, 26
 MHAKernel::algo_comm_class_t, 987

get_var_spectrum
 Communication between algorithms, 25

get_var_vfloat
 Communication between algorithms, 27

get_var_waveform
 Communication between algorithms, 25

get_varname
 plugins::hoertech::acrec::acwriter_t, 1382

get_vF
 DynComp::gaintable_t, 460

get_vL
 DynComp::gaintable_t, 460

get_weights
 MHAFilter::complex_bandpass_t, 873
 MHAFilter::gamma_flt_t, 893, 894

get_window
 MHAParser::window_t, 1135, 1136

get_window_data
 windowselector_t, 1514

get_write_event
 MHA_TCP::Connection, 806

get_write_ptr
 mha_fifo_t< T >, 780

get_xlimits
 MHATableLookup::xy_table_t, 1285

get_xruns
 MHAJack::client_t, 974

get_xruns_reset
 MHAJack::client_t, 975

get_zeropadding

wave2spec_t, 1494

getcipd
 coherence, 84

getdata
 MHASignal::uint_vector_t, 1258

getfullname
 PluginLoader::mhaplugloader_t, 1369

getmodulename
 dynamiclib_t, 448

Getmsg
 mha_error.hh, 1588

getname
 dynamiclib_t, 448
 MHA_AC::ac2matrix_t, 723

getusername
 MHA_AC::ac2matrix_t, 723

getvar
 acmon::ac_monitor_t, 205
 MHA_AC::ac2matrix_helper_t, 721

GF
 MHAFilter::gamma_flt_t, 894

gf
 gtfb_simple_rt_t, 568

gf_internals
 gtfb_simple_t, 573

GITCOMMITHASH
 mha_git_commit_hash.cpp, 1596

GLR
 smooth_cepstrum::smooth_cepstrum_t,
 1444

GLRDebug
 noise_psd_estimator::noise_psd_estimator_t,
 1313

GLRexp
 noise_psd_estimator::noise_psd_estimator_t,
 1314
 smooth_cepstrum::smooth_cepstrum_t,
 1444

Grad
 gsc_adaptive_stage::gsc_adaptive_stage,
 542

grad
 gsc_adaptive_stage::gsc_adaptive_stage,
 542

GREETING_TEXT
 mhamain.cpp, 1685

groupdelay
 delaysum_spec::delaysum_spec_if_t, 418

groupdelay_t
 MHASignal::schroeder_t, 1241

gsc_adaptive_stage, 95

DELT, 96
gsc_adaptive_stage::gsc_adaptive_stage, 537
gsc_adaptive_stage.cpp, 1553
gsc_adaptive_stage.hh, 1553
gsc_adaptive_stage::gsc_adaptive_stage, 536
~gsc_adaptive_stage, 538
ac, 539
alp, 541
bufSize, 539
d, 542
desired_chan, 540
doCircularComp, 540
E, 542
e, 542
E2, 542
e_out, 543
frac_old, 539
Grad, 542
grad, 542
gsc_adaptive_stage, 537
insert, 539
lenNewSamps, 539
lenOldSamps, 539
mha_fft, 540
mu, 540
nchan, 540
nfreq, 540
P, 543
process, 538
Psum, 543
useVAD, 541
vadName, 541
W, 541
X, 541
x, 541
Y, 541
y, 542
gsc_adaptive_stage::gsc_adaptive_stage_if, 544
~gsc_adaptive_stage_if, 545
alp, 548
doCircularComp, 547
gsc_adaptive_stage_if, 545
lenOldSamps, 547
mu, 547
on_model_param_valuechanged, 547
patchbay, 547
prepare, 546
process, 546
release, 546
update_cfg, 547
useVAD, 548
vadName, 548
gsc_adaptive_stage_if
gsc_adaptive_stage::gsc_adaptive_stage_if, 545
gsc_adaptive_stage_if.cpp, 1553
gsc_adaptive_stage_if.hh, 1553
gt
dc::dc_t, 386
gtdata
dc::dc_vars_t, 389
gtfb_analyzer, 96
gtfb_analyzer.cpp, 1553
filter_complex, 1554
filter_real, 1555
gtfb_analyzer::gtfb_analyzer_cfg_t, 548
~gtfb_analyzer_cfg_t, 550
bands, 550
channels, 550
coeff, 551
cvalue, 551
frames, 550
gtfb_analyzer_cfg_t, 549
norm_phase, 551
order, 551
s_out, 551
state, 552
states, 550
gtfb_analyzer::gtfb_analyzer_t, 552
coeff, 555
gtfb_analyzer_t, 554
norm_phase, 555
order, 555
patchbay, 554
prepare, 554
prepared, 555
process, 554
update_cfg, 554
gtfb_analyzer_cfg_t
gtfb_analyzer::gtfb_analyzer_cfg_t, 549
gtfb_analyzer_t
gtfb_analyzer::gtfb_analyzer_t, 554
gtfb_simd.cpp, 1555
add4f, 1557
check_alignment, 1557
filter_simd, 1559
filter_sisd_complex, 1558
filter_sisd_real, 1558
mul4f, 1557

MXCSR_DAZ, 1557
 MXCSR_FTZ, 1557
 sub4f, 1557
gtfb_simd_cfg_t, 555
 ~gtfb_simd_cfg_t, 557
 bands, 558
 bandsXchannels, 558
 channels, 558
 get_bands, 557
 get_channels, 557
 get_frames, 557
gtfb_simd_cfg_t, 556, 557
 icoefficients, 559
 iiinputs, 559
 istates, 559
 large_array, 560
 norm_phase, 559
 operator=, 558
 order, 558
 process, 558
 rcoefficients, 559
 rinputs, 559
 rstates, 559
 s_out, 560
 sout_buf, 560
gtfb_simd_t, 561
 coeff, 563
gtfb_simd_t, 562
 norm_phase, 563
 order, 563
 patchbay, 562
 prepare, 562
 prepared, 562
 process, 562
 update_cfg, 562
gtfb_simple_bridge.cpp, 1561
gtfb_simple_rt_t, 563
 _ac, 568
 _order, 566
 _pre_stages, 567
 ac_resynthesis_gain, 568
 acbw, 567
 accf, 567
 cLTASS, 568
 duplicate_vector, 566
 element_gain_name_, 568
 get_gf, 566
 gf, 568
gtfb_simple_rt_t, 564
 imag, 567
 input, 567
 nbands, 567
 output, 567
 post_plugin, 565
 pre_plugin, 565
gtfb_simple_t, 569
 cLTASS, 573
 desired_delay, 572
 element_gain_name, 572
 fscale, 572
 gf_internals, 573
gtfb_simple_t, 570
 name_, 573
 order, 572
 plug, 572
 prepare, 571
 prestages, 572
 process, 571
 release, 571
 resynthesis_gain, 573
 setlock, 571
gtmin
 dc::dc_vars_t, 389
gtstep
 dc::dc_vars_t, 389
h
 dynamiclib_t, 449
 MHASignal::hilbert_t, 1215
H_ERRNO
 MHA_TCP, 102
hamming
 MHAWindow, 155
hamming_t
 MHAWindow::hamming_t, 1293
handle
 algo_comm_t, 281
handler
 osc_variable_t, 1325, 1326
hann
 hann.cpp, 1562
 hann.h, 1562
 MHAOvlFilter::ShapeFun, 121
hann.cpp, 1561
 hann, 1562
 hannf, 1562
 PI, 1562
hann.h, 1562
 hann, 1562
 hannf, 1562
hann1
 plingploing::plingploing_t, 1345
hann2

plingploing::plingploing_t, 1345
hannf
 hann.cpp, 1562
 hann.h, 1562
hanning
 MHAWindow, 155
hanning_ramps_t, 573
 ~hanning_ramps_t, 574
 hanning_ramps_t, 574
 len_a, 574
 len_b, 574
 operator(), 574
 ramp_a, 574
 ramp_b, 575
hanning_t
 MHAWindow::hanning_t, 1294
hardlimit
 softclipper_t, 1457
 softclipper_variables_t, 1459
has_modified
 dc_simple::dc_if_t, 395
has_entry
 MHAParser::parser_t, 1103
has_inner_error
 analysepath_t, 309
has_key
 MHAKernel::comm_var_map_t, 990
has_parser
 io_wrapper, 642
 plug_wrapper, 1348
 plug_wrapperl, 1350
 PluginLoader::mhaplugloader_t, 1368
has_process
 io_wrapper, 643
 plug_wrapper, 1349
 plug_wrapperl, 1351
 PluginLoader::mhaplugloader_t, 1368
head_model_sphere_radius_cm
 rohBeam::rohBeam, 1399
header
 io_asterisk_sound_t, 598
 io_tcp_sound_t, 634
headModel
 rohBeam::rohConfig, 1405
help
 MHAParser::base_t, 1038
HELP_TEXT
 mhamain.cpp, 1685
hhCorrXpXp
 rohBeam::rohConfig, 1407
hifftwin
 doasvm_feature_extraction_config, 431
 hifftwin_sum
 doasvm_feature_extraction_config, 431
 high_side_flat
 MHAOvlFilter::band_descriptor_t, 998
 hilbert
 MHASignal::hilbert_fftw_t, 1213
 hilbert_fftw_t
 MHASignal::hilbert_fftw_t, 1212
 hilbert_shifter_t
 fshift_hilbert::hilbert_shifter_t, 518
 hilbert_t
 MHASignal::hilbert_t, 1215
HINSTANCE
 mha_plugin.hh, 1617
HL
 rmslevel, 159
host
 ac2osc_t, 183
 osc2ac_t, 1320
host_port_to_sock_addr
 mha_tcp.cpp, 1639
hostApi
 MHAIOPortAudio::device_info_t, 955
Hs
 MHAFilter::fftfilterbank_t, 885
HSTRERROR
 MHA_TCP, 102
HTL
 AuditoryProfile::parser_t::ear_t, 329
 AuditoryProfile::profile_t::ear_t, 333
hton
 io_asterisk_sound_t, 598
 io_tcp_sound_t, 635
hw
 MHAFilter::fftfilterbank_t, 885
hwin
 doasvm_feature_extraction_config, 431
hz2bark
 MHAOvlFilter::FreqScaleFun, 118
hz2bark_analytic
 MHAOvlFilter::FreqScaleFun, 119
hz2bark_t
 MHAOvlFilter::barkscale::hz2bark_t, 1000
hz2erb
 MHAOvlFilter::FreqScaleFun, 119
hz2erb_glasberg1990
 MHAOvlFilter::FreqScaleFun, 119
hz2hz
 MHAOvlFilter::FreqScaleFun, 118
 speechnoise.cpp, 1699

hz2khz
 MHAOvlFilter::FreqScaleFun, 118

hz2log
 MHAOvlFilter::FreqScaleFun, 119

hz2octave
 MHAOvlFilter::FreqScaleFun, 118

hz2third_octave
 MHAOvlFilter::FreqScaleFun, 118

hz2unit
 MHAOvlFilter::scale_var_t, 1025

i
 io_tcp_sound_t::float_union, 637

icoefficients
 gtfb_simd_cfg_t, 559

id
 equalize::freqgains_t, 465
 mha_channel_info_t, 744

id_str
 MHAParser::base_t, 1038

id_string
 MHAParser::parser_t, 1103

identity
 MHASignal::schroeder_t, 1243

identity.cpp, 1563

identity_t, 575
 identity_t, 576
 prepare, 576
 process, 576
 release, 576

idstr
 mha_channel_info_t, 744

idx
 level_matching::channel_pair, 649

if_t
 plingploing::if_t, 1338
 testplugin::if_t, 1481
 windnoise::if_t, 1509

iface
 MHA_TCP::Server, 817

ifft
 doasvm_feature_extraction_config, 431

ifftshift
 ifftshift.cpp, 1563
 ifftshift.h, 1563

ifftshift.cpp, 1563
 ifftshift, 1563

ifftshift.h, 1563
 ifftshift, 1563

Ignore
 lsl2ac, 97

ignore
 MHA_TCP::Event_Watcher, 812

ignored_by
 MHA_TCP::Wakeup_Event, 836

iinputs
 gtfb_simd_cfg_t, 559

iir_filter_state_t
 MHAFilter::iir_filter_state_t, 896

iir_filter_t
 MHAFilter::iir_filter_t, 897

iir_ord1_real_t
 MHAFilter::iir_ord1_real_t, 901

iirfilter.cpp, 1563

iirfilter_t, 577
 iirfilter_t, 577
 prepare_, 578
 process, 578
 release_, 578

ilen
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 320
 fader_wave::level_adapt_t, 492

im
 mha_complex_t, 745

imag
 acsave::mat4head_t, 225
 fftfilterbank::fftfb_plug_t, 508
 gtfb_simple_rt_t, 567
 MHASignal::matrix_t, 1228–1230

imagfb
 MHAOvlFilter::overlap_save_filterbank_analytic_t, 1019

impulse_response
 MHAFilter::polyphase_resampling_t, 925
 MHAFilter::transfer_function_t, 938

in_buf
 wave2spec_t, 1496

in_cfg
 rohBeam::rohConfig, 1405
 smooth_cepstrum::smooth_params, 1446

in_spec
 doasvm_feature_extraction_config, 432
 shadowfilter_end::cfg_t, 1425

in_spec_copy
 shadowfilter_begin::cfg_t, 1421

inbuf
 MHA_TCP::Connection, 809

inch
 mconv::MConv, 719
 MHAJack::client_t, 979

increase_condition
 mha_fifo_posix_threads_t, 775

increment 1157
 mha_fifo_posix_threads_t, 774
 mha_fifo_thread_platform_t, 786
index
 MHAParser::keyword_list_t, 1069
index_t
 MHAFilter::partitioned_convolution_t::index_itput
 919, 920
info
 ac2lsl::save_var_base_t, 170
 ac2lsl::save_var_t< mha_complex_t >, 177
 177
 ac2lsl::save_var_t< T >, 174
 lsl2ac::save_var_t, 688
init_dynamic
 rohBeam::rohConfig, 1404
Init_mha_ruby
 mha_ruby.cpp, 1622
init_peer_data
 MHA_TCP::Connection, 804
initialize
 MHA_TCP::Server, 816
inner2outer_resampling
 MHAPPlugin_Resampling::resampling_t, 1157
inner_ac_copy
 analysepath_t, 308
inner_error
 analysepath_t, 308
 mha_dblbuf_t< FIFO >, 753
inner_fragsize
 MHAPPlugin_Resampling::resampling_t, 1156
inner_in
 MHASignal::doublebuffer_t, 1206
inner_input
 analysepath_t, 308
 dbasync_native::dbasync_t, 379
inner_out
 MHASignal::doublebuffer_t, 1206
inner_out_domain
 analysepath_t, 308
inner_process
 db_t, 373
 MHASignal::doublebuffer_t, 1205
inner_process_wave2spec
 analysepath_t, 307
inner_process_wave2wave
 analysepath_t, 307
inner_signal
 MHAPPlugin_Resampling::resampling_t, 1157
inner_size
 mha_dblbuf_t< FIFO >, 752
inner_srata
 MHAPPlugin_Resampling::resampling_t, 1156
 1156
 ac_proc::interface_t, 198
 gtfb_simple_rt_t, 567
 io_parser_t, 619
 mha_dblbuf_t< FIFO >, 751
 MHAJack::port_t, 981
 MHASignal::loop_wavefragment_t, 1218
input_cfg
 MHAPPlugin::plugin_t< runtime_cfg_t >, 1150
input_cfg_<
 MHAPPlugin::plugin_t< runtime_cfg_t >, 1151
input_channels
 adm_if_t, 275
 mha_dblbuf_t< FIFO >, 753
input_domain
 PluginLoader::mhaplugloader_t, 1368
input_fifo
 mha_dblbuf_t< FIFO >, 753
input_level
 dc::dc_vars_t, 391
input_portnames
 MHAJack::client_t, 980
input_signal_spec
 MHAFilter::partitioned_convolution_t, 917
input_signal_wave
 MHAFilter::partitioned_convolution_t, 917
input_spec
 testplugin::signal_parser_t, 1484
input_to_process
 analysepath_t, 309
input_wave
 testplugin::signal_parser_t, 1484
inputchannels
 MHAFilter::fftfilterbank_t, 885
inputPow
 noise_psd_estimator::noise_psd_estimator_t, 1313
inputSpec
 noise_psd_estimator::noise_psd_estimator_t, 1314
insert
 acPooling_wave_config, 216
 acSteer_config, 233

adaptive_feedback_canceller_config, 246
 coherence::cohflt_t, 351
 fftfilterbank::fftfb_plug_t, 507
 gsc_adaptive_stage::gsc_adaptive_stage, 539
 lpc_config, 676
 MHA_AC::ac2matrix_t, 724
 MHA_AC::acspace2matrix_t, 727
 MHA_AC::double_t, 729
 MHA_AC::float_t, 731
 MHA_AC::int_t, 733
 MHA_AC::spectrum_t, 736
 MHA_AC::stat_t, 737
 MHA_AC::waveform_t, 740
 MHAoVlFilter::fftfb_ac_info_t, 1001
 multibandcompressor::fftfb_plug_t, 1299
 noise_psd_estimator::noise_psd_estimator_t, 1312
 rmslevel::rmslevel_t, 1392
 rt_nlms_t, 1414
 windnoise::if_t, 1510
 insert_config_vars
 matlab_wrapper::matlab_wrapper_t, 700
 insert_item
 MHAParser::parser_t, 1099
 insert_items
 windowselector_t, 1514
 insert_member
 mha_parser.hh, 1615
 insert_monitors
 matlab_wrapper::matlab_wrapper_t, 700
INSERT_PATCH
 acConcat_wave.cpp, 1519
 acPooling_wave.cpp, 1520
 acSteer.cpp, 1523
 acTransform_wave.cpp, 1524
 adaptive_feedback_canceller.cpp, 1524
 doasvm_classification.cpp, 1541
 doasvm_feature_extraction.cpp, 1542
 level_matching.cpp, 1564
 lpc.cpp, 1565
 lpc_bl_predictor.cpp, 1566
 lpc_burg-lattice.cpp, 1567
 smooth_cepstrum.cpp, 1696
 steerbf.cpp, 1703
INSERT_VAR
 smooth_cepstrum.cpp, 1696
 insert_var
 algo_comm_t, 281
 MHAKernel::algo_comm_class_t, 986
 testplugin::ac_parser_t, 1476
 insert_var_double
 algo_comm_t, 282
 MHAKernel::algo_comm_class_t, 987
 insert_var_float
 algo_comm_t, 282
 MHAKernel::algo_comm_class_t, 986
 insert_var_int
 algo_comm_t, 281
 MHAKernel::algo_comm_class_t, 986
 insert_var_vfloat
 MHAKernel::algo_comm_class_t, 986
 insert_variable
 osc_server_t, 1322
 insert_vars
 lsl2ac::save_var_t, 689
 inspect
 MHAFilter::complex_bandpass_t, 874
 MHAFilter::gamma_flt_t, 894
 MHASignal::delay_t, 1201
 inst_name
 fw_t, 529
 int_data
 testplugin::ac_parser_t, 1476
 int_mon_t
 MHAParser::int_mon_t, 1062
 int_t
 MHA_AC::int_t, 733
 MHAParser::int_t, 1065
 integrate
 Vector and matrix processing toolbox, 43
 intensity
 mha_signal.cpp, 1625
 interface_t
 ac_proc::interface_t, 197
 delay::interface_t, 410
 fftfilter::interface_t, 501
 multibandcompressor::interface_t, 1301
 route::interface_t, 1409
 interleaved
 combc_if_t, 358
 interleaved_
 combc_t, 360
 intermic_distance_cm
 rohBeam::rohBeam, 1399
 intern_level
 MHASignal::loop_wavefragment_t, 1221
 internal_fir
 MHAFilter::smoothspec_t, 931
 internal_start
 MHAJack::client_t, 976
 internal_stop

MHAJack::client_t, 976
interp
 MHATableLookup::linear_table_t, 1276
 MHATableLookup::table_t, 1280
 MHATableLookup::xy_table_t, 1283
interp1
 DynComp, 90
interp2
 DynComp, 91
inv_scale
 MHAOvlFilter::FreqScaleFun, 120
INVALID_SOCKET
 mha_tcp.cpp, 1638
INVALID_THREAD_PRIORITY
 dbasync_native, 86
 MHAPlugin_Split, 136
invalidate_window_data
 windowselector_t, 1515
invert
 coherence::vars_t, 355
invgain
 alsa_t< T >, 294
inwave
 lpc_config, 677
io
 MHAJack, 112
 MHAJack::client_avg_t, 964
 MHAJack::client_noncont_t, 968
io_alsa_t, 578
 alsa_start_counter, 583
 b_process, 581
 dev_in, 582
 dev_out, 582
 format, 583
 fw fragsize, 581
 fw samplerate, 581
 io_alsa_t, 580
 p_in, 583
 p_out, 583
 patchbay, 583
 pcmlink, 583
 prepare, 580, 581
 priority, 583
 proc_event, 582
 proc_handle, 582
 proc_thread, 582
 process, 581
 release, 580
 start, 580
 start_event, 582
 start_handle, 582
 stop, 581
 stop_event, 582
 stop_handle, 582
 thread_start, 581
io_asterisk_fwcbs_t, 583
 ~io_asterisk_fwcbs_t, 585
 io_asterisk_fwcbs_t, 584
 io_err, 588
 proc_err, 587
 proc_event, 587
 proc_handle, 587
 process, 585
 set_errno, 586
 start, 585
 start_event, 587
 start_handle, 587
 stop, 586
 stop_event, 587
 stop_handle, 587
io_asterisk_parser_t, 588
 ~io_asterisk_parser_t, 590
 connected, 594
 debug, 594
 debug_file, 595
 debug_filename, 595
 get_connected, 592
 get_local_address, 590
 get_local_port, 590
 get_server_port_open, 591
 io_asterisk_parser_t, 590
 local_address, 594
 local_port, 594
 peer_address, 595
 peer_port, 595
 server_port_open, 594
 set_connected, 592
 set_local_port, 591
 set_new_peer, 593
 set_server_port_open, 591
io_asterisk_sound_t, 595
 ~io_asterisk_sound_t, 597
 chunkbytes_in, 597
 fragsize, 598
 header, 598
 hton, 598
 io_asterisk_sound_t, 596
 ntoh, 598
 num_inchannels, 599
 num_outchannels, 599
 output_data, 599
 prepare, 597

release, 597
 s_in, 599
 samplerate, 598
io_asterisk_t, 599
 ~io_asterisk_t, 601
 accept_loop, 602
 connection_loop, 602
 fwcb, 603
 io_asterisk_t, 600
 notify_release, 603
 notify_start, 603
 notify_stop, 603
 parse, 602
 parser, 602
 prepare, 601
 release, 601
 server, 603
 sound, 602
 start, 601
 stop, 601
 thread, 603
io_context
 mha_tcp::server_t, 823
io_err
 io_asterisk_fwcbs_t, 588
 io_tcp_fwcbs_t, 624
io_error
 fw_t, 530
IO_ERROR_JACK
 mhajack.h, 1684
IO_ERROR_MHAJACKLIB
 mhajack.h, 1684
io_file_t, 604
 ~io_file_t, 606
 b_prepared, 609
 filename_input, 608
 filename_output, 608
 fragsize, 607
io_file_t, 605
 length, 609
 nchannels_file_in, 607
 nchannels_in, 607
 nchannels_out, 607
 output_sample_format, 608
 prepare, 606
 proc_event, 608
 proc_handle, 608
 release, 606
 s_file_in, 609
 s_in, 609
 s_out, 609
 samplerate, 607
 setlock, 607
 sf_in, 609
 sf_out, 610
 sfinf_in, 610
 sfinf_out, 610
 start, 606
 start_event, 608
 start_handle, 608
 startsample, 609
 stop, 606
 stop_event, 608
 stop_handle, 608
 stopped, 606
 strict_channel_match, 609
 strict_srate_match, 609
 total_read, 610
io_jack_t
 MHAIOJack::io_jack_t, 941
 MHAIOJackdb::io_jack_t, 948
io_lib
 fw_t, 529
io_lib_t, 610
 ~io_lib_t, 612
 io_lib_t, 612
 IODestroy_cb, 614
 IOInit_cb, 614
 IOPrepare_cb, 614
 IOResume_cb, 614
 IOSetVar_cb, 614
 IOStart_cb, 614
 IOStop_cb, 614
 IOStrError_cb, 614
 lib_data, 613
 lib_err, 613
 lib_handle, 613
 lib_str_error, 613
 prepare, 612
 release, 613
 start, 613
 stop, 613
 test_error, 613
io_name
 fw_t, 528
io_parser_t, 615
 ~io_parser_t, 616
 b_fw_started, 619
 b_prepared, 619
 b_starting, 619
 b_stopped, 619
 fragsize, 618

input, 619
io_parser_t, 616
nchannels_in, 618
nchannels_out, 618
output, 619
patchbay, 619
prepare, 617
proc_event, 618
proc_handle, 618
process_frame, 617
release, 617
s_in, 619
s_out, 619
start, 617
start_event, 618
start_handle, 618
started, 617
stop, 617
stop_event, 618
stop_handle, 618
stopped, 617
io_portaudio_t
 MHAIOPortAudio::io_portaudio_t, 958
io_tcp_fwcb_t, 620
 ~io_tcp_fwcb_t, 621
io_err, 624
io_tcp_fwcb_t, 621
proc_err, 623
proc_event, 623
proc_handle, 623
process, 621
set_ernos, 622
start, 621
start_event, 623
start_handle, 623
stop, 622
stop_event, 623
stop_handle, 623
io_tcp_parser_t, 624
 ~io_tcp_parser_t, 626
connected, 630
debug, 630
debug_file, 631
debug_filename, 631
get_connected, 628
get_local_address, 626
get_local_port, 626
get_server_port_open, 627
io_tcp_parser_t, 626
local_address, 630
local_port, 630
peer_address, 631
peer_port, 631
server_port_open, 630
set_connected, 628
set_local_port, 627
set_new_peer, 629
set_server_port_open, 627
io_tcp_sound_t, 631
 ~io_tcp_sound_t, 633
check_sound_data_type, 633
chunkbytes_in, 634
fragsize, 635
header, 634
hton, 635
io_tcp_sound_t, 633
ntoh, 634
num_inchannels, 636
num_outchannels, 636
prepare, 634
release, 634
s_in, 636
samplerate, 635
io_tcp_sound_t::float_union, 636
c, 637
f, 637
i, 637
io_tcp_t, 637
 ~io_tcp_t, 638
accept_loop, 639
connection_loop, 639
fwcb, 640
io_tcp_t, 638
notify_release, 641
notify_start, 640
notify_stop, 641
parse, 640
parser, 640
prepare, 638
release, 639
server, 640
sound, 640
start, 639
stop, 639
thread, 640
io_wrapper, 641
 ~io_wrapper, 642
get_categories, 642
get_documentation, 643
has_parser, 642
has_process, 643
io_wrapper, 642

parse, 642

iob
 MHAJack::port_t, 984

IODestroy
 MHAIOalsa.cpp, 1647, 1649
 MHAIOAsterisk.cpp, 1652, 1654
 MHAIOFile.cpp, 1657, 1658
 MHAIOJack.cpp, 1661, 1662
 MHAIOJackdb.cpp, 1665, 1666
 MHAIOParser.cpp, 1669, 1670
 MHAIOPortAudio.cpp, 1673, 1675
 MHAIOTCP.cpp, 1679, 1681

IODestroy_cb
 io_lib_t, 614

IODestroy_t
 mha_io_ifc.h, 1599

IOInit
 MHAIOalsa.cpp, 1646, 1647
 MHAIOAsterisk.cpp, 1651, 1653
 MHAIOFile.cpp, 1656, 1657
 MHAIOJack.cpp, 1660, 1661
 MHAIOJackdb.cpp, 1664, 1665
 MHAIOParser.cpp, 1668, 1669
 MHAIOPortAudio.cpp, 1672, 1674
 MHAIOTCP.cpp, 1678, 1680

IOInit_cb
 io_lib_t, 614

IOInit_t
 mha_io_ifc.h, 1598

IOPrepare
 MHAIOalsa.cpp, 1646, 1648
 MHAIOAsterisk.cpp, 1652, 1653
 MHAIOFile.cpp, 1656, 1657
 MHAIOJack.cpp, 1660, 1661
 MHAIOJackdb.cpp, 1664, 1665
 MHAIOParser.cpp, 1668, 1669
 MHAIOPortAudio.cpp, 1673, 1674
 MHAIOTCP.cpp, 1678, 1680

IOPrepare_cb
 io_lib_t, 614

IOPrepare_t
 mha_io_ifc.h, 1598

IOProcessEvent_inner
 MHAIOJackdb::io_jack_t, 948

IOProcessEvent_t
 mha_io_ifc.h, 1598

IOResume
 MHAIOalsa.cpp, 1646, 1648
 MHAIOAsterisk.cpp, 1652, 1653
 MHAIOFile.cpp, 1656, 1658
 MHAIOJack.cpp, 1661, 1662

MHAIOJackdb.cpp, 1665, 1666

MHAIOParser.cpp, 1669, 1670

MHAIOPortAudio.cpp, 1673, 1674

MHAIOTCP.cpp, 1679, 1680

IOResume_cb
 io_lib_t, 614

IOResume_t
 mha_io_ifc.h, 1599

IOSetVar
 MHAIOalsa.cpp, 1647, 1648
 MHAIOAsterisk.cpp, 1652, 1654
 MHAIOFile.cpp, 1656, 1658
 MHAIOJack.cpp, 1661, 1662
 MHAIOJackdb.cpp, 1665, 1666
 MHAIOParser.cpp, 1669, 1670
 MHAIOPortAudio.cpp, 1673, 1675
 MHAIOTCP.cpp, 1679, 1680

IOSetVar_cb
 io_lib_t, 614

IOSetVar_t
 mha_io_ifc.h, 1599

IOStart
 MHAIOalsa.cpp, 1646, 1648
 MHAIOAsterisk.cpp, 1652, 1653
 MHAIOFile.cpp, 1656, 1657
 MHAIOJack.cpp, 1660, 1662
 MHAIOJackdb.cpp, 1664, 1666
 MHAIOParser.cpp, 1668, 1670
 MHAIOPortAudio.cpp, 1673, 1674
 MHAIOTCP.cpp, 1678, 1680

IOStart_cb
 io_lib_t, 614

IOStart_t
 mha_io_ifc.h, 1598

IOStartedEvent_t
 mha_io_ifc.h, 1598

IOStop
 MHAIOalsa.cpp, 1646, 1648
 MHAIOAsterisk.cpp, 1652, 1653
 MHAIOFile.cpp, 1656, 1658
 MHAIOJack.cpp, 1660, 1662
 MHAIOJackdb.cpp, 1664, 1666
 MHAIOParser.cpp, 1668, 1670
 MHAIOPortAudio.cpp, 1673, 1674
 MHAIOTCP.cpp, 1679, 1680

IOStop_cb
 io_lib_t, 614

IOStop_t
 mha_io_ifc.h, 1599

IOStoppedEvent
 MHAJack::client_avg_t, 965

MHAJack::client_noncont_t, 969
IOStoppedEvent_t
 mha_io_ifc.h, 1598
IOSError
 MHAIOalsa.cpp, 1647, 1648
 MHAIOAsterisk.cpp, 1652, 1654
 MHAIOFile.cpp, 1657, 1658
 MHAIOJack.cpp, 1661, 1662
 MHAIOJackdb.cpp, 1665, 1666
 MHAIOParser.cpp, 1669, 1670
 MHAIOPortAudio.cpp, 1673, 1675
 MHAIOTCP.cpp, 1679, 1681
IOSError_cb
 io_lib_t, 614
IOSError_t
 mha_io_ifc.h, 1599
irs
 fftfilter::interface_t, 502
 mconv::MConv, 719
irs_length
 fftfilter, 93
irs_validator
 fftfilter, 93
irslen
 fftfilter::fftfilter_t, 499
 fshift_hilbert::frequency_translator_t, 517
irslen_inner2outer
 MHAParser_Resampling::resampling_if_t, 1154
irslen_outer2inner
 MHAParser_Resampling::resampling_if_t, 1154
irswnd
 MHAOvLFilter::overlap_save_filterbank_t::vars_t, 1023
 smoothgains_bridge::overlapadd_if_t, 1450
is_complex
 MHA_AC::ac2matrix_helper_t, 721
 mha_audio_descriptor_t, 741
 plugins::hoertech::acrec::acwriter_t, 1384
is_denormal
 MHAUtils, 151, 152
is_first_run
 ac2lsl::ac2lsl_t, 166
 ac2osc_t, 185
is_multiple_of
 MHAUtils, 151
is_multiple_of_by_power_of_two
 MHAUtils, 151
is_num_channels_known
 plugins::hoertech::acrec::acwriter_t, 1384
is_playback_active
 MHASignal::loop_wavefragment_t, 1220
is_power_of_two
 MHAUtils, 151
is_prepared
 adm_if_t, 273
 double2acvar::double2acvar_t, 435
 MHAJack::client_t, 976
 MHAParser::plugin_t< runtime_cfg_t >, 1150
 PluginLoader::mhaplugloader_t, 1370
is_prepared_<
 MHAParser::plugin_t< runtime_cfg_t >, 1151
is_running
 osc_server_t, 1323
is_same_size
 MHASignal::matrix_t, 1227
is_var
 algo_comm_t, 283
 MHAKernel::algo_comm_class_t, 987
iscomplex
 MHASignal::matrix_t, 1227
isempty
 AuditoryProfile::fmap_t, 326
 gaintable.cpp, 1550
 MHAFilter::transfer_function_t, 937
istates
 gtfb_simd_cfg_t, 559
isval
 MHAParser::kw_t, 1072
iter
 adaptive_feedback_canceller_config, 247
iterate_lnn
 audiometerbackend::lnn3rdoct_t, 322
iterator
 MHA_TCP::Event_Watcher, 811
j0
 rohBeam, 159
jack_error_handler
 mhajack.cpp, 1681
jack_proc_cb
 MHAJack::client_t, 977
jack_xrun_cb
 MHAJack::client_t, 977
jc
 MHAJack::client_t, 979
 MHAJack::port_t, 984
jstate_prev
 MHAJack::client_t, 979

k_inner
 MHASignal::doublebuffer_t, 1206

k_outer
 MHASignal::doublebuffer_t, 1207

kappa
 lpc_burglattice_config, 674

kappa_block
 lpc_burglattice_config, 674

kappa_const
 smooth_cepstrum::smooth_cepstrum_if_t,
 1436

 smooth_cepstrum::smooth_params, 1447

keyword_list_t
 MHParse::keyword_list_t, 1068

kick_condition
 MHAParser_Split::posix_threads_t, 1169

kick_thread
 MHAParser_Split::dummy_threads_t,
 1165

 MHAParser_Split::posix_threads_t, 1168

 MHAParser_Split::thread_platform_t,
 1186

kicked
 MHAParser_Split::posix_threads_t, 1170

km
 lpc_bl_predictor_config, 669

kmax
 fshift::fshift_config_t, 510

 fshift_hilbert::hilbert_shifter_t, 521

kmin
 fshift::fshift_config_t, 510

 fshift_hilbert::hilbert_shifter_t, 520

kth_smallest
 MHASignal, 144

kw_index2type
 transducers.cpp, 1705

kw_t
 MHParse::kw_t, 1071, 1072

L

 AuditoryProfile::parser_t, 327

 AuditoryProfile::profile_t, 332

l_new
 addsndfile::level_adapt_t, 256

 audiometerbackend::level_adapt_t, 320

 fader_wave::level_adapt_t, 492

l_old
 addsndfile::level_adapt_t, 257

 audiometerbackend::level_adapt_t, 320

 fader_wave::level_adapt_t, 493

lambda
 lpc_burglattice, 672

 lpc_burglattice_config, 674

lambda_ceps
 smooth_cepstrum::smooth_cepstrum_t,
 1443

lambda_ceps_prev
 smooth_cepstrum::smooth_cepstrum_t,
 1443

lambda_ml_ceps
 smooth_cepstrum::smooth_cepstrum_t,
 1442

lambda_ml_full
 smooth_cepstrum::smooth_cepstrum_t,
 1442

lambda_ml_smooth
 smooth_cepstrum::smooth_cepstrum_t,
 1442

lambda_smoothing_power
 nlms_t, 1308

lambda_spec
 smooth_cepstrum::smooth_cepstrum_t,
 1443

lambda_thresh
 smooth_cepstrum::smooth_cepstrum_if_t,
 1436

 smooth_cepstrum::smooth_params, 1446

large_array
 gtfb_simd_cfg_t, 560

last_complex_bin
 MHASignal::subsample_delay_t, 1254

last_errormsg
 MHParse::parser_t, 1103

last_jack_err
 mhajack.cpp, 1682

last_jack_err_msg
 mhajack.cpp, 1682

 mhajack.h, 1684

last_name
 MHParse::mhapluginloader_t, 1089

latex_doc_t, 643

 ac, 646

 get_ac, 645

 get_categories, 645

 get_latex_doc, 645

 get_main_category, 645

 get_parser_tab, 646

 get_parser_var, 646

 latex_doc_t, 644

 latex_pluginname, 646

 loader, 646

 parsername, 646

 plugin_macro, 647

pluginame, 646
 strdom, 645

latex_pluginame
 latex_doc_t, 646

len
 MHA_AC::acspace2matrix_t, 728
 MHAFilter::filter_t, 891
 MHATableLookup::linear_table_t, 1278
 plingploing::plingploing_t, 1343

len_A
 MHAFilter::filter_t, 890

len_a
 hanning_ramps_t, 574

len_B
 MHAFilter::filter_t, 890

len_b
 hanning_ramps_t, 574

length
 io_file_t, 609
 MHASignal::uint_vector_t, 1258

lenNewSamps
 gsc_adaptive_stage::gsc_adaptive_stage,
 539

lenOldSamps
 gsc_adaptive_stage::gsc_adaptive_stage,
 539
 gsc_adaptive_stage::gsc_adaptive_stage_if,
 547

lev
 noise_t, 1317
 sine_t, 1432

LEVEL
 dc_simple, 87

level
 addsndfile::addsndfile_if_t, 253
 audiometerbackend::audiometer_if_t, 317
 plingploing::if_t, 1339
 plingploing::plingploing_t, 1345

level_adapt_t
 addsndfile::level_adapt_t, 255
 audiometerbackend::level_adapt_t, 319
 fader_wave::level_adapt_t, 491

level_adaptor
 addsndfile, 80
 audiometerbackend, 83
 fader_wave, 92

level_in_db
 dc::dc_t, 387

level_in_db_adjusted
 dc::dc_t, 387

level_matching, 96

 level_matching.cpp, 1564
 INSERT_PATCH, 1564
 PATCH_VAR, 1564

 level_matching.hh, 1564

 level_matching::channel_pair, 647
 channel_pair, 647, 648
 get_idx, 649
 get_mismatch, 649
 idx, 649
 mismatch, 650
 update_mismatch, 648, 649

 level_matching::level_matching_config_t, 650
 ~level_matching_config_t, 651
 ffflen, 652
 level_matching_config_t, 651
 lp, 652
 pairings, 652
 process, 651, 652
 range, 652
 tmp, 652

 level_matching::level_matching_t, 653
 ~level_matching_t, 654
 channels, 656
 level_matching_t, 654
 lp_level_tau, 656
 lp_signal_fpass, 656
 lp_signal_fstop, 656
 lp_signal_order, 656
 patchbay, 656
 prepare, 655
 process, 654, 655
 range, 656
 release, 655
 update_cfg, 655

 level_matching_config_t
 level_matching::level_matching_config_t,
 651

 level_matching_t
 level_matching::level_matching_t, 654

 level_mode_t
 MHASignal::loop_wavefragment_t, 1217

 level_mon
 droptect_t, 442

 level_smoothen_t
 dc_simple::level_smoothen_t, 407

 level_spec
 dc_simple::level_smoothen_t, 408

 level_wave
 dc_simple::level_smoothen_t, 408

 levelmeter.cpp, 1564
 PASCAL, 1565

levelmeter_t, 657
 levelmeter_t, 658
 mode, 659
 patchbay, 659
 peak, 659
 prepare, 658
 process, 658
 query_peak, 659
 query_rms, 658
 rms, 659
 tau, 659
 update_tau, 658
 levelmode
 addsndfile::addsndfile_if_t, 253
 Levinson2
 lpc.cpp, 1566
 lib_data
 io_lib_t, 613
 PluginLoader::mhaplugloader_t, 1371
 lib_err
 io_lib_t, 613
 PluginLoader::mhaplugloader_t, 1371
 lib_handle
 io_lib_t, 613
 PluginLoader::mhaplugloader_t, 1371
 lib_str_error
 io_lib_t, 613
 libdata
 analysepath_t, 308
 MHAParser::c_ifc_parser_t, 1047
 liberr
 MHAParser::c_ifc_parser_t, 1047
 libname
 analysispath_if_t, 312
 PluginLoader::config_file_splitter_t, 1361
 library_handle
 matlab_wrapper::matlab_wrapper_t::wrappedapi_t, 707
 library_name
 matlab_wrapper::matlab_wrapper_t, 701
 library_paths
 pluginbrowser_t, 1355
 like_ratio
 acPooling_wave_config, 217
 like_ratio_name
 acPooling_wave, 215
 limit
 coherence::cohflt_t, 352
 coherence::vars_t, 355
 MHASignal, 144
 MHASignal::quantizer_t, 1236
 MHASignal::waveform_t, 1269
 limiter
 dc_simple::dc_t, 399
 limiter_threshold
 dc_simple::dc_t, 399
 dc_simple::dc_vars_t, 403
 lin2db
 MHASignal, 139, 140
 line_t
 dc_simple::dc_t::line_t, 400, 401
 linear
 MHAOvIFilter::ShapeFun, 121
 softclipper_t, 1457
 softclipper_variables_t, 1459
 linear_table_t
 MHATableLookup::linear_table_t, 1276
 Linearphase_FIR
 ADM::Linearphase_FIR< F >, 269
 list_dir
 mha_os.cpp, 1602
 mha_os.h, 1606
 lnn3rdoct_t
 audiometerbackend::lnn3rdoct_t, 321
 lo_addr
 ac2osc_t, 185
 load_io_lib
 fw_t, 527
 load_lib
 dynamiclib_t, 448
 matlab_wrapper::matlab_wrapper_t, 700
 load_plug
 MHAParser::mhaplugloader_t, 1088
 load_proc_lib
 fw_t, 526
 loader
 latex_doc_t, 646
 mhaplug_t,
 analysispath_if_t, 312
 local_address
 io_asterisk_parser_t, 594
 io_tcp_parser_t, 630
 local_get_entries
 MHAKernel::algo_comm_class_t, 989
 local_get_var
 MHAKernel::algo_comm_class_t, 989
 local_insert_var
 MHAKernel::algo_comm_class_t, 988
 local_is_var
 MHAKernel::algo_comm_class_t, 989
 local_port
 io_asterisk_parser_t, 594

io_tcp_parser_t, 630
local_remove_ref
 MHAKernel::algo_comm_class_t, 988
local_remove_var
 MHAKernel::algo_comm_class_t, 988
locate
 MHAIQJackdb::io_jack_t, 952
locate_end
 MHASignal::loop_wavefragment_t, 1220
lock_channels
 fw_vars_t, 531
lock_srate_fragsize
 fw_vars_t, 531
locked
 MHAParser::variable_t, 1115
log_down
 MHASignal::schroeder_t, 1244
log_interp
 dc::dc_t, 387
 dc::dc_vars_t, 390
log_lambda_spec
 smooth_cepstrum::smooth_cepstrum_t,
 1443
log_up
 MHASignal::schroeder_t, 1243
logfile
 mhaserver_t, 1193
logGLRFact
 noise_psd_estimator::noise_psd_estimator_t,
 1314
 smooth_cepstrum::smooth_cepstrum_t,
 1444
logstring
 mhaserver_t, 1192
longmsg
 MHA_Error, 765
lookup
 MHATableLookup::linear_table_t, 1276
 MHATableLookup::table_t, 1280
 MHATableLookup::xy_table_t, 1282
loop
 addsndfile::addsndfile_if_t, 252
loop_wavefragment_t
 MHASignal::loop_wavefragment_t, 1218
lost
 osc_server_t, 1323
low_incl
 MHAParser::range_var_t, 1107
low_limit
 MHAParser::range_var_t, 1107
low_side_flat
 MHAOvIFilter::band_descriptor_t, 998
low_thresh
 acPooling_wave_config, 218
lower_threshold
 acPooling_wave, 214
lowpass_quotient
 windnoise::if_t, 1511
lowpass_quotient_acname
 windnoise::if_t, 1512
LowPassCutOffFrequency
 windnoise::if_t, 1511
LowPassFraction
 windnoise::cfg_t, 1507
 windnoise::if_t, 1511
LowPassWindGain
 windnoise::cfg_t, 1507
 windnoise::if_t, 1511
lp
 DynComp::dc_afterburn_rt_t, 451
 level_matching::level_matching_config_t,
 652
lp1i
 coherence::cohflt_t, 353
lp1tg
 coherence::cohflt_t, 353
lp1r
 coherence::cohflt_t, 352
lp_coeffs
 adm_rtconfig_t, 279
lp_level_tau
 level_matching::level_matching_t, 656
lp_order
 adm_if_t, 274
lp_signal_fpass
 level_matching::level_matching_t, 656
lp_signal_fstop
 level_matching::level_matching_t, 656
lp_signal_order
 level_matching::level_matching_t, 656
lpc, 660
 ~lpc, 661
 algo_name, 662
 comp_each_iter, 662
 lpc, 661
 lpc_buffer_size, 662
 lpc_order, 662
 norm, 663
 patchbay, 663
 prepare, 661
 process, 661
 release, 662

shift, 662
 update_cfg, 662
lpc.cpp, 1565
 INSERT_PATCH, 1565
 Levinson2, 1566
 PATCH_VAR, 1565
lpc.h, 1566
lpc_bl_predictor, 663
 ~lpc_bl_predictor, 664
 lpc_bl_predictor, 664
lpc_order, 666
 name_b, 666
 name_f, 666
 name_kappa, 666
 name_lpc_b, 666
 name_lpc_f, 666
 patchbay, 666
 prepare, 665
 process, 665
 release, 665
 update_cfg, 665
lpc_bl_predictor.cpp, 1566
 INSERT_PATCH, 1566
 PATCH_VAR, 1566
lpc_bl_predictor.h, 1567
 EPSILON, 1567
lpc_bl_predictor_config, 667
 ~lpc_bl_predictor_config, 667
 ac, 668
 b_est, 668
 backward, 668
 f_est, 668
 forward, 668
 km, 669
lpc_bl_predictor_config, 667
lpc_order, 668
 name_b, 669
 name_f, 668
 name_km, 668
 process, 667
 s_b, 669
 s_f, 669
lpc_buffer_size
 lpc, 662
 lpc_config, 677
lpc_burg-lattice.cpp, 1567
 INSERT_PATCH, 1567
 PATCH_VAR, 1567
lpc_burg-lattice.h, 1568
 EPSILON, 1568
lpc_burglattice, 669
 ~lpc_burglattice, 670
 lambda, 672
lpc_burglattice, 670
lpc_order, 672
 name_b, 672
 name_f, 672
 name_kappa, 672
 patchbay, 672
 prepare, 671
 process, 671
 release, 671
 update_cfg, 671
lpc_burglattice_config, 672
 ~lpc_burglattice_config, 673
 ac, 673
 backward, 674
 dm, 674
 forward, 674
 kappa, 674
 kappa_block, 674
 lambda, 674
lpc_burglattice_config, 673
lpc_order, 674
 name_b, 675
 name_f, 674
 nm, 674
 process, 673
 s_b, 675
 s_f, 675
lpc_config, 675
 ~lpc_config, 676
 A, 677
 comp_each_iter, 677
 comp_iter, 677
 corr_out, 678
 insert, 676
 inwave, 677
lpc_buffer_size, 677
lpc_config, 676
lpc_out, 678
 N, 677
 norm, 676
 order, 677
 process, 676
 R, 677
 sample, 677
 shift, 676
lpc_order
 adaptive_feedback_canceller, 243
 lpc, 662
 lpc_bl_predictor, 666

lpc_bl_predictor_config, 668
lpc_burglattice, 672
lpc_burglattice_config, 674
lpc_out
 lpc_config, 678
lsl2ac, 96
 Discard, 97
 Ignore, 97
 overrun_behavior, 97
lsl2ac.cpp, 1568
lsl2ac.hh, 1568
lsl2ac::cfg_t, 678
 cfg_t, 679
 process, 679
 varlist, 679
lsl2ac::lsl2ac_t, 680
 activate, 683
 available_streams, 685
 buffersize, 684
 chunksize, 684
 get_all_stream_names, 683
lsl2ac_t, 682
 nchannels, 684
 nsamples, 684
 overrun_behavior, 684
patchbay, 684
prepare, 682
process, 682
release, 682
setlock, 683
streams, 683
update, 683
lsl2ac::save_var_t, 685
 ~save_var_t, 687
 ac, 690
 buf, 689
 bufsize, 692
 chunksize, 691
 cv, 690
 get_time_correction, 689
 info, 688
 insert_vars, 689
 n_new_samples, 692
 name, 691
 nchannels, 691
 new_name, 690
 nsamples, 691
 ob, 691
 operator=, 688
 pull_samples_discard, 688
 pull_samples_ignore, 688
receive_frame, 688
save_var_t, 687
skip, 691
stream, 689
tc, 690
tc_name, 690
tic, 691
ts, 690
ts_buf, 689
ts_name, 690
lsl2ac_t
 lsl2ac::lsl2ac_t, 682
LTASS_combined
 speechnoise_t, 1467
LTASS_female
 speechnoise_t, 1467
LTASS_male
 speechnoise_t, 1467
ltgcomp
 coherence::vars_t, 356
ltgtau
 coherence::vars_t, 356
lval
 MHAParser::expression_t, 1056
m
 dc_simple::dc_t::line_t, 401
 matrixmixer::cfg_t, 713
m_alpha
 ADM::Linearphase_FIR< F >, 270
m_beta
 ADM::ADM< F >, 264
m_coeff
 ADM::Delay< F >, 267
m_decomb
 ADM::ADM< F >, 264
m_delay_back
 ADM::ADM< F >, 264
m_delay_front
 ADM::ADM< F >, 264
m_df
 fshift::fshift_t, 514
m_fmax
 fshift::fshift_t, 514
m_fmin
 fshift::fshift_t, 513
m_fullsamples
 ADM::Delay< F >, 267
m_lp_bf
 ADM::ADM< F >, 264
m_lp_result
 ADM::ADM< F >, 264

m_mu_beta
ADM::ADM< F >, 264

m_norm
ADM::Delay< F >, 268

m_now
ADM::Linearphase_FIR< F >, 270

m_now_in
ADM::Delay< F >, 268

m_order
ADM::Linearphase_FIR< F >, 270

m_output
ADM::Linearphase_FIR< F >, 270

M_PI
mha_defs.h, 1583
mha_signal.hh, 1634

m_powerfilter_coeff
ADM::ADM< F >, 265

m_powerfilter_norm
ADM::ADM< F >, 265

m_powerfilter_state
ADM::ADM< F >, 265

m_state
ADM::Delay< F >, 268

magResp
rohBeam::rohConfig, 1407

main
analysemhaplugin.cpp, 1530
browsemhaplugins.cpp, 1533
generatemhaplugindoc.cpp, 1552
mha.cpp, 1570
MHAPlugIn_Split::posix_threads_t, 1169
testalsadevice.c, 1704

make_friendly_number
MHAFilter, 106
mhajack.cpp, 1682

make_friendly_number_by_limiting
adaptive_feedback_canceller.cpp, 1525
nlms_wave.cpp, 1690

mapping
addsndfile::addsndfile_if_t, 253
coherence::vars_t, 355

matlab_wrapper, 97

matlab_wrapper.cpp, 1569

matlab_wrapper.hh, 1569
MHAPLUGIN_OVERLOAD_OUTDOMAIN, 1569

matlab_wrapper::callback, 692
callback, 693
on_writeaccess, 693
parent, 694
user_config, 693

var, 694

matlab_wrapper::matlab_wrapper_rt_cfg_t, 694
~matlab_wrapper_rt_cfg_t, 695
matlab_wrapper_rt_cfg_t, 695
user_config, 695

matlab_wrapper::matlab_wrapper_t, 696
callback, 701
callbacks, 701
cb_patchbay, 701
insert_config_vars, 700
insert_monitors, 700
library_name, 701
load_lib, 700
matlab_wrapper_t, 698
monitors, 702
patchbay, 701
plug, 701
prepare, 699
process, 698, 699
release, 700
update, 700
update_monitors, 700
vars, 701

matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 702
~wrapped_plugin_t, 704
fcn_init, 707
fcn_prepare, 708
fcn_process_ss, 708
fcn_process_sw, 708
fcn_process_ws, 708
fcn_process_ww, 707
fcn_release, 708
fcn_terminate, 707
library_handle, 707
mha_spec_out, 710
mha_wave_out, 709
prepare, 706
process_ss, 705
process_sw, 706
process_ws, 705
process_ww, 704
release, 706
signal_dimensions, 709
spec_in, 709
spec_out, 709
state, 707
user_config, 707
wave_in, 709
wave_out, 709

wrapped_plugin_t, 704
matlab_wrapper::types< MHA_SPECTRUM >, 710
array_type, 710
class_signal_type, 711
signal_type, 710
matlab_wrapper::types< MHA_WAVEFORM >, 711
array, 711
class_signal_type, 711
signal_type, 711
matlab_wrapper::types< T >, 710
matlab_wrapper_rt_cfg_t
matlab_wrapper::matlab_wrapper_rt_cfg_t, MAX_USER_ERR
695
matlab_wrapper_t
matlab_wrapper::matlab_wrapper_t, 698
matmix_t
matrixmixer::matmix_t, 714
matrix_t
MHASignal::matrix_t, 1223, 1225
matrixmixer, 98
matrixmixer.cpp, 1569
matrixmixer::cfg_t, 712
cfg_t, 712
m, 713
process, 712
sout, 713
wout, 713
matrixmixer::matmix_t, 713
ci, 715
co, 715
matmix_t, 714
mixer, 715
patchbay, 715
prepare, 714
process, 714, 715
update_m, 715
MAX
mha_defs.h, 1583
max
spec2wave.cpp, 1697
Vector and matrix processing toolbox, 56
max_clipped
softclipper_variables_t, 1459
max_frames
acsave::cfg_t, 225
max_lag
doasvm_feature_extraction, 428
max_p_ind_name
doasvm_classification, 423
max_pool_ind_name
acPooling_wave, 215
max_q
smooth_cepstrum::smooth_cepstrum_t, 1444
max_sleep_time
dropgen_t, 438
MAX_TCP_PORT
MHAIOAsterisk.cpp, 1651
MHAIOTCP.cpp, 1678
MAX_TCP_PORT_STR
MHAIOAsterisk.cpp, 1651
MHAIOTCP.cpp, 1678
MAX_USER_ERR
MHAIOalsa.cpp, 1646
MHAIOAsterisk.cpp, 1651
MHAIOFile.cpp, 1656
MHAIOJack.cpp, 1660
MHAIOJackdb.cpp, 1664
MHAIOParser.cpp, 1668
MHAIOPortAudio.cpp, 1672
MHAIOTCP.cpp, 1677
mhajack.h, 1684
max_val
smooth_cepstrum::smooth_cepstrum_t, 1444
maxabs
Vector and matrix processing toolbox, 54, 55
maxframe
acsave::save_var_t, 228
maxgain
dc_simple::dc_t, 400
dc_simple::dc_vars_t, 403
DynComp::dc_afterburn_rt_t, 451
DynComp::dc_afterburn_vars_t, 456
maximum_reader_xruns_in_succession_before_stop
mha_drifter_fifo_t< T >, 762
maximum_writer_xruns_in_succession_before_stop
mha_drifter_fifo_t< T >, 762
maxInputChannels
MHAIOPortAudio::device_info_t, 955
maxlen
plingploing::if_t, 1340
maxlen_
plingploing::plingploing_t, 1343
maxLim
rohBeam::rohConfig, 1407
maxOutputChannels
MHAIOPortAudio::device_info_t, 955
mcomplex_mon_t

MHParse::mcomplex_mon_t, 1075
 mcomplex_t
 MHParse::mcomplex_t, 1077
 MConv
 mconv::MConv, 717
 mconv, 98
 mconv.cpp, 1570
 mconv::MConv, 716
 fragsize, 719
 inch, 719
 irs, 719
 MConv, 717
 nchannels_in, 719
 nchannels_out, 719
 outch, 719
 patchbay, 719
 prepare, 718
 process, 718
 release, 718
 update, 718
 update_irs, 718
 mean
 MHA_AC::stat_t, 737
 MHASignal, 147
 MHASignal::stat_t, 1251
 mean_std
 MHASignal::stat_t, 1251
 median
 MHASignal, 145
 mfloat_mon_t
 MHParse::mfloat_mon_t, 1079
 mfloat_t
 MHParse::mfloat_t, 1081
 mha
 mhahost_t::tcp_server_t, 1195
 spechnoise_t, 1467
 mha.cpp, 1570
 main, 1570
 mhamain, 1570
 mha.hh, 1570
 algo_comm_t, 1576
 MHA_AC_CHAR, 1575
 MHA_AC_DOUBLE, 1576
 MHA_AC_FLOAT, 1576
 MHA_AC_INT, 1575
 MHA_AC_MHACOMPLEX, 1576
 MHA_AC_MHAREAL, 1576
 MHA_AC_UNKNOWN, 1575
 MHA_AC_USER, 1576
 MHA_AC_VEC_FLOAT, 1576
 MHA_CALLBACK_TEST, 1573
 MHA_CALLBACK_TEST_PREFIX, 1573
 MHA_DOMAIN_MAX, 1575
 mha_domain_t, 1576
 MHA_DOMAIN_UNKNOWN, 1575
 MHA_RELEASE_VERSION_STRING,
 1575
 MHA_SPECTRUM, 1575
 MHA_STRF, 1573
 MHA_STRUCT_SIZEMATCH, 1574
 MHA_VERSION, 1574
 MHA_VERSION_BUILD, 1574
 MHA_VERSION_MAJOR, 1574
 MHA_VERSION_MINOR, 1574
 MHA_VERSION_RELEASE, 1574
 MHA_VERSION_STRING, 1574
 MHA_WAVEFORM, 1575
 MHA_XSTRF, 1573
 MHADestroy_t, 1577
 MHAGetVersion_t, 1576
 MHAInit_t, 1577
 MHAPluginCategory_t, 1578
 MHAPluginDocumentation_t, 1578
 MHAPrepare_t, 1577
 MHAProc_spec2spec_t, 1578
 MHAProc_spec2wave_t, 1577
 MHAProc_wave2spec_t, 1577
 MHAProc_wave2wave_t, 1577
 MHARelease_t, 1577
 MHASet_t, 1577
 MHASetcpp_t, 1577
 MHAStrError_t, 1578
 MHA_AC, 98
 MHA_AC::ac2matrix_helper_t, 720
 ac, 721
 ac2matrix_helper_t, 721
 acvar, 722
 getvar, 721
 is_complex, 721
 name, 721
 size, 721
 username, 721
 MHA_AC::ac2matrix_t, 722
 ac2matrix_t, 723
 getname, 723
 getusername, 723
 insert, 724
 update, 723
 MHA_AC::acspace2matrix_t, 724
 ~acspace2matrix_t, 726
 acspace2matrix_t, 725
 data, 728

frame, 727
frameno, 728
insert, 727
len, 728
operator=, 726
operator[], 726, 727
size, 727
update, 727
MHA_AC::double_t, 728
~double_t, 729
ac, 730
data, 729
double_t, 729
insert, 729
name, 730
MHA_AC::float_t, 730
~float_t, 731
ac, 731
data, 731
float_t, 731
insert, 731
name, 732
MHA_AC::int_t, 732
~int_t, 733
ac, 733
data, 733
insert, 733
int_t, 733
name, 733
MHA_AC::spectrum_t, 734
~spectrum_t, 735
ac, 736
insert, 736
name, 736
spectrum_t, 735
MHA_AC::stat_t, 736
insert, 737
mean, 737
stat_t, 737
std, 738
update, 737
MHA_AC::waveform_t, 738
~waveform_t, 739
ac, 740
insert, 740
name, 740
waveform_t, 739
MHA_AC_CHAR
mha.hh, 1575
MHA_AC_DOUBLE
mha.hh, 1576
MHA_AC_FLOAT
mha.hh, 1576
MHA_AC_INT
mha.hh, 1575
MHA_AC_MHACOMPLEX
mha.hh, 1576
MHA_AC_MHAREAL
mha.hh, 1576
MHA_AC_UNKNOWN
mha.hh, 1575
MHA_AC_USER
mha.hh, 1576
MHA_AC_VEC_FLOAT
mha.hh, 1576
mha_algo_comm.cpp, 1578
AC_DIM_MISMATCH, 1579
AC_INVALID_HANDLE, 1579
AC_INVALID_NAME, 1579
AC_INVALID_OUTPTR, 1579
AC_STRING_TRUNCATED, 1579
AC_SUCCESS, 1579
AC_TYPE_MISMATCH, 1579
algo_comm_default, 1579
mha_algo_comm.h, 1580
mha_algo_comm.hh, 1581
algo_comm_default, 1582
ALGO_COMM_ID_STR, 1581
mha_alloc
mha_ruby.cpp, 1622
MHA_assert
Error handling in the openMHA, 29
MHA_assert_equal
Error handling in the openMHA, 29
mha_audio_descriptor_t, 740
cf, 741
chdir, 742
dt, 741
is_complex, 741
n_channels, 741
n_freqs, 741
n_samples, 741
mha_audio_t, 742
cdata, 743
descriptor, 742
rdata, 743
MHA_CALLBACK_TEST
mha.hh, 1573
MHA_CALLBACK_TEST_PREFIX
mha.hh, 1573
mha_channel_info_t, 743
dir, 744

id, 744
 idstr, 744
 peaklevel, 744
 side, 744
mha_complex
 Complex arithmetics in the openMHA, 60
mha_complex_t, 744
 im, 745
 re, 745
mha_complex_test_array_t, 745
 c, 746
mha_dbdbuf_t
 mha_dbdbuf_t< FIFO >, 748
mha_dbdbuf_t< FIFO >, 746
 ~mha_dbdbuf_t, 749
 delay, 752
 fifo_size, 753
 get_delay, 749
 get_fifo_size, 749
 get_inner_error, 750
 get_inner_size, 749
 get_input_channels, 749
 get_input_fifo_fill_count, 750
 get_input_fifo_space, 750
 get_outer_size, 749
 get_output_channels, 750
 get_output_fifo_fill_count, 750
 get_output_fifo_space, 750
 inner_error, 753
 inner_size, 752
 input, 751
 input_channels, 753
 input_fifo, 753
 mha_dbdbuf_t, 748
 outer_error, 754
 outer_size, 752
 output, 752
 output_channels, 753
 output_fifo, 753
 process, 751
 provoke_inner_error, 750
 provoke_outer_error, 751
 value_type, 748
mha_debug
 Error handling in the openMHA, 29
 mha_error.cpp, 1587
mha_defs.h, 1582
 __MHA_FUN__, 1582
 __declspec, 1583
 CHECK_EXPR, 1583
 CHECK_VAR, 1583
 M_PI, 1583
 MAX, 1583
 MHA_EAR_LEFT, 1583
 MHA_EAR_MAX, 1584
 MHA_EAR_RIGHT, 1584
 MIN, 1583
mha_delenv
 mha_os.cpp, 1601
 mha_os.h, 1606
mha_direction_t, 754
 azimuth, 754
 distance, 755
 elevation, 755
MHA_DOMAIN_MAX
 mha.hh, 1575
mha_domain_t
 mha.hh, 1576
MHA_DOMAIN_UNKNOWN
 mha.hh, 1575
mha_drifter_fifo_t
 mha_drifter_fifo_t< T >, 757
mha_drifter_fifo_t< T >, 755
 desired_fill_count, 761
 get_available_space, 759
 get_des_fill_count, 759
 get_fill_count, 759
 get_min_fill_count, 760
 maximum_reader_xruns_in_succession_before_stop,
 762
 maximum_writer_xruns_in_succession_before_stop,
 762
 mha_drifter_fifo_t, 757
 minimum_fill_count, 760
 null_data, 762
 read, 758
 reader_started, 761
 reader_xruns_in_succession, 762
 reader_xruns_since_start, 762
 reader_xruns_total, 761
 starting, 760
 startup_zeros, 763
 stop, 760
 write, 758
 writer_started, 761
 writer_xruns_in_succession, 762
 writer_xruns_since_start, 761
 writer_xruns_total, 761
MHA_EAR_LEFT
 mha_defs.h, 1583
MHA_EAR_MAX
 mha_defs.h, 1584

MHA_EAR_RIGHT
 mha_defs.h, 1584

MHA_ERR_INVALID_HANDLE
 mha_errno.h, 1586

MHA_ERR_NULL
 mha_errno.h, 1586

MHA_ERR_SUCCESS
 mha_errno.h, 1585

MHA_ERR_UNKNOWN
 mha_errno.h, 1586

MHA_ERR_USER
 mha_errno.h, 1586

MHA_ERR_VARFMT
 mha_errno.h, 1586

MHA_ERR_VARRANGE
 mha_errno.h, 1586

mha_errno.c, 1584
 cstr_strerror, 1585
 mha_set_user_error, 1585
 mha_strerror, 1584
 next_except_str, 1585
 STRLEN, 1584

mha_errno.h, 1585
 MHA_ERR_INVALID_HANDLE, 1586
 MHA_ERR_NULL, 1586
 MHA_ERR_SUCCESS, 1585
 MHA_ERR_UNKNOWN, 1586
 MHA_ERR_USER, 1586
 MHA_ERR_VARFMT, 1586
 MHA_ERR_VARRANGE, 1586
 mha_set_user_error, 1586
 mha_strerror, 1586

MHA_Error, 763
 ~MHA_Error, 764
 get_longmsg, 765
 get_msg, 765
 longmsg, 765
 MHA_Error, 764
 msg, 765
 operator=, 765
 what, 765

mha_error.cpp, 1587
 mha_debug, 1587

mha_error.hh, 1587
 Getmsg, 1588

mha_error_helpers, 99
 digits, 99
 snprintf_required_length, 100

MHA_ErrorMsg
 Error handling in the openMHA, 28

MHA_ErrorMsg2

 MHAIOAsterisk.cpp, 1651
 MHAIOTCP.cpp, 1677

MHA_ErrorMsg3
 MHAIOAsterisk.cpp, 1651
 MHAIOTCP.cpp, 1678

mha_event_emitter.h, 1588

mha_events.cpp, 1589

mha_events.h, 1589

mha_exit_request
 mha_ruby.cpp, 1622

mha_fft
 gsc_adaptive_stage::gsc_adaptive_stage, 540
 smooth_cepstrum::smooth_cepstrum_t, 1440

mha_fft_backward
 Fast Fourier Transform functions, 75

mha_fft_backward_scale
 Fast Fourier Transform functions, 76

mha_fft_forward
 Fast Fourier Transform functions, 75

mha_fft_forward_scale
 Fast Fourier Transform functions, 76

mha_fft_free
 Fast Fourier Transform functions, 72

mha_fft_new
 Fast Fourier Transform functions, 71

mha_fft_spec2wave
 Fast Fourier Transform functions, 73, 74

mha_fft_spec2wave_scale
 Fast Fourier Transform functions, 77

mha_fft_t
 Fast Fourier Transform functions, 71

mha_fft_wave2spec
 Fast Fourier Transform functions, 72, 73

mha_fft_wave2spec_scale
 Fast Fourier Transform functions, 76

mha_fftfb.cpp, 1589
 BARKSCALE_ENTRIES, 1590
 filtershapefun, 1590

mha_fftfb.hh, 1591

mha_fifo.cpp, 1592

mha_fifo.h, 1592
 mha_fifo_thread_platform_implementation_t, 1593

mha_fifo_lf_t
 mha_fifo_lf_t< T >, 767

mha_fifo_lf_t< T >, 766
 atomic_read_ptr, 769
 atomic_write_ptr, 769
 get_available_space, 768

get_fill_count, 768
 mha_fifo_lf_t, 767
 read, 768
 write, 767
 mha_fifo_lw_t
 mha_fifo_lw_t< T >, 770
 mha_fifo_lw_t< T >, 769
 ~mha_fifo_lw_t, 770
 error, 772
 mha_fifo_lw_t, 770
 read, 771
 set_error, 772
 sync, 772
 write, 771
 mha_fifo_posix_threads_t, 773
 ~mha_fifo_posix_threads_t, 773
 aquire_mutex, 774
 decrease_condition, 775
 decrement, 774
 increase_condition, 775
 increment, 774
 mha_fifo_posix_threads_t, 773
 mutex, 775
 release_mutex, 774
 wait_for_decrease, 774
 wait_for_increase, 774
 mha_fifo_t
 mha_fifo_t< T >, 777, 778
 mha_fifo_t< T >, 775
 ~mha_fifo_t, 778
 buf, 781
 clear, 780
 get_available_space, 779
 get_fill_count, 779, 781
 get_max_fill_count, 780
 get_read_ptr, 781
 get_write_ptr, 780
 mha_fifo_t, 777, 778
 operator=, 780
 read, 779
 read_ptr, 782
 value_type, 777
 write, 778
 write_ptr, 781
 mha_fifo_thread_guard_t, 782
 ~mha_fifo_thread_guard_t, 783
 mha_fifo_thread_guard_t, 782
 sync, 783
 mha_fifo_thread_platform_implementation_t
 mha_fifo.h, 1593
 mha_fifo_thread_platform_t, 783
 ~mha_fifo_thread_platform_t, 784
 aquire_mutex, 785
 decrement, 786
 increment, 786
 mha_fifo_thread_platform_t, 784, 785
 operator=, 786
 release_mutex, 785
 wait_for_decrease, 785
 wait_for_increase, 786
 mha_filter.cpp, 1593
 diff_coeffs, 1593
 mha_filter.hh, 1593
 mha_fragsize
 MHAIOJackdb::io_jack_t, 950
 mha_free
 mha_ruby.cpp, 1621
 mha_freelib
 mha_os.h, 1603
 mha_freelib_success
 mha_os.h, 1603
 mha_generic_chain.cpp, 1595
 mhaconfig_compare, 1595
 mha_generic_chain.h, 1595
 MHAPLUGIN_OVERLOAD_OUTDOMAIN,
 1596
 mha_getenv
 mha_os.cpp, 1601
 mha_os.h, 1605
 mha_getlibfun
 mha_os.h, 1603
 mha_getlibfun_checked
 mha_os.h, 1604
 mha_git_commit_hash
 mha_git_commit_hash.cpp, 1596
 mha_git_commit_hash.hh, 1597
 mha_git_commit_hash.cpp, 1596
 GITCOMMITHASH, 1596
 mha_git_commit_hash, 1596
 mha_git_commit_hash.hh, 1597
 mha_git_commit_hash, 1597
 mha_hasenv
 mha_os.cpp, 1600
 mha_os.h, 1605
 mha_hton
 mha_os.h, 1606, 1607
 MHA_ID_MATRIX
 mha_signal.cpp, 1625
 MHA_ID_UINT_VECTOR
 mha_signal.cpp, 1624
 mha_io_ifc.h, 1597
 IODestroy_t, 1599

IOInit_t, 1598
IOPrepare_t, 1598
IOProcessEvent_t, 1598
IOResume_t, 1599
IOSetVar_t, 1599
IOStart_t, 1598
IOStartedEvent_t, 1598
IOStop_t, 1599
IOStoppedEvent_t, 1598
IOStrError_t, 1599
mha_io_utils.cpp, 1599
mha_io_utils.hh, 1599
mha_lib_extension
 mha_os.h, 1604
mha_libhandle_t
 mha_os.h, 1605
mha_library_paths
 mha_os.cpp, 1602
 mha_os.h, 1606
mha_loadlib
 mha_os.h, 1603
mha_loadlib_error
 mha_os.h, 1604
mha_min_1
 mha_signal.hh, 1635
mha_msleep
 mha_os.h, 1604
mha_multisrc.cpp, 1600
mha_multisrc.h, 1600
mha_ntoh
 mha_os.h, 1607
mha_os.cpp, 1600
 list_dir, 1602
 mha_delenv, 1601
 mha_getenv, 1601
 mha_hasenv, 1600
 mha_library_paths, 1602
 mha_setenv, 1601
mha_os.h, 1602
 FMTsz, 1604
 list_dir, 1606
 mha_delenv, 1606
 mha_freelib, 1603
 mha_freelib_success, 1603
 mha_getenv, 1605
 mha_getlibfun, 1603
 mha_getlibfun_checked, 1604
 mha_hasenv, 1605
 mha_hton, 1606, 1607
 mha_lib_extension, 1604
 mha_libhandle_t, 1605
 mha_library_paths, 1606
 mha_loadlib, 1603
 mha_loadlib_error, 1604
 mha_msleep, 1604
 mha_ntoh, 1607
 MHA_RESOLVE, 1604
 MHA_RESOLVE_CHECKED, 1604
 mha_setenv, 1606
mha_parse
 mha_ruby.cpp, 1622
mha_parser.cpp, 1607
 check_parenthesis_complex, 1609
 check_sign_complex, 1609
 MHAPLATFORM, 1608
 parse_1_complex, 1610
 parse_1_float, 1608
 write_float, 1608
mha_parser.hh, 1611
 DEFAULT_RETURNSIZE, 1615
 insert_member, 1615
mha_platform_tic
 mha_profiling.c, 1620
 mha_profiling.h, 1621
mha_platform_tictoc_t
 mha_profiling.h, 1620
mha_platform_toc
 mha_profiling.c, 1620
 mha_profiling.h, 1621
mha_plugin.cpp, 1615
mha_plugin.hh, 1615
 __declspec, 1616
 HINSTANCE, 1617
 MHAPLUGIN_CALLBACKS, 1618
 MHAPLUGIN_CALLBACKS_PREFIX,
 1617
 MHAPLUGIN_DOCUMENTATION, 1619
 MHAPLUGIN_DOCUMENTATION_PREFIX,
 1618
 MHAPLUGIN_INIT_CALLBACKS, 1618
 MHAPLUGIN_INIT_CALLBACKS_PREFIX,
 1617
 MHAPLUGIN_PROC_CALLBACK, 1618
 MHAPLUGIN_PROC_CALLBACK_PREFIX,
 1617
 MHAPLUGIN_SETCPP_CALLBACK_PREFIX,
 1617
 WINAPI, 1616
mha_profiling.c, 1619
 mha_platform_tic, 1620
 mha_platform_toc, 1620
 mha_tic, 1619

mha_toc, 1620
 mha_profiling.h, 1620
 mha_platform_tic, 1621
 mha_platform_tictoc_t, 1620
 mha_platform_toc, 1621
 mha_real_t
 Vector and matrix processing toolbox, 37
 mha_real_test_array_t, 787
 r, 787
 MHA_RELEASE_VERSION_STRING
 mha.hh, 1575
 MHA_RESOLVE
 mha_os.h, 1604
 MHA_RESOLVE_CHECKED
 mha_os.h, 1604
 mha_round
 mha_signal.hh, 1634
 mha_rt_fifo_element_t
 mha_rt_fifo_element_t< T >, 788
 mha_rt_fifo_element_t< T >, 787
 ~mha_rt_fifo_element_t, 788
 abandonned, 789
 data, 789
 mha_rt_fifo_element_t, 788
 next, 788
 mha_rt_fifo_t
 mha_rt_fifo_t< T >, 790
 mha_rt_fifo_t< T >, 789
 ~mha_rt_fifo_t, 790
 current, 792
 mha_rt_fifo_t, 790
 poll, 791
 poll_1, 791
 push, 791
 remove_abandonned, 792
 remove_all, 792
 root, 792
 mha_ruby.cpp, 1621
 Init_mha_ruby, 1622
 mha_alloc, 1622
 mha_exit_request, 1622
 mha_free, 1621
 mha_parse, 1622
 rb_f_t, 1621
 mha_samplerate
 MHAIOJackdb::io_jack_t, 950
 mha_set_user_error
 mha_errno.c, 1585
 mha_errno.h, 1586
 mha_setenv
 mha_os.cpp, 1601

mha_os.h, 1606
 mha_signal.cpp, 1622
 ASSERT_EQUAL_DIM, 1625
 ASSERT_EQUAL_DIM_PTR, 1625
 intensity, 1625
 MHA_ID_MATRIX, 1625
 MHA_ID_UINT_VECTOR, 1624
 safe_div, 1625
 set_minabs, 1625
 mha_signal.hh, 1626
 M_PI, 1634
 mha_min_1, 1635
 mha_round, 1634
 operator<<, 1635
 operator>>, 1636
 safe_div, 1635
 set_minabs, 1635
 value, 1635
 mha_signal_fft.h, 1636
 mha_spec_out
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,
 710
 mha_spec_t, 793
 buf, 794
 channel_info, 794
 num_channels, 794
 num_frames, 794
 MHA_SPECTRUM
 mha.hh, 1575
 mha_stash_environment_variable_t, 795
 ~mha_stash_environment_variable_t,
 796
 existed_before, 796
 mha_stash_environment_variable_t, 795
 original_content, 796
 variable_name, 796
 mha_strerror
 mha_errno.c, 1584
 mha_errno.h, 1586
 MHA_STRF
 mha.hh, 1573
 MHA_STRUCT_SIZEMATCH
 mha.hh, 1574
 mha_tablelookup.cpp, 1636
 mha_tablelookup.hh, 1636
 MHA_TCP, 100
 dtimes, 103
 G_ERRNO, 102
 H_ERRNO, 102
 HSTRERROR, 102
 N_ERRNO, 102

sock_initializer, 103
SOCKET, 102
stime, 103
STRERROR, 102
mha_tcp, 103
mha_tcp.cpp, 1637
ASYNC_CONNECT_STARTED, 1638
closesocket, 1638
host_port_to_sock_addr, 1639
INVALID_SOCKET, 1638
SOCKET, 1639
SOCKET_ERROR, 1638
tcp_connect_to, 1639
tcp_connect_to_with_timeout, 1639
thread_start_func, 1639
mha_tcp.hh, 1640
Sleep, 1641
MHA_TCP::Async_Notify, 797
~Async_Notify, 798
Async_Notify, 797
pipe, 798
reset, 798
set, 798
mha_tcp::buffered_socket_t, 798
current_message, 800
get_buffer, 799
next_message, 800
queue_write, 799
streambuf, 800
MHA_TCP::Client, 800
Client, 801
MHA_TCP::Connection, 802
~Connection, 804
buffered_incoming_bytes, 809
buffered_outgoing_bytes, 809
can_read_bytes, 807
can_read_line, 806
can_sysread, 805
can_syswrite, 805
closed, 810
Connection, 804
eof, 806
fd, 810
get_fd, 806
get_peer_address, 806
get_peer_port, 806
get_read_event, 806
get_write_event, 806
inbuf, 809
init_peer_data, 804
needs_write, 809
outbuf, 809
peer_addr, 810
read_bytes, 808
read_event, 809
read_line, 807
sysread, 805
syswrite, 805
try_write, 808
write, 808
write_event, 809
MHA_TCP::Event_Watcher, 810
~Event_Watcher, 812
Events, 811
events, 812
ignore, 812
iterator, 811
observe, 812
wait, 812
MHA_TCP::OS_EVENT_TYPE, 813
fd, 814
mode, 814
R, 813
T, 813
timeout, 814
W, 813
X, 813
MHA_TCP::Server, 814
~Server, 815
accept, 816
accept_event, 817
get_accept_event, 816
get_interface, 816
get_port, 816
iface, 817
initialize, 816
port, 817
Server, 815
serversocket, 817
sock_addr, 817
try_accept, 816
mha_tcp::server_t, 818
~server_t, 820
acceptor, 823
add_connection, 823
async_accept_has_been_triggered, 823
connections, 823
get_address, 820
get_context, 822
get_endpoint, 820
get_num_accepted_connections, 821
get_port, 820

io_context, 823
 num_accepted_connections, 823
 on_received_line, 821
 post_trigger_read_line, 822
 run, 821
 server_t, 819
 shutdown, 822
 trigger_accept, 822
 trigger_read_line, 822
MHA_TCP::sock_init_t, 824
 sock_init_t, 824
MHA_TCP::Sockaccept_Event, 824
 Sockaccept_Event, 825
MHA_TCP::Sockread_Event, 825
 Sockread_Event, 826
MHA_TCP::Sockwrite_Event, 826
 Sockwrite_Event, 827
MHA_TCP::Thread, 827
 ~Thread, 830
 arg, 830
 error, 831
 FINISHED, 829
 PREPARED, 829
 return_value, 831
 run, 830
 RUNNING, 829
 state, 831
 thr_f, 828
 Thread, 829
 thread_arg, 831
 thread_attr, 830
 thread_finish_event, 831
 thread_func, 831
 thread_handle, 830
MHA_TCP::Timeout_Event, 832
 end_time, 833
 get_os_event, 832
 Timeout_Event, 832
MHA_TCP::Timeout_Watcher, 833
 ~Timeout_Watcher, 834
 timeout, 834
 Timeout_Watcher, 834
MHA_TCP::Wakeup_Event, 835
 ~Wakeup_Event, 836
 get_os_event, 837
 ignored_by, 836
 observed_by, 836
 observers, 837
 os_event, 837
 os_event_valid, 838
 reset, 837
 status, 837
 Wakeup_Event, 836
mha_tcp_server.cpp, 1641
mha_tcp_server.hh, 1641
mha_test_struct_size
 PluginLoader::mhaplugloader_t, 1370
mha_tic
 mha_profiling.c, 1619
mha_tictoc_t, 838
 t, 838
 tv1, 838
 tv2, 838
 tz, 838
mha_toc
 mha_profiling.c, 1620
mha_toolbox.h, 1642
mha_utils.cpp, 1642
mha_utils.hh, 1642
MHA_VERSION
 mha.hh, 1574
MHA_VERSION_BUILD
 mha.hh, 1574
MHA_VERSION_MAJOR
 mha.hh, 1574
MHA_VERSION_MINOR
 mha.hh, 1574
MHA_VERSION_RELEASE
 mha.hh, 1574
MHA_VERSION_STRING
 mha.hh, 1574
mha_wave_out
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,
 709
mha_wave_t, 839
 buf, 840
 channel_info, 840
 num_channels, 840
 num_frames, 840
MHA_WAVEFORM
 mha.hh, 1575
mha_windowparser.cpp, 1642
 wnd_funcs, 1643
mha_windowparser.h, 1643
MHA_XSTRF
 mha.hh, 1573
mhachain, 104
mhachain.cpp, 1644
mhachain::chain_base_t, 841
 algos, 843
 b_prepared, 844
 bprofiling, 843

cfin, 844
cfout, 844
chain, 844
chain_base_t, 842
old_algos, 843
patchbay, 844
prepare, 843
process, 842, 843
release, 843
update, 843
mhachain::mhachain_t, 845
 mhachain_t, 845
mhachain::plugs_t, 846
 ~plugs_t, 847
 ac, 848
 algos, 848
 alloc_plugs, 847
 b_prepared, 848
 b_use_profiling, 850
 chain, 848
 cleanup_plugs, 848
 parser, 848
 plugs_t, 846
 prepare, 847
 prepared, 847
 proc_cnt, 849
 process, 847
 prof_algos, 849
 prof_cfg, 850
 prof_init, 849
 prof_load_con, 850
 prof_prepare, 849
 prof_process, 849
 prof_process_load, 849
 prof_process_tt, 849
 prof_release, 849
 prof_tt_con, 850
 profiling, 848
 release, 847
 tictoc, 850
 update_proc_load, 848
mhachain_t
 mhachain::mhachain_t, 845
mhachannels
 addsndfile::addsndfile_if_t, 254
mhaconfig_compare
 mha_generic_chain.cpp, 1595
 PluginLoader, 157
mhaconfig_in
 MHAParser::plugin_t< runtime_cfg_t >, 1152
mhaconfig_mon_t
 MHAParser::mhaconfig_mon_t, 1084
mhaconfig_out
 MHAParser::plugin_t< runtime_cfg_t >, 1152
mhaconfig_t, 850
 channels, 851
 domain, 851
 ffflen, 852
 fragsize, 851
 srate, 852
 wndlen, 852
MHADestroy_cb
 PluginLoader::mhaplugloader_t, 1371
MHADestroy_t
 mha.hh, 1577
MHAEvents, 104
MHAEvents::connector_base_t, 852
 ~connector_base_t, 853
 connector_base_t, 853
 emit_event, 853, 854
 emitter_die, 854
 emitter_is_alive, 854
MHAEvents::connector_t< receiver_t >, 855
 ~connector_t, 856
 connector_t, 856
 emit_event, 856, 857
 emitter, 857
 eventhandler, 857
 eventhandler_s, 857
 eventhandler_suu, 858
 receiver, 857
MHAEvents::emitter_t, 858
 ~emitter_t, 859
 connect, 859
 connections, 860
 disconnect, 859
 operator(), 859
MHAEvents::patchbay_t< receiver_t >, 860
 ~patchbay_t, 861
 connect, 861, 862
 cons, 862
mhafft
 fshift_hilbert::hilbert_shifter_t, 520
MHAFilter, 104
 butter_stop_ord1, 107
 fir_lp, 107
 gcd, 108
 make_friendly_number, 106
 o1_lp_coeffs, 106
 resampling_factors, 109

sinc, 109
 spec2fir, 108
MHAFilter::adapt_filter_param_t, 862
 adapt_filter_param_t, 863
 err_in, 863
 mu, 863
MHAFilter::adapt_filter_state_t, 863
 adapt_filter_state_t, 864
 filter, 864
 nchannels, 864
 ntaps, 864
 od, 865
 oy, 865
 W, 864
 X, 864
MHAFilter::adapt_filter_t, 865
 adapt_filter_t, 866
 connector, 867
 err_in, 867
 filter, 866
 mu, 867
 nchannels, 867
 ntaps, 867
 set_channelcnt, 866
 update_mu, 866
 update_ntaps, 867
MHAFilter::blockprocessing_polyphase_resampling_get_irs, 867
 ~blockprocessing_polyphase_resampling_t, 869
blockprocessing_polyphase_resampling_t, 868
 can_read, 870
 fragsize_in, 870
 fragsize_out, 870
 num_channels, 871
 read, 870
 resampling, 870
 write, 869
MHAFilter::complex_bandpass_t, 871
 A_, 874
 B_, 875
 complex_bandpass_t, 872
 creator_A, 872
 creator_B, 873
 filter, 873, 874
 get_weights, 873
 inspect, 874
 set_state, 873
 set_weights, 873
 Yn, 875
MHAFilter::diff_t, 875
 diff_t, 876
MHAFilter::fftfilter_t, 876
 ~fftfilter_t, 877
 channels, 880
 fft, 881
 fftfilter_t, 877
 fftlen, 880
 filter, 878, 879
 fragsize, 879
 sInput, 880
 sWeights, 880
 update_coeffs, 878
 wInput, 880
 wInput_fft, 880
 wIRS_fft, 880
 wOutput, 880
 wOutput_fft, 880
MHAFilter::fftfilterbank_t, 881
 ~fftfilterbank_t, 883
 fft, 886
 fftfilterbank_t, 882
 fftlen, 885
 filter, 883, 884
 firchannels, 885
 fragsize, 885
Hs, 885
hw, 885
 inputchannels, 885
 outputchannels, 885
 tail, 886
 update_coeffs, 883
 Xs, 885
 xw, 885
 Ys, 886
 yw, 886
 yw_temp, 886
MHAFilter::filter_t, 886
 ~filter_t, 888
 A, 890
 B, 890
 channels, 891
 filter, 889
 filter_t, 888
 get_len_A, 890
 get_len_B, 890
 len, 891
 len_A, 890
 len_B, 890
 state, 891

MHAFilter::gamma_flt_t, 891
 ~gamma_flt_t, 892
 A, 894
 bw_, 895
 cf_, 895
 delay, 894
 envelope_delay, 895
 gamma_flt_t, 892
 get_A, 894
 get_resynthesis_gain, 894
 get_weights, 893, 894
 GF, 894
 inspect, 894
 operator(), 892, 893
 phase_correction, 893
 reset_state, 894
 resynthesis_gain, 895
 set_weights, 893
 srate_, 895
MHAFilter::iir_filter_state_t, 895
 iir_filter_state_t, 896
MHAFilter::iir_filter_t, 896
 A, 899
 B, 899
 connector, 900
 filter, 898
 iir_filter_t, 897
 nchannels, 900
 resize, 899
 update_filter, 899
MHAFilter::iir_ord1_real_t, 900
 A_, 903
 B_, 903
 iir_ord1_real_t, 901
 operator(), 902
 set_state, 901, 902
 Yn, 903
MHAFilter::o1_ar_filter_t, 903
 c1_a, 907
 c1_r, 907
 c2_a, 907
 c2_r, 907
 fs, 907
 o1_ar_filter_t, 905
 operator(), 906
 set_tau_attack, 905
 set_tau_release, 906
MHAFilter::o1flt_lowpass_t, 908
 get_c1, 910
 get_last_output, 910
 o1flt_lowpass_t, 909
 set_tau, 909
MHAFilter::o1flt_maxtrack_t, 910
 o1flt_maxtrack_t, 911
 set_tau, 912
MHAFilter::o1flt_mintrack_t, 912
 o1flt_mintrack_t, 913
 set_tau, 913, 914
MHAFilter::partitioned_convolution_t, 914
 ~partitioned_convolution_t, 916
 bookkeeping, 918
 current_input_signal_buffer_half_index,
 917
 current_output_partition_index, 918
 fft, 918
 filter_partitions, 917
 fragsize, 916
 frequency_response, 918
 input_signal_spec, 917
 input_signal_wave, 917
 nchannels_in, 916
 nchannels_out, 917
 output_partitions, 917
 output_signal_spec, 918
 output_signal_wave, 918
 partitioned_convolution_t, 915
 process, 916
MHAFilter::partitioned_convolution_t::index_t,
 919
 delay, 920
 index_t, 919, 920
 source_channel_index, 920
 target_channel_index, 920
MHAFilter::polyphase_resampling_t, 921
 downsampling_factor, 925
 impulse_response, 925
 now_index, 925
 polyphase_resampling_t, 922
 read, 924
 readable_frames, 924
 ringbuffer, 925
 underflow, 925
 upsampling_factor, 924
 write, 923
MHAFilter::resampling_filter_t, 926
 fragsize, 928
 fragsize_validator, 927
 resampling_filter_t, 927
MHAFilter::smoothspec_t, 928
 _linphase_asym, 932
 ~smoothspec_t, 930
 fft, 932

fftlen, 931
 internal_fir, 931
 minphase, 932
 nchannels, 931
 smoothspec, 930
 smoothspec_t, 929
 spec2fir, 931
 tmp_spec, 932
 tmp_wave, 932
 window, 931
MHAFilter::thirddoctave_analyzer_t, 932
 bw_generator, 934
 cf, 934
 cf_generator, 934
 cfg_, 934
 dup, 934
 fb, 934
 get_cf_hz, 933
 nbands, 933
 nchannels, 933
 out_chunk, 934
 out_chunk_im, 935
 process, 933
 thirddoctave_analyzer_t, 933
MHAFilter::transfer_function_t, 935
 impulse_response, 938
 isempty, 937
 non_empty_partitions, 937
 partitions, 936
 source_channel_index, 938
 target_channel_index, 938
 transfer_function_t, 936
MHAFilter::transfer_matrix_t, 938
 non_empty_partitions, 939
 partitions, 939
mhfaw_lib.cpp, 1644
mhfaw_lib.h, 1644
MHAGetVersion_cb
 PluginLoader::mhapluginloader_t, 1371
MHAGetVersion_t
 mha.hh, 1576
MHAInit_cb
 PluginLoader::mhapluginloader_t, 1371
MHAInit_t
 mha.hh, 1577
MHAIOalsa.cpp, 1644
 DBG, 1645
 dummy_interface_test, 1647
 ERR_IHANDLE, 1646
 ERR_SUCCESS, 1646
 ERR_USER, 1646
 IODestroy, 1647, 1649
 IOInit, 1646, 1647
 IOPrepare, 1646, 1648
 IORelease, 1646, 1648
 IOSetVar, 1647, 1648
 IOStart, 1646, 1648
 IOStop, 1646, 1648
 IOStrError, 1647, 1648
 MAX_USER_ERR, 1646
 user_err_msg, 1649
MHAIOAsterisk.cpp, 1649
 copy_error, 1653
 dummy_interface_test, 1652
 ERR_IHANDLE, 1650
 ERR_SUCCESS, 1650
 ERR_USER, 1650
 IODestroy, 1652, 1654
 IOInit, 1651, 1653
 IOPrepare, 1652, 1653
 IORelease, 1652, 1653
 IOSetVar, 1652, 1654
 IOStart, 1652, 1653
 IOStop, 1652, 1653
 IOStrError, 1652, 1654
 MAX_TCP_PORT, 1651
 MAX_TCP_PORT_STR, 1651
 MAX_USER_ERR, 1651
 MHA_ErrorMsg2, 1651
 MHA_ErrorMsg3, 1651
 MIN_TCP_PORT, 1651
 MIN_TCP_PORT_STR, 1651
 thread_startup_function, 1653
 user_err_msg, 1654
MHAIOFile.cpp, 1654
 DEBUG, 1655
 dummy_interface_test, 1657
 ERR_IHANDLE, 1656
 ERR_SUCCESS, 1655
 ERR_USER, 1656
 IODestroy, 1657, 1658
 IOInit, 1656, 1657
 IOPrepare, 1656, 1657
 IORelease, 1656, 1658
 IOSetVar, 1656, 1658
 IOStart, 1656, 1657
 IOStop, 1656, 1658
 IOStrError, 1657, 1658
 MAX_USER_ERR, 1656
 user_err_msg, 1658
MHAIOJack, 109
MHAIOJack.cpp, 1659

dummy_interface_test, 1661
ERR_IHANDLE, 1660
ERR_SUCCESS, 1660
ERR_USER, 1660
IODestroy, 1661, 1662
IOInit, 1660, 1661
IOPrepare, 1660, 1661
IOResume, 1661, 1662
IOSetVar, 1661, 1662
IOStart, 1660, 1662
IOStop, 1660, 1662
IOStrError, 1661, 1662
MAX_USER_ERR, 1660
user_err_msg, 1663
MHAIOJack::io_jack_t, 939
 clientname, 943
 connections_in, 943
 connections_out, 944
 delays_in, 943
 delays_out, 944
 fw_fragsize, 943
 fw_samplerate, 943
 get_all_input_ports, 942
 get_all_output_ports, 942
 get_delays_in, 942
 get_delays_out, 942
 get_physical_input_ports, 942
 get_physical_output_ports, 942
 io_jack_t, 941
 patchbay, 945
 portnames_in, 944
 portnames_out, 944
 ports_in_all, 944
 ports_in_physical, 944
 ports_out_all, 944
 ports_out_physical, 944
 ports_parser, 945
 prepare, 941
 read_get_cpu_load, 942
 read_get_scheduler, 943
 read_get_xruns, 943
 reconnect_inports, 941
 reconnect_outports, 942
 release, 941
 servername, 943
 state_cpupload, 945
 state_parser, 945
 state_priority, 945
 state_scheduler, 945
 state_xruns, 945
MHAIOJackdb, 110
MHAIOJackdb.cpp, 1663
 dummy_interface_test, 1665
 ERR_IHANDLE, 1664
 ERR_SUCCESS, 1664
 ERR_USER, 1664
 IODestroy, 1665, 1666
 IOInit, 1664, 1665
 IOPrepare, 1664, 1665
 IOResume, 1665, 1666
 IOSetVar, 1665, 1666
 IOStart, 1664, 1666
 IOStop, 1664, 1666
 IOStrError, 1665, 1666
 MAX_USER_ERR, 1664
 user_err_msg, 1667
MHAIOJackdb::io_jack_t, 946
 clientname, 951
 connections_in, 951
 connections_out, 951
 fail_on_async_jackerr, 951
 fail_on_async_jackerror, 948
 fragsize_ratio, 950
 get_all_input_ports, 949
 get_all_output_ports, 949
 get_physical_input_ports, 949
 get_physical_output_ports, 949
 io_jack_t, 948
 IOProcessEvent_inner, 948
 locate, 952
 mha_fragsize, 950
 mha_samplerate, 950
 patchbay, 953
 portnames_in, 951
 portnames_out, 951
 ports_in_all, 952
 ports_in_physical, 952
 ports_out_all, 952
 ports_out_physical, 952
 ports_parser, 952
 prepare, 948
 proc_event, 950
 proc_handle, 950
 pwinner_out, 953
 read_get_cpu_load, 949
 read_get_scheduler, 950
 read_get_xruns, 949
 reconnect_inports, 949
 reconnect_outports, 949
 release, 948
 server_fragsize, 952
 server_srate, 952

servername, 951
 set_locate, 950
 set_use_jack_transport, 950
 state_cpupload, 953
 state_parser, 953
 state_priority, 953
 state_scheduler, 953
 state_xruns, 953
 use_jack_transport, 951
MHAIOParser.cpp, 1667
 dummy_interface_test, 1669
 ERR_IHANDLE, 1668
 ERR_SUCCESS, 1668
 ERR_USER, 1668
 IODestroy, 1669, 1670
 IOInit, 1668, 1669
 IOPrepare, 1668, 1669
 IOResume, 1669, 1670
 IOSetVar, 1669, 1670
 IOStart, 1668, 1670
 IOStop, 1668, 1670
 IOStrError, 1669, 1670
 MAX_USER_ERR, 1668
 user_err_msg, 1671
MHAIOPortAudio, 110
 parserFriendlyName, 110
MHAIOPortAudio.cpp, 1671
 dummy_interface_test, 1673
 ERR_IHANDLE, 1672
 ERR_SUCCESS, 1672
 ERR_USER, 1672
 IODestroy, 1673, 1675
 IOInit, 1672, 1674
 IOPrepare, 1673, 1674
 IOResume, 1673, 1674
 IOSetVar, 1673, 1675
 IOStart, 1673, 1674
 IOStop, 1673, 1674
 IOStrError, 1673, 1675
 MAX_USER_ERR, 1672
 portaudio_callback, 1674, 1675
 user_err_msg, 1675
MHAIOPortAudio::device_info_t, 954
 defaultHighInputLatency, 956
 defaultHighOutputLatency, 956
 defaultLowInputLatency, 956
 defaultLowOutputLatency, 956
 defaultSampleRate, 956
 device_info_t, 954
 fill_info, 955
 hostApi, 955
 maxInputChannels, 955
 maxOutputChannels, 955
 name, 955
 numDevices, 955
 structVersion, 955
MHAIOPortAudio::io_portaudio_t, 957
 ~io_portaudio_t, 958
 cmd_prepare, 959
 cmd_release, 960
 cmd_start, 959
 cmd_stop, 959
 device_index_in, 962
 device_index_in_updated, 959
 device_index_out, 962
 device_index_out_updated, 959
 device_info, 960
 device_name_in, 962
 device_name_in_updated, 959
 device_name_out, 962
 device_name_out_updated, 959
 fragsize, 961
 io_portaudio_t, 958
 nchannels_in, 961
 nchannels_out, 960
 patchbay, 962
 portaudio_callback, 960
 portaudio_stream, 962
 proc_event, 961
 proc_handle, 961
 s_in, 960
 s_out, 960
 samplerate, 960
 start_event, 961
 start_handle, 961
 stop_event, 961
 stop_handle, 961
MHAIOTCP.cpp, 1676
 copy_error, 1679
 dummy_interface_test, 1679
 ERR_IHANDLE, 1677
 ERR_SUCCESS, 1677
 ERR_USER, 1677
 IODestroy, 1679, 1681
 IOInit, 1678, 1680
 IOPrepare, 1678, 1680
 IOResume, 1679, 1680
 IOSetVar, 1679, 1680
 IOStart, 1678, 1680
 IOStop, 1679, 1680
 IOStrError, 1679, 1681
 MAX_TCP_PORT, 1678

MAX_TCP_PORT_STR, 1678
MAX_USER_ERR, 1677
MHA_ErrorMsg2, 1677
MHA_ErrorMsg3, 1678
MIN_TCP_PORT, 1678
MIN_TCP_PORT_STR, 1678
thread_startup_function, 1679
user_err_msg, 1681
mhaioutils, 111
 to_int_clamped, 111
MHAJack, 111
 get_port_capture_latency, 112
 get_port_capture_latency_int, 112
 get_port_playback_latency, 114
 get_port_playback_latency_int, 114
 io, 112
mhajack.cpp, 1681
 dummy_jack_proc_cb, 1682
 jack_error_handler, 1681
 last_jack_err, 1682
 last_jack_err_msg, 1682
 make_friendly_number, 1682
mhajack.h, 1682
 IO_ERROR_JACK, 1684
 IO_ERROR_MHAJACKLIB, 1684
 last_jack_err_msg, 1684
 MAX_USER_ERR, 1684
 MHAJACK_FW_STARTED, 1683
 MHAJACK_STARTING, 1684
 MHAJACK_STOPPED, 1684
MHAJack::client_avg_t, 963
 b_ready, 967
 b_stopped, 966
 client_avg_t, 964
 frag_out, 966
 io, 964
 IOStoppedEvent, 965
 n, 966
 name, 966
 nrep, 966
 pos, 966
 proc, 965
 sn_in, 966
 sn_out, 966
MHAJack::client_noncont_t, 967
 b_stopped, 969
 client_noncont_t, 968
 frag_out, 970
 io, 968
 IOStoppedEvent, 969
 name, 970
 pos, 969
 proc, 969
 sn_in, 969
 sn_out, 969
 MHAJack::client_t, 970
 b_prepared, 979
 client_t, 972
 connect_input, 974
 connect_output, 974
 fail_on_async_jackerror, 980
 flags, 979
 fragsize, 977
 get_cpu_load, 976
 get_fragsize, 974
 get_my_input_ports, 975
 get_my_output_ports, 975
 get_ports, 975
 get_srate, 974
 get_xruns, 974
 get_xruns_reset, 975
 inch, 979
 input_portnames, 980
 internal_start, 976
 internal_stop, 976
 is_prepared, 976
 jack_proc_cb, 977
 jack_xrun_cb, 977
 jc, 979
 jstate_prev, 979
 nchannels_in, 978
 nchannels_out, 978
 num_xruns, 977
 outch, 979
 output_portnames, 980
 prepare, 973
 prepare_impl, 976
 proc_event, 978
 proc_handle, 978
 release, 974
 s_in, 979
 s_out, 979
 samplerate, 978
 set_input_portnames, 975
 set_output_portnames, 975
 set_use_jack_transport, 976
 start, 974
 start_event, 978
 start_handle, 978
 stop, 974
 stop_event, 978
 stop_handle, 978

stopped, 977
 str_error, 975
 use_jack_transport, 979
MHAJack::port_t, 980
 ~port_t, 982
 connect_to, 983
 dir_t, 981
 dir_type, 983
 get_short_name, 983
 input, 981
 iob, 984
 jc, 984
 mute, 983
 output, 981
 port, 983
 port_t, 981, 982
 read, 982
 write, 982
MHAJACK_FW_STARTED
 mhajack.h, 1683
MHAJACK_STARTING
 mhajack.h, 1684
MHAJACK_STOPPED
 mhajack.h, 1684
MHAKernel, 114
 algo_comm_safe_cast, 115
MHAKernel::algo_comm_class_t, 984
 ~algo_comm_class_t, 986
 ac, 989
 algo_comm_class_t, 985
 algo_comm_id_string, 989
 algo_comm_id_string_len, 989
 get_c_handle, 986
 get_entries, 988
 get_error, 988
 get_var, 987
 get_var_double, 988
 get_var_float, 988
 get_var_int, 987
 insert_var, 986
 insert_var_double, 987
 insert_var_float, 986
 insert_var_int, 986
 insert_var_vfloat, 986
 is_var, 987
 local_get_entries, 989
 local_get_var, 989
 local_insert_var, 988
 local_is_var, 989
 local_remove_ref, 988
 local_remove_var, 988
 remove_ref, 987
 remove_var, 987
 size, 989
 vars, 990
MHAKernel::comm_var_map_t, 990
 has_key, 990
mhamain
 mha.cpp, 1570
 mhamain.cpp, 1686
 mhamain.cpp, 1684
 create_lock, 1685
 GREETING_TEXT, 1685
 HELP_TEXT, 1685
 mhamain, 1686
 NORELEASE_WARNING, 1685
 remove_lock, 1686
 VERSION_EXTENSION, 1685
MHAMultiSrc, 115
MHAMultiSrc::base_t, 991
 ac, 992
 base_t, 992
 select_source, 992
MHAMultiSrc::channel_t, 993
 channel, 993
 name, 993
MHAMultiSrc::channels_t, 993
 channels_t, 993
MHAMultiSrc::spectrum_t, 994
 spectrum_t, 995
 update, 995
MHAMultiSrc::waveform_t, 996
 update, 997
 waveform_t, 996
MHAOvlFilter, 115
 scale_fun_t, 116
MHAOvlFilter::band_descriptor_t, 997
 cf, 998
 cf_h, 998
 cf_l, 997
 ef_h, 998
 ef_l, 998
 high_side_flat, 998
 low_side_flat, 998
MHAOvlFilter::barkscale, 116
 vbark, 117
 vfreq, 117
MHAOvlFilter::barkscale::bark2hz_t, 999
 ~bark2hz_t, 999
 bark2hz_t, 999
MHAOvlFilter::barkscale::hz2bark_t, 1000
 ~hz2bark_t, 1000

hz2bark_t, 1000
MHAOvlFilter::fftfb_ac_info_t, 1001
 bwv, 1002
 cfv, 1001
 cLTASS, 1002
 efv, 1002
 fftfb_ac_info_t, 1001
 insert, 1001
MHAOvlFilter::fftfb_t, 1002
 ~fftfb_t, 1004
 apply_gains, 1004
 bin1, 1004
 bin2, 1005
 fftfb_t, 1003
 ffflen, 1006
 get_fbpower, 1004
 get_fbpower_db, 1004
 get_ffflen, 1005
 get_ltass_gain_db, 1004
 samplingrate, 1006
 shape, 1006
 vbin1, 1005
 vbin2, 1005
 w, 1005
MHAOvlFilter::fftfb_vars_t, 1006
 cf, 1009
 cLTASS, 1010
 ef, 1010
 f, 1009
 fail_on_nonmonotonic, 1009
 fail_on_unique_bins, 1009
 fftfb_vars_t, 1008
 flag_allow_empty_bands, 1009
 fscale, 1008
 ftype, 1008
 normalize, 1009
 ovltype, 1008
 plateau, 1008
 shapes, 1010
MHAOvlFilter::FreqScaleFun, 117
 hz2bark, 118
 hz2bark_analytic, 119
 hz2erb, 119
 hz2erb_glasberg1990, 119
 hz2hz, 118
 hz2khz, 118
 hz2log, 119
 hz2octave, 118
 hz2third_octave, 118
 inv_scale, 120
MHAOvlFilter::fscale_bw_t, 1010
 bw, 1012
 bw_hz, 1012
 fscale_bw_t, 1011
 get_bw_hz, 1011
 update_hz, 1011
 updater, 1012
MHAOvlFilter::fscale_t, 1012
 f, 1014
 f_hz, 1014
 fscale_t, 1013
 get_f_hz, 1013
 unit, 1013
 update_hz, 1013
 updater, 1014
MHAOvlFilter::fspacing_t, 1014
 bands, 1017
 cf2bands, 1016
 ef2bands, 1016
 equidist2bands, 1016
 fail_on_nonmonotonic_cf, 1016
 fail_on_unique_fftbins, 1016
 fs_, 1017
 fspacing_t, 1015
 get_cf_fftbin, 1015
 get_cf_hz, 1016
 get_ef_hz, 1016
 nbands, 1016
 nfft_, 1017
 symmetry_scale, 1017
MHAOvlFilter::overlap_save_filterbank_analytic_t,
 1018
 filter_analytic, 1019
 imagfb, 1019
 overlap_save_filterbank_analytic_t, 1018
MHAOvlFilter::overlap_save_filterbank_t,
 1019
 channelconfig_out_, 1021
 get_channelconfig, 1021
 overlap_save_filterbank_t, 1021
MHAOvlFilter::overlap_save_filterbank_t::vars_t,
 1022
 ffflen, 1023
 irswnd, 1023
 phasemodel, 1023
 vars_t, 1022
MHAOvlFilter::scale_var_t, 1024
 add_fun, 1025
 fun, 1026
 get_fun, 1025
 get_name, 1025
 hz2unit, 1025

names, 1026
 scale_var_t, 1025
 unit2hz, 1025
MHAOvlFilter::ShapeFun, 120
 expfit, 122
 gauss, 122
 hann, 121
 linear, 121
 rect, 120
MHAParser, 122
 all_dump, 127
 all_ids, 127
 c_parse_cmd_t, 126
 c_parse_err_t, 128
 cfg_dump, 126
 cfg_dump_short, 126
 commentate, 126
 entry_map_t, 126
 envreplace, 127
 get_precision, 126
 mon_dump, 127
 opact_map_t, 125
 opact_t, 125
 query_map_t, 126
 query_t, 125
 strreplace, 127
 trim, 126
MHAParser::base_t, 1026
 ~base_t, 1030
 activate_query, 1036
 add_parent_on_insert, 1036
 add_replace_pair, 1037
 base_t, 1030
 data_is_initialized, 1038
 fullname, 1036
 help, 1038
 id_str, 1038
 nested_lock, 1038
 notify, 1036
 op_query, 1032
 op_setval, 1032
 op_subparse, 1032
 operators, 1038
 oplist, 1037
 parent, 1038
 parse, 1031, 1032
 prereadaccess, 1037
 queries, 1038
 query_addsubst, 1035
 query_cmds, 1035
 query_dump, 1032
 query_entries, 1032
 query_help, 1035
 query_id, 1035
 query_listids, 1034
 query_perm, 1033
 query_range, 1033
 query_readfile, 1034
 query_savefile, 1034
 query_savefile_compact, 1034
 query_savemons, 1034
 query_subst, 1035
 query_type, 1033
 query_val, 1033
 query_version, 1035
 readaccess, 1037
 repl_list, 1038
 repl_list_t, 1030
 rm_parent_on_remove, 1036
 set_help, 1036
 set_node_id, 1035
 thefullname, 1039
 valuechanged, 1037
 writeaccess, 1037
MHAParser::base_t::replace_t, 1039
 a, 1040
 b, 1040
 get_a, 1040
 get_b, 1040
 replace, 1039
 replace_t, 1039
MHAParser::bool_mon_t, 1040
 bool_mon_t, 1041
 data, 1042
 query_type, 1042
 query_val, 1041
MHAParser::bool_t, 1042
 bool_t, 1043
 data, 1044
 op_setval, 1044
 query_type, 1044
 query_val, 1044
MHAParser::c_ifc_parser_t, 1045
 ~c_ifc_parser_t, 1046
 c_ifc_parser_t, 1046
 c_parse_cmd, 1047
 c_parse_err, 1047
 libdata, 1047
 liberr, 1047
 modulename, 1047
 op_query, 1046
 op_setval, 1046

op_subparse, 1046
ret_size, 1047
retv, 1047
set_parse_cb, 1046
test_error, 1047
MHParse::commit_t< receiver_t >, 1048
commit_t, 1049
extern_connector, 1049
MHParse::complex_mon_t, 1050
complex_mon_t, 1051
data, 1051
query_type, 1051
query_val, 1051
MHParse::complex_t, 1052
complex_t, 1053
data, 1054
op_setval, 1053
query_type, 1053
query_val, 1053
MHParse::entry_t, 1054
entry, 1055
entry_t, 1054
name, 1054
MHParse::expression_t, 1055
expression_t, 1055, 1056
lval, 1056
op, 1056
rval, 1056
MHParse::float_mon_t, 1056
data, 1058
float_mon_t, 1057
query_type, 1058
query_val, 1057
MHParse::float_t, 1058
data, 1061
float_t, 1060
op_setval, 1060
query_type, 1060
query_val, 1061
MHParse::int_mon_t, 1061
data, 1063
int_mon_t, 1062
query_type, 1063
query_val, 1063
MHParse::int_t, 1064
data, 1066
int_t, 1065
op_setval, 1065
query_type, 1066
query_val, 1066
MHParse::keyword_list_t, 1066
add_entry, 1069
empty_string, 1070
entries, 1070
get_entries, 1069
get_index, 1069
get_value, 1068
index, 1069
keyword_list_t, 1068
set_entries, 1068
set_index, 1069
set_value, 1068
size_t, 1067
validate, 1069
MHParse::kw_t, 1070
data, 1073
isval, 1072
kw_t, 1071, 1072
op_setval, 1072
query_range, 1073
query_type, 1073
query_val, 1073
set_range, 1072
validate, 1072
MHParse::mcomplex_mon_t, 1074
data, 1075
mcomplex_mon_t, 1075
query_type, 1075
query_val, 1075
MHParse::mcomplex_t, 1076
data, 1078
mcomplex_t, 1077
op_setval, 1077
query_type, 1077
query_val, 1077
MHParse::mfloat_mon_t, 1078
data, 1080
mfloat_mon_t, 1079
query_type, 1079
query_val, 1079
MHParse::mfloat_t, 1080
data, 1082
mfloat_t, 1081
op_setval, 1082
query_type, 1082
query_val, 1082
MHParse::mhacconfig_mon_t, 1083
channels, 1084
domain, 1084
ffflen, 1085
fragsize, 1084
mhacconfig_mon_t, 1084

srate, 1085
 update, 1084
 wndlen, 1084
MHAParser::mhapluginloader_t, 1085
 ~mhapluginloader_t, 1087
 ac_, 1089
 bookkeeping, 1089
 cf_in_, 1089
 cf_out_, 1089
 connector, 1089
 get_cfin, 1088
 get_cfout, 1088
 get_last_name, 1088
 last_name, 1089
 load_plug, 1088
 mhapluginloader_t, 1087
 parent_, 1088
 plug, 1088
 plugname, 1089
 plugname_name_, 1089
 prefix_, 1089
 prepare, 1087
 process, 1087, 1088
 release, 1087
MHAParser::mint_mon_t, 1090
 data, 1092
 mint_mon_t, 1091
 query_type, 1091
 query_val, 1091
MHAParser::mint_t, 1092
 data, 1094
 mint_t, 1093
 op_setval, 1094
 query_type, 1094
 query_val, 1094
MHAParser::monitor_t, 1095
 monitor_t, 1095, 1096
 op_query, 1096
 query_dump, 1096
 query_perm, 1096
MHAParser::parser_t, 1097
 ~parser_t, 1099
 entries, 1103
 force_remove_item, 1100
 has_entry, 1103
 id_string, 1103
 insert_item, 1099
 last_errormsg, 1103
 op_query, 1101
 op_setval, 1101
 op_subparse, 1100
 parser_t, 1099
 query_dump, 1101
 query_entries, 1101
 query_listids, 1102
 query_readfile, 1101
 query_savefile, 1102
 query_savefile_compact, 1102
 query_savemons, 1102
 query_type, 1101
 query_val, 1102
 remove_item, 1099, 1100
 set_id_string, 1102
MHAParser::range_var_t, 1103
 check_low, 1107
 check_range, 1107
 check_up, 1107
 low_incl, 1107
 low_limit, 1107
 query_range, 1105
 range_var_t, 1105
 set_range, 1105
 up_incl, 1107
 up_limit, 1107
 validate, 1105, 1106
MHAParser::StrCnv, 128
 bracket_balance, 130
 num_brackets, 129
 str2val, 130, 131
 str2val< mha_real_t >, 131
 val2str, 132–134
MHAParser::string_mon_t, 1108
 data, 1110
 query_type, 1109
 query_val, 1109
 string_mon_t, 1109
MHAParser::string_t, 1110
 data, 1112
 op_setval, 1112
 query_type, 1112
 query_val, 1112
 string_t, 1111
MHAParser::variable_t, 1113
 locked, 1115
 op_setval, 1114
 query_perm, 1114
 setlock, 1114
 variable_t, 1114
MHAParser::vcomplex_mon_t, 1115
 data, 1117
 query_type, 1116
 query_val, 1116

vcomplex_mon_t, 1116
MHPARSER::vcomplex_t, 1117
 data, 1119
 op_setval, 1118
 query_type, 1119
 query_val, 1119
 vcomplex_t, 1118
MHPARSER::vfloat_mon_t, 1120
 data, 1121
 query_type, 1121
 query_val, 1121
 vfloat_mon_t, 1121
MHPARSER::vfloat_t, 1122
 data, 1124
 op_setval, 1124
 query_type, 1124
 query_val, 1124
 vfloat_t, 1123
MHPARSER::vint_mon_t, 1125
 data, 1126
 query_type, 1126
 query_val, 1126
 vint_mon_t, 1126
MHPARSER::vint_t, 1127
 data, 1129
 op_setval, 1128
 query_type, 1128
 query_val, 1128
 vint_t, 1128
MHPARSER::vstring_mon_t, 1129
 data, 1131
 query_type, 1131
 query_val, 1130
 vstring_mon_t, 1130
MHPARSER::vstring_t, 1131
 data, 1133
 op_setval, 1132
 query_type, 1132
 query_val, 1133
 vstring_t, 1132
MHPARSER::window_t, 1133
 get_type, 1136
 get_window, 1135, 1136
 setlock, 1136
 user, 1137
 window_t, 1135
 wnd_bartlett, 1135
 wnd_blackman, 1135
 wnd_hamming, 1135
 wnd_hann, 1135
 wnd_rect, 1135
 wnd_user, 1135
 wtype, 1137
 wtype_t, 1135
MHAPLATFORM
 mha_parser.cpp, 1608
mhaplug_cfg_t, 1137
 ~mhaplug_cfg_t, 1138
 mhaplug_cfg_t, 1137
MHAPLUGIN, 134
MHAPLUGIN::cfg_node_t< runtime_cfg_t >, 1138
 ~cfg_node_t, 1139
 cfg_node_t, 1139
 data, 1140
 next, 1139
 not_in_use, 1140
MHAPLUGIN::config_t< runtime_cfg_t >, 1141
 ~config_t, 1143
 cfg, 1145
 cfg_node_current, 1146
 cfg_root, 1146
 cleanup_unused_cfg, 1145
 config_t, 1143
 peek_config, 1144
 poll_config, 1144
 push_config, 1145
 remove_all_cfg, 1145
MHAPLUGIN::plugin_t< runtime_cfg_t >, 1146
 ~plugin_t, 1148
 ac, 1151
 input_cfg, 1150
 input_cfg_, 1151
 is_prepared, 1150
 is_prepared_, 1151
 mhaconfig_in, 1152
 mhaconfig_out, 1152
 output_cfg, 1150
 output_cfg_, 1151
 plugin_t, 1148
 prepare, 1148
 prepare_, 1150
 release, 1149
 release_, 1150
 tftype, 1151
MHAPLUGIN_CALLBACKS
 mha_plugin.hh, 1618
MHAPLUGIN_CALLBACKS_PREFIX
 mha_plugin.hh, 1617
MHAPLUGIN_DOCUMENTATION
 mha_plugin.hh, 1619
MHAPLUGIN_DOCUMENTATION_PREFIX

mha_plugin.hh, 1618
MHAPLUGIN_INIT_CALLBACKS
 mha_plugin.hh, 1618
MHAPLUGIN_INIT_CALLBACKS_PREFIX
 mha_plugin.hh, 1617
MHAPLUGIN_OVERLOAD_OUTDOMAIN
 altconfig.hh, 1528
 altplugs.cpp, 1529
 matlab_wrapper.hh, 1569
 mha_generic_chain.h, 1596
 split.cpp, 1703
 wave2spec.hh, 1706
MHAPLUGIN_PROC_CALLBACK
 mha_plugin.hh, 1618
MHAPLUGIN_PROC_CALLBACK_PREFIX
 mha_plugin.hh, 1617
MHAPlugin_Resampling, 134
MHAPlugin_Resampling::resampling_if_t,
 1152
 algo, 1155
 chain, 1155
 fragsize, 1154
 irslen_inner2outer, 1154
 irslen_outer2inner, 1154
 nyquist_ratio, 1154
 plugloader, 1154
 prepare, 1153
 process, 1153
 release, 1154
 resampling_if_t, 1153
 srate, 1154
MHAPlugin_Resampling::resampling_t, 1155
 inner2outer_resampling, 1157
 inner_fragsize, 1156
 inner_signal, 1157
 inner_srate, 1156
 nchannels_in, 1156
 nchannels_out, 1157
 outer2inner_resampling, 1157
 outer_fragsize, 1156
 outer_srate, 1156
 output_signal, 1157
 plugloader, 1157
 process, 1156
 resampling_t, 1155
MHAPLUGIN_SETCPP_CALLBACK_PREFIX
 mha_plugin.hh, 1617
MHAPlugin_Split, 135
 INVALID_THREAD_PRIORITY, 136
MHAPlugin_Split::domain_handler_t, 1158
 ~domain_handler_t, 1160
 deallocate_domains, 1161
 domain_handler_t, 1159
 get_signal, 1162
 operator=, 1160
 process, 1163
 processor, 1164
 put_signal, 1161
 set_input_domain, 1160
 set_output_domain, 1160
 spec_in, 1164
 spec_out, 1164
 wave_in, 1163
 wave_out, 1163
MHAPlugin_Split::dummy_threads_t, 1164
 catch_thread, 1166
 dummy_threads_t, 1165
 kick_thread, 1165
MHAPlugin_Split::posix_threads_t, 1166
 ~posix_threads_t, 1168
 attr, 1170
 catch_condition, 1170
 catch_thread, 1168
 current_thread_priority, 1169
 current_thread_scheduler, 1169
 kick_condition, 1169
 kick_thread, 1168
 kicked, 1170
 main, 1169
 mutex, 1169
 posix_threads_t, 1168
 priority, 1170
 processing_done, 1170
 scheduler, 1170
 termination_request, 1171
 thread, 1170
 thread_start, 1169
MHAPlugin_Split::split_t, 1171
 ~split_t, 1173
 algos, 1176
 chains, 1177
 channels, 1176
 clear_chains, 1174
 collect_result, 1175
 copy_output_spec, 1175
 copy_output_wave, 1174
 delay, 1177
 framework_thread_priority, 1177
 framework_thread_scheduler, 1177
 patchbay, 1176
 prepare_, 1174
 process, 1174

release_, 1174
signal_out, 1175
spec_out, 1177
split_t, 1173
thread_platform, 1176
trigger_processing, 1175
update, 1174
wave_out, 1177
worker_thread_priority, 1176
worker_thread_scheduler, 1176
MHAPlugin_Split::splitted_part_t, 1178
~splitted_part_t, 1180
collect_result, 1182
domain, 1183
operator=, 1181
parse, 1182
plug, 1183
prepare, 1181
release, 1181
splitted_part_t, 1179, 1180
thread, 1183
trigger_processing, 1182
MHAPlugin_Split::thread_platform_t, 1183
~thread_platform_t, 1186
catch_thread, 1186
kick_thread, 1186
operator=, 1186
processor, 1187
thread_platform_t, 1184
MHAPlugin_Split::uni_processor_t, 1187
~uni_processor_t, 1188
process, 1188
MHAPluginCategory_t
 mha.hh, 1578
MHAPluginDocumentation_t
 mha.hh, 1578
mhapluginloader.cpp, 1686
mhapluginloader.h, 1686
mhapluginloader_t
 MHAParser::mhapluginloader_t, 1087
 PluginLoader::mhapluginloader_t, 1367
MHAPrepare_cb
 PluginLoader::mhapluginloader_t, 1371
MHAPrepare_t
 mha.hh, 1577
MHAProc_spec2spec_cb
 PluginLoader::mhapluginloader_t, 1372
MHAProc_spec2spec_t
 mha.hh, 1578
MHAProc_spec2wave_cb
 PluginLoader::mhapluginloader_t, 1372
MHAProc_spec2wave_t
 mha.hh, 1577
 PluginLoader::mhapluginloader_t, 1372
MHAProc_wave2spec_cb
 PluginLoader::mhapluginloader_t, 1372
MHAProc_wave2spec_t
 mha.hh, 1577
MHAProc_wave2wave_cb
 PluginLoader::mhapluginloader_t, 1372
MHAProc_wave2wave_t
 mha.hh, 1577
MHARelease_cb
 PluginLoader::mhapluginloader_t, 1371
MHARelease_t
 mha.hh, 1577
mhaserver_t, 1189
 ~mhaserver_t, 1191
 acceptor_started, 1191
 ack_fail, 1192
 ack_ok, 1192
 announce_port, 1193
 b_interactive, 1193
 logfile, 1193
 logstring, 1192
 mhaserver_t, 1190
 on_received_line, 1191
 pid_mon, 1193
 port, 1193
 run, 1192
 send_port_announcement, 1191
 set_announce_port, 1191
 start_stdin_thread, 1191
 tcpserver, 1192
mhaserver_t::tcp_server_t, 1193
 mha, 1195
 on_received_line, 1194
 tcp_server_t, 1194
MHASet_cb
 PluginLoader::mhapluginloader_t, 1372
MHASet_t
 mha.hh, 1577
MHASetcpp_cb
 PluginLoader::mhapluginloader_t, 1372
MHASetcpp_t
 mha.hh, 1577
MHASignal, 136
 copy_permuted, 149
 db2lin, 140
 db2sq, 141
 dbspl2pa, 142
 dbspl2pa2, 143
 for_each, 139

kth_smallest, 144
 limit, 144
 lin2db, 139, 140
 mean, 147
 median, 145
 pa22dbspl, 142
 pa2dbspl, 141, 142
 quantile, 147
 saveas_mat4, 148, 149
 scale, 144
 sec2smp, 144
 signal_counter, 149
 smp2sec, 143
 sq2db, 140
MHASignal::async_rmslevel_t, 1195
 async_rmslevel_t, 1196
 filled, 1198
 peaklevel, 1197
 pos, 1197
 process, 1197
 rmslevel, 1197
MHASignal::delay_spec_t, 1198
 ~delay_spec_t, 1198
 buffer, 1199
 delay, 1199
 delay_spec_t, 1198
 pos, 1199
 process, 1199
MHASignal::delay_t, 1199
 ~delay_t, 1200
 buffer, 1201
 channels, 1201
 delay_t, 1200
 delays, 1201
 inspect, 1201
 pos, 1201
 process, 1201
MHASignal::delay_wave_t, 1202
 ~delay_wave_t, 1202
 buffer, 1203
 delay, 1203
 delay_wave_t, 1202
 pos, 1203
 process, 1203
MHASignal::doublebuffer_t, 1203
 ~doublebuffer_t, 1205
 ch, 1207
 doublebuffer_t, 1204
 inner_in, 1206
 inner_out, 1206
 inner_process, 1205
 k_inner, 1206
 k_outer, 1207
 min, 1206
 outer_out, 1206
 outer_process, 1205
 this_outer_out, 1206
MHASignal::fft_t, 1207
 ~fft_t, 1208
 backward, 1209
 backward_scale, 1210
 buf_in, 1211
 buf_out, 1211
 fft_t, 1208
 fftw_plan_fft, 1211
 fftw_plan_ifft, 1211
 fftw_plan_spec2wave, 1211
 fftw_plan_wave2spec, 1211
 forward, 1209
 forward_scale, 1209
 n_im, 1210
 n_re, 1210
 nfft, 1210
 scale, 1211
 sort_fftw2spec, 1210
 sort_spec2fftw, 1210
 spec2wave, 1208, 1209
 spec2wave_scale, 1209
 wave2spec, 1208
 wave2spec_scale, 1209
MHASignal::hilbert_fftw_t, 1212
 ~hilbert_fftw_t, 1212
 buf_c_in, 1213
 buf_c_out, 1213
 buf_r_in, 1213
 buf_r_out, 1213
 hilbert, 1213
 hilbert_fftw_t, 1212
 n, 1213
 p1, 1213
 p2, 1213
 sc, 1214
MHASignal::hilbert_t, 1214
 ~hilbert_t, 1215
 h, 1215
 hilbert_t, 1215
 operator(), 1215
MHASignal::loop_wavefragment_t, 1216
 add, 1218
 b_loop, 1221
 get_mapping, 1219
 input, 1218

intern_level, 1221
is_playback_active, 1220
level_mode_t, 1217
locate_end, 1220
loop_wavefragment_t, 1218
mute, 1218
peak, 1218
playback, 1219, 1220
playback_channels, 1220
playback_mode_t, 1218
pos, 1221
relative, 1218
replace, 1218
rewind, 1220
rms, 1218
rms_limit40, 1218
set_level_db, 1220
set_level_lin, 1220

MHASignal::matrix_t, 1221
~matrix_t, 1226
cdata, 1232
complex_ofs, 1232
dimension, 1226
get_cdata, 1232
get_comm_var, 1226
get_index, 1231
get_nelements, 1227
get_nreals, 1231
get_rdata, 1231
imag, 1228–1230
is_same_size, 1227
iscomplex, 1227
matrix_t, 1223, 1225
nelements, 1232
numbytes, 1231
operator(), 1228–1230
operator=, 1226
rdata, 1232
real, 1227–1230
size, 1227
write, 1231

MHASignal::minphase_t, 1233
minphase_t, 1233
operator(), 1234
phase, 1234

MHASignal::quantizer_t, 1234
downscale, 1236
limit, 1236
operator(), 1235
quantizer_t, 1235
up_limit, 1236

upscale, 1236
MHASignal::ringbuffer_t, 1236
contained_frames, 1238
discard, 1239
next_read_frame_index, 1240
next_write_frame_index, 1240
ringbuffer_t, 1237
value, 1238
write, 1239

MHASignal::schroeder_t, 1240
down, 1242
groupdelay_t, 1241
identity, 1243
log_down, 1244
log_up, 1243
schroeder_t, 1242, 1243
sign_t, 1242
up, 1242

MHASignal::spectrum_t, 1244
~spectrum_t, 1246
copy, 1247
copy_channel, 1248
export_to, 1248
operator(), 1246
operator[], 1247
scale, 1248
scale_channel, 1250
spectrum_t, 1245, 1246
value, 1247

MHASignal::stat_t, 1250
mean, 1251
mean_std, 1251
n, 1252
push, 1251, 1252
stat_t, 1251
sum, 1252
sum2, 1252

MHASignal::subsample_delay_t, 1252
last_complex_bin, 1254
phase_gains, 1254
process, 1253, 1254
subsample_delay_t, 1253

MHASignal::uint_vector_t, 1255
~uint_vector_t, 1256
data, 1258
get_length, 1257
getdata, 1258
length, 1258
numbytes, 1258
operator=, 1257
operator==, 1257

operator[], 1257
 uint_vector_t, 1256
 write, 1258
MHASignal::waveform_t, 1259
 ~waveform_t, 1262
 assign, 1266, 1267
 assign_channel, 1267
 assign_frame, 1267
 copy, 1267, 1268
 copy_channel, 1268
 copy_from_at, 1268
 export_to, 1269
 flatten, 1262
 get_size, 1271
 limit, 1269
 operator std::vector< mha_real_t >, 1262
 operator(), 1263, 1264
 operator=, 1263
 operator[], 1263
 power, 1269
 powspec, 1270
 scale, 1270
 scale_channel, 1271
 scale_frame, 1271
 sum, 1265
 sum_channel, 1266
 sumsqr, 1266
 value, 1263, 1264
 waveform_t, 1261, 1262
MHASndFile, 150
 mhasndfile.cpp, 1687
 validator_channels, 1687
 validator_length, 1687
 write_wave, 1687
 mhasndfile.h, 1687
 write_wave, 1688
MHASndFile::sf_t, 1272
 ~sf_t, 1272
 sf, 1273
 sf_t, 1272
MHASndFile::sf_wave_t, 1273
 sf_wave_t, 1274
mhastrdomain
 PluginLoader, 157
MHAStrError_cb
 PluginLoader::mhaplugloader_t, 1372
MHAStrError_t
 mha.hh, 1578
MHATableLookup, 150
MHATableLookup::linear_table_t, 1274
 ~linear_table_t, 1276
 add_entry, 1277
 clear, 1278
 interp, 1276
 len, 1278
 linear_table_t, 1276
 lookup, 1276
 prepare, 1277
 scalefac, 1279
 set_xmax, 1277
 set_xmin, 1277
 vec_y, 1278
 vy, 1278
 xmax, 1279
 xmin, 1278
MHATableLookup::table_t, 1279
 ~table_t, 1280
 clear, 1280
 interp, 1280
 lookup, 1280
 table_t, 1280
MHATableLookup::xy_table_t, 1281
 add_entry, 1283
 clear, 1284
 get_xlimits, 1285
 interp, 1283
 lookup, 1282
 mXY, 1285
 set_xfun, 1284
 set_xyfun, 1285
 set_yfun, 1284
 xfun, 1285
 xy_table_t, 1282
 xyfun, 1285
 yfun, 1285
MHAUtils, 150
 is_denormal, 151, 152
 is_multiple_of, 151
 is_multiple_of_by_power_of_two, 151
 is_power_of_two, 151
 remove, 151
 spl2hl, 153
 strip, 151
MHAWindow, 153
 bartlett, 154
 blackman, 155
 hamming, 155
 hanning, 155
 rect, 154
MHAWindow::bartlett_t, 1286
 bartlett_t, 1287
MHAWindow::base_t, 1287

base_t, 1288
operator(), 1288
ramp_begin, 1289
ramp_end, 1289
MHAWindow::blackman_t, 1289
blackman_t, 1290
MHAWindow::fun_t, 1291
fun_t, 1291
MHAWindow::hamming_t, 1292
hamming_t, 1293
MHAWindow::hanning_t, 1293
hanning_t, 1294
MHAWindow::rect_t, 1295
rect_t, 1296
MHAWindow::user_t, 1296
user_t, 1297
mic_azimuth_degrees_vec
rohBeam::rohBeam, 1399
MIN
mha_defs.h, 1583
min
MHASignal::doublebuffer_t, 1206
spec2wave.cpp, 1697
Vector and matrix processing toolbox, 56
min_sleep_time
dropgen_t, 438
MIN_TCP_PORT
MHAIOAsterisk.cpp, 1651
MHAIOTCP.cpp, 1678
MIN_TCP_PORT_STR
MHAIOAsterisk.cpp, 1651
MHAIOTCP.cpp, 1678
minimum_fill_count
mha_drifter_fifo_t< T >, 760
minlen
plingploing::if_t, 1340
minlen_
plingploing::plingploing_t, 1343
minLim
rohBeam::rohConfig, 1407
minphase
MHAFilter::smoothspec_t, 932
minphase_t
MHASignal::minphase_t, 1233
mint_mon_t
MHPParser::mint_mon_t, 1091
mint_t
MHPParser::mint_t, 1093
minw_
wavwriter_t, 1503
minwrite
plugins::hoertech::acrec::acrec_t, 1378
wavrec_t, 1499
mismatch
level_matching::channel_pair, 650
mix
sine_cfg_t, 1429
mixer
matrixmixer::matmix_t, 715
mixw_ref
fshift_hilbert::hilbert_shifter_t, 520
mixw_shift
fshift_hilbert::hilbert_shifter_t, 520
mode
ac2osc_t, 183
addsndfile::addsndfile_if_t, 253
audiometerbackend::audiometer_if_t, 317
levelmeter_t, 659
MHA_TCP::OS_EVENT_TYPE, 814
noise_t, 1317
sine_t, 1432
smoothgains_bridge::overlapadd_if_t,
1450
modified
dc::dc_vars_t, 391
dc_simple::dc_if_t, 396
modulename
dynamiclib_t, 449
MHPParser::c_ifc_parser_t, 1047
mon
acmon::ac_monitor_t, 206
mon_complex
acmon::ac_monitor_t, 206
mon_dump
MHPParser, 127
mon_g
dc_simple::dc_if_t, 396
dc_simple::dc_t, 400
mon_l
dc_simple::dc_if_t, 396
dc_simple::dc_t, 400
mon_mat
acmon::ac_monitor_t, 206
mon_mat_complex
acmon::ac_monitor_t, 206
mon_t
rmslevel::mon_t, 1386
monitor variable, 4
monitor_t
MHPParser::monitor_t, 1095, 1096
monitors
matlab_wrapper::matlab_wrapper_t, 702

rmslevel::rmslevel_t, 1392
 mpo
 DynComp::dc_afterburn_vars_t, 456
 mpo_inv
 DynComp::dc_afterburn_rt_t, 451
 msg
 MHA_Error, 765
 mu
 gsc_adaptive_stage::gsc_adaptive_stage,
 540
 gsc_adaptive_stage::gsc_adaptive_stage_if_n
 547
 MHAFilter::adapt_filter_param_t, 863
 MHAFilter::adapt_filter_t, 867
 mu_beta
 adm_if_t, 274
 mul4f
 gtfb_simd.cpp, 1557
 multibandcompressor, 155
 multibandcompressor.cpp, 1688
 multibandcompressor::fftfb_plug_t, 1298
 bwv, 1299
 cfv, 1299
 efv, 1299
 fftfb_plug_t, 1298
 insert, 1299
 multibandcompressor::interface_t, 1300
 algo, 1302
 burn, 1302
 interface_t, 1301
 num_channels, 1302
 patchbay, 1302
 plug, 1302
 plug_sigs, 1302
 prepare, 1301
 process, 1301
 release, 1301
 update_cfg, 1301
 multibandcompressor::plugin_signals_t, 1303
 apply_gains, 1303
 gain, 1304
 plug_level, 1304
 plug_output, 1304
 plugin_signals_t, 1303
 update_levels, 1303
 mute
 MHAJack::port_t, 983
 MHASignal::loop_wavefragment_t, 1218
 mutex
 mha_fifo_posix_threads_t, 775
 MHAPlugin_Split::posix_threads_t, 1169
 MXCSR_DAZ
 gtfb_simd.cpp, 1557
 MXCSR_FTZ
 gtfb_simd.cpp, 1557
 mXY
 MHATableLookup::xy_table_t, 1285
 mylogf
 dc_afterburn.cpp, 1538
 N
 ipc_config, 677
 MHAJack::client_avg_t, 966
 MHASignal::hilbert_fftw_t, 1213
 MHASignal::stat_t, 1252
 n_channels
 mha_audio_descriptor_t, 741
 N_ERRNO
 MHA_TCP, 102
 n_freqs
 mha_audio_descriptor_t, 741
 n_im
 MHASignal::fft_t, 1210
 n_new_samples
 lsl2ac::save_var_t, 692
 n_no_update
 adaptive_feedback_canceller, 244
 nlms_t, 1308
 n_no_update_
 adaptive_feedback_canceller_config, 247
 rt_nlms_t, 1416
 n_pad1
 overlapadd::overlapadd_t, 1334
 n_pad2
 overlapadd::overlapadd_t, 1335
 n_re
 MHASignal::fft_t, 1210
 n_samples
 mha_audio_descriptor_t, 741
 n_zero
 overlapadd::overlapadd_t, 1334
 name
 ac2lsl::type_info, 179
 ac2wave_if_t, 187
 ac2wave_t, 190
 acmon::ac_monitor_t, 205
 accsave::save_var_t, 228
 fftfbpow::fftfbpow_interface_t, 495
 lsl2ac::save_var_t, 691
 MHA_AC::ac2matrix_helper_t, 721
 MHA_AC::double_t, 730
 MHA_AC::float_t, 732

MHA_AC::int_t, 733
MHA_AC::spectrum_t, 736
MHA_AC::waveform_t, 740
MHAIOPortAudio::device_info_t, 955
MHAJack::client_avg_t, 966
MHAJack::client_noncont_t, 970
MHAMultiSrc::channel_t, 993
MHAParser::entry_t, 1054
noise_psd_estimator::noise_psd_estimator_if_t, acsSave::mat4head_t, 226
names
 name_u
 nlms_t, 1307
 name_u_
 rt_nlms_t, 1416
 namelen
 names
 MHAOvIFilter::scale_var_t, 1026
 nangle
 acSteer_config, 233
 steerbf_config, 1473
 native_thread_platform_type
 split.cpp, 1703
 naudiochannels
 dc::dc_t, 387
 nbands
 coherence::cohflt_t, 352
 combc_t, 360
 dc::dc_t, 387
 dc_simple::dc_t, 400
 dc_simple::level_smoothen_t, 408
 DynComp::gaintable_t, 460
 fftfilterbank::fftfb_interface_t, 505
 gtfb_simple_rt_t, 567
 MHAFilter::thirdoctave_analyzer_t, 933
 MHAOvIFilter::fspacing_t, 1016
 nbits
 calibrator_variables_t, 342
 nch
 dc::dc_t, 387
 shadowfilter_begin::cfg_t, 1421
 shadowfilter_begin::shadowfilter_begin_t,
 1424
 spec_fader_t, 1465
 nch_out
 shadowfilter_end::cfg_t, 1425
 nchan
 acSteer_config, 233
 gsc_adaptive_stage::gsc_adaptive_stage,
 540
 smooth_cepstrum::smooth_cepstrum_t,
 1441
 steerbf_config, 1473
 nchan_block
 rohBeam::rohConfig, 1405
 nchannels
 DynComp::gaintable_t, 460
 equalize::cfg_t, 463

fftfilterbank::fftfb_interface_t, 505
 lsl2ac::lsl2ac_t, 684
 lsl2ac::save_var_t, 691
 MHAFilter::adapt_filter_state_t, 864
 MHAFilter::adapt_filter_t, 867
 MHAFilter::iir_filter_t, 900
 MHAFilter::smoothspec_t, 931
 MHAFilter::thirdoctave_analyzer_t, 933
 nchannels_file_in
 io_file_t, 607
 nchannels_in
 io_file_t, 607
 io_parser_t, 618
 mconv::MConv, 719
 MHAFilter::partitioned_convolution_t, 916
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 978
 MHAPlugin_Resampling::resampling_t,
 1156
 nchannels_out
 fw_t, 528
 io_file_t, 607
 io_parser_t, 618
 mconv::MConv, 719
 MHAFilter::partitioned_convolution_t, 917
 MHAIOPortAudio::io_portaudio_t, 960
 MHAJack::client_t, 978
 MHAPlugin_Resampling::resampling_t,
 1157
 NDEBUG
 rohBeam.hh, 1694
 ndim
 acsave::save_var_t, 228
 needs_write
 MHA_TCP::Connection, 809
 neigh
 acPooling_wave_config, 218
 neighbourhood
 acPooling_wave, 214
 nelements
 MHASignal::matrix_t, 1232
 nested_lock
 MHAParser::base_t, 1038
 new_name
 lsl2ac::save_var_t, 690
 newgains
 fader_if_t, 488
 next
 mha_rt_fifo_element_t< T >, 788
 MHAPlugin::cfg_node_t< runtime_cfg_t
 >, 1139
 next_except_str
 mha_errno.c, 1585
 next_message
 mha_tcp::buffered_socket_t, 800
 next_read_frame_index
 MHASignal::ringbuffer_t, 1240
 next_write_frame_index
 MHASignal::ringbuffer_t, 1240
 nextXpYf
 rohBeam::rohConfig, 1407
 nfft
 MHASignal::fft_t, 1210
 overlapadd::overlapadd_if_t, 1330
 shadowfilter_end::cfg_t, 1425
 spec2wave_t, 1464
 wave2spec_if_t, 1490
 nfft_
 MHAOvIFilter::fspacing_t, 1017
 nframes
 acsave::save_var_t, 228
 nfreq
 acSteer_config, 233
 gsc_adaptive_stage::gsc_adaptive_stage,
 540
 rohBeam::rohConfig, 1404
 smooth_cepstrum::smooth_cepstrum_t,
 1440
 steerbf_config, 1473
 nlms_t, 1305
 algo, 1308
 c, 1307
 estimtype, 1307
 lambda_smoothing_power, 1308
 n_no_update, 1308
 name_d, 1307
 name_e, 1308
 name_f, 1308
 name_u, 1307
 nlms_t, 1306
 normtype, 1307
 ntaps, 1307
 patchbay, 1308
 prepare, 1306
 process, 1306
 release, 1306
 rho, 1307
 update, 1307
 nlms_wave.cpp, 1688
 ESTIM_CUR, 1690
 ESTIM_PREV, 1689
 ESTIMATION_TYPES, 1689

make_friendly_number_by_limiting, 1690
NORM_DEFAULT, 1689
NORM_NONE, 1689
NORM_SUM, 1689
NORMALIZATION_TYPES, 1689
nm
 lpc_burglattice_config, 674
no_iter
 adaptive_feedback_canceller_config, 247
 rt_nlms_t, 1417
noise.cpp, 1690
noise_field_model
 rohBeam::rohBeam, 1400
noise_integrate_hrtf
 rohBeam::rohBeam, 1397
noise_psd_estimator, 155
noise_psd_estimator.cpp, 1690
 POWSPEC_FACTOR, 1691
noise_psd_estimator::noise_psd_estimator_if_t, noiseFuncPtr
 1309
 alphaPH1mean, 1310
 alphaPSD, 1311
 name, 1311
 noise_psd_estimator_if_t, 1310
 patchbay, 1311
 prepare, 1310
 process, 1310
 q, 1311
 update_cfg, 1310
 xiOptDb, 1311
noise_psd_estimator::noise_psd_estimator_t, 1311
 alphaPH1mean_, 1314
 alphaPSD_, 1314
 estimateDebug, 1314
 frameno, 1315
 GLRDebug, 1313
 GLRexp, 1314
 inputPow, 1313
 inputSpec, 1314
 insert, 1312
 logGLRFact, 1314
 noise_psd_estimator_t, 1312
 noisePow, 1313
 noisyPer, 1313
 PH1Debug, 1313
 PH1mean, 1313
 priorFact, 1314
 process, 1312
 snrPost1Debug, 1313
 xiOpt, 1314
noise_psd_estimator_if_t
 noise_psd_estimator::noise_psd_estimator_if_t, 1310
noise_psd_estimator_t
 noise_psd_estimator::noise_psd_estimator_t, 1312
noise_t, 1315
 frozennoise_length, 1317
 lev, 1317
 mode, 1317
 noise_t, 1316
 patchbay, 1317
 prepare, 1316
 process, 1316
 seed, 1317
 update_cfg, 1316
noise_type_t
 speechnoise_t, 1467
noiseFuncPtr
 rohBeam::rohBeam, 1399
noiseModelExport
 rohBeam::rohBeam, 1401
noisePow
 noise_psd_estimator::noise_psd_estimator_t, 1313
 smooth_cepstrum::smooth_cepstrum_t, 1442
noisePow_name
 smooth_cepstrum::smooth_cepstrum_if_t, 1437
 smooth_cepstrum::smooth_params, 1448
noisyPer
 noise_psd_estimator::noise_psd_estimator_t, 1313
non_empty_partitions
 MHAFilter::transfer_function_t, 937
 MHAFilter::transfer_matrix_t, 939
nondefault_labels
 altpugs_t, 304
NORELEASE_WARNING
 mhomain.cpp, 1685
norm
 lpc, 663
 lpc_config, 676
NORM_DEFAULT
 nlms_wave.cpp, 1689
NORM_NONE
 nlms_wave.cpp, 1689
norm_phase
 gtfb_analyzer::gtfb_analyzer_cfg_t, 551
 gtfb_analyzer::gtfb_analyzer_t, 555

gtfb_simd_cfg_t, 559
 gtfb_simd_t, 563
NORM_SUM
 nlms_wave.cpp, 1689
NORMALIZATION_TYPES
 nlms_wave.cpp, 1689
normalize
 Complex arithmetics in the openMHA, 68
 MHAOvIFilter::ffftfb_vars_t, 1009
normtype
 nlms_t, 1307
not_in_use
 MHAPlugin::cfg_node_t< runtime_cfg_t >, 1140
not_zero
 dc_simple, 88
notify
 MHAParser::base_t, 1036
notify_release
 io_asterisk_t, 603
 io_tcp_t, 641
notify_start
 io_asterisk_t, 603
 io_tcp_t, 640
notify_stop
 io_asterisk_t, 603
 io_tcp_t, 641
now_index
 MHAFilter::polyphase_resampling_t, 925
npad1
 spec2wave_t, 1463
 wave2spec_t, 1496
npad2
 spec2wave_t, 1463
 wave2spec_t, 1496
nperiods
 alsa_dev_par_parser_t, 290
nrefmic
 acSteer, 231
 acSteer_config, 233
nrep
 MHAJack::client_avg_t, 966
nsamples
 lsl2ac::lsl2ac_t, 684
 lsl2ac::save_var_t, 691
nsteerchan
 acSteer, 231
 acSteer_config, 233
ntaps
 adaptive_feedback_canceller, 243
 adaptive_feedback_canceller_config, 246
MHAFilter::adapt_filter_state_t, 864
MHAFilter::adapt_filter_t, 867
 nlms_t, 1307
 rt_nlms_t, 1415
ntoh
 io_asterisk_sound_t, 598
 io_tcp_sound_t, 634
ntracks
 shadowfilter_begin::cfg_t, 1422
 shadowfilter_begin::shadowfilter_begin_t, 1424
 shadowfilter_end::cfg_t, 1425
null_data
 mha_drifter_fifo_t< T >, 762
num_AC
 acConcat_wave, 201
num_accepted_connections
 mha_tcp::server_t, 823
num_adms
 adm_rtconfig_t, 278
num_bins
 equalize::cfg_t, 463
num_brackets
 MHAParser::StrCnv, 129
num_channels
 ac_mul_t, 195
 calibrator_variables_t, 343
 DynComp::gaintable_t, 461
 mha_spec_t, 794
 mha_wave_t, 840
 MHAFilter::blockprocessing_polyphase_resampling_t, 871
 multibandcompressor::interface_t, 1302
 plugins::hoertech::acrec::acwriter_t, 1384
NUM_ENTR_LTASS
 speechnoise.cpp, 1699
NUM_ENTR_MHAORIG
 speechnoise.cpp, 1699
NUM_ENTR_OLNOISE
 speechnoise.cpp, 1699
num_entries
 ac2lsl::save_var_base_t, 171
 ac2lsl::save_var_t< mha_complex_t >, 177
 ac2lsl::save_var_t< T >, 174
 comm_var_t, 361
 testplugin::ac_parser_t, 1476
num_F
 DynComp::gaintable_t, 461
num_frames
 ac_mul_t, 195

mha_spec_t, 794
mha_wave_t, 840
num_inchannels
 io_asterisk_sound_t, 599
 io_tcp_sound_t, 636
num_L
 DynComp::gaintable_t, 461
num_outchannels
 io_asterisk_sound_t, 599
 io_tcp_sound_t, 636
num_xruns
 MHAJack::client_t, 977
numbytes
 MHASignal::matrix_t, 1231
 MHASignal::uint_vector_t, 1258
numchannels
 acConcat_wave, 201
 addsndfile::addsndfile_if_t, 253
numDevices
 MHAIOPortAudio::device_info_t, 955
numsamples
 acPooling_wave, 213
 acTransform_wave, 237
numSamples_AC
 acConcat_wave_config, 203
nupsample
 doasvm_feature_extraction, 428
nvars
 acsave::cfg_t, 224
nwnd
 overlapadd::overlapadd_if_t, 1330
 wave2spec_if_t, 1490
 wave2spec_t, 1495
nwndshift
 spec2wave_t, 1464
 wave2spec_t, 1495
nyquist_ratio
 MHAPlugin_Resampling::resampling_if_t, 1154
o1_ar_filter_t
 MHAFilter::o1_ar_filter_t, 905
o1_lp_coeffs
 MHAFilter, 106
o1flt_lowpass_t
 MHAFilter::o1flt_lowpass_t, 909
o1flt_maxtrack_t
 MHAFilter::o1flt_maxtrack_t, 911
o1flt_mintrack_t
 MHAFilter::o1flt_mintrack_t, 913
ob
 lsl2ac::save_var_t, 691
observe
 MHA_TCP::Event_Watcher, 812
observed_by
 MHA_TCP::Wakeup_Event, 836
observers
 MHA_TCP::Wakeup_Event, 837
od
 MHAFilter::adapt_filter_state_t, 865
offset
 acTransform_wave_config, 239
 dc::dc_t, 386
 dc::dc_vars_t, 390
ola_powspec_scale
 smooth_cepstrum::smooth_cepstrum_t, 1441
old_algos
 mhachain::chain_base_t, 843
olnoise
 speechnoise_t, 1467
on_configuration_update
 double2acvar::double2acvar_t, 435
on_model_param_valuechanged
 gsc_adaptive_stage::gsc_adaptive_stage_if, 547
 rohBeam::rohBeam, 1398
 smooth_cepstrum::smooth_cepstrum_if_t, 1435
on_preadaccess
 example3_t, 474
 example4_t, 478
on_received_line
 mha_tcp::server_t, 821
 mhaserver_t, 1191
 mhaserver_t::tcp_server_t, 1194
on_scale_ch_readaccess
 example3_t, 474
 example4_t, 478
on_scale_ch_valuechanged
 example3_t, 474
 example4_t, 478
on_scale_ch_writeaccess
 example3_t, 473
 example4_t, 478
on_set_algos
 altconfig_t, 297
on_set_select
 altconfig_t, 297
on_writeaccess
 matlab_wrapper::callback, 693
op
 MHAParser::expression_t, 1056

op_query
 MHParse::base_t, 1032
 MHParse::c_ifc_parser_t, 1046
 MHParse::monitor_t, 1096
 MHParse::parser_t, 1101

op_setval
 MHParse::base_t, 1032
 MHParse::bool_t, 1044
 MHParse::c_ifc_parser_t, 1046
 MHParse::complex_t, 1053
 MHParse::float_t, 1060
 MHParse::int_t, 1065
 MHParse::kw_t, 1072
 MHParse::mcomplex_t, 1077
 MHParse::mfloat_t, 1082
 MHParse::mint_t, 1094
 MHParse::parser_t, 1101
 MHParse::string_t, 1112
 MHParse::variable_t, 1114
 MHParse::vcomplex_t, 1118
 MHParse::vfloat_t, 1124
 MHParse::vint_t, 1128
 MHParse::vstring_t, 1132

op_subparse
 MHParse::base_t, 1032
 MHParse::c_ifc_parser_t, 1046
 MHParse::parser_t, 1100

opact_map_t
 MHParse, 125

opact_t
 MHParse, 125

operator std::vector< mha_real_t >
 MHASignal::waveform_t, 1262

operator!=
 Complex arithmetics in the openMHA, 67

operator<
 Complex arithmetics in the openMHA, 69

operator<<
 mha_signal.hh, 1635

operator>>
 mha_signal.hh, 1636

operator*
 Complex arithmetics in the openMHA, 64,
 65

operator*=
 Complex arithmetics in the openMHA, 64
 Vector and matrix processing toolbox, 50,
 51

operator^=
 Vector and matrix processing toolbox, 52

operator()
 rmslevel::mon_t, 1386

dc_simple::dc_t::line_t, 401
 hanning_ramps_t, 574
 MHAEvents::emitter_t, 859
 MHAFilter::gamma_flt_t, 892, 893
 MHAFilter::iir_ord1_real_t, 902
 MHAFilter::o1_ar_filter_t, 906
 MHASignal::hilbert_t, 1215
 MHASignal::matrix_t, 1228–1230
 MHASignal::minphase_t, 1234
 MHASignal::quantizer_t, 1235
 MHASignal::spectrum_t, 1246
 MHASignal::waveform_t, 1263, 1264
 MHAWindow::base_t, 1288

operator+
 Complex arithmetics in the openMHA, 62,
 63

operator+=
 Complex arithmetics in the openMHA, 62
 Vector and matrix processing toolbox, 50,
 52

operator-
 Complex arithmetics in the openMHA, 63,
 66

operator-=
 Complex arithmetics in the openMHA, 63
 Vector and matrix processing toolbox, 50

operator/
 Complex arithmetics in the openMHA, 65,
 66

operator/=
 Complex arithmetics in the openMHA, 65,
 66
 Vector and matrix processing toolbox, 51,
 52

operator=

- equalize::cfg_t, 462
- gtfb_simd_cfg_t, 558
- lsl2ac::save_var_t, 688
- MHA_AC::acspace2matrix_t, 726
- MHA_Error, 765
- mha_fifo_t< T >, 780
- mha_fifo_thread_platform_t, 786
- MHAPlugin_Split::domain_handler_t,
 1160
- MHAPlugin_Split::splitted_part_t, 1181
- MHAPlugin_Split::thread_platform_t,
 1186
- MHASignal::matrix_t, 1226
- MHASignal::uint_vector_t, 1257
- MHASignal::waveform_t, 1263

rohBeam::rohConfig, 1404
smooth_cepstrum::smooth_cepstrum_t, 1440
operator==
 Complex arithmetics in the openMHA, 66
 MHASignal::uint_vector_t, 1257
operator[]
 MHA_AC::acspace2matrix_t, 726, 727
 MHASignal::spectrum_t, 1247
 MHASignal::uint_vector_t, 1257
 MHASignal::waveform_t, 1263
operators
 MHAParser::base_t, 1038
oplist
 MHAParser::base_t, 1037
order
 gtfb_analyzer::gtfb_analyzer_cfg_t, 551
 gtfb_analyzer::gtfb_analyzer_t, 555
 gtfb_simd_cfg_t, 558
 gtfb_simd_t, 563
 gtfb_simple_t, 572
 lpc_config, 677
original_content
 mha_stash_environment_variable_t, 796
origname
 PluginLoader::config_file_splitter_t, 1361
os_event
 MHA_TCP::Wakeup_Event, 837
os_event_valid
 MHA_TCP::Wakeup_Event, 838
osc2ac.cpp, 1691
osc2ac_t, 1318
 host, 1320
 osc2ac_t, 1319
 patchbay, 1320
 port, 1320
 prepare, 1319
 process, 1319
 release, 1319
 setlock, 1320
 size, 1320
 srv, 1320
 vars, 1320
osc_data
 osc_variable_t, 1327
osc_server_t, 1321
 ~osc_server_t, 1321
 ac_insert, 1322
 error_h, 1322
 insert_variable, 1322
 is_running, 1323
lost, 1323
osc_server_t, 1321
pVars, 1322
server_start, 1322
server_stop, 1322
sync_osc2ac, 1322
osc_variable_t, 1323
 ac_data, 1326
 ac_insert, 1325
 acname, 1326
 handler, 1325, 1326
 name_, 1327
 osc_data, 1327
 osc_variable_t, 1324
 oscaddr, 1326
 sync_osc2ac, 1325
oscaddr
 osc_variable_t, 1326
out
 adm_if_t, 273
 delaysum::delaysum_wave_t, 415
out_buf
 overlapadd::overlapadd_t, 1334
 spec2wave_t, 1464
out_cfg
 rohBeam::rohConfig, 1405
out_chunk
 MHAFilter::thirdoctave_analyzer_t, 934
out_chunk_im
 MHAFilter::thirdoctave_analyzer_t, 935
out_spec
 shadowfilter_begin::cfg_t, 1421
 shadowfilter_end::cfg_t, 1426
outbuf
 MHA_TCP::Connection, 809
outch
 mconv::MConv, 719
 MHAJack::client_t, 979
outchannel
 audiometerbackend::audiometer_if_t, 318
outchannels
 combc_if_t, 358
outer2inner_resampling
 MHAPlugin_Resampling::resampling_t, 1157
outer_ac
 analysepath_t, 308
outer_ac_copy
 analysepath_t, 308
outer_error
 mha_dblbuf_t< FIFO >, 754

outer_fragsize
 MHAPlugin_Resampling::resampling_t,
 1156

outer_out
 MHASignal::doublebuffer_t, 1206

outer_output
 dbasync_native::dbasync_t, 379

outer_process
 dbasync_native::dbasync_t, 378
 MHASignal::doublebuffer_t, 1205

outer_size
 mha_dbdbuf_t< FIFO >, 752

outer_srate
 MHAPlugin_Resampling::resampling_t,
 1156

outfile
 plugins::hoertech::acrec::acwriter_t, 1384

output
 delaysum_spec::delaysum_t, 419
 gtfb_simple_rt_t, 567
 io_parser_t, 619
 mha_dbdbuf_t< FIFO >, 752
 MHAJack::port_t, 981

output_cfg
 MHAPlugin::plugin_t< runtime_cfg_t >,
 1150

output_cfg_
 MHAPlugin::plugin_t< runtime_cfg_t >,
 1151

output_channels
 mha_dbdbuf_t< FIFO >, 753

output_data
 io_asterisk_sound_t, 599

output_domain
 PluginLoader::mhapluginloader_t, 1368

output_fifo
 mha_dbdbuf_t< FIFO >, 753

output_partitions
 MHAFilter::partitioned_convolution_t, 917

output_portnames
 MHAJack::client_t, 980

output_sample_format
 io_file_t, 608
 wavrec_t, 1499

output_signal
 MHAPlugin_Resampling::resampling_t,
 1157

output_signal_spec
 MHAFilter::partitioned_convolution_t, 918

output_signal_wave
 MHAFilter::partitioned_convolution_t, 918

output_spec
 testplugin::signal_parser_t, 1484

output_type
 plugins::hoertech::acrec::acwriter_t, 1381

output_wave
 testplugin::signal_parser_t, 1484

outputchannels
 MHAFilter::fftfilterbank_t, 885

outSpec
 rohBeam::rohConfig, 1406
 steerbf_config, 1473

overlap_save_filterbank_analytic_t
 MHAOvIFilter::overlap_save_filterbank_analytic_t,
 1018

overlap_save_filterbank_t
 MHAOvIFilter::overlap_save_filterbank_t,
 1021

overlapadd, 156

overlapadd.cpp, 1691

overlapadd.hh, 1691

overlapadd::overlapadd_if_t, 1327
 ~overlapadd_if_t, 1329
 algo, 1331
 cf_in, 1331
 cf_out, 1331
 nfft, 1330
 nwnd, 1330
 overlapadd_if_t, 1328
 plugloader, 1331
 postscale, 1331
 prepare, 1329
 prescale, 1331
 process, 1329
 release, 1329
 setlock, 1329
 strict_window_ratio, 1330
 update, 1329
 window, 1330
 wndexp, 1330
 wndpos, 1330
 zerowindow, 1330

overlapadd::overlapadd_t, 1332
 ~overlapadd_t, 1332
 calc_out, 1334
 fft, 1333
 n_pad1, 1334
 n_pad2, 1335
 n_zero, 1334
 out_buf, 1334
 overlapadd_t, 1332
 postwnd, 1334

prewnd, 1333
spec2wave, 1333
spec_in, 1334
wave2spec, 1333
wave2spec_apply_window, 1333
wave2spec_compute_fft, 1333
wave2spec_hop_forward, 1333
wave_in1, 1334
wave_out1, 1334
write_buf, 1334
overlapadd_if_t
overlapadd::overlapadd_if_t, 1328
smoothgains_bridge::overlapadd_if_t, 1449
overlapadd_t
overlapadd::overlapadd_t, 1332
overrun_behavior
 lsl2ac, 97
 lsl2ac::lsl2ac_t, 684
ovltype
 MHAOvlFilter::fftfb_vars_t, 1008
oy
 MHAFilter::adapt_filter_state_t, 865

P
 gsc_adaptive_stage::gsc_adaptive_stage, 543
p
 acPooling_wave_config, 217
 doasvm_classification_config, 424
 pluginbrowser_t, 1355
p1
 MHASignal::hilbert_fftw_t, 1213
p2
 MHASignal::hilbert_fftw_t, 1213
p_biased
 acPooling_wave_config, 217
p_biased_name
 acPooling_wave, 214
p_in
 io_alsa_t, 583
p_max
 acPooling_wave_config, 217
 doasvm_classification_config, 424
p_name
 acPooling_wave, 214
 doasvm_classification, 423
p_out
 io_alsa_t, 583
p_parser
 acmon::ac_monitor_t, 206
P_Sum
 rt_nlms_t, 1416
pa22dbspl
 MHASignal, 142
pa2dbspl
 MHASignal, 141, 142
pairings
 level_matching::level_matching_config_t, 652
params
 smooth_cepstrum::smooth_cepstrum_t, 1440
parent
 matlab_wrapper::callback, 694
 MHAParser::base_t, 1038
parent_
 MHAParser::mhapluginloader_t, 1088
parse
 altplugs_t, 302
 io_asterisk_t, 602
 io_tcp_t, 640
 io_wrapper, 642
 MHAParser::base_t, 1031, 1032
 MHAParser_Split::splitted_part_t, 1182
 plug_wrapper, 1348
 plug_wrapperl, 1350
 PluginLoader::fourway_processor_t, 1365
 PluginLoader::mhapluginloader_t, 1368
parse_1_complex
 mha_parser.cpp, 1610
parse_1_float
 mha_parser.cpp, 1608
parser
 io_asterisk_t, 602
 io_tcp_t, 640
 mhachain::plugs_t, 848
parser_algos
 altconfig_t, 299
parser_int_dyn, 1335
 parser_int_dyn, 1336
 set_max_angle_ind, 1336
parser_plugs
 altplugs_t, 303
parser_t
 AuditoryProfile::parser_t, 327
 MHAParser::parser_t, 1099
parserFriendlyName
 MHAIOPortAudio, 110
parsename
 latex_doc_t, 646
parserstate
 fw_t, 528

partitioned_convolution_t
 MHAFilter::partitioned_convolution_t, 915
 partitions
 MHAFilter::transfer_function_t, 936
 MHAFilter::transfer_matrix_t, 939
 PASCALE
 levelmeter.cpp, 1565
 PATCH_VAR
 acConcat_wave.cpp, 1519
 acPooling_wave.cpp, 1520
 acSteer.cpp, 1523
 acTransform_wave.cpp, 1523
 adaptive_feedback_canceller.cpp, 1524
 doasvm_classification.cpp, 1541
 doasvm_feature_extraction.cpp, 1542
 level_matching.cpp, 1564
 lpc.cpp, 1565
 lpc_bl_predictor.cpp, 1566
 lpc_burg-lattice.cpp, 1567
 smooth_cepstrum.cpp, 1696
 steerbf.cpp, 1703
 patchbay
 ac2lsl::ac2lsl_t, 165
 ac2osc_t, 184
 ac2wave_if_t, 188
 acConcat_wave, 202
 acmon::acmon_t, 210
 acPooling_wave, 215
 acsave::acsave_t, 223
 acSteer, 232
 acTransform_wave, 237
 adaptive_feedback_canceller, 244
 addsndfile::addsndfile_if_t, 254
 adm_if_t, 275
 altconfig_t, 299
 altplugs_t, 304
 analysispath_if_t, 312
 audiometerbackend::audiometer_if_t, 318
 AuditoryProfile::parser_t::fmap_t, 331
 calibrator_t, 341
 coherence::cohfilt_if_t, 349
 complex_scale_channel_t, 364
 cpupload::cpupload_if_t, 369
 db_if_t, 371
 dc::dc_if_t, 383
 dc_simple::dc_if_t, 397
 delay::interface_t, 410
 delaysum::delaysum_wave_if_t, 413
 delaysum_spec::delaysum_spec_if_t, 418
 doasvm_classification, 423
 doasvm_feature_extraction, 428
 double2acvar::double2acvar_t, 435
 dropgen_t, 438
 DynComp::dc_afterburn_t, 454
 equalize::freqgains_t, 465
 example3_t, 475
 example4_t, 480
 example6_t, 483
 fader_if_t, 487
 fader_wave::fader_wave_if_t, 490
 fftfbpow::fftfbpow_interface_t, 496
 fftfilter::interface_t, 502
 fftfilterbank::fftfb_interface_t, 505
 fshift::fshift_t, 513
 fshift_hilbert::frequency_translator_t, 516
 fw_t, 530
 gain::gain_if_t, 534
 gsc_adaptive_stage::gsc_adaptive_stage_if, 547
 gtfb_analyzer::gtfb_analyzer_t, 554
 gtfb_simd_t, 562
 io_alsa_t, 583
 io_parser_t, 619
 level_matching::level_matching_t, 656
 levelmeter_t, 659
 lpc, 663
 lpc_bl_predictor, 666
 lpc_burglattice, 672
 lsl2ac::lsl2ac_t, 684
 matlab_wrapper::matlab_wrapper_t, 701
 matrixmixer::matmix_t, 715
 mconv::MConv, 719
 mhachain::chain_base_t, 844
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 MHAIOPortAudio::io_portaudio_t, 962
 MHAPlugin_Split::split_t, 1176
 multibandcompressor::interface_t, 1302
 nlms_t, 1308
 noise_psd_estimator::noise_psd_estimator_if_t, 1311
 noise_t, 1317
 osc2ac_t, 1320
 plingloing::if_t, 1339
 plugin_interface_t, 1353
 plugins::hoertech::acrec::acrec_t, 1378
 rmslevel::rmslevel_if_t, 1389
 rohBeam::rohBeam, 1401
 route::interface_t, 1410
 sine_t, 1432
 smooth_cepstrum::smooth_cepstrum_if_t, 1438

smoothgains_bridge::overlapadd_if_t, 1450
softclip_t, 1455
steerbf, 1471
testplugin::ac_parser_t, 1477
testplugin::if_t, 1482
wavrec_t, 1499
windnoise::if_t, 1510
windowselector_t, 1516
path
 addsndfile::addsndfile_if_t, 252
pcm
 alsa_base_t, 288
pcm_format
 alsa_t< T >, 294
pcmlink
 io_alsa_t, 583
peak
 levelmeter_t, 659
 MHASignal::loop_wavefragment_t, 1218
peaklevel
 calibrator_variables_t, 342
 mha_channel_info_t, 744
 MHASignal::async_rmslevel_t, 1197
peek_config
 MHAPlugin::config_t< runtime_cfg_t >, 1144
peer_addr
 MHA_TCP::Connection, 810
peer_address
 io_asterisk_parser_t, 595
 io_tcp_parser_t, 631
peer_port
 io_asterisk_parser_t, 595
 io_tcp_parser_t, 631
period
 droptect_t, 442
permute
 ac_proc::interface_t, 198
pfragmentsize
 fw_vars_t, 532
PH1Debug
 noise_psd_estimator::noise_psd_estimator_t, 1313
PH1mean
 noise_psd_estimator::noise_psd_estimator_t, 1313
phase
 cpupload::cpupload_cfg_t, 366
 MHASignal::minphase_t, 1234
phase_correction
 MHAFilter::gamma_flt_t, 893
phase_div_2pi
 sine_t, 1432
phase_gains
 MHASignal::subsample_delay_t, 1254
phase_increment_div_2pi
 sine_cfg_t, 1429
phasemode
 fshift_hilbert::frequency_translator_t, 517
phasemode
 MHAOvlFilter::overlap_save_filterbank_t::vars_t, 1023
phasereconstruction
 rohBeam::rohConfig, 1404
PI
 ADM, 82
 hann.cpp, 1562
pid_mon
 mhaserver_t, 1193
pinchannels
 fw_vars_t, 532
pink
 speechnoise_t, 1467
pipe
 MHA_TCP::Async_Notify, 798
pitch
 plingploing::if_t, 1339
pitch_
 plingploing::plingploing_t, 1343
pitch_set_first
 smooth_cepstrum::smooth_cepstrum_t, 1444
pitch_set_last
 smooth_cepstrum::smooth_cepstrum_t, 1444
plan_spec2analytic
 fshift_hilbert::hilbert_shifter_t, 520
plateau
 MHAOvlFilter::fftfb_vars_t, 1008
playback
 MHASignal::loop_wavefragment_t, 1219, 1220
 playback_channels
 MHASignal::loop_wavefragment_t, 1220
 playback_mode_t
 MHASignal::loop_wavefragment_t, 1218
 plingploing, 156
 drand, 156
 plingploing.cpp, 1691
 plingploing::if_t, 1337
 bassmod, 1340

bassperiod, 1341
 bpm, 1340
 fun1_key, 1339
 fun1_range, 1340
 fun2_key, 1340
 fun2_range, 1340
 if_t, 1338
 level, 1339
 maxlen, 1340
 minlen, 1340
 patchbay, 1339
 pitch, 1339
 prepare, 1339
 process, 1339
 update, 1339
plingploing::plingploing_t, 1341
 alph, 1345
 bass, 1343
 bassmod_, 1345
 bassperiod_, 1345
 bt, 1343
 cf, 1343
 dist, 1344
 dist1, 1344
 dur_, 1343
 freq, 1344
 fun1, 1344
 fun1_key, 1344
 fun1_range, 1344
 fun2, 1344
 fun2_key, 1344
 fun2_range, 1344
 hann1, 1345
 hann2, 1345
 len, 1343
 level, 1345
 maxlen_, 1343
 minlen_, 1343
 pitch_, 1343
plingploing_t, 1342
 process, 1342
 rms, 1345
 t, 1343
plingploing_t
plingploing::plingploing_t, 1342
plug
 ac_proc::interface_t, 198
 analysispath_if_t, 313
 gtfb_simple_t, 572
 matlab_wrapper::matlab_wrapper_t, 701
 MHAParser::mhapluginloader_t, 1088
 MHAPlugin_Split::splitted_part_t, 1183
 multibandcompressor::interface_t, 1302
 testplugin::if_t, 1482
plug_level
 multibandcompressor::plugin_signals_t,
 1304
plug_output
 multibandcompressor::plugin_signals_t,
 1304
plug_sigs
 multibandcompressor::interface_t, 1302
plug_t, 1346
 ~plug_t, 1346
 get_ac, 1347
 get_handle, 1347
 get_process_spec, 1347
 get_process_wave, 1346
 plug_t, 1346
plug_wrapper, 1347
 ~plug_wrapper, 1348
 get_categories, 1348
 get_documentation, 1348
 has_parser, 1348
 has_process, 1349
 parse, 1348
 plug_wrapper, 1348
plug_wrapperl, 1349
 ~plug_wrapperl, 1350
 get_categories, 1350
 get_documentation, 1350
 has_parser, 1350
 has_process, 1351
 parse, 1350
 plug_wrapperl, 1350
plugin_categories
 PluginLoader::mhapluginloader_t, 1373
plugin_documentation
 PluginLoader::mhapluginloader_t, 1373
plugin_extension
 pluginbrowser_t, 1355
plugin_interface_t, 1351
 factor, 1353
 patchbay, 1353
 plugin_interface_t, 1352
 prepare, 1352
 process, 1352
 scale_ch, 1353
 update_cfg, 1353
plugin_macro
 latex_doc_t, 647
plugin_paths

fw_t, 529
plugin_signals_t
 multibandcompressor::plugin_signals_t,
 1303
plugin_t
 MHAPlugin::plugin_t< runtime_cfg_t >, 1148
pluginbrowser.cpp, 1692
pluginbrowser.h, 1692
pluginbrowser_t, 1353
 add_plugin, 1354
 add_plugins, 1354
 clear_plugins, 1354
 get_paths, 1354
 get_plugins, 1355
 library_paths, 1355
 p, 1355
 plugin_extension, 1355
 pluginbrowser_t, 1354
 plugins, 1355
 scan_plugin, 1354
 scan_plugins, 1354
plugindescription_t, 1356
 categories, 1356
 documentation, 1356
 fullname, 1356
 name, 1356
 queries, 1357
 query_cmds, 1357
 spec2spec, 1357
 spec2wave, 1357
 wave2spec, 1356
 wave2wave, 1356
pluginlib_t, 1357
 ~pluginlib_t, 1358
 pluginlib_t, 1358
 resolve, 1359
PluginLoader, 157
 mhaconfig_compare, 157
 mhastrdomain, 157
PluginLoader::config_file_splitter_t, 1359
 config_file_splitter_t, 1360
 configfile, 1361
 configname, 1361
 get_configfile, 1361
 get_configname, 1360
 get_libname, 1360
 get_origname, 1360
 libname, 1361
 origname, 1361
PluginLoader::fourway_processor_t, 1362
 ~fourway_processor_t, 1363
 parse, 1365
 prepare, 1364
 process, 1363, 1364
 release, 1365
 PluginLoader::mhapluginloader_t, 1365
 ~mhapluginloader_t, 1368
 ac, 1371
 b_check_version, 1373
 b_is_prepared, 1373
 cf_input, 1372
 cf_output, 1373
 get_categories, 1370
 get_documentation, 1370
 getfullname, 1369
 has_parser, 1368
 has_process, 1368
 input_domain, 1368
 is_prepared, 1370
 lib_data, 1371
 lib_err, 1371
 lib_handle, 1371
 mha_test_struct_size, 1370
 MHADestroy_cb, 1371
 MHAGetVersion_cb, 1371
 MHAInit_cb, 1371
 mhapluginloader_t, 1367
 MHAPrepare_cb, 1371
 MHAProc_spec2spec_cb, 1372
 MHAProc_spec2wave_cb, 1372
 MHAProc_wave2spec_cb, 1372
 MHAProc_wave2wave_cb, 1372
 MHARelease_cb, 1371
 MHASet_cb, 1372
 MHASetcpp_cb, 1372
 MHAStrError_cb, 1372
 output_domain, 1368
 parse, 1368
 plugin_categories, 1373
 plugin_documentation, 1373
 prepare, 1368
 process, 1369
 release, 1369
 resolve_and_init, 1370
 test_error, 1370
 test_version, 1370
 pluginloader_t, 1373
 ~pluginloader_t, 1374
 pluginloader_t, 1374
 plugins, 158
 fw_t, 529

pluginbrowser_t, 1355
 plugins::hoertech, 158
 plugins::hoertech::acrec, 158
 to_iso8601, 158
 plugins::hoertech::acrec::acrec_t, 1374
 ac, 1379
 acrec_t, 1376
 cv, 1379
 fifolen, 1378
 minwrite, 1378
 patchbay, 1378
 prefix, 1378
 prepare, 1377
 process, 1376
 record, 1378
 release, 1377
 start_new_session, 1377
 use_date, 1378
 varname, 1378
 plugins::hoertech::acrec::acwriter_t, 1379
 ~acwriter_t, 1381
 active, 1383
 acwriter_t, 1381
 close_session, 1383
 create_datafile, 1382
 disk_write_threshold_min_num_samples, 1383
 diskbuffer, 1384
 exit_request, 1382
 fifo, 1383
 get_varname, 1382
 is_complex, 1384
 is_num_channels_known, 1384
 num_channels, 1384
 outfile, 1384
 output_type, 1381
 process, 1382
 varname, 1384
 write_thread, 1382
 writethread, 1383
 plugloader
 bbcilib_interface_t, 336
 db_if_t, 372
 db_t, 373
 dbasync_native::db_if_t, 376
 dbasync_native::dbasync_t, 379
 MHAParser_Resampling::resampling_if_t, 1154
 MHAParser_Resampling::resampling_t, 1157
 overlapadd::overlapadd_if_t, 1331
 smoothgains_bridge::overlapadd_if_t, 1450
 plugname
 latex_doc_t, 646
 MHAParser::mhapluginloader_t, 1089
 plugname_name_
 MHAParser::mhapluginloader_t, 1089
 plugs
 altplugs_t, 304
 plugs_t
 mhachain::plugs_t, 846
 pmode
 audiometerbackend::audiometer_if_t, 318
 calibrator_runtime_layer_t, 338
 poll
 mha_rt_fifo_t< T >, 791
 poll_1
 mha_rt_fifo_t< T >, 791
 poll_config
 MHAParser::config_t< runtime_cfg_t >, 1144
 poll_latest_value_and_reinsert
 double2acvar::double2acvar_t, 434
 polyphase_resampling_t
 MHAFilter::polyphase_resampling_t, 922
 pool
 acPooling_wave_config, 218
 pool_name
 acPooling_wave, 214
 pooling_ind
 acPooling_wave_config, 217
 pooling_option
 acPooling_wave_config, 217
 pooling_size
 acPooling_wave_config, 218
 pooling_type
 acPooling_wave, 214
 pooling_wndlen
 acPooling_wave, 214
 port
 ac2osc_t, 183
 MHA_TCP::Server, 817
 MHAJack::port_t, 983
 mhaserver_t, 1193
 osc2ac_t, 1320
 port_t
 MHAJack::port_t, 981, 982
 portaudio_callback
 MHAIOPortAudio.cpp, 1674, 1675
 MHAIOPortAudio::io_portaudio_t, 960
 portaudio_stream

MHAIOPortAudio::io_portaudio_t, 962
portnames_in
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 951
portnames_out
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 951
ports_in_all
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 952
ports_in_physical
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 952
ports_out_all
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 952
ports_out_physical
 MHAIOJack::io_jack_t, 944
 MHAIOJackdb::io_jack_t, 952
ports_parser
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 952
pos
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 320
 cfg_t, 347
 fader_wave::level_adapt_t, 492
 MHAJack::client_avg_t, 966
 MHAJack::client_noncont_t, 969
 MHASignal::async_rmslevel_t, 1197
 MHASignal::delay_spec_t, 1199
 MHASignal::delay_t, 1201
 MHASignal::delay_wave_t, 1203
 MHASignal::loop_wavefragment_t, 1221
posix_threads_t
 MHAPlugin_Split::posix_threads_t, 1168
posixthreads
 split.cpp, 1703
post_plugin
 gtfb_simple_rt_t, 565
post_trigger_read_line
 mha_tcp::server_t, 822
postfilter
 rohBeam::rohConfig, 1404
postscale
 overlapadd::overlapadd_if_t, 1331
postwindow
 spec2wave_t, 1464
postwnd
 overlapadd::overlapadd_t, 1334
power
 MHASignal::waveform_t, 1269
powSpec
 smooth_cepstrum::smooth_cepstrum_t,
 1442
powspec
 MHASignal::waveform_t, 1270
 windnoise::cfg_t, 1507
POWSPEC_FACTOR
 noise_psd_estimator.cpp, 1691
pre_plugin
 gtfb_simple_rt_t, 565
prefix
 plugins::hoertech::acrec::acrec_t, 1378
 wavrec_t, 1499
prefix_
 MHAParser::mhapluginloader_t, 1089
prefix_names_AC
 acConcat_wave, 201
prepare
 ac2lsl::ac2lsl_t, 163
 ac2osc_t, 182
 ac2wave_if_t, 187
 ac_proc::interface_t, 197
 acConcat_wave, 200
 acmon::acmon_t, 208
 acPooling_wave, 213
 acsave::acsave_t, 220
 acSteer, 230
 acTransform_wave, 236
 adaptive_feedback_canceller, 242
 addsndfile::addsndfile_if_t, 251
 adm_if_t, 273
 altconfig_t, 297
 altplugs_t, 301
 analysispath_if_t, 311
 attenuate20_t, 315
 audiometerbackend::audiometer_if_t, 317
 bbcalib_interface_t, 335
 calibrator_t, 340
 coherence::cohfilt_if_t, 349
 combc_if_t, 357
 complex_scale_channel_t, 363
 cpupload::cpupload_if_t, 368
 db_if_t, 371
 dbasync_native::db_if_t, 375
 dc::dc_if_t, 382
 dc_simple::dc_if_t, 394
 delay::interface_t, 410
 delaysum::delaysum_wave_if_t, 412
 delaysum_spec::delaysum_spec_if_t, 417
 doasvm_classification, 421

doasvm_feature_extraction, 427
 dropgen_t, 437
 droptect_t, 440
 ds_t, 444
 equalize::freqgains_t, 464
 example1_t, 467
 example2_t, 470
 example3_t, 474
 example4_t, 478
 example6_t, 482
 example7_t, 485
 fader_if_t, 487
 fader_wave::fader_wave_if_t, 489
 fftfbpow::fftfbpow_interface_t, 494
 fftfilter::interface_t, 501
 fftfilterbank::fftfb_interface_t, 504
 fshift::fshift_t, 512
 fshift_hilbert::frequency_translator_t, 516
 fw_t, 525
 gain::gain_if_t, 534
 gsc_adaptive_stage::gsc_adaptive_stage_if, 546
 gtfb_analyzer::gtfb_analyzer_t, 554
 gtfb_simd_t, 562
 gtfb_simple_t, 571
 identity_t, 576
 io_alsa_t, 580, 581
 io_asterisk_sound_t, 597
 io_asterisk_t, 601
 io_file_t, 606
 io_lib_t, 612
 io_parser_t, 617
 io_tcp_sound_t, 634
 io_tcp_t, 638
 level_matching::level_matching_t, 655
 levelmeter_t, 658
 lpc, 661
 lpc_bl_predictor, 665
 lpc_burglattice, 671
 lsl2ac::lsl2ac_t, 682
 matlab_wrapper::matlab_wrapper_t, 699
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 706
 matrixmixer::matmix_t, 714
 mconv::MConv, 718
 mhachain::chain_base_t, 843
 mhachain::plugs_t, 847
 MHAIOJack::io_jack_t, 941
 MHAIOJackdb::io_jack_t, 948
 MHAJack::client_t, 973
 MHAParser::mhaplugloader_t, 1087
 MHAPlugin::plugin_t< runtime_cfg_t >, 1148
 MHAPlugin_Resampling::resampling_if_t, 1153
 MHAPlugin_Split::splitted_part_t, 1181
 MHATableLookup::linear_table_t, 1277
 multibandcompressor::interface_t, 1301
 nlms_t, 1306
 noise_psd_estimator::noise_psd_estimator_if_t, 1310
 noise_t, 1316
 osc2ac_t, 1319
 overlapadd::overlapadd_if_t, 1329
 plingploing::if_t, 1339
 plugin_interface_t, 1352
 PluginLoader::fourway_processor_t, 1364
 PluginLoader::mhaplugloader_t, 1368
 plugins::hoertech::acrec::acrec_t, 1377
 rmslevel::rmslevel_if_t, 1388
 rohBeam::rohBeam, 1396
 route::interface_t, 1409
 save_spec_t, 1418
 save_wave_t, 1420
 shadowfilter_begin::shadowfilter_begin_t, 1423
 shadowfilter_end::shadowfilter_end_t, 1427
 sine_t, 1431
 smooth_cepstrum::smooth_cepstrum_if_t, 1435
 smoothgains_bridge::overlapadd_if_t, 1449
 softclip_t, 1454
 spec2wave_if_t, 1461
 steerbf, 1470
 testplugin::if_t, 1481
 us_t, 1485
 wave2spec_if_t, 1488
 wavrec_t, 1498
 windnoise::if_t, 1509
 prepare_,
 prepare_plugin_t, 193
 double2acvar::double2acvar_t, 434
 iirfilter_t, 578
 MHAPlugin::plugin_t< runtime_cfg_t >, 1150
 MHAPlugin_Split::split_t, 1174
 prepare_impl
 MHAJack::client_t, 976
 prepare_vars
 fw_t, 527

PREPARED
 MHA_TCP::Thread, 829

prepared
 ac2wave_if_t, 188
 altplugs_t, 305
 calibrator_t, 341
 dc_simple::dc_if_t, 397
 example3_t, 475
 example4_t, 479
 fader_wave::fader_wave_if_t, 490
 fftfilterbank::fftfb_interface_t, 505
 gtfb_analyzer::gtfb_analyzer_t, 555
 gtfb_simd_t, 562
 mhachain::plugs_t, 847
 rohBeam::rohBeam, 1401
 route::interface_t, 1411
 smooth_cepstrum::smooth_cepstrum_if_t, 1438

prereadaccess
 MHAParser::base_t, 1037

prescale
 overlapadd::overlapadd_if_t, 1331

preset
 dc::dc_vars_t, 391
 dc_simple::dc_if_t, 396

prestages
 gtfb_simple_t, 572

prewnd
 overlapadd::overlapadd_t, 1333

print_ac
 analysemhaplugin.cpp, 1529

print_plugin_references
 generatemhaplugindoc.cpp, 1551

prior_q
 smooth_cepstrum::smooth_cepstrum_if_t, 1437
 smooth_cepstrum::smooth_params, 1447

priorFact
 noise_psd_estimator::noise_psd_estimator_t, 1314
 smooth_cepstrum::smooth_cepstrum_t, 1444

priority
 analysepath_t, 309
 analysispath_if_t, 312
 dbasync_native::dbasync_t, 379
 io_alsa_t, 583
 MHAPlugin_Split::posix_threads_t, 1170

prob_bias
 acPooling_wave, 215

prob_bias_func

 acPooling_wave_config, 218

proc
 MHAJack::client_avg_t, 965
 MHAJack::client_noncont_t, 968, 969

proc_1
 smoothgains_bridge::smoothspec_wrap_t, 1452

proc_2
 smoothgains_bridge::smoothspec_wrap_t, 1452

proc_cnt
 mhachain::plugs_t, 849

proc_err
 io_asterisk_fwcb_t, 587
 io_tcp_fwcb_t, 623

proc_error
 fw_t, 530

proc_error_string
 fw_t, 530

proc_event
 io_alsa_t, 582
 io_asterisk_fwcb_t, 587
 io_file_t, 608
 io_parser_t, 618
 io_tcp_fwcb_t, 623
 MHAIOJackdb::io_jack_t, 950
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 978

proc_handle
 io_alsa_t, 582
 io_asterisk_fwcb_t, 587
 io_file_t, 608
 io_parser_t, 618
 io_tcp_fwcb_t, 623
 MHAIOJackdb::io_jack_t, 950
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 978

proc_lib
 fw_t, 529

proc_name
 fw_t, 528

proc_ramp
 altplugs_t, 303

proc_thread
 io_alsa_t, 582

proc_wave
 doasvm_feature_extraction_config, 431

process
 ac2lsl::ac2lsl_t, 164
 ac2lsl::cfg_t, 168
 ac2osc_t, 182

ac2wave_if_t, 186, 187
 ac2wave_t, 189
 ac_mul_t, 193, 194
 ac_proc::interface_t, 197, 198
 acConcat_wave, 200
 acConcat_wave_config, 203
 acmon::acmon_t, 209
 acPooling_wave, 213
 acPooling_wave_config, 216
 acsave::acsave_t, 221
 acSteer, 230
 acTransform_wave, 235
 acTransform_wave_config, 238
 adaptive_feedback_canceller, 241
 adaptive_feedback_canceller_config, 245
 addsndfile::addsndfile_if_t, 251
 ADM::ADM< F >, 263
 ADM::Delay< F >, 267
 ADM::Linearphase_FIR< F >, 270
 adm_if_t, 272
 altplugs_t, 302
 analysispath_if_t, 311
 attenuate20_t, 315
 audiometerbackend::audiometer_if_t, 316
 bbcalib_interface_t, 335
 calibrator_runtime_layer_t, 337
 calibrator_t, 340
 cfg_t, 346
 coherence::cohflt_if_t, 349
 coherence::cohflt_t, 351
 combc_if_t, 357
 combc_t, 359
 complex_scale_channel_t, 363
 cpupload::cpupload_cfg_t, 365
 cpupload::cpupload_if_t, 368
 db_if_t, 371
 dbasync_native::db_if_t, 375
 dc::dc_if_t, 382
 dc::dc_t, 385
 dc_simple::dc_if_t, 395
 dc_simple::dc_t, 399
 dc_simple::level_smoothen_t, 407
 delay::interface_t, 410
 delaysum::delaysum_wave_if_t, 412
 delaysum::delaysum_wave_t, 415
 delaysum_spec::delaysum_spec_if_t, 417
 delaysum_spec::delaysum_t, 419
 doasvm_classification, 421
 doasvm_classification_config, 424
 doasvm_feature_extraction, 426
 doasvm_feature_extraction_config, 430
 double2acvar::double2acvar_t, 434
 dropgen_t, 437
 droptect_t, 441
 ds_t, 444
 equalize::freqgains_t, 464
 example1_t, 468
 example2_t, 471
 example3_t, 474
 example4_t, 479
 example5_t, 480
 example6_t, 482
 example7_t, 485
 fader_if_t, 487
 fader_wave::fader_wave_if_t, 489
 fftfbpow::fftfbpow_interface_t, 495
 fftfilter::fftfilter_t, 499
 fftfilter::interface_t, 501
 fftfilterbank::fftfb_interface_t, 504, 505
 fftfilterbank::fftfb_plug_t, 507
 fshift::fshift_config_t, 509
 fshift::fshift_t, 512
 fshift_hilbert::frequency_translator_t, 515
 fshift_hilbert::hilbert_shifter_t, 519
 fw_t, 526
 gain::gain_if_t, 533
 gsc_adaptive_stage::gsc_adaptive_stage, 538
 gsc_adaptive_stage::gsc_adaptive_stage_if, 546
 gtfb_analyzer::gtfb_analyzer_t, 554
 gtfb_simd_cfg_t, 558
 gtfb_simd_t, 562
 gtfb_simple_t, 571
 identity_t, 576
 iirfilter_t, 578
 io_alsa_t, 581
 io_asterisk_fwcb_t, 585
 io_tcp_fwcb_t, 621
 level_matching::level_matching_config_t, 651, 652
 level_matching::level_matching_t, 654, 655
 levelmeter_t, 658
 lpc, 661
 lpc_bl_predictor, 665
 lpc_bl_predictor_config, 667
 lpc_burglattice, 671
 lpc_burglattice_config, 673
 lpc_config, 676
 lsl2ac::cfg_t, 679
 lsl2ac::lsl2ac_t, 682

matlab_wrapper::matlab_wrapper_t, 698, 699
matrixmixer::cfg_t, 712
matrixmixer::matmix_t, 714, 715
mconv::MConv, 718
mha_dbdbuf_t< FIFO >, 751
mhachain::chain_base_t, 842, 843
mhachain::plugs_t, 847
MHAFilter::partitioned_convolution_t, 916
MHAFilter::thirddoctave_analyzer_t, 933
MHAParser::mhaplugloader_t, 1087, 1088
MHAParser_Resampling::resampling_if_t, 1153
MHAParser_Resampling::resampling_t, 1156
MHAParser_Split::domain_handler_t, 1163
MHAParser_Split::split_t, 1174
MHAParser_Split::uni_processor_t, 1188
MHASignal::async_rmslevel_t, 1197
MHASignal::delay_spec_t, 1199
MHASignal::delay_t, 1201
MHASignal::delay_wave_t, 1203
MHASignal::subsample_delay_t, 1253, 1254
multibandcompressor::interface_t, 1301
nlms_t, 1306
noise_psd_estimator::noise_psd_estimator_if_t, 1310
noise_psd_estimator::noise_psd_estimator_process_cc, 1312
noise_t, 1316
osc2ac_t, 1319
overlapadd::overlapadd_if_t, 1329
plingploing::if_t, 1339
plingploing::plingploing_t, 1342
plugin_interface_t, 1352
PluginLoader::fourway_processor_t, 1363, 1364
PluginLoader::mhaplugloader_t, 1369
plugins::hoertech::acrec::acrec_t, 1376
plugins::hoertech::acrec::acwriter_t, 1382
rmslevel::rmslevel_if_t, 1388
rmslevel::rmslevel_t, 1391
rohBeam::rohBeam, 1396
rohBeam::rohConfig, 1404
route::interface_t, 1409, 1410
route::process_t, 1412
rt_nlms_t, 1414
save_spec_t, 1418
save_wave_t, 1420
shadowfilter_begin::cfg_t, 1421
shadowfilter_begin::shadowfilter_begin_t, 1423
shadowfilter_end::cfg_t, 1425
shadowfilter_end::shadowfilter_end_t, 1427
sine_t, 1431
smooth_cepstrum::smooth_cepstrum_if_t, 1434
smooth_cepstrum::smooth_cepstrum_t, 1440
smoothgains_bridge::overlapadd_if_t, 1450
softclip_t, 1454
softclipper_t, 1456
spec2wave_if_t, 1461
spec2wave_t, 1463
steerbf, 1470
steerbf_config, 1472
testplugin::if_t, 1481
us_t, 1485
wave2spec_if_t, 1488, 1489
wave2spec_t, 1494
wavrec_t, 1498
wavwriter_t, 1501
windnoise::cfg_t, 1505
windnoise::if_t, 1510
process_cc
 ac_mul_t, 194
process_cr
 ac_mul_t, 194
process_frame
 io_parser_t, 617
process_rc
 ac_mul_t, 194
process_rr
 ac_mul_t, 194
process_ss
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 705
process_sw
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 706
process_t
 route::process_t, 1411
process_ws
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 705
process_ww
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,

704
processing_done
 MHAPlugin_Split::posix_threads_t, 1170
ProcessMutex
 analysepath_t, 309
processor
 MHAPlugin_Split::domain_handler_t, 1164
 MHAPlugin_Split::thread_platform_t, 1187
prof_algos
 mhachain::plugs_t, 849
prof_cfg
 mhachain::plugs_t, 850
prof_init
 mhachain::plugs_t, 849
prof_load_con
 mhachain::plugs_t, 850
prof_prepare
 mhachain::plugs_t, 849
prof_process
 mhachain::plugs_t, 849
prof_process_load
 mhachain::plugs_t, 849
prof_process_tt
 mhachain::plugs_t, 849
prof_release
 mhachain::plugs_t, 849
prof_tt_con
 mhachain::plugs_t, 850
profiling
 mhachain::plugs_t, 848
prop_type
 rohBeam::rohBeam, 1399
propExport
 rohBeam::rohBeam, 1401
provoke_inner_error
 mha_dbdbuf_t< FIFO >, 750
provoke_outer_error
 mha_dbdbuf_t< FIFO >, 751
PSD_Lowpass
 windnoise::cfg_t, 1507
PSD_val
 adaptive_feedback_canceller_config, 247
psrate
 fw_vars_t, 532
Psum
 gsc_adaptive_stage::gsc_adaptive_stage, 543
Pu
 adaptive_feedback_canceller_config, 246
rt_nlms_t, 1415
publish_ac_variables
 wave2spec_t, 1494
pull_samples_discard
 lsl2ac::save_var_t, 688
pull_samples_ignore
 lsl2ac::save_var_t, 688
push
 mha_rt_fifo_t< T >, 791
 MHASignal::stat_t, 1251, 1252
push_config
 MHAPlugin::config_t< runtime_cfg_t >, 1145
put_signal
 MHAPlugin_Split::domain_handler_t, 1161
pVars
 osc_server_t, 1322
pwinner_out
 MHAIOLockdb::io_jack_t, 953
q
 noise_psd_estimator::noise_psd_estimator_if_t, 1311
q_high
 smooth_cepstrum::smooth_cepstrum_t, 1441
q_low
 smooth_cepstrum::smooth_cepstrum_t, 1441
quant
 calibrator_runtime_layer_t, 338
quantile
 MHASignal, 147
quantizer_t
 MHASignal::quantizer_t, 1235
queries
 MHParse::base_t, 1038
 plugindescription_t, 1357
query_addsubst
 MHParse::base_t, 1035
query_cmds
 MHParse::base_t, 1035
 plugindescription_t, 1357
query_dump
 MHParse::base_t, 1032
 MHParse::monitor_t, 1096
 MHParse::parser_t, 1101
query_entries
 MHParse::base_t, 1032
 MHParse::parser_t, 1101
query_help

MHAParser::base_t, 1035
query_id
 MHAParser::base_t, 1035
query_listids
 MHAParser::base_t, 1034
 MHAParser::parser_t, 1102
query_map_t
 MHAParser, 126
query_peak
 levelmeter_t, 659
query_perm
 MHAParser::base_t, 1033
 MHAParser::monitor_t, 1096
 MHAParser::variable_t, 1114
query_range
 MHAParser::base_t, 1033
 MHAParser::kw_t, 1073
 MHAParser::range_var_t, 1105
query_readfile
 MHAParser::base_t, 1034
 MHAParser::parser_t, 1101
query_rms
 levelmeter_t, 658
query_savefile
 MHAParser::base_t, 1034
 MHAParser::parser_t, 1102
query_savefile_compact
 MHAParser::base_t, 1034
 MHAParser::parser_t, 1102
query_savemons
 MHAParser::base_t, 1034
 MHAParser::parser_t, 1102
query_subst
 MHAParser::base_t, 1035
query_t
 MHAParser, 125
query_type
 MHAParser::base_t, 1033
 MHAParser::bool_mon_t, 1042
 MHAParser::bool_t, 1044
 MHAParser::complex_mon_t, 1051
 MHAParser::complex_t, 1053
 MHAParser::float_mon_t, 1058
 MHAParser::float_t, 1060
 MHAParser::int_mon_t, 1063
 MHAParser::int_t, 1066
 MHAParser::kw_t, 1073
 MHAParser::mcomplex_mon_t, 1075
 MHAParser::mcomplex_t, 1077
 MHAParser::mfloat_mon_t, 1079
 MHAParser::mfloat_t, 1082
 MHAParser::mint_mon_t, 1091
 MHAParser::mint_t, 1094
 MHAParser::parser_t, 1101
 MHAParser::string_mon_t, 1109
 MHAParser::string_t, 1112
 MHAParser::vcomplex_mon_t, 1116
 MHAParser::vcomplex_t, 1119
 MHAParser::vfloat_mon_t, 1121
 MHAParser::vfloat_t, 1124
 MHAParser::vint_mon_t, 1126
 MHAParser::vint_t, 1128
 MHAParser::vstring_mon_t, 1131
 MHAParser::vstring_t, 1132
query_val
 MHAParser::base_t, 1033
 MHAParser::bool_mon_t, 1041
 MHAParser::bool_t, 1044
 MHAParser::complex_mon_t, 1051
 MHAParser::complex_t, 1053
 MHAParser::float_mon_t, 1057
 MHAParser::float_t, 1061
 MHAParser::int_mon_t, 1063
 MHAParser::int_t, 1066
 MHAParser::kw_t, 1073
 MHAParser::mcomplex_mon_t, 1075
 MHAParser::mcomplex_t, 1077
 MHAParser::mfloat_mon_t, 1079
 MHAParser::mfloat_t, 1082
 MHAParser::mint_mon_t, 1091
 MHAParser::mint_t, 1094
 MHAParser::parser_t, 1102
 MHAParser::string_mon_t, 1109
 MHAParser::string_t, 1112
 MHAParser::vcomplex_mon_t, 1116
 MHAParser::vcomplex_t, 1119
 MHAParser::vfloat_mon_t, 1121
 MHAParser::vfloat_t, 1124
 MHAParser::vint_mon_t, 1126
 MHAParser::vint_t, 1128
 MHAParser::vstring_mon_t, 1131
 MHAParser::vstring_t, 1132
query_version
 MHAParser::base_t, 1035
queue_write
 mha_tcp::buffered_socket_t, 799
quit
 fw_t, 525

R

AuditoryProfile::parser_t, 327
AuditoryProfile::profile_t, 332
lpc_config, 677

r
 MHA_TCP::OS_EVENT_TYPE, 813
 r
 dropgen_t, 438
 mha_real_test_array_t, 787
 rad2smp
 Vector and matrix processing toolbox, 40
 ramp_a
 hanning_ramps_t, 574
 ramp_b
 hanning_ramps_t, 575
 ramp_begin
 MHAWindow::base_t, 1289
 ramp_counter
 altplugs_t, 305
 ramp_end
 MHAWindow::base_t, 1289
 ramp_len
 altplugs_t, 305
 ramplen
 addsndfile::addsndfile_if_t, 253
 altplugs_t, 304
 audiometerbackend::audiometer_if_t, 318
 fader_wave::fader_wave_if_t, 490
 spec2wave_if_t, 1462
 ramps
 spec2wave_t, 1464
 rand_dist
 cfg_t, 347
 random
 audiometerbackend::Inn3rdoct_t, 322
 random_engine
 dropgen_t, 438
 range
 level_matching::level_matching_config_t, 652
 level_matching::level_matching_t, 656
 Vector and matrix processing toolbox, 38
 range_var_t
 MHAParser::range_var_t, 1105
 ratio
 ds_t, 444
 us_t, 1486
 raw_p_max_name
 acTransform_wave, 237
 acTransform_wave_config, 239
 raw_p_name
 acPooling_wave_config, 217
 acTransform_wave, 236
 acTransform_wave_config, 239
 rb_f_t
 mha_ruby.cpp, 1621
 rcoefficients
 gtfb_simd_cfg_t, 559
 rdata
 mha_audio_t, 743
 MHASignal::matrix_t, 1232
 re
 mha_complex_t, 745
 read
 alsal_base_t, 288
 alsal_t< T >, 293
 mha_drifter_fifo_t< T >, 758
 mha_fifo_lf_t< T >, 768
 mha_fifo_lw_t< T >, 771
 mha_fifo_t< T >, 779
 MHAFilter::blockprocessing_polyphase_resampling_t, 870
 MHAFilter::polyphase_resampling_t, 924
 MHAJack::port_t, 982
 read_bytes
 MHA_TCP::Connection, 808
 read_event
 MHA_TCP::Connection, 809
 read_get_cpu_load
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949
 read_get_scheduler
 MHAIOJack::io_jack_t, 943
 MHAIOJackdb::io_jack_t, 950
 read_get_xruns
 MHAIOJack::io_jack_t, 943
 MHAIOJackdb::io_jack_t, 949
 read_levels
 calibrator_t, 341
 read_line
 MHA_TCP::Connection, 807
 read_modified
 dc_simple::dc_if_t, 395
 read_ptr
 mha_fifo_t< T >, 782
 readable_frames
 MHAFilter::polyphase_resampling_t, 924
 readaccess
 MHAParser::base_t, 1037
 reader_started
 mha_drifter_fifo_t< T >, 761
 reader_xruns_in_succession
 mha_drifter_fifo_t< T >, 762
 reader_xruns_since_start
 mha_drifter_fifo_t< T >, 762
 reader_xruns_total
 mha_drifter_fifo_t< T >, 761

real
 MHASignal::matrix_t, 1227–1230

rear_channel
 adm_rtconfig_t, 278

rear_channels
 adm_if_t, 273
 adm_rtconfig_t, 279

rec_frames
 acsave::cfg_t, 224

receive_frame
 lsl2ac::save_var_t, 688

receiver
 MHAEvents::connector_t< receiver_t >, 857

reciprocal
 Complex arithmetics in the openMHA, 67

reclen
 acsave::acsave_t, 222

recmode
 acmon::acmon_t, 210

reconnect_inports
 MHAIOJack::io_jack_t, 941
 MHAIOJackdb::io_jack_t, 949

reconnect_outports
 MHAIOJack::io_jack_t, 942
 MHAIOJackdb::io_jack_t, 949

record
 plugins::hoertech::acrec::acrec_t, 1378
 wavrec_t, 1499

rect
 MHAOvIFilter::ShapeFun, 120
 MHAWindow, 154

rect_t
 MHAWindow::rect_t, 1296

refL
 rohBeam, 160

refR
 rohBeam, 160

register_configuration_variable
 windnoise.cpp, 1707

relative
 MHASignal::loop_wavefragment_t, 1218

release
 ac2lsl::ac2lsl_t, 164
 ac2osc_t, 182
 ac2wave_if_t, 187
 ac_proc::interface_t, 197
 acConcat_wave, 201
 acmon::acmon_t, 209
 acPooling_wave, 213
 acsave::acsave_t, 221

acSteer, 231

acTransform_wave, 236

adaptive_feedback_canceller, 242

addsndfile::addsndfile_if_t, 252

adm_if_t, 273

altconfig_t, 297

altparts_t, 301

analysispath_if_t, 312

attenuate20_t, 315

bbcalib_interface_t, 335

calibrator_t, 340

coherence::cohflt_if_t, 349

db_if_t, 371

dbasync_native::db_if_t, 376

dc_simple::dc_if_t, 394

delaysum::delaysum_wave_if_t, 413

doasvm_classification, 422

doasvm_feature_extraction, 428

dropgen_t, 437

droptect_t, 441

ds_t, 444

example1_t, 467

example2_t, 470

example3_t, 474

example4_t, 478

example7_t, 485

fader_wave::fader_wave_if_t, 489

fftfilterbank::fftfb_interface_t, 504

fshift::fshift_t, 513

fshift_hilbert::frequency_translator_t, 516

fw_t, 525

gain::gain_if_t, 534

gsc_adaptive_stage::gsc_adaptive_stage_if, 546

gtfb_simple_t, 571

identity_t, 576

io_alsa_t, 580

io_asterisk_sound_t, 597

io_asterisk_t, 601

io_file_t, 606

io_lib_t, 613

io_parser_t, 617

io_tcp_sound_t, 634

io_tcp_t, 639

level_matching::level_matching_t, 655

lpc, 662

lpc_bl_predictor, 665

lpc_burglattice, 671

lsl2ac::lsl2ac_t, 682

matlab_wrapper::matlab_wrapper_t, 700

matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,

706
 mconv::MConv, [718](#)
 mhachain::chain_base_t, [843](#)
 mhachain::plugs_t, [847](#)
 MHAIOJack::io_jack_t, [941](#)
 MHAIOJackdb::io_jack_t, [948](#)
 MHAJack::client_t, [974](#)
 MHAParser::mhapluginloader_t, [1087](#)
 MHAParser::plugin_t< runtime_cfg_t >, [1149](#)
 MHAParser_Resampling::resampling_if_t, [1154](#)
 MHAParser_Split::splitted_part_t, [1181](#)
 multibandcompressor::interface_t, [1301](#)
 nlms_t, [1306](#)
 osc2ac_t, [1319](#)
 overlapadd::overlapadd_if_t, [1329](#)
 PluginLoader::fourway_processor_t, [1365](#)
 PluginLoader::mhapluginloader_t, [1369](#)
 plugins::hoertech::acrec::acrec_t, [1377](#)
 rohBeam::rohBeam, [1396](#)
 route::interface_t, [1409](#)
 smooth_cepstrum::smooth_cepstrum_if_t, [1435](#)
 smoothgains_bridge::overlapadd_if_t, [1449](#)
 spec2wave_if_t, [1461](#)
 steerbf, [1471](#)
 us_t, [1485](#)
 wave2spec_if_t, [1488](#)
 wavrec_t, [1498](#)
 windnoise::if_t, [1510](#)
 release_
 ac_mul_t, [193](#)
 double2acvar::double2acvar_t, [435](#)
 iirfilter_t, [578](#)
 MHAParser::plugin_t< runtime_cfg_t >, [1150](#)
 MHAParser_Split::split_t, [1174](#)
 release_mutex
 mha_fifo_posix_threads_t, [774](#)
 mha_fifo_thread_platform_t, [785](#)
 remapping
 windnoise::cfg_t, [1506](#)
 remove
 MHAUtils, [151](#)
 remove_abandonned
 mha_rt_fifo_t< T >, [792](#)
 remove_all
 mha_rt_fifo_t< T >, [792](#)
 remove_all_cfg
 MHAPlugin::config_t< runtime_cfg_t >, [1145](#)
 remove_item
 MHAParser::parser_t, [1099, 1100](#)
 remove_lock
 mhamain.cpp, [1686](#)
 remove_ref
 algo_comm_t, [283](#)
 MHAKernel::algo_comm_class_t, [987](#)
 remove_var
 algo_comm_t, [282](#)
 MHAKernel::algo_comm_class_t, [987](#)
 repl_list
 MHAParser::base_t, [1038](#)
 repl_list_t
 MHAParser::base_t, [1030](#)
 replace
 MHAParser::base_t::replace_t, [1039](#)
 MHASignal::loop_wavefragment_t, [1218](#)
 replace_
 cfg_t, [346](#)
 replace_t
 MHAParser::base_t::replace_t, [1039](#)
 res_c
 ac_mul_t, [195](#)
 res_r
 ac_mul_t, [195](#)
 resampled_num_frames
 addsndfile, [81](#)
 resampled_soundfile_t
 addsndfile::resampled_soundfile_t, [258](#)
 resampling
 MHAFilter::blockprocessing_polyphase_resampling_t, [870](#)
 resampling.cpp, [1692](#)
 resampling_factors
 MHAFilter, [109](#)
 resampling_filter_t
 MHAFilter::resampling_filter_t, [927](#)
 resampling_if_t
 MHAParser_Resampling::resampling_if_t, [1153](#)
 resampling_t
 MHAParser_Resampling::resampling_t, [1155](#)
 resamplingmode
 addsndfile::addsndfile_if_t, [253](#)
 reset
 droptect_t, [441](#)
 MHA_TCP::Async_Notify, [798](#)
 MHA_TCP::Wakeups_Event, [837](#)

reset_state
 MHAFilter::gamma_flt_t, 894

resize
 MHAFilter::iir_filter_t, 899

resolution
 acTransform_wave_config, 239

resolve
 dynamiclib_t, 447
 pluginlib_t, 1359

resolve_and_init
 PluginLoader::mhaplugloader_t, 1370

resolve_checked
 dynamiclib_t, 447

restore_state
 altconfig_t, 298

result
 cpupload::cpupload_cfg_t, 366

resynthesis_gain
 gtfb_simple_t, 573
 MHAFilter::gamma_flt_t, 895

ret_size
 MHAParser::c_ifc_parser_t, 1047

return_imag
 fftfilterbank::fftfb_interface_t, 505

return_imag_
 fftfilterbank::fftfb_plug_t, 508

return_sig
 audiometerbackend, 83

return_value
 MHA_TCP::Thread, 831

return_wave
 wave2spec_if_t, 1490

retv
 MHAParser::c_ifc_parser_t, 1047

rewind
 MHASignal::loop_wavefragment_t, 1220

rho
 adaptive_feedback_canceller, 242
 nlms_t, 1307

ringbuffer
 MHAFilter::polyphase_resampling_t, 925

ringbuffer_t
 MHASignal::ringbuffer_t, 1237

rinputs
 gtfb_simd_cfg_t, 559

rm_parent_on_remove
 MHAParser::base_t, 1036

rms
 levelmeter_t, 659
 MHASignal::loop_wavefragment_t, 1218
 plingploing::plingploing_t, 1345

rms_limit40
 MHASignal::loop_wavefragment_t, 1218

rmsdb
 example6_t, 483

rmslevel, 158
 calibrator_variables_t, 343
 dc::dc_t, 386
 HL, 159
 MHASignal::async_rmslevel_t, 1197
 SPL, 159
 UNIT, 159
 Vector and matrix processing toolbox, 53,
 55

rmslevel.cpp, 1692

rmslevel::mon_t, 1385
 mon_t, 1386
 operator=, 1386
 store, 1386

rmslevel::rmslevel_if_t, 1387
 name, 1389
 patchbay, 1389
 prepare, 1388
 process, 1388
 rmslevel_if_t, 1388
 unit, 1389
 update, 1389

rmslevel::rmslevel_t, 1389
 ~rmslevel_t, 1391
 domain, 1392
 ffflen, 1392
 fill, 1391
 freq_offsets, 1392
 insert, 1392
 monitors, 1392
 process, 1391
 rmslevel_t, 1390
 sum_and_fill, 1391, 1392
 unit, 1392

rmslevel_if_t
 rmslevel::rmslevel_if_t, 1388

rmslevel_t
 rmslevel::rmslevel_t, 1390

rmslevelmeter
 transducers.cpp, 1705

rng
 audiometerbackend::Inn3rdoct_t, 322
 cfg_t, 347

rohBeam, 159
 CONST_C, 160
 j0, 159
 refL, 160

refR, 160
 rohBeam::rohBeam, 1396
 scalarify, 160
 rohBeam.cpp, 1693
 rohBeam.hh, 1693
 NDEBUG, 1694
 rohBeam::configOptions, 1393
 alpha_blocking_XkXi, 1393
 alpha_blocking_XkY, 1394
 alpha_postfilter, 1393
 binaural_type_index, 1393
 enable_adaptive_beam, 1393
 rohBeam::rohBeam, 1394
 ~rohBeam, 1396
 beamExport, 1401
 binaural_type, 1400
 compute_beamW, 1398
 compute_delaycomp_vec, 1397
 compute_diff2D, 1398
 compute_diff3D, 1398
 compute_head_model_alpha, 1397
 compute_head_model_mat, 1397
 compute_head_model_T, 1397
 compute_uncorr, 1397
 compute_wng, 1398
 diag_loading_mu, 1400
 enable_adaptive_beam, 1400
 enable_export, 1400
 enable_wng_optimization, 1400
 export_beam_design, 1398
 get_noise_model_func, 1398
 head_model_sphere_radius_cm, 1399
 intermic_distance_cm, 1399
 mic_azimuth_degrees_vec, 1399
 noise_field_model, 1400
 noise_integrate_hrtf, 1397
 noiseFuncPtr, 1399
 noiseModelExport, 1401
 on_model_param_valuechanged, 1398
 patchbay, 1401
 prepare, 1396
 prepared, 1401
 process, 1396
 prop_type, 1399
 propExport, 1401
 release, 1396
 rohBeam, 1396
 sampled_hrir_path, 1399
 solve_MVDR, 1397
 source_azimuth_degrees, 1399
 tau_blocking_XkXi_ms, 1400
 tau_blocking_XkY_ms, 1401
 tau_postfilter_ms, 1400
 update_cfg, 1396
 rohBeam::rohConfig, 1402
 ~rohConfig, 1403
 alpha_blocking_XkXi, 1406
 alpha_blocking_XkY, 1406
 alpha_postfilter, 1406
 beam1, 1405
 beamA, 1406
 beamW, 1405
 binaural_type_index, 1405
 blockSpec, 1406
 blockXp, 1407
 copyfixedfboutput, 1404
 corrLL, 1407
 corrRR, 1407
 corrXpXp, 1406
 corrXpYf, 1406
 corrZZ, 1406
 delayComp, 1405
 enable_adaptive_beam, 1405
 freqResp, 1407
 headModel, 1405
 hhCorrXpXp, 1407
 in_cfg, 1405
 init_dynamic, 1404
 magResp, 1407
 maxLim, 1407
 minLim, 1407
 nchan_block, 1405
 nextXpYf, 1407
 nfreq, 1404
 operator=, 1404
 out_cfg, 1405
 outSpec, 1406
 phasereconstruction, 1404
 postfilter, 1404
 process, 1404
 rohConfig, 1403
 rohConfig
 rohBeam::rohConfig, 1403
 root
 mha_rt_fifo_t< T >, 792
 rotated_i
 acTransform_wave_config, 239
 rotated_p
 acTransform_wave_config, 239
 rotated_p_max_name
 acTransform_wave, 237
 rotated_p_name

acTransform_wave, 237
route, 160
route.cpp, 1694
route::interface_t, 1408
 algo, 1411
 cfac, 1410
 cfin, 1410
 cfout, 1410
 interface_t, 1409
 patchbay, 1410
 prepare, 1409
 prepared, 1411
 process, 1409, 1410
 release, 1409
 route_ac, 1410
 route_out, 1410
 stopped, 1411
 update, 1410
route::process_t, 1411
 process, 1412
 process_t, 1411
 sout, 1412
 sout_ac, 1412
 wout, 1412
 wout_ac, 1412
route_ac
 route::interface_t, 1410
route_out
 route::interface_t, 1410
rows
 acsave::mat4head_t, 225
rstates
 gtfb_simd_cfg_t, 559
rt_nlms_t, 1413
 ~rt_nlms_t, 1414
 ac, 1414
 channels, 1415
 F, 1415
 frames, 1415
 fu, 1415
 fu_previous, 1416
 fuflt, 1416
 insert, 1414
 n_no_update_, 1416
 name_d_, 1416
 name_e_, 1416
 name_u_, 1416
 no_iter, 1417
 ntaps, 1415
 P_Sum, 1416
 process, 1414
Pu, 1415
rt_nlms_t, 1414
s_E, 1417
U, 1415
Uflt, 1415
y_previous, 1416
rt_process
 analysepath_t, 307
rt_strict
 ac2lsl::ac2lsl_t, 165
 ac2osc_t, 184
rtcalibrator
 transducers.cpp, 1705
rtmem
 ac2osc_t, 184
run
 mha_tcp::server_t, 821
 MHA_TCP::Thread, 830
 mhaserver_t, 1192
RUNNING
 MHA_TCP::Thread, 829
runtime configuration, 4
rval
 MHAParser::expression_t, 1056
s_b
 lpc_bl_predictor_config, 669
 lpc_burglattice_config, 675
s_E
 adaptive_feedback_canceller_config, 246
 rt_nlms_t, 1417
s_E_afc_delay
 adaptive_feedback_canceller_config, 247
s_f
 lpc_bl_predictor_config, 669
 lpc_burglattice_config, 675
s_file_in
 io_file_t, 609
s_in
 ac_proc::interface_t, 199
 io_asterisk_sound_t, 599
 io_file_t, 609
 io_parser_t, 619
 io_tcp_sound_t, 636
 MHAIOPortAudio::io_portaudio_t, 960
 MHAJack::client_t, 979
s_in_perm
 ac_proc::interface_t, 198
s_LPC
 adaptive_feedback_canceller_config, 249
s_out
 ac_proc::interface_t, 198

coherence::cohflt_t, 353
 combc_t, 360
 fftfilterbank::fftfb_plug_t, 508
 gtfb_analyzer::gtfb_analyzer_cfg_t, 551
 gtfb_simd_cfg_t, 560
 io_file_t, 609
 io_parser_t, 619
 MHAIOPortAudio::io_portaudio_t, 960
 MHAJack::client_t, 979
s_U
 adaptive_feedback_canceller_config, 247
s_U_delay
 adaptive_feedback_canceller_config, 248
s_U_delayflt
 adaptive_feedback_canceller_config, 248
s_Usmpl
 adaptive_feedback_canceller_config, 249
s_W
 adaptive_feedback_canceller_config, 248
s_Wflt
 adaptive_feedback_canceller_config, 248
s_Y_delay
 adaptive_feedback_canceller_config, 248
s_Y_delayflt
 adaptive_feedback_canceller_config, 248
safe_div
 Complex arithmetics in the openMHA, 66
 mha_signal.cpp, 1625
 mha_signal.hh, 1635
sample
 lpc_config, 677
sampled_hrir_path
 rohBeam::rohBeam, 1399
samplerate
 io_asterisk_sound_t, 598
 io_file_t, 607
 io_tcp_sound_t, 635
 MHAIOPortAudio::io_portaudio_t, 960
 MHAJack::client_t, 978
samples_AC
 acConcat_wave, 201
samplingrate
 MHAOvlFilter::fftfb_t, 1006
save_m
 acsave::save_var_t, 227
save_mat4
 acsave::save_var_t, 227
save_spec.cpp, 1694
save_spec_t, 1417
 basename, 1418
 prepare, 1418
 process, 1418
 save_spec_t, 1418
save_state
 altconfig_t, 298
save_txt
 acsave::save_var_t, 227
save_var_t
 ac2lsl::save_var_t< mha_complex_t >, 176
 ac2lsl::save_var_t< T >, 172
 acsave::save_var_t, 226
 lsl2ac::save_var_t, 687
save_vars
 acmon::acmon_t, 209
save_wave.cpp, 1694
save_wave_t, 1419
 basename, 1420
 prepare, 1420
 process, 1420
 save_wave_t, 1419
saveas_mat4
 MHASignal, 148, 149
sc
 MHASignal::hilbert_fftw_t, 1214
 spec2wave_t, 1464
scalarify
 rohBeam, 160
scale
 cfg_t, 346
 delaysum_spec::delaysum_t, 419
 example5_t, 481
 MHASignal, 144
 MHASignal::fft_t, 1211
 MHASignal::spectrum_t, 1248
 MHASignal::waveform_t, 1270
scale_ch
 complex_scale_channel_t, 364
 example2_t, 471
 example3_t, 475
 example4_t, 479
 plugin_interface_t, 1353
scale_channel
 MHASignal::spectrum_t, 1250
 MHASignal::waveform_t, 1271
scale_frame
 MHASignal::waveform_t, 1271
scale_fun_t
 MHAOvlFilter, 116
scale_var_t
 MHAOvlFilter::scale_var_t, 1025
scalefac

MHATableLookup::linear_table_t, 1279
scaler_t
 gain::scaler_t, 535
scan_dir
 addsndfile::addsndfile_if_t, 252
scan_plugin
 pluginbrowser_t, 1354
scan_plugins
 pluginbrowser_t, 1354
scan_syntax
 ac_mul_t, 193
scheduler
 analysepath_t, 309
 dbasync_native::dbasync_t, 379
 MHAPlugin_Split::posix_threads_t, 1170
schroeder_t
 MHASignal::schroeder_t, 1242, 1243
search_pattern
 addsndfile::addsndfile_if_t, 254
search_result
 addsndfile::addsndfile_if_t, 254
sec2smp
 MHASignal, 144
seed
 noise_t, 1317
select_plug
 altconfig_t, 299
 altpugs_t, 304
select_source
 MHAMultiSrc::base_t, 992
selectall
 altconfig_t, 299
selected_plug
 altpugs_t, 304
send_frame
 ac2lsl::save_var_base_t, 170
 ac2lsl::save_var_t< mha_complex_t >, 178
 ac2lsl::save_var_t< T >, 174
send_osc_float
 ac2osc_t, 182
send_port_announcement
 mhaserver_t, 1191
Server
 MHA_TCP::Server, 815
server
 io_asterisk_t, 603
 io_tcp_t, 640
server_fragsize
 MHAIOJackdb::io_jack_t, 952
server_port_open

 io_asterisk_parser_t, 594
 io_tcp_parser_t, 630
server_srate
 MHAIOJackdb::io_jack_t, 952
server_start
 osc_server_t, 1322
server_stop
 osc_server_t, 1322
server_t
 mha_tcp::server_t, 819
servername
 MHAIOJack::io_jack_t, 943
 MHAIOJackdb::io_jack_t, 951
serversocket
 MHA_TCP::Server, 817
set
 Complex arithmetics in the openMHA, 60, 61
 MHA_TCP::Async_Notify, 798
 testplugin::config_parser_t, 1478
set_announce_port
 mhaserver_t, 1191
set_buf_address
 ac2lsl::save_var_base_t, 170
 ac2lsl::save_var_t< mha_complex_t >, 177
 ac2lsl::save_var_t< T >, 173
set_channelcnt
 MHAFilter::adapt_filter_t, 866
set_connected
 io_asterisk_parser_t, 592
 io_tcp_parser_t, 628
set_entries
 MHAParser::keyword_list_t, 1068
set_errno
 io_asterisk_fwcb_t, 586
 io_tcp_fwcb_t, 622
set_error
 mha_fifo_lw_t< T >, 772
set_fb_pars
 DynComp::dc_afterburn_t, 453
set_format
 wavwriter_t, 1502
set_help
 MHAParser::base_t, 1036
set_id_string
 MHAParser::parser_t, 1102
set_index
 MHAParser::keyword_list_t, 1069
set_input_domain
 MHAPlugin_Split::domain_handler_t,

1160
set_input_portnames
 MHAJack::client_t, 975
set_level
 addsndfile::addsndfile_if_t, 252
 audiometerbackend::audiometer_if_t, 317
 fader_wave::fader_wave_if_t, 490
set_level_db
 MHASignal::loop_wavefragment_t, 1220
set_level_lin
 MHASignal::loop_wavefragment_t, 1220
set_local_port
 io_asterisk_parser_t, 591
 io_tcp_parser_t, 627
set_locate
 MHAIOJackdb::io_jack_t, 950
set_max_angle_ind
 parser_int_dyn, 1336
set_minabs
 mha_signal.cpp, 1625
 mha_signal.hh, 1635
set_new_peer
 io_asterisk_parser_t, 593
 io_tcp_parser_t, 629
set_node_id
 MHAParser::base_t, 1035
set_output_domain
 MHAPlugin_Split::domain_handler_t,
 1160
set_output_portnames
 MHAJack::client_t, 975
set_parse_cb
 MHAParser::c_ifc_parser_t, 1046
set_range
 MHAParser::kw_t, 1072
 MHAParser::range_var_t, 1105
set_server_port_open
 io_asterisk_parser_t, 591
 io_tcp_parser_t, 627
set_state
 MHAFilter::complex_bandpass_t, 873
 MHAFilter::iir_ord1_real_t, 901, 902
set_tau
 MHAFilter::o1flt_lowpass_t, 909
 MHAFilter::o1flt_maxtrack_t, 912
 MHAFilter::o1flt_mintrack_t, 913, 914
set_tau_attack
 MHAFilter::o1_ar_filter_t, 905
set_tau_release
 MHAFilter::o1_ar_filter_t, 906
set_use_jack_transport
 MHAIOJackdb::io_jack_t, 950
 MHAJack::client_t, 976
set_value
 MHAParser::keyword_list_t, 1068
set_weights
 MHAFilter::complex_bandpass_t, 873
 MHAFilter::gamma_flt_t, 893
set_xfun
 MHATableLookup::xy_table_t, 1284
set_xmax
 MHATableLookup::linear_table_t, 1277
set_xmin
 MHATableLookup::linear_table_t, 1277
set_xyfun
 MHATableLookup::xy_table_t, 1285
set_yfun
 MHATableLookup::xy_table_t, 1284
setlock
 gtfb_simple_t, 571
 io_file_t, 607
 lsl2ac::lsl2ac_t, 683
 MHAParser::variable_t, 1114
 MHAParser::window_t, 1136
 osc2ac_t, 1320
 overlapadd::overlapadd_if_t, 1329
 spec2wave_if_t, 1461
 testplugin::config_parser_t, 1478
 wave2spec_if_t, 1489
 windowselector_t, 1514
sf
 MHASndFile::sf_t, 1273
 wavwriter_t, 1502
sf_in
 io_file_t, 609
sf_out
 io_file_t, 610
sf_t
 MHASndFile::sf_t, 1272
sf_wave_t
 MHASndFile::sf_wave_t, 1274
sfinf_in
 io_file_t, 610
sfinf_out
 io_file_t, 610
shadowfilter_begin
 160
shadowfilter_begin.cpp
 1695
shadowfilter_begin::cfg_t
 1420
 cfg_t, 1421
 in_spec_copy, 1421
 nch, 1421
 ntracks, 1422

out_spec, 1421
process, 1421
shadowfilter_begin::shadowfilter_begin_t,
 1422
basename, 1423
nch, 1424
ntracks, 1424
prepare, 1423
process, 1423
 shadowfilter_begin_t, 1423
shadowfilter_begin_t
 shadowfilter_begin::shadowfilter_begin_t,
 1423
shadowfilter_end, 161
shadowfilter_end.cpp, 1695
shadowfilter_end::cfg_t, 1424
 ac, 1425
 cfg_t, 1424
 gains, 1426
 in_spec, 1425
 name, 1425
 nch_out, 1425
 nfft, 1425
 ntracks, 1425
 out_spec, 1426
 process, 1425
shadowfilter_end::shadowfilter_end_t, 1426
 basename, 1428
 prepare, 1427
 process, 1427
 shadowfilter_end_t, 1427
shadowfilter_end_t
 shadowfilter_end::shadowfilter_end_t,
 1427
shape
 MHAOvIFilter::fftfb_t, 1006
shapes
 MHAOvIFilter::fftfb_vars_t, 1010
shift
 lpc, 662
 lpc_config, 676
shifted
 fshift_hilbert::hilbert_shifter_t, 519
shutdown
 mha_tcp::server_t, 822
side
 mha_channel_info_t, 744
sign_t
 MHASignal::schroeder_t, 1242
signal
 testplugin::if_t, 1482
signal_counter
 MHASignal, 149
signal_dimensions
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,
 709
signal_gen_t
 audiometerbackend::signal_gen_t, 323
signal_out
 MHAPlugin_Split::split_t, 1175
signal_parser_t
 testplugin::signal_parser_t, 1483
signal_type
 matlab_wrapper::types< MHA_SPECTRUM
 >, 710
 matlab_wrapper::types< MHA_WAVEFORM
 >, 711
sigtype
 audiometerbackend::audiometer_if_t, 317
sin125
 speechnoise_t, 1467
sin1k
 speechnoise_t, 1467
sin250
 speechnoise_t, 1467
sin2k
 speechnoise_t, 1467
sin4k
 speechnoise_t, 1467
sin500
 speechnoise_t, 1467
sin8k
 speechnoise_t, 1467
sinc
 MHAFilter, 109
sine.cpp, 1695
sine_cfg_t, 1428
 amplitude, 1429
 channels, 1429
 mix, 1429
 phase_increment_div_2pi, 1429
 sine_cfg_t, 1429
sine_t, 1430
 audiometerbackend::sine_t, 324
 channels, 1432
 frequency, 1432
 lev, 1432
 mode, 1432
 patchbay, 1432
 phase_div_2pi, 1432
 prepare, 1431
 process, 1431

sine_t, 1431
 update_cfg, 1432
 sInput
 MHAFilter::fftfilter_t, 880
 size
 MHA_AC::ac2matrix_helper_t, 721
 MHA_AC::acspace2matrix_t, 727
 MHAKernel::algo_comm_class_t, 989
 MHASignal::matrix_t, 1227
 osc2ac_t, 1320
 Vector and matrix processing toolbox, 43,
 44
 size_t
 MHAParser::keyword_list_t, 1067
 skip
 ac2lsl::ac2lsl_t, 165
 ac2lsl::cfg_t, 168
 ac2osc_t, 184
 lsl2ac::save_var_t, 691
 skipcnt
 ac2lsl::cfg_t, 168
 ac2osc_t, 184
 Sleep
 mha_tcp.hh, 1641
 slope
 softclipper_t, 1457
 softclipper_variables_t, 1459
 slope_db
 softclip_t, 1455
 smooth_cepstrum, 161
 smooth_cepstrum.cpp, 1695
 INSERT_PATCH, 1696
 INSERT_VAR, 1696
 PATCH_VAR, 1696
 smooth_cepstrum.hh, 1696
 smooth_cepstrum::smooth_cepstrum_if_t,
 1433
 alpha_const_limits_hz, 1437
 alpha_const_vals, 1437
 alpha_pitch, 1436
 beta_const, 1436
 delta_pitch, 1436
 f0_high, 1436
 f0_low, 1436
 gain_min_db, 1437
 kappa_const, 1436
 lambda_thresh, 1436
 noisePow_name, 1437
 on_model_param_valuechanged, 1435
 patchbay, 1438
 prepare, 1435
 prepared, 1438
 prior_q, 1437
 process, 1434
 release, 1435
 smooth_cepstrum_if_t, 1434
 spp, 1437
 update_cfg, 1435
 win_f0, 1437
 xi_min_db, 1436
 xi_opt_db, 1437
 smooth_cepstrum::smooth_cepstrum_t, 1438
 ~smooth_cepstrum_t, 1439
 ac, 1440
 alpha_const, 1441
 alpha_frame, 1443
 alpha_hat, 1442
 alpha_prev, 1441
 ffflen, 1440
 gain_min, 1441
 gain_wiener, 1443
 gamma_post, 1442
 GLR, 1444
 GLRexp, 1444
 lambda_ceps, 1443
 lambda_ceps_prev, 1443
 lambda_ml_ceps, 1442
 lambda_ml_full, 1442
 lambda_ml_smooth, 1442
 lambda_spec, 1443
 log_lambda_spec, 1443
 logGLRFact, 1444
 max_q, 1444
 max_val, 1444
 mha_fft, 1440
 nchan, 1441
 nfreq, 1440
 noisePow, 1442
 ola_powspec_scale, 1441
 operator=, 1440
 params, 1440
 pitch_set_first, 1444
 pitch_set_last, 1444
 powSpec, 1442
 priorFact, 1444
 process, 1440
 q_high, 1441
 q_low, 1441
 smooth_cepstrum_t, 1439
 spec_out, 1443
 winF0, 1441
 xi_est, 1443

xi_min, 1441
xi_ml, 1442
xiOpt, 1444
smooth_cepstrum::smooth_params, 1445
 alpha_const_limits_hz, 1447
 alpha_const_vals, 1447
 alpha_pitch, 1447
 beta_const, 1447
 delta_pitch, 1446
 f0_high, 1446
 f0_low, 1446
 gain_min_db, 1447
 in_cfg, 1446
 kappa_const, 1447
 lambda_thresh, 1446
 noisePow_name, 1448
 prior_q, 1447
 smooth_params, 1445
 winF0, 1447
 xi_min_db, 1446
 xi_opt_db, 1447
smooth_cepstrum_if_t
 smooth_cepstrum::smooth_cepstrum_if_t, 1434
smooth_cepstrum_t
 smooth_cepstrum::smooth_cepstrum_t, 1439
smooth_params
 smooth_cepstrum::smooth_params, 1445
smoothgains_bridge, 161
smoothgains_bridge.cpp, 1696
smoothgains_bridge::overlapadd_if_t, 1448
 ~overlapadd_if_t, 1449
algo, 1451
cf_in, 1451
cf_out, 1451
epsilon, 1450
irswnd, 1450
mode, 1450
overlapadd_if_t, 1449
patchbay, 1450
plugloader, 1450
prepare, 1449
process, 1450
release, 1449
update, 1450
smoothgains_bridge::smoothspec_wrap_t, 1451
 proc_1, 1452
 proc_2, 1452
 smoothspec, 1452
smoothspec_epsilon, 1453
smoothspec_wrap_t, 1452
spec_in_copy, 1452
use_smoothspec, 1452
smoothspec
 MHAFilter::smoothspec_t, 930
 smoothgains_bridge::smoothspec_wrap_t, 1452
smoothspec_epsilon
 smoothgains_bridge::smoothspec_wrap_t, 1453
smoothspec_t
 MHAFilter::smoothspec_t, 929
smoothspec_wrap_t
 smoothgains_bridge::smoothspec_wrap_t, 1452
smp2rad
 Vector and matrix processing toolbox, 40
smp2sec
 MHASignal, 143
smpl
 adaptive_feedback_canceller_config, 249
sn_in
 MHAJack::client_avg_t, 966
 MHAJack::client_noncont_t, 969
sn_out
 MHAJack::client_avg_t, 966
 MHAJack::client_noncont_t, 969
sndfile_t
 addsndfile::sndfile_t, 259
snprintf_required_length
 mha_error_helpers, 100
snrPost1Debug
 noise_psd_estimator::noise_psd_estimator_t, 1313
sock_addr
 MHA_TCP::Server, 817
sock_init_t
 MHA_TCP::sock_init_t, 824
sock_initializer
 MHA_TCP, 103
Sockaccept_Event
 MHA_TCP::Sockaccept_Event, 825
SOCKET
 MHA_TCP, 102
 mha_tcp.cpp, 1639
SOCKET_ERROR
 mha_tcp.cpp, 1638
Sockread_Event
 MHA_TCP::Sockread_Event, 826
Sockwrite_Event

MHA_TCP::Sockwrite_Event, 827
softclip
 calibrator_runtime_layer_t, 338
 calibrator_variables_t, 344
softclip.cpp, 1697
softclip_t, 1453
 attack, 1455
 decay, 1455
 patchbay, 1455
 prepare, 1454
 process, 1454
 slope_db, 1455
 softclip_t, 1454
 start_limit, 1455
 tftype, 1455
 update, 1454
softclipper_t, 1455
 attack, 1456
 cliptometer, 1456
 decay, 1456
 hardlimit, 1457
 linear, 1457
 process, 1456
 slope, 1457
 softclipper_t, 1456
 threshold, 1457
softclipper_variables_t, 1457
 clipped, 1459
 hardlimit, 1459
 linear, 1459
 max_clipped, 1459
 slope, 1459
 softclipper_variables_t, 1458
 tau_attack, 1458
 tau_clip, 1458
 tau_decay, 1458
 threshold, 1459
solve_MVDR
 rohBeam::rohBeam, 1397
sort_fftw2spec
 MHASignal::fft_t, 1210
sort_spec2fftw
 MHASignal::fft_t, 1210
sound
 io_asterisk_t, 602
 io_tcp_t, 640
source_azimuth_degrees
 rohBeam::rohBeam, 1399
source_channel_index
 MHAFilter::partitioned_convolution_t::index_spec_fader_t, 1465
 920
 ~spec_fader_t, 1465
source_id
 ac2lsl::ac2lsl_t, 165
 ac2lsl::cfg_t, 168
sout
 matrixmixer::cfg_t, 713
 route::process_t, 1412
sout_ac
 route::process_t, 1412
sout_buf
 gtfb_simd_cfg_t, 560
spec2fir
 MHAFilter, 108
 MHAFilter::smoothspec_t, 931
spec2spec
 plugindescription_t, 1357
spec2wave
 MHASignal::fft_t, 1208, 1209
 overlapadd::overlapadd_t, 1333
 plugindescription_t, 1357
spec2wave.cpp, 1697
 max, 1697
 min, 1697
spec2wave_if_t, 1460
 prepare, 1461
 process, 1461
 ramplen, 1462
 release, 1461
 setlock, 1461
 spec2wave_if_t, 1461
 update, 1461
 window_config, 1462
spec2wave_scale
 MHASignal::fft_t, 1209
spec2wave_t, 1462
 ~spec2wave_t, 1463
 calc_out, 1464
 ft, 1463
 nfft, 1464
 npad1, 1463
 npad2, 1463
 nwndshift, 1464
 out_buf, 1464
 postwindow, 1464
 process, 1463
 ramps, 1464
 sc, 1464
 spec2wave_t, 1463
 write_buf, 1464
 ~spec_fader_t, 1465

fr, 1465
gains, 1465
nch, 1465
spec_fader_t, 1465
spec_in
matlab_wrapper::matlab_wrapper_t::wrapped_phoneme_type_t, 1467
709
MHAPlugin_Split::domain_handler_t, 1164
overlapadd::overlapadd_t, 1334
wave2spec_t, 1496
spec_in_copy
smoothgains_bridge::smoothspec_wrap_t, 1452
spec_out
matlab_wrapper::matlab_wrapper_t::wrapped_pspectrohnoise_t, 1467, 1468
709
MHAPlugin_Split::domain_handler_t, 1164
MHAPlugin_Split::split_t, 1177
smooth_cepstrum::smooth_cepstrum_t, 1443
specSteer1
acSteer_config, 233
specSteer2
acSteer_config, 233
spectrum_t
MHA_AC::spectrum_t, 735
MHAMultiSrc::spectrum_t, 995
MHASignal::spectrum_t, 1245, 1246
speechnoise
calibrator_runtime_layer_t, 338
speechnoise.cpp, 1697
bandw_correction, 1700
erb_hz_f_hz, 1699
fHz2bandno, 1699
hz2hz, 1699
NUM_ENTR_LTASS, 1699
NUM_ENTR_MHAORIG, 1699
NUM_ENTR_OLNOISE, 1699
vLTASS_combined_lev, 1701
vLTASS_female_lev, 1701
vLTASS_freq, 1700
vLTASS_male_lev, 1701
vMHAOrigFreq, 1700
vMHAOrigSpec, 1700
vOlnoiseFreq, 1701
vOlnoiseLev, 1701
speechnoise.h, 1701
speechnoise_t, 1466
brown, 1467
creator, 1468
LTASS_combined, 1467
LTASS_female, 1467
LTASS_male, 1467
mha, 1467
olnoise, 1467
pink, 1467
sin125, 1467
sin1k, 1467
sin250, 1467
sin2k, 1467
sin4k, 1467
sin500, 1467
sin8k, 1467
pspectrohnoise_t, 1467
TEN_SPL, 1467
TEN_SPL_250_8k, 1467
TEN_SPL_50_16k, 1467
white, 1467
SPL
rmslevel, 159
spl2hl
MHAUtils, 153
split.cpp, 1702
MHAPLUGIN_OVERLOAD_OUTDOMAIN, 1703
native_thread_platform_type, 1703
posixthreads, 1703
split_t
MHAPlugin_Split::split_t, 1173
splitted_part_t
MHAPLUGIN_SPLIT::splitted_part_t, 1179, 1180
spnoise_channels
calibrator_variables_t, 343
spnoise_level
calibrator_variables_t, 343
spnoise_mode
calibrator_variables_t, 343
spnoise_parser
calibrator_variables_t, 343
spp
smooth_cepstrum::smooth_cepstrum_if_t, 1437
sq2db
MHASignal, 140
srate
ac2lsl::cfg_t, 168
adm_if_t, 275
calibrator_variables_t, 343

mhaconfig_t, 852
 MHAParser::mhaconfig_mon_t, 1085
 MHAPlugin_Resampling::resampling_if_t, 1154
 testplugin::config_parser_t, 1479
 srate_
 MHAFilter::gamma_flt_t, 895
 srv
 osc2ac_t, 1320
 start
 alsa_base_t, 287
 alsa_t< T >, 293
 fw_t, 525
 io_alsa_t, 580
 io_asterisk_fwcb_t, 585
 io_asterisk_t, 601
 io_file_t, 606
 io_lib_t, 613
 io_parser_t, 617
 io_tcp_fwcb_t, 621
 io_tcp_t, 639
 MHAJack::client_t, 974
 START_BETA
 ADM, 82
 start_event
 io_alsa_t, 582
 io_asterisk_fwcb_t, 587
 io_file_t, 608
 io_parser_t, 618
 io_tcp_fwcb_t, 623
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 978
 start_handle
 io_alsa_t, 582
 io_asterisk_fwcb_t, 587
 io_file_t, 608
 io_parser_t, 618
 io_tcp_fwcb_t, 623
 MHAIOPortAudio::io_portaudio_t, 961
 MHAJack::client_t, 978
 start_limit
 softclip_t, 1455
 start_lin
 cfg_t, 347
 start_new_session
 plugins::hoertech::acrec::acrec_t, 1377
 wavrec_t, 1499
 start_stdin_thread
 mhaserver_t, 1191
 started
 fw_t, 526
 io_parser_t, 617
 starting
 ma_drifter_fifo_t< T >, 760
 startpos
 addsndfile::addsndfile_if_t, 253
 startsample
 io_file_t, 609
 startup_zeros
 ma_drifter_fifo_t< T >, 763
 stat_t
 MHA_AC::stat_t, 737
 MHASignal::stat_t, 1251
 state
 fw_t, 530
 gtfb_analyzer::gtfb_analyzer_cfg_t, 552
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 707
 MHA_TCP::Thread, 831
 MHAFilter::filter_t, 891
 state_cupload
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 state_parser
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 state_priority
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 state_scheduler
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 state_t
 fw_t, 524
 state_xruns
 MHAIOJack::io_jack_t, 945
 MHAIOJackdb::io_jack_t, 953
 states
 gtfb_analyzer::gtfb_analyzer_cfg_t, 550
 staticgain
 coherence::cohflt_t, 354
 coherence::vars_t, 356
 status
 MHA_TCP::Wakeup_Event, 837
 std
 MHA_AC::stat_t, 738
 std_vector_float
 Vector and matrix processing toolbox, 49
 std_vector_vector_complex
 Vector and matrix processing toolbox, 49
 std_vector_vector_float
 Vector and matrix processing toolbox, 49

stdcomplex
Complex arithmetics in the openMHA, 61

steerbf, 1469
~steerbf, 1470
angle_ind, 1471
angle_src, 1471
bf_src, 1471
patchbay, 1471
prepare, 1470
process, 1470
release, 1471
steerbf, 1470
update_cfg, 1471

steerbf.cpp, 1703
INSERT_PATCH, 1703
PATCH_VAR, 1703

steerbf.h, 1704

steerbf_config, 1472
_steerbf, 1473
~steerbf_config, 1472

ac, 1473
bf_src_copy, 1473
bf_vec, 1473
nangle, 1473
nchan, 1473
nfreq, 1473
outSpec, 1473
process, 1472
steerbf_config, 1472

steerFile
acSteer, 231

stime
MHA_TCP, 103

stop
alsa_base_t, 287
alsa_t< T >, 293
fw_t, 525
io_alsa_t, 581
io_asterisk_fwcb_t, 586
io_asterisk_t, 601
io_file_t, 606
io_lib_t, 613
io_parser_t, 617
io_tcp_fwcb_t, 622
io_tcp_t, 639
mha_drifter_fifo_t< T >, 760
MHAJack::client_t, 974

stop_event
io_alsa_t, 582
io_asterisk_fwcb_t, 587
io_file_t, 608

io_parser_t, 618
io_tcp_fwcb_t, 623
MHAIOPortAudio::io_portaudio_t, 961
MHAJack::client_t, 978

stop_handle
io_alsa_t, 582
io_asterisk_fwcb_t, 587
io_file_t, 608
io_parser_t, 618
io_tcp_fwcb_t, 623
MHAIOPortAudio::io_portaudio_t, 961
MHAJack::client_t, 978

stopped
fw_t, 525, 526
io_file_t, 606
io_parser_t, 617
MHAJack::client_t, 977
route::interface_t, 1411

store
rmslevel::mon_t, 1386

store_frame
acsave::cfg_t, 224
acsave::save_var_t, 227

str2val
MHAParser::StrCnv, 130, 131

str2val< mha_real_t >
MHAParser::StrCnv, 131

str_a
ac_mul_t, 195

str_b
ac_mul_t, 195

str_error
MHAJack::client_t, 975

strdom
analysemhaplugin.cpp, 1529
latex_doc_t, 645

stream
ac2lsl::save_var_t< mha_complex_t >, 178
ac2lsl::save_var_t< T >, 175
lsl2ac::save_var_t, 689

stream_dir
alsa_dev_par_parser_t, 290

streambuf
mha_tcp::buffered_socket_t, 800

streams
lsl2ac::lsl2ac_t, 683

STRERROR
MHA_TCP, 102

strict_channel_match
io_file_t, 609

strict_srate_match
 io_file_t, 609
 strict_window_ratio
 overlapadd::overlapadd_if_t, 1330
 wave2spec_if_t, 1490
 stride
 comm_var_t, 362
 testplugin::ac_parser_t, 1476
 string_mon_t
 MHAParser::string_mon_t, 1109
 string_t
 MHAParser::string_t, 1111
 strip
 MHAUtils, 151
 STRLEN
 mha_errno.c, 1584
 strNames_AC
 acConcat_wave_config, 203
 strreplace
 MHAParser, 127
 structVersion
 MHAIOPortAudio::device_info_t, 955
 sub4f
 gtfb_simd.cpp, 1557
 sub_ac
 dbasync_native::db_if_t, 376
 subsample_delay_t
 MHASignal::subsample_delay_t, 1253
 subsampledelay_coeff
 ADM, 81
 sum
 MHASignal::stat_t, 1252
 MHASignal::waveform_t, 1265
 sum2
 MHASignal::stat_t, 1252
 sum_and_fill
 rmslevel::rmslevel_t, 1391, 1392
 sum_channel
 MHASignal::waveform_t, 1266
 sumsqr
 MHASignal::waveform_t, 1266
 sumsqr_channel
 Vector and matrix processing toolbox, 56
 sumsqr_frame
 Vector and matrix processing toolbox, 57
 svc
 analysepath_t, 307
 dbasync_native::dbasync_t, 379
 sWeights
 MHAFilter::fftfilter_t, 880
 symmetry_scale

MHAOvlFilter::fspacing_t, 1017
 sync
 mha_fifo_lw_t< T >, 772
 mha_fifo_thread_guard_t, 783
 sync_osc2ac
 osc_server_t, 1322
 osc_variable_t, 1325
 sysread
 MHA_TCP::Connection, 805
 syswrite
 MHA_TCP::Connection, 805

T

t
 acsave::mat4head_t, 225
 mha_tictoc_t, 838
 plingploing::plingploing_t, 1343
 table
 cpupload::cpupload_cfg_t, 366
 table_size
 cpupload::cpupload_if_t, 369
 table_t
 MHATableLookup::table_t, 1280
 tail
 MHAFilter::fftfilterbank_t, 886
 target_channel_index
 MHAFilter::partitioned_convolution_t::index_t, 920
 MHAFilter::transfer_function_t, 938
 tau
 coherence::vars_t, 355
 droptect_t, 442
 fader_if_t, 487
 levelmeter_t, 659
 tau_attack
 softclipper_variables_t, 1458
 tau_beta
 adm_if_t, 274
 tau_blocking_XkXi_ms
 rohBeam::rohBeam, 1400
 tau_blocking_XkY_ms
 rohBeam::rohBeam, 1401
 tau_clip
 softclipper_variables_t, 1458
 tau_decay
 softclipper_variables_t, 1458
 tau_level
 calibrator_variables_t, 343
 tau_Lowpass
 windnoise::if_t, 1511
 tau_postfilter_ms

rohBeam::rohBeam, 1400
tau_unit
 coherence::vars_t, 355
tauattack
 dc::dc_vars_t, 389
 dc_simple::dc_vars_t, 403
taudecay
 dc::dc_vars_t, 390
 dc_simple::dc_vars_t, 403
taugain
 DynComp::dc_afterburn_vars_t, 456
taurmslevel
 dc::dc_vars_t, 389
tc
 lsl2ac::save_var_t, 690
tc_name
 lsl2ac::save_var_t, 690
tcp_connect_to
 mha_tcp.cpp, 1639
tcp_connect_to_with_timeout
 mha_tcp.cpp, 1639
tcp_server_t
 mhserver_t::tcp_server_t, 1194
tcpserver
 mhserver_t, 1192
TEN_SPL
 speechnoise_t, 1467
TEN_SPL_250_8k
 speechnoise_t, 1467
TEN_SPL_50_16k
 speechnoise_t, 1467
termination_request
 MHAPlugin_Split::posix_threads_t, 1171
test_error
 io_lib_t, 613
 MHAParser::c_ifc_parser_t, 1047
 PluginLoader::mhapluginloader_t, 1370
test_fail
 dc_simple, 87
test_prepare
 testplugin::if_t, 1481
test_process
 testplugin::if_t, 1481
test_version
 PluginLoader::mhapluginloader_t, 1370
testsalsadevice.c, 1704
 main, 1704
testplugin, 161
testplugin.cpp, 1704
testplugin::ac_parser_t, 1474
 _MHA_AC_CHAR, 1475
 _MHA_AC_DOUBLE, 1475
 _MHA_AC_FLOAT, 1475
 _MHA_AC_INT, 1475
 _MHA_AC_MHACOMPLEX, 1475
 _MHA_AC_MHAREAL, 1475
 unknown, 1475
ac_parser_t, 1475
char_data, 1476
complex_data, 1476
data_type, 1476
data_type_t, 1475
do_get_var, 1475
do_insert_var, 1475
float_data, 1476
get_var, 1476
insert_var, 1476
int_data, 1476
num_entries, 1476
patchbay, 1477
stride, 1476
testplugin::config_parser_t, 1477
 channels, 1478
 config_parser_t, 1478
 domain, 1479
 ffflen, 1479
 fragsize, 1479
 get, 1478
 set, 1478
 setlock, 1478
 srate, 1479
 wndlen, 1479
testplugin::if_t, 1480
 _prepare, 1482
 ac, 1482
 config_in, 1482
 config_out, 1482
 if_t, 1481
 patchbay, 1482
 plug, 1482
 prepare, 1481
 process, 1481
 signal, 1482
 test_prepare, 1481
 test_process, 1481
testplugin::signal_parser_t, 1483
 input_spec, 1484
 input_wave, 1484
 output_spec, 1484
 output_wave, 1484
 signal_parser_t, 1483
tftype

MHAParser::plugin_t< runtime_cfg_t >, 1151
 softclip_t, 1455
 The MHA Framework interface, 22
 The openMHA configuration language, 30
 The openMHA Toolbox library, 31
 thefullname
 MHAParser::base_t, 1039
 thirddoctave_analyzer_t
 MHAFilter::thirddoctave_analyzer_t, 933
 this_outer_out
 MHASignal::doublebuffer_t, 1206
 thr_f
 MHA_TCP::Thread, 828
 Thread
 MHA_TCP::Thread, 829
 thread
 analysepath_t, 309
 dbasync_native::dbasync_t, 379
 io_asterisk_t, 603
 io_tcp_t, 640
 MHAParser::posix_threads_t, 1170
 MHAParser::splitted_part_t, 1183
 thread_arg
 MHA_TCP::Thread, 831
 thread_attr
 MHA_TCP::Thread, 830
 thread_finish_event
 MHA_TCP::Thread, 831
 thread_func
 MHA_TCP::Thread, 831
 thread_handle
 MHA_TCP::Thread, 830
 thread_platform
 MHAParser::split_t, 1176
 thread_platform_t
 MHAParser::thread_platform_t, 1184
 thread_start
 analysispath.cpp, 1530
 dbasync_native, 86
 io_alsa_t, 581
 MHAParser::posix_threads_t, 1169
 thread_start_func
 mha_tcp.cpp, 1639
 thread_startup_function
 MHAIOAsterisk.cpp, 1653
 MHAIOTCP.cpp, 1679
 threshold
 droptect_t, 442
 softclipper_t, 1457
 softclipper_variables_t, 1459
 threshold_compare
 windnoise::cfg_t, 1505
 tic
 lsl2ac::save_var_t, 691
 tictoc
 mhachain::plugs_t, 850
 timeout
 MHA_TCP::OS_EVENT_TYPE, 814
 MHA_TCP::Timeout_Watcher, 834
 Timeout_Event
 MHA_TCP::Timeout_Event, 832
 Timeout_Watcher
 MHA_TCP::Timeout_Watcher, 834
 timeshift
 Vector and matrix processing toolbox, 45
 tmp
 level_matching::level_matching_config_t, 652
 tmp_spec
 MHAFilter::smoothspec_t, 932
 tmp_wave
 MHAFilter::smoothspec_t, 932
 to_from
 acTransform_wave, 237
 acTransform_wave_config, 239
 to_int_clamped
 mhaioutils, 111
 to_iso8601
 plugins::hoertech::acrec, 158
 total_read
 io_file_t, 610
 transducers.cpp, 1705
 kw_index2type, 1705
 rmslevelmeter, 1705
 rtcalibrator, 1705
 vint_0123n1, 1706
 transfer_function_t
 MHAFilter::transfer_function_t, 936
 trigger_accept
 mha_tcp::server_t, 822
 trigger_processing
 MHAParser::split_t, 1175
 MHAParser::splitted_part_t, 1182
 trigger_read_line
 mha_tcp::server_t, 822
 trim
 MHAParser, 126
 try_accept
 MHA_TCP::Server, 816
 try_write

MHA_TCP::Connection, 808
ts
 lsl2ac::save_var_t, 690
ts_buf
 lsl2ac::save_var_t, 689
ts_name
 lsl2ac::save_var_t, 690
ttl
 ac2osc_t, 183
tv1
 mha_tictoc_t, 838
tv2
 mha_tictoc_t, 838
types
 ac2lsl, 78
tz
 mha_tictoc_t, 838

U
 rt_nlms_t, 1415
UbufferPrew
 adaptive_feedback_canceller_config, 248
UCL
 AuditoryProfile::parser_t::ear_t, 329
 AuditoryProfile::profile_t::ear_t, 334
Uflit
 rt_nlms_t, 1415
uint_mode
 addsndfile::addsndfile_if_t, 254
uint_vector_t
 MHASignal::uint_vector_t, 1256
underflow
 MHAFilter::polyphase_resampling_t, 925
UNIT
 rmslevel, 159
unit
 MHAOvlFilter::fscale_t, 1013
 rmslevel::rmslevel_if_t, 1389
 rmslevel::rmslevel_t, 1392
unit2hz
 MHAOvlFilter::scale_var_t, 1025
unlock_channels
 fw_vars_t, 531
unlock_srate_fragsize
 fw_vars_t, 531
unset_fb_pars
 DynComp::dc_afterburn_t, 453
up
 MHASignal::schroeder_t, 1242
up_incl
 MHAParser::range_var_t, 1107
up_limit

MHAParser::range_var_t, 1107
MHASignal::quantizer_t, 1236
up_thresh
 acPooling_wave_config, 218
update
 ac2lsl::ac2lsl_t, 165
 ac2wave_if_t, 187
 addsndfile::addsndfile_if_t, 252
 adm_if_t, 273
 audiometerbackend::audiometer_if_t, 317
 calibrator_t, 340
 coherence::cohflt_if_t, 349
 cpuload::cpuload_if_t, 368
 dc::dc_if_t, 382
 delay::interface_t, 410
 DynComp::dc_afterburn_t, 453
 DynComp::gaintable_t, 459
 fftfilter::interface_t, 502
 fshift_hilbert::frequency_translator_t, 516
 lsl2ac::lsl2ac_t, 683
 matlab_wrapper::matlab_wrapper_t, 700
 mconv::MConv, 718
 MHA_AC::ac2matrix_t, 723
 MHA_AC::acspace2matrix_t, 727
 MHA_AC::stat_t, 737
 mhachain::chain_base_t, 843
 MHAMultiSrc::spectrum_t, 995
 MHAMultiSrc::waveform_t, 997
 MHAParser::mhaconfig_mon_t, 1084
 MHAPlugin_Split::split_t, 1174
 nlms_t, 1307
 overlapadd::overlapadd_if_t, 1329
 plingploing::if_t, 1339
 rmslevel::rmslevel_if_t, 1389
 route::interface_t, 1410
 smoothgains_bridge::overlapadd_if_t,
 1450
 softclip_t, 1454
 spec2wave_if_t, 1461
 wave2spec_if_t, 1489
 windnoise::if_t, 1510
update_burner
 DynComp::dc_afterburn_t, 453
update_cfg
 acConcat_wave, 201
 acPooling_wave, 213
 acSteer, 231
 acTransform_wave, 236
 adaptive_feedback_canceller, 242
 complex_scale_channel_t, 364
 delaysum::delaysum_wave_if_t, 413

delaysum_spec::delaysum_spec_if_t, 417
 doasvm_classification, 422
 doasvm_feature_extraction, 428
 example6_t, 483
 fader_if_t, 487
 fftfbpow::fftfbpow_interface_t, 495
 fftfilterbank::fftfb_interface_t, 505
 fshift::fshift_t, 513
 gsc_adaptive_stage::gsc_adaptive_stage_if_t, 547
 gtfb_analyzer::gtfb_analyzer_t, 554
 gtfb_simd_t, 562
 level_matching::level_matching_t, 655
 lpc, 662
 lpc_bl_predictor, 665
 lpc_burglattice, 671
 multibandcompressor::interface_t, 1301
 noise_psd_estimator::noise_psd_estimator_if_t, 1310
 noise_t, 1316
 plugin_interface_t, 1353
 rohBeam::rohBeam, 1396
 sine_t, 1432
 smooth_cepstrum::smooth_cepstrum_if_t, 1435
 steerbf, 1471
 update_coeffs
 MHAFilter::fftfilter_t, 878
 MHAFilter::fftfilterbank_t, 883
 update_dc
 dc_simple::dc_if_t, 395
 update_filter
 MHAFilter::iir_filter_t, 899
 update_frame
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 319
 fader_wave::level_adapt_t, 492
 update_gain
 gain::gain_if_t, 534
 update_gain_mon
 dc_simple::dc_if_t, 395
 update_gains
 equalize::freqgains_t, 465
 update_hz
 MHAOvIFilter::fscale_bw_t, 1011
 MHAOvIFilter::fscale_t, 1013
 update_id
 equalize::freqgains_t, 465
 update_irs
 mconv::MConv, 718
 update_level
 dc_simple::dc_if_t, 395
 update_level_mon
 dc_simple::dc_if_t, 395
 update_levels
 multibandcompressor::plugin_signals_t, 1303
 update_m
 matrixmixer::matmix_t, 715
 update_minmax
 gain::gain_if_t, 534
 update_mismatch
 level_matching::channel_pair, 648, 649
 update_mode
 ac2osc_t, 183
 update_monitors
 dc::dc_if_t, 382
 matlab_wrapper::matlab_wrapper_t, 700
 update_mu
 MHAFilter::adapt_filter_t, 866
 update_ntaps
 MHAFilter::adapt_filter_t, 867
 update_parser
 windowselector_t, 1515
 update_proc_load
 mhachain::plugs_t, 848
 update_PSD_Lowpass
 windnoise::cfg_t, 1505
 update_rampen
 altpicks_t, 303
 update_recmode
 acmon::acmon_t, 209
 update_selector_list
 altpicks_t, 303
 update_tau
 levelmeter_t, 658
 update_tau_level
 calibrator_t, 341
 update_varlist
 ac2lsl::cfg_t, 168
 updated
 windowselector_t, 1515
 updater
 MHAOvIFilter::fscale_bw_t, 1012
 MHAOvIFilter::fscale_t, 1014
 upper_threshold
 acPooling_wave, 214
 UPrew
 adaptive_feedback_canceller_config, 249
 UPrewW
 adaptive_feedback_canceller_config, 249
 upsample.cpp, 1706

upsampling_factor
 MHAFilter::polyphase_resampling_t, 924

upscale
 MHASignal::quantizer_t, 1236

us_t, 1484
 antialias, 1486
 prepare, 1485
 process, 1485
 ratio, 1486
 release, 1485
 us_t, 1485

use_date
 plugins::hoertech::acrec::acrec_t, 1378
 wavrec_t, 1499

use_frozen_
 cfg_t, 346

use_jack_transport
 MHAIOJackdb::io_jack_t, 951
 MHAJack::client_t, 979

use_mat
 acmon::ac_monitor_t, 206

use_own_ac
 altplugs_t, 303

use_sine
 cpuload::cpuload_cfg_t, 366
 cpuload::cpuload_if_t, 369

use_smoothspec
 smoothgains_bridge::smoothspec_wrap_t,
 1452

UseChannel_LF_attenuation
 windnoise::cfg_t, 1506
 windnoise::if_t, 1511

user
 MHAParser::window_t, 1137

user_config
 matlab_wrapper::callback, 693
 matlab_wrapper::matlab_wrapper_rt_cfg_t,
 695
 matlab_wrapper::matlab_wrapper_t::wrapped_p

user_err_msg
 MHAIOalsa.cpp, 1649
 MHAIOAsterisk.cpp, 1654
 MHAIOFile.cpp, 1658
 MHAIOJack.cpp, 1663
 MHAIOJackdb.cpp, 1667
 MHAIOParser.cpp, 1671
 MHAIOPortAudio.cpp, 1675
 MHAIOTCP.cpp, 1681

user_t
 MHAWindow::user_t, 1297

username
 MHA_AC::ac2matrix_helper_t, 721

userwnd
 windowselector_t, 1516

useVAD
 gsc_adaptive_stage::gsc_adaptive_stage,
 541
 gsc_adaptive_stage::gsc_adaptive_stage_if,
 548

v_G
 adaptive_feedback_canceller_config, 247

vadName
 gsc_adaptive_stage::gsc_adaptive_stage,
 541
 gsc_adaptive_stage::gsc_adaptive_stage_if,
 548

val2str
 MHAParser::StrCnv, 132–134

VAL_COMPLEX
 ac_mul.hh, 1519

VAL_REAL
 ac_mul.hh, 1519

val_type_t
 ac_mul.hh, 1518

validate
 AuditoryProfile::parser_t::fmap_t, 330
 MHAParser::keyword_list_t, 1069
 MHAParser::kw_t, 1072
 MHAParser::range_var_t, 1105, 1106

validator_channels
 mhasndfile.cpp, 1687

validator_length
 mhasndfile.cpp, 1687

value
 AuditoryProfile::parser_t::fmap_t, 331
 mha_signal.hh, 1635
 MHASignal::ringbuffer_t, 1238
 MHASignal::spectrum_t, 1247
 MHASignal::waveform_t, 1263, 1264
 Vector and matrix processing toolbox, 46–
 49

value_type
 mha_dblbuf_t< FIFO >, 748
 mha_fifo_t< T >, 777

valuechanged
 MHAParser::base_t, 1037

var
 matlab_wrapper::callback, 694

variable, 4

variable_name
 mha_stash_environment_variable_t, 796

variable_t
 MHAParser::variable_t, 1114

variables, 4
 acsave::acsave_t, 222

varlist
 ac2lsl::cfg_t, 168
 acmon::acmon_t, 209
 acsave::acsave_t, 222
 acsave::cfg_t, 224
 lsl2ac::cfg_t, 679

varlist_t
 acsave::acsave_t, 220

varname
 plugins::hoertech::acrec::acrec_t, 1378
 plugins::hoertech::acrec::acwriter_t, 1384

vars
 ac2lsl::ac2lsl_t, 165
 ac2osc_t, 183
 acmon::acmon_t, 210
 analysispath_if_t, 313
 calibrator_t, 341
 coherence::cohfilt_if_t, 350
 matlab_wrapper::matlab_wrapper_t, 701
 MHAKernel::algo_comm_class_t, 990
 osc2ac_t, 1320

vars_t
 coherence::vars_t, 355
 MHAOvIFilter::overlap_save_filterbank_t::vars_t,DynComp::gaintable_t, 461
 1022

v bark
 MHAOvIFilter::barkscale, 117

vbin1
 MHAOvIFilter::fftfb_t, 1005

vbin2
 MHAOvIFilter::fftfb_t, 1005

vcomplex_mon_t
 MHAParser::vcomplex_mon_t, 1116

vcomplex_t
 MHAParser::vcomplex_t, 1118

vec_y
 MHATableLookup::linear_table_t, 1278

Vector and matrix processing toolbox, 33
 assign, 44, 45
 bin2freq, 39
 channels, 38
 clear, 44
 colored_intensity, 54
 conjugate, 57
 copy_channel, 52, 53
 dupvec, 41
 dupvec_chk, 41

equal_dim, 42
 freq2bin, 39
 integrate, 43
 max, 56
 maxabs, 54, 55
 mha_real_t, 37
 min, 56
 operator*=, 50, 51
 operator^=, 52
 operator+=, 50, 52
 operator-=, 50
 operator/=, 51, 52
 rad2smp, 40
 range, 38
 rmslevel, 53, 55
 size, 43, 44
 smp2rad, 40
 std_vector_float, 49
 std_vector_vector_complex, 49
 std_vector_vector_float, 49
 sumsqr_channel, 56
 sumsqr_frame, 57
 timeshift, 45
 value, 46–49

VERSION_EXTENSION
 mhamain.cpp, 1685

vF

vfloat_mon_t
 MHAParser::vfloat_mon_t, 1121

vfloat_t
 MHAParser::vfloat_t, 1123

vFlag
 DynComp::gaintable_t, 461

vfreq
 MHAOvIFilter::barkscale, 117

vGCC
 acConcat_wave_config, 203
 doasvm_feature_extraction_config, 431

vGCC_ac
 doasvm_feature_extraction_config, 431

vGCC_con
 acConcat_wave_config, 203

vGCC_name
 doasvm_classification, 423
 doasvm_feature_extraction, 428

vint_0123n1
 transducers.cpp, 1706

vint_mon_t
 MHAParser::vint_mon_t, 1126

vint_t

MHAParser::vint_t, 1128
vL
 DynComp::gaintable_t, 461
vLTASS_combined_lev
 speechnoise.cpp, 1701
vLTASS_female_lev
 speechnoise.cpp, 1701
vLTASS_freq
 speechnoise.cpp, 1700
vLTASS_male_lev
 speechnoise.cpp, 1701
vmax
 gain::gain_if_t, 535
vMHAOrigFreq
 speechnoise.cpp, 1700
vMHAOrigSpec
 speechnoise.cpp, 1700
vmin
 gain::gain_if_t, 534
vOlnoiseFreq
 speechnoise.cpp, 1701
vOlnoiseLev
 speechnoise.cpp, 1701
vstring_mon_t
 MHAParser::vstring_mon_t, 1130
vstring_t
 MHAParser::vstring_t, 1132
vy
 MHATableLookup::linear_table_t, 1278

W

 gsc_adaptive_stage::gsc_adaptive_stage, 541
 MHA_TCP::OS_EVENT_TYPE, 813
 MHAFilter::adapt_filter_state_t, 864

w

 ac2wave_t, 190
 doasvm_classification, 422
 MHAOvlFilter::fftfb_t, 1005

w_out

 combc_t, 360

wait

 MHA_TCP::Event_Watcher, 812

wait_for_decrease

 mha_fifo_posix_threads_t, 774
 mha_fifo_thread_platform_t, 785

wait_for_increase

 mha_fifo_posix_threads_t, 774
 mha_fifo_thread_platform_t, 786

Wakeup_Event

 MHA_TCP::Wakeup_Event, 836

wave

 alsa_t< T >, 294
wave2spec
 MHASignal::fft_t, 1208
 overlapadd::overlapadd_t, 1333
 plugindescription_t, 1356
wave2spec.cpp, 1706
wave2spec.hh, 1706
 MHAPLUGIN_OVERLOAD_OUTDOMAIN, 1706
wave2spec_apply_window
 overlapadd::overlapadd_t, 1333
wave2spec_compute_fft
 overlapadd::overlapadd_t, 1333
wave2spec_hop_forward
 overlapadd::overlapadd_t, 1333
wave2spec_if_t, 1486
 algo, 1491
 nfft, 1490
 nwnd, 1490
 prepare, 1488
 process, 1488, 1489
 release, 1488
 return_wave, 1490
 setlock, 1489
 strict_window_ratio, 1490
 update, 1489
 wave2spec_if_t, 1488
 window_config, 1490
 wndpos, 1490
 zeropadding, 1491
wave2spec_scale
 MHASignal::fft_t, 1209
wave2spec_t, 1491
 ~wave2spec_t, 1493
 ac_wndshape_name, 1496
 calc_in, 1496
 calc_pre_wnd, 1495
 ft, 1495
 get_zeropadding, 1494
 in_buf, 1496
 npad1, 1496
 npad2, 1496
 nwnd, 1495
 nwndshift, 1495
 process, 1494
 publish_ac_variables, 1494
 spec_in, 1496
 wave2spec_t, 1493
 window, 1496
wave2wave
 plugindescription_t, 1356

wave_fifo
 analysepath_t, 308

wave_in
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_thread, 1501
 709

MHAPlugin_Split::domain_handler_t, 1163

wave_in1
 overlapadd::overlapadd_t, 1334

wave_out
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t, 1501
 709

MHAPlugin_Split::domain_handler_t, 1163

MHAPlugin_Split::split_t, 1177

wave_out1
 overlapadd::overlapadd_t, 1334

wave_reader
 addsndfile, 80

waveform_proxy_t
 addsndfile::waveform_proxy_t, 261

waveform_t
 MHA_AC::waveform_t, 739
 MHAMultiSrc::waveform_t, 996
 MHASignal::waveform_t, 1261, 1262

wavrec.cpp, 1707

wavrec_t, 1497
 fifolen, 1499
 minwrite, 1499
 output_sample_format, 1499
 patchbay, 1499
 prefix, 1499
 prepare, 1498
 process, 1498
 record, 1499
 release, 1498
 start_new_session, 1499
 use_date, 1499
 wavrec_t, 1498

wavwriter_t, 1500
 ~wavwriter_t, 1501
 act_, 1502
 cf_, 1502
 close_session, 1502
 create_soundfile, 1501
 data, 1503
 exit_request, 1501
 fifo, 1502
 format_name, 1503
 minw_, 1503
 process, 1501

set_format, 1502
 sf, 1502
 wavwriter_t, 1501
 writethread, 1503

weights
 delaysum::delaysum_wave_if_t, 413
 delaysum::delaysum_wave_t, 415

what
 MHA_Error, 765

win_f0
 smooth_cepstrum::smooth_cepstrum_if_t, 1437

WINAPI
 mha_plugin.hh, 1616

windnoise, 161

windnoise.cpp, 1707
 register_configuration_variable, 1707

windnoise.hh, 1707

windnoise::cfg_t, 1503
 alpha_Lowpass, 1506
 cfg_t, 1504
 compensation, 1506
 FrequencyBinLowPass, 1506
 LowPassFraction, 1507
 LowPassWindGain, 1507
 powspec, 1507
 process, 1505
 PSD_Lowpass, 1507
 remapping, 1506
 threshold_compare, 1505
 update_PSD_Lowpass, 1505
 UseChannel_LF_attenuation, 1506

windnoise::if_t, 1508
 detected, 1511
 detected_acname, 1511
 if_t, 1509
 insert, 1510
 lowpass_quotient, 1511
 lowpass_quotient_acname, 1512
 LowPassCutOffFrequency, 1511
 LowPassFraction, 1511
 LowPassWindGain, 1511
 patchbay, 1510
 prepare, 1509
 process, 1510
 release, 1510
 tau_Lowpass, 1511
 update, 1510

UseChannel_LF_attenuation, 1511
 WindNoiseDetector, 1511
WindNoiseDetector
 windnoise::if_t, 1511
window
 MHAFilter::smoothspec_t, 931
 overlapadd::overlapadd_if_t, 1330
 wave2spec_t, 1496
window_config
 spec2wave_if_t, 1462
 wave2spec_if_t, 1490
window_t
 MHAParser::window_t, 1135
windowselector.cpp, 1707
windowselector.h, 1707
windowselector_t, 1512
 ~windowselector_t, 1514
 get_window_data, 1514
 insert_items, 1514
 invalidate_window_data, 1515
 patchbay, 1516
 setlock, 1514
 update_parser, 1515
 updated, 1515
 userwnd, 1516
 windowselector_t, 1513
 wnd, 1515
 wndexp, 1516
 wndtype, 1515
winFO
 smooth_cepstrum::smooth_cepstrum_t,
 1441
 smooth_cepstrum::smooth_params, 1447
wlInput
 MHAFilter::fftfilter_t, 880
wlInput_fft
 MHAFilter::fftfilter_t, 880
wIRS_fft
 MHAFilter::fftfilter_t, 880
wnd
 addsndfile::level_adapt_t, 256
 audiometerbackend::level_adapt_t, 320
 fader_wave::level_adapt_t, 492
 windowselector_t, 1515
wnd_bartlett
 MHAParser::window_t, 1135
wnd_blackman
 MHAParser::window_t, 1135
wnd_funs
 mha_windowparser.cpp, 1643
wnd_hamming
 MHAParser::window_t, 1135
 wnd_hann
 MHAParser::window_t, 1135
 wnd_rect
 MHAParser::window_t, 1135
 wnd_user
 MHAParser::window_t, 1135
wndexp
 overlapadd::overlapadd_if_t, 1330
 windowselector_t, 1516
wndlen
 doasvm_feature_extraction_config, 430
 mhaconfig_t, 852
 MHAParser::mhaconfig_mon_t, 1084
 testplugin::config_parser_t, 1479
wndpos
 overlapadd::overlapadd_if_t, 1330
 wave2spec_if_t, 1490
wndtype
 windowselector_t, 1515
worker_thread_priority
 dbasync_native::db_if_t, 376
 MHAPlugin_Split::split_t, 1176
worker_thread_scheduler
 dbasync_native::db_if_t, 376
 MHAPlugin_Split::split_t, 1176
wout
 matrixmixer::cfg_t, 713
 route::process_t, 1412
wout_ac
 route::process_t, 1412
wOutput
 MHAFilter::fftfilter_t, 880
wOutput_fft
 MHAFilter::fftfilter_t, 880
wrapped_plugin_t
 matlab_wrapper::matlab_wrapper_t::wrapped_plugin_t,
 704
write
 alsa_base_t, 288
 alsa_t< T >, 293
 mha_drifter_fifo_t< T >, 758
 mha_fifo_lf_t< T >, 767
 mha_fifo_lw_t< T >, 771
 mha_fifo_t< T >, 778
 MHA_TCP::Connection, 808
 MHAFilter::blockprocessing_polyphase_resampling_t,
 869
 MHAFilter::polyphase_resampling_t, 923
 MHAJack::port_t, 982
 MHASignal::matrix_t, 1231

MHASignal::ringbuffer_t, 1239
 MHASignal::uint_vector_t, 1258

write_buf
 overlapadd::overlapadd_t, 1334
 spec2wave_t, 1464

write_event
 MHA_TCP::Connection, 809

write_float
 mha_parser.cpp, 1608

write_ptr
 mha_fifo_t< T >, 781

write_thread
 plugins::hoertech::acrec::acwriter_t, 1382
 wavwriter_t, 1501

write_to_table
 cpupload::cpupload_cfg_t, 366

write_wave
 mhasndfile.cpp, 1687
 mhasndfile.h, 1688

writeaccess
 MHAParser::base_t, 1037

writer_started
 mha_drifter_fifo_t< T >, 761

writer_xruns_in_succession
 mha_drifter_fifo_t< T >, 762

writer_xruns_since_start
 mha_drifter_fifo_t< T >, 761

writer_xruns_total
 mha_drifter_fifo_t< T >, 761

writethread
 plugins::hoertech::acrec::acwriter_t, 1383
 wavwriter_t, 1503

Writing openMHA Plugins. A step-by-step tutorial, 6

wtype
 MHAParser::window_t, 1137

wtype_t
 MHAParser::window_t, 1135

X

gsc_adaptive_stage::gsc_adaptive_stage,
 541

MHA_TCP::OS_EVENT_TYPE, 813

MHAFilter::adapt_filter_state_t, 864

x

doasvm_classification, 422

gsc_adaptive_stage::gsc_adaptive_stage,
 541

xfun
 MHATableLookup::xy_table_t, 1285

xi_est

smooth_cepstrum::smooth_cepstrum_t,
 1443

xi_min
 smooth_cepstrum::smooth_cepstrum_t,
 1441

xi_min_db
 smooth_cepstrum::smooth_cepstrum_if_t,
 1436

smooth_cepstrum::smooth_params, 1446

xi_ml
 smooth_cepstrum::smooth_cepstrum_t,
 1442

xi_opt_db
 smooth_cepstrum::smooth_cepstrum_if_t,
 1437

smooth_cepstrum::smooth_params, 1447

xiOpt
 noise_psd_estimator::noise_psd_estimator_t,
 1314

smooth_cepstrum::smooth_cepstrum_t,
 1444

xiOptDb
 noise_psd_estimator::noise_psd_estimator_if_t,
 1311

xmax
 MHATableLookup::linear_table_t, 1279

xmin
 MHATableLookup::linear_table_t, 1278

Xs
 MHAFilter::fftfilterbank_t, 885

xw
 MHAFilter::fftfilterbank_t, 885

xy_table_t
 MHATableLookup::xy_table_t, 1282

xyfun
 MHATableLookup::xy_table_t, 1285

Y

gsc_adaptive_stage::gsc_adaptive_stage,
 541

y

doasvm_classification, 422

gsc_adaptive_stage::gsc_adaptive_stage,
 542

y0
 dc_simple::dc_t::line_t, 401

y_previous
 rt_nlms_t, 1416

yfun
 MHATableLookup::xy_table_t, 1285

Yn
 MHAFilter::complex_bandpass_t, 875

MHAFilter::iir_ord1_real_t, [903](#)
YPrew
adaptive_feedback_canceller_config, [249](#)
Ys
MHAFilter::fftfilterbank_t, [886](#)
yw
MHAFilter::fftfilterbank_t, [886](#)
yw_temp
MHAFilter::fftfilterbank_t, [886](#)

zeropadding
wave2spec_if_t, [1491](#)
zeros
ac2wave_if_t, [188](#)
zerowindow
overlapadd::overlapadd_if_t, [1330](#)